나만의 인공지능 프로그래밍

kt × MechaSolution

나만의 인공지능 프로그래밍

초판발행 2019년 8월 5일
지 은 이 메카솔루션
펴 낸 이 메카솔루션
펴 낸 곳 메카솔루션
주 소 대구광역시 달서구 성서공단로 11길 62, 2호관 323
전 화 053)588-4080
팩 스 053)289-5499
등록번호

편 집 메카솔루션
디 자 인 메카솔루션
마 케 팅 메카솔루션

홈페이지 www.mechasolution.com
이 메 일 officems4080@gmail.com

KT AI 코딩팩 교구는 별도 판매합니다.
구 매 처 053)588-4080
홈페이지 www.mechasolution.com

들어가기 전

기술의 발전으로 사람들은 인간과 소통이 가능한 컴퓨터를 꿈꿔왔습니다. '말'은 사람에게 가장 익숙한 소통 방식이자 가장 기본적인 소통 방식입니다. 최초의 컴퓨터, 약어로 ABC라고 불리는 아타나 소프-베리 컴퓨터(Atanasoff - Berry Computer)는 1937년부터 1942년에 개발되었습니다. ABC를 시작으로 컴퓨터는 성능의 면에서나, 크기의 면에서 굉장한 발전을 해왔습니다. 컴퓨터가 발전함에 따라 사람들은 인간에게 복잡한 방식보다는 직관적이고 쉬운 방식을 찾기 시작했습니다. 이러한 과정 속에서 자연스레 인간을 흉내 낼 수 있는 컴퓨터를 연구하기 시작했고, 인공지능이라는 개념이 등장했습니다.

현재 우리에게는 익숙한 인공지능 음성비서는 인간을 모방하려는 노력의 결과물입니다. 이를 이용해 기가 지니, 카카오 미니, 에코 등과 같은 여러 인공지능 스피커들이 출시되었습니다. KT AI 코딩팩은 인공지능 스피커를 누구나 쉽게 만들 수 있도록 제작된 키트입니다. 본 교재에서 나오는 라즈베리 파이는 다른 마이크로컴퓨터 기판과 달리 PC로 사용이 가능합니다. 또한 파이썬은 명령줄에서 동작하는 작은 프로그램부터 GUI를 사용한 본격적인 애플리케이션까지 모두 개발할 수 있는 강력한 프로그래밍 언어입니다. 이러한 라즈베리파이와 파이썬을 이용하여 나만의 음성인식 AI 스피커를 제작하게 됩니다. 제공되는 예제 프로그램을 따라 하면 누구나 쉽게 제작할 수 있습니다. 6단원의 '6단원 제목'을 참고하여 여러 가지 응용 콘텐츠로 다양한 메이킹을 할 수도 있습니다.

집필 기간 내내 가장 초점을 두었던 부분은 '누구나 알기 쉽게'였습니다. 중고급 교육 교재로 책을 집필하긴 했지만, 프로그래밍이 아직은 어색한 분들도 쉽게 따라 할 수 있도록 내용과 디자인을 신경 써서 진행하였습니다. 어렵고 복잡한 프로그래밍이 아닌 즐기면서 만들어보는 음성인식 AI 스피커 만들기 안내서가 되었으면 좋겠습니다. 이 한 권의 책이 많은 학생들과 독자들에게 조금이라도 도움이 되기를 바랍니다. 즐거운 프로그래밍 하세요!

임홍준

책 소개

Q 누구를 위한 책인가요?

A 중급 책이 고등~대학까지의 대상의 책이었다면초급 책은 초등~중등까지의 대상으로 만들어져요. 초등학교 고학년~중2 까지라고 생각하는데 애런과 데이빗이 더 자세히 알고 있어요.

Q '인공지능 프로그래밍'이 뭐예요?

A 이번에 만드는 책은 인공지능 프로그래밍 책은 아니에요. 인공지능 프로그램 일이라면 컴퓨터가 스스로 학습하여 결과를 만들어 내는 것을 말해요. 예를 들면 대북, 날씨 예측 등과 같은 음성인식 또한 인공지능 프로그램으로 이루어져 있지만 책에서 설명한 내용은 음성인식하는 프로그램을 만든다기보다는 이미 구현되어 있는 인공지능을 사용해 인공지능 스피커를 활용하는 방법을 알아보는 것이에요.

Q 왜 라즈베리파이와 파이썬을 사용하나요?

A 라즈베리 파이를 이용하는 이유는 여러 가지가있는데 첫 번째로는 가격이 있어요. 컴퓨터 본체를 구매한다고 했을 때 쓸 만한 컴퓨터를 사겠다고 하면 약 20~30만 원 정도가 들어요. 하지만 라즈베리 파이는 4~6만 원이면 구매할 수 있어요 두 번째로는 GPIO 즉 전자회로를 통한 입출력이 가능하며 아두이노를 사용하는 것과 같이 LED를 키거나 모터를 돌릴 수 있어요. 세 번째는 낮은 전력을 사용한다는 점이 있어요. 컴퓨터를 사용한다면 220V 전원과 큰 용량의 파워서플라이를 사용해야 하지만 라즈베리 파이는 보조배터리를 연결하거나 최대 10h~15w 정도의 충전기만 있다면 사용할 수 있어요.,
네 번째는 라즈베리 파이의 운영체제에 있어요. 우리가 일 번작으로 사용하는 window는 프로그래밍을 하는 데 있어 리눅스(유닉스

목차

My own artificial intelligence programming

나만의 인공지능 프로그래밍
Contents

총 6단원

1 4차산업혁명과 인공지능

2 라즈베리파이로 Python 공부하기

3 코딩팩 기본기 익히기

목차

4 코딩팩에서 웹 데이터 이용하기

5 코딩팩으로 사물인터넷 구현하기

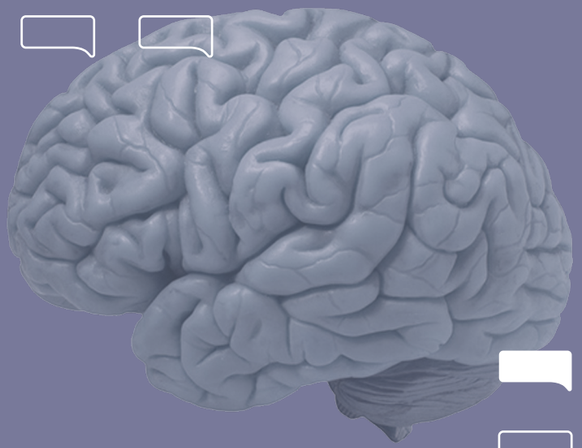
My own artificial intelligence programming

나만의 인공지능 프로그래밍
Contents

총 6단원

4차 산업혁명과
인공지능

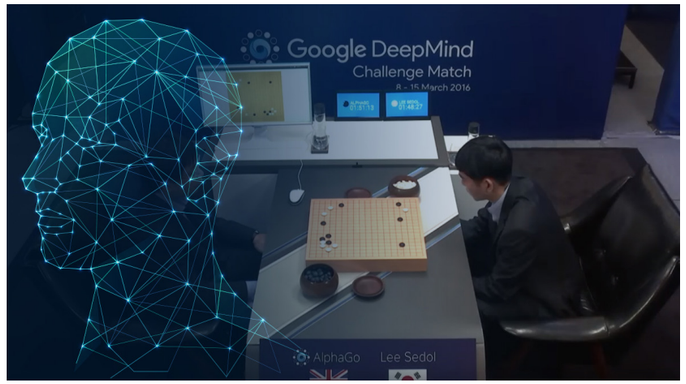
1



01
4차 산업혁명에 대해 알아보기

우리가 4차 산업혁명이라는 용어를 사용하게 된 지는 그리 오래되지 않았습니다. 2016년 1월 말 세계경제포럼World Economic Forum, 다보스 포럼에서 세계 학자들이 '4차 산업혁명의 이해' 라는 주제로 논쟁을 벌였고, 이 논쟁으로 인해 4차산업 혁명이 많은 사람들에게 널리 알려지게 되었습니다. 우리나라에서는 같은 해 3월에 있었던 바둑기사 이세돌 9단과 알파고의 바둑경기가 이슈화 되면서 많은 사람들에게 기억되었습니다.

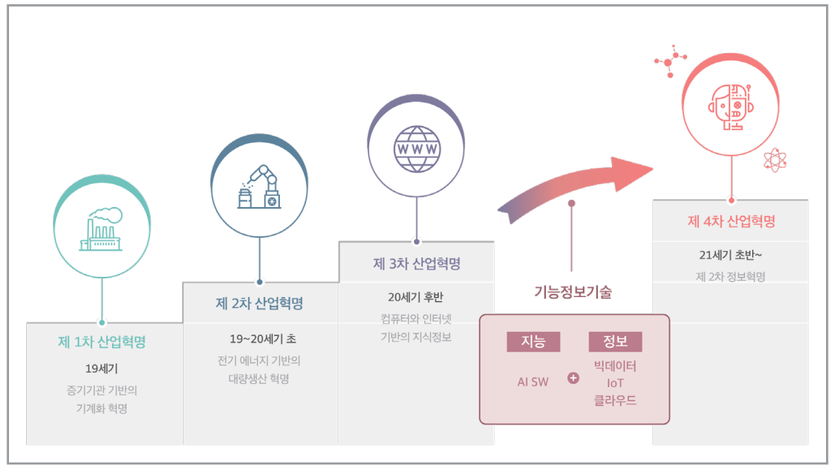
인간(이세돌 9단)과 인공지능(알파고)의 대결은 우리나라뿐만 아니라 세상의 이목과 관심을 받은 경기였습니다. 모두가 인간이 이길 것이라 예상했던 경기는 4:1이라는 결과로 인공지능의 승리로 끝났습니다. 많은 사람들은 이 결과에 충격을 받았고, 그 여파로 4차 산업혁명에 대한 관심이 증가했습니다.



이세돌·알파고 '세기의 대결'
사진출처<한국기원 제공>

자, 그럼 4차 산업혁명이란 무엇일까요? 4차 산업혁명 용어를 전 세계에 퍼뜨린 세계경제포럼 클라우스 슈밥 회장은 3차 산업혁명을 기반으로 디지털, 생물학, 물리학 등의 경계가 없어지고 모든 기술이 서로 융합되는 기술 혁명이라고 말했습니다. 너무 어려운 말인가요? 조금 쉽게 이야기해 보면 로봇, 산업장비, 의료기기 등 현실 속 제품을 뜻하는 물리적인 세계Physical System 와 인터넷, 가상공간을 뜻하는 사이버 세계Cyber System 가 기술 간 융합으로 인해 하나의 네트워크로 연결되어 서로 데이터를 공유하고 모아진 데이터를 분석 및 활용할 수 있게 되었습니다. 이러한 변화는 사물의 무인화를 가능하게 합니다.

4차 산업혁명은 '지능화', '초연결성'이라는 두 가지 특징을 가지고 있습니다. 이 두 가지 특징은 4차 산업혁명이 우리를 어떠한 사회로 이끌 것인가를 잘 보여줍니다.



4차 산업혁명의 단계

첫째, 인공지능, 빅데이터, 사물인터넷 등 4차 산업혁명의 주요 변화 동인 기술의 융합으로 기술 및 산업 구조가 '지능화' 되었습니다. 사물이 데이터를 기반으로 하는 지능을 가지게 되었고, 그 결과 앞으로의 사회는 자동화를 넘어서 무인화로 바뀌게 될 것입니다.

둘째, 가상현실, 증강현실 기술로 인해 현실세계와 가상세계가 연결되고 무선 네트워크를 통해 인간과 인간, 인간과 사물, 사물과 사물이 연결되어 실시간 소통이 가능해졌습니다. 그 결과 앞으로 지구촌에 존재하는 대부분의 온·오프라인 기기가 무선 네트워크로 연결되고, 인간의 제어 없이 주어진 상황을 파악하고 그에 걸맞은 행동을 취하는 지능화된 디지털 기기로 탈바꿈할 것입니다.

02
왜 우리는 인공지능을 공부해야 하나?

앞서 4차 산업혁명에 관해 이야기할 때 기술융합이라는 단어를 언급했습니다. 4차 산업혁명에서 기술융합은 매우 중요합니다. 4차 산업혁명을 이야기할 때 인공지능, 빅데이터, 사물인터넷, 자율주행 등의 기술들이 자연스럽게 떠오를 것입니다. 그렇다면 인공지능, 빅데이터, 사물인터넷, 자율주행 등의 기술 중 딱 한 가지의 기술만 적용해 사용하는 사례가 있을까요? 몇 가지 실사례를 보도록 하겠습니다.

1) 알파고

인공지능 빅데이터

알파고는 구글이 만들어 낸 인공지능 바둑 프로그램입니다. 바둑의 역사를 봤을 때, 지금까지 바둑 기보를 모으면 수백만 가지 혹은 수천만 가지가 될 것입니다. 여기서 기보를 바둑 데이터로 볼 수 있고 다른 말로 바둑 빅데이터라 표현 할 수 있습니다. 구글은 바둑 빅데이터를 알파고에 학습시켰고, 그 결과 알파고는 이세돌의 한 수를 입력하면 이길 가능성이 가장 높은 다음 수를 도출해 낼 수 있었습니다.

알파고 LOGO
사진출처<deepmind 사이트 제공>

2) 질병 진단 보조

인공지능 빅데이터

폐렴 환자의 X-Ray 사진 수 만장을 모아놓은 빅데이터가 있다고 가정해 봅시다. 인간이 폐렴이 의심되는 환자의 X-Ray 사진을 가지고 빅데이터 안으로 들어가 일일이 비교하며 폐렴인지 아닌지의 여부를 가리는 것은 불가능에 가까운 일입니다. 하지만 인공지능에게 폐렴 환자의 X-Ray 사진을 모아놓은 빅데이터를 학습시킨 후 이 작업을 맡긴다면 몇 초 이내에 환자의 폐렴 유무를 알려 줄 것입니다.

사진출처<www.freepik.com>

3) 스마트홈

인공지능 빅데이터 사물인터넷

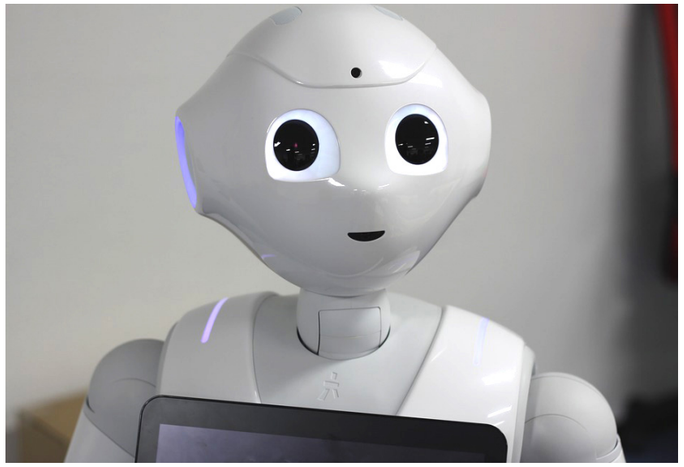
요즘 새롭게 지어지는 아파트 광고들을 보면 '첨단 기술이 접목된 아파트'라는 문구를 쉽게 접할 수 있습니다. 여기서 말하는 첨단 기술이란 무엇을 뜻하는 걸까요? 바로 인공지능과 사물인터넷의 기술융합을 뜻합니다. 목소리로 실내 전자기기를 제어하고, 외출 시 불필요하게 소비되는 에너지(수도, 전기, 냉난방 등)를 자동으로 차단합니다. 또 인공지능은 사용자의 취향(온도, 시간 등)을 학습하고, 이에 맞는 환경을 사용자에게 제공합니다.

사진출처<www.freepik.com>

4) 감정을 인식하는 로봇 페퍼(Pepper)

인공지능 빅데이터 자율주행

페퍼는 감정을 인식하는 인공지능 로봇으로 일본 소프트뱅크사에서 만들었습니다. 페퍼는 원래 가정용 로봇으로 만들어졌으나 한국, 일본, 유럽 등지에서는 접객 용도로 주로 사용되고 있습니다. 이마트에서 페퍼는 행사 정보나 휴점일 등 자주 묻는 말에 대한 답을 해주기도 하고 이마트를 돌아다니며 서성이는 고객에게 다가가 어떤 요리를 하고 싶은지 질문을 하며 고객이 답변한 요리에 대한 정보를 제공하기도 합니다.



사진출처
<www.pixabay.com>

위에서 설명한 모든 사례에 인공지능과 빅데이터 기술이 사용된 것을 확인할 수 있습니다. 필자가 인공지능과 빅데이터가 들어간 사례만 가져온 것은 아닐까? 라는 의문을 가지는 사람도 분명히 있을 것입니다. 하지만 절대 그렇지않습니다. 4차 산업혁명에서 인공지능과 빅데이터는 매우 중요한 기술이고 핵심입니다. 특히 "인공지능의 발달이 없었다면 4차 산업혁명은 없었을 수도 있다." 라고 말할 수 있을 만큼 인공지능은 4차 산업혁명에서 매우 중요한 기술입니다. 한번 생각해보도록 하겠습니다. 만약 인공지능이 없었다면 3차 산업혁명의 결과인 데이터를 어떻게 활용할 수 있었을까요? 무수히 많은 데이터를 사용하기 위해 인간이 학습하는데 얼마나 많은 시간이 걸릴까요? 한 조사 기관에 따르면 전 세계에서 하루 평균 생성되는 데이터의 양은 2.5엑사바이트(EB)로 해리포터 책 6500억권에 해당하는 분량이라고 합니다. 만약 학습을 완료했다 하더라도 그 사이 또 다른 새로운 데이터가 만들어져 있을 것입니다. 모든 사물이 네트워크를 활용하여 각종 데이터를 쏟아내고 있는 오늘날 인공지능은 이를 효과적으로 관리하고 분석하므로 새로운 차원의 산업혁명을 가능하게 합니다.

+

1) 인공지능이란?

인간의 학습능력, 추론능력, 지각능력, 자연언어의 이해능력 등을 컴퓨터 프로그램으로 실현한 기술로, 학습된 데이터를 기반으로 인간의 지능적인 행동을 모방합니다. 인공지능을 학습시키기 위해서는 기계학습Machine Learning을 많이 사용합니다. 1959년 아서사무엘이란 학자는 기계학습을 "컴퓨터에 명시적인 프로그램 없이 배울 수 있는 능력을 부여하는 연구 분야" 라고 정의했습니다. 즉 인간이 공부하듯이, 컴퓨터에 데이터들을 제공하고 학습하게 함으로써 새로운 지식을 얻게 합니다.

2) 빅데이터란?

빅데이터는 '빅(Big) + 데이터(Data)'의 단순한 합성어가 아닙니다. 빅데이터를 '어마어마하게 많은 데이터'라는 식으로 받아들이면 본질적인 의미와 가치를 잃어버리게 됩니다. 빅데이터에서의 데이터는 우리가 기존에 알고 있던 문자, 숫자, 기호로 정형화된 데이터뿐만 아니라 GPS 정보, 센서 정보 등의 반정형화된 데이터와 사진, 동영상 등의 비정형 데이터 모두를 포함하고 있습니다. 시장조사기관인 IDC는 "빅데이터 기술은 다양한 형태로 구성된 방대한 크기의 데이터로부터 경제적으로 필요한 가치를 추출할 수 있도록 디자인된 차세대 기술이다"라 정의했고, 미국의 IT 리서치 회사인 가트너는 크기Volume, 속도Velocity, 다양성Variety 3V로 빅데이터의 특징을 요약했습니다. 여기서 크기는 데이터의 물리적인 크기, 속도는 데이터 처리 능력, 다양성은 데이터의 형태를 뜻합니다. 정리해보면, 빅데이터는 단순히 대용량 데이터 그 자체만을 지칭하는 것이 아니라 그 데이터를 효과적으로 처리하고 분석해 가치를 생성하는 것에 초점을 두고 있습니다.

3) 사물인터넷이란?

모든 사물에 통신 기능을 장착해 인터넷으로 연결하여 사람과 사물, 사물과 사물간의 정보를 상호 소통할 수 있게 하는 기술입니다. 다양한 보고서에서는 사물인터넷이라는 용어의 정의를 두고 각기 다르게 정의하고 있지만 대개 다음 세 가지 공통 요소를 가지고 있습니다.

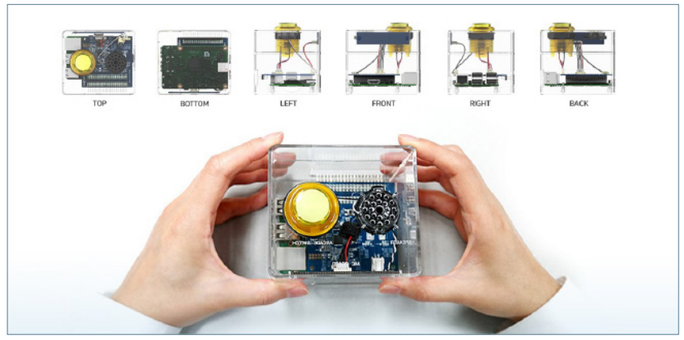
1. 각각의 사물은 '스스로 행동할 수 있는 지능'을 가져야 한다. 여기서 말하는 스스로 행동할 수 있는 지능이란 사람의 제어 없이 사물이 정보를 수집하고 수집된 결과에 따라 주체적으로 어떠한 행위를 할 수 있어야 한다는 것을 의미합니다.

2. 각각의 사물은 인터넷으로 연결되어 사람 혹은 다른 사물과 소통할 수 있어야 한다.

3. 사람 혹은 사물과의 소통의 결과로 발생 되는 정보는 새로운 가치 및 서비스를 제공할 수 있어야 한다.

03
KT AI 코딩팩

KT에서 출시한 KT AI 코딩팩(이하 코딩팩)은 어렵게 느껴지던 인공지능이라는 분야를 라즈베리파이Raspberry Pi라는 저렴한 소형 컴퓨터와 KT의 인공지능 개발 인프라를 기반으로 "누구나 쉽게", 보다 "저렴하게", 그리고 다양한 콘텐츠 공유를 통해 많은 사람들이 즐길 수 있도록 개발되었습니다. 코딩팩은 "인공지능 교육"을 위한 요소와 "메이커스 교육"을 위한 요소가 모두 담겨 있습니다. 인공지능 교육을 위해서 음성 인식, 음성 합성, 대화와 같은 요소를 직접 학습하고 개발할 수 있으며 카메라 비전을 활용한 사람과 사물을 인식하는 기능도 가능합니다. 또한, 전기·전자 기초, 소자 및 센서, 그리고 모터를 사용하여 창의적인 프로젝트를 만들 수 있는 메이커 교육도 가능하며 인공지능 요소가 가미되어 더욱 더 재미있고 창의적인 활동도 해볼 수 있습니다.



KT AI 코딩팩 제품 상세 이미지

+

코딩팩 동작 원리

코딩팩의 마이크를 통해 입력된 아날로그 음성 신호는❷ 보이스 키트 쉴드를 거치면서 컴퓨터가 처리할 수 있도록 디지털 음성 신호❸로 변환됩니다. 변환된 디지털 음성 신호는 라즈베리파이❹와 연결된 네트워크를 통해 KT AI 서버의 '서비스 처리 단계'❻로 전송됩니다. 각 서버에서는 맡은 역할에 따라 신호를 변환 혹은 해석합니다. 모든 단계를 거쳐 완성된 데이터는 사용자에게 들려주기위해 다시 네트워크를 통해 라즈베리파이로 전송되고, 라즈베리파이는 보이스키트를 통해 스피커로 데이터를 출력합니다.

❻ 서비스 처리

입력된 신호가 음성 신호인지 문자 신호인지를 판단하여 각각의 신호 형태에 맞는 서버로 신호를 전송합니다.

❼ 음성인식 서버

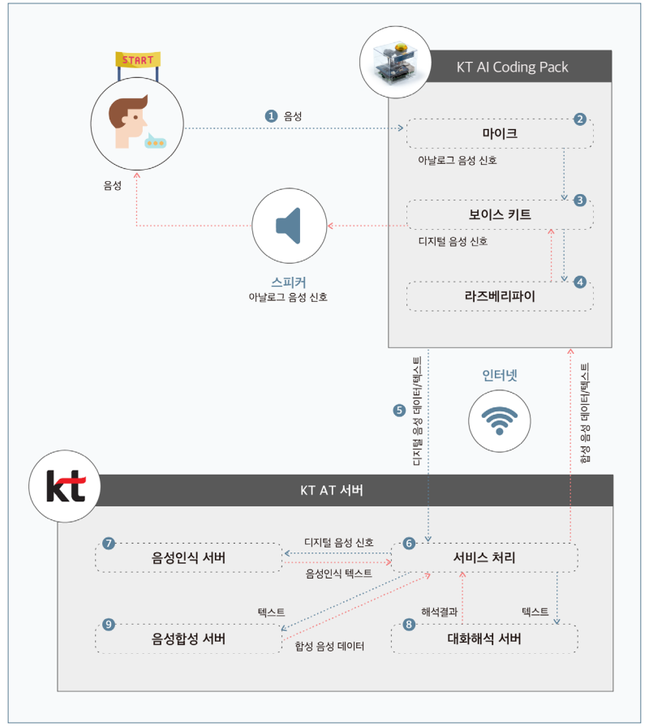
학습된 데이터를 기반으로 입력된 음성을 문자로 바꿔 다시 '서비스 처리 단계'로 전송합니다.

❽ 대화해석 서버

학습된 데이터를 기반으로 입력된 문자를 분석해 의미를 해석하고 해석결과를 '서비스 처리 단계'로 전송합니다.

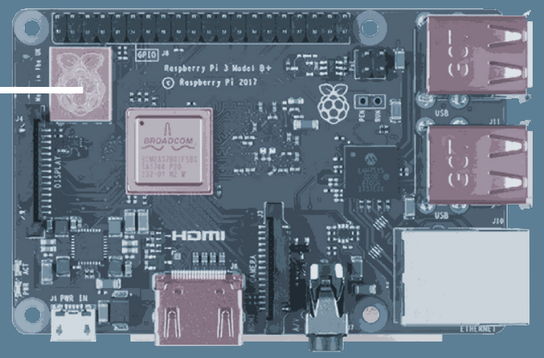
❾ 음성합성 서버

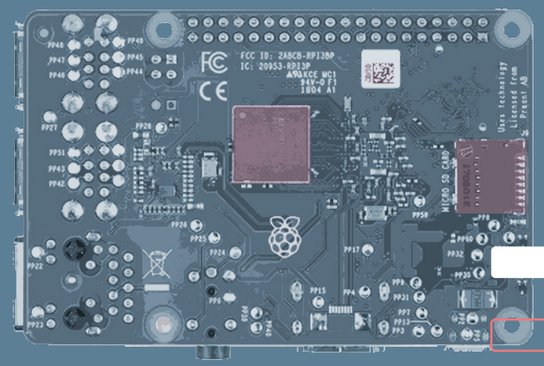
처리 결과를 음성으로 출력하기 위해 문자를 음성으로 변환하여 '서비스 처리 단계'로 전송합니다.



라즈베리파이로
Python 공부하기

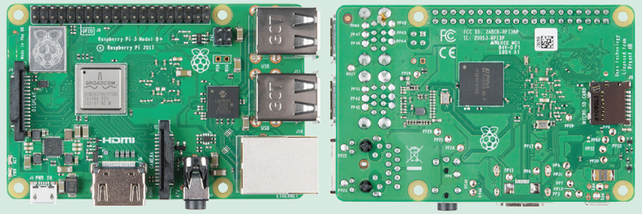
2





01
라즈베리파이와 코딩팩 준비하기

라즈베리파이는 영국 라즈베리파이 재단에서 기초 컴퓨터 과학 교육 목적으로 개발된 소형 컴퓨터입니다. 크기는 일반 신용카드와 유사하고 모니터, 키보드, 마우스를 연결하면 일반 컴퓨터가 할 수 있는 대부분의 작업을 할 수 있습니다. 물론 고성능이 요구되는 게임, 동영상 편집등 높은 사양이 요구되는 작업을 할 수 없으나 문서작업, 인터넷, 프로그래밍 학습, 사물인터넷 장치 만들기등 기본적인 작업을 하는데는 충분히 사용할 수 있습니다.



라즈베리파이 3B+

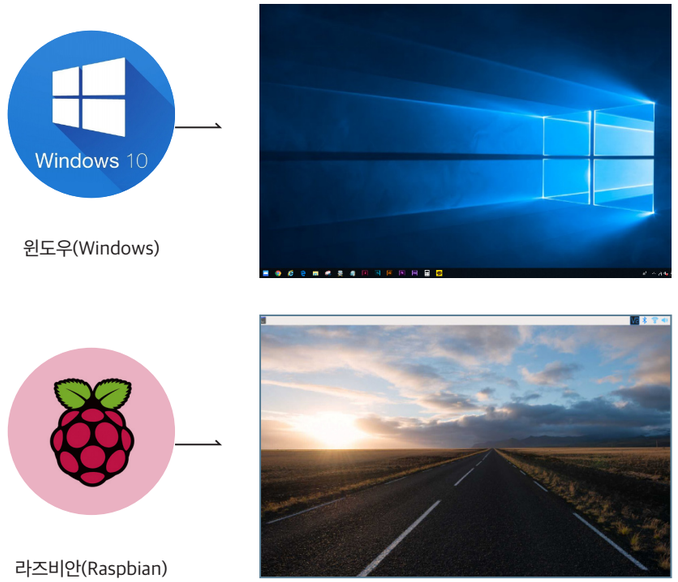
1) OS 다운로드

라즈베리파이를 사용하기 위한 필수 요소인 운영체제(OS)를 설치하는 방법에 대해 알아보도록 하겠습니다. 운영체제란 사용자가 컴퓨터를 사용할 수 있도록 중재 역할을 해주는 프로그램을 말합니다. 우리가 사용하는 컴퓨터는 대개 '윈도우(Windows)'라는 운영체제를 사용하고 있습니다. 하지만 지금부터 사용하게 될 라즈베리파이는 '라즈비안(Raspbian)'이라는 운영체제를 사용합니다. 라즈비안 운영체제는 'OS 이미지' 파일을 다운받아 라즈베리파이의 하드디스크 역할을 하는 마이크로SD 카드에 설치해서 사용할 수 있습니다.

TIP

※ 라즈비안 운영체제는 공식 홈페이지(www.raspberrypi.org)에서 무료로 다운받을 수 있습니다.

※ 이 책에서는 KT에서 제공하는 코딩팩 전용 운영체제를 사용하여 진행하였습니다. 본 단원은 라즈베리파이 전용 운영체제인 라즈비안을 사용하여 진행하셔도 무관하나, 이후에는 코딩팩을 사용해야하기에 코딩팩 전용 운영체제를 권장합니다.

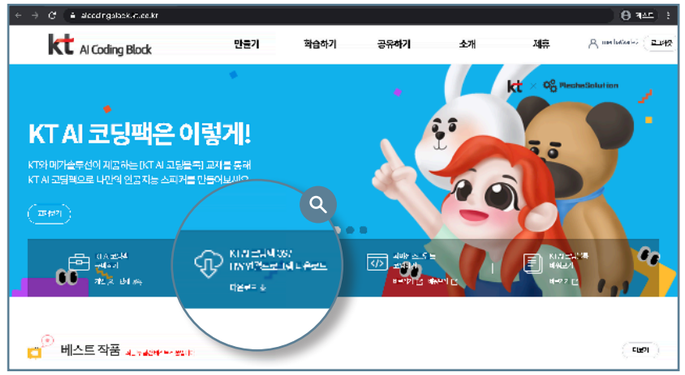


윈도우 운영체제와
라즈비안 운영체제

⚠ 이 단계는 개인 PC를 이용해 진행하시기 바랍니다.

(1) OS 이미지 다운로드

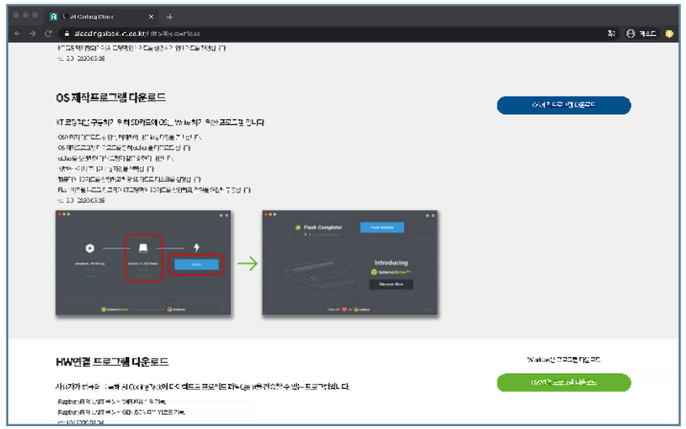
AI Coding Block 홈페이지에 접속하여 KT AI 코딩팩 OS / HW 연결프로그램 다운로드 항목을 클릭합니다.



1 OS 이미지 다운로드 버튼을 클릭하여 이미지 다운로드를 진행합니다. (인터넷 상황에 따라 속도가 달라질 수 있으며 5~10분 내로 다운로드됩니다)

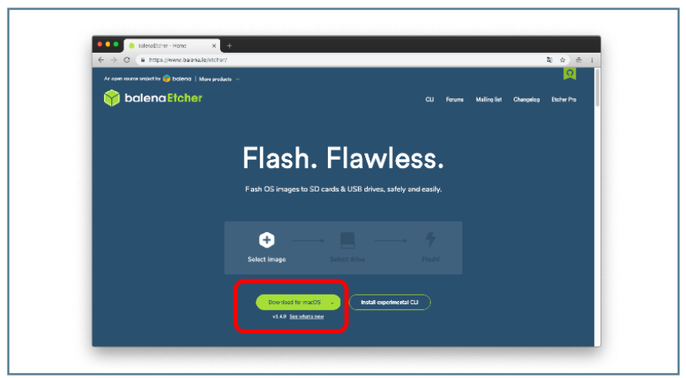
다운로드가 완료되면 압축을 풀어줍니다. (img 확장자 파일이 OS 이미지입니다.)

OS 제작프로그램 다운로드 버튼을 클릭하여 Etcher 프로그램을 다운로드할 수 있는 홈페이지로 이동합니다.



(2) OS 이미지 레코딩

이미지 레코딩 프로그램인 Etcher를 설치합니다.



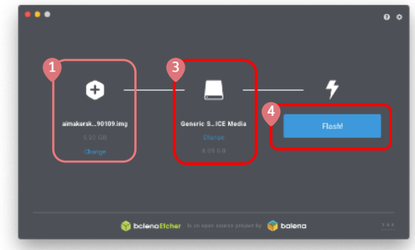
TIP

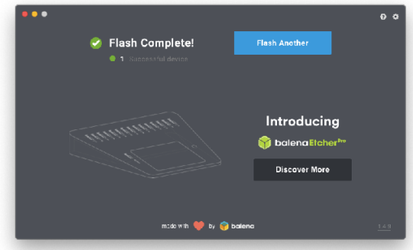
본 교재는 Etcher을 사용하였지만 사용하는 운영체제에 맞춰 다른 프로그램을 설치하여도 무관합니다. (사용하는 운영체제에 맞는 버전을 다운로드 받습니다.)
Etcher 다운로드 링크 : https://www.balena.io/etcher

1 'Select image'에서 압축을 풀어놓은 img파일을 선택하고

2 'Select drive' 버튼을 누릅니다. 앞서 PC에 연결한 3 마이크로 SD카드를 선택한 후

4 ' Flash! ' 버튼을 클릭하여 OS 이미지 레코딩을 시작합니다.

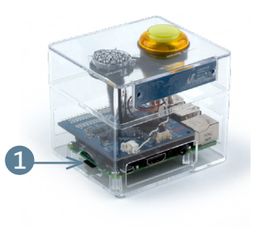




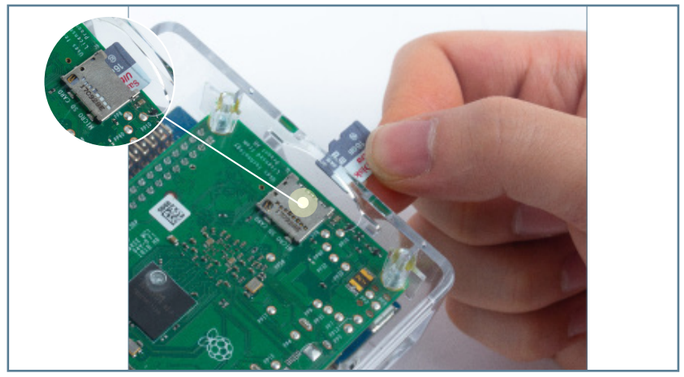
(완료)

2) 라즈베리파이 연결하기

1 마이크로SD 카드 삽입

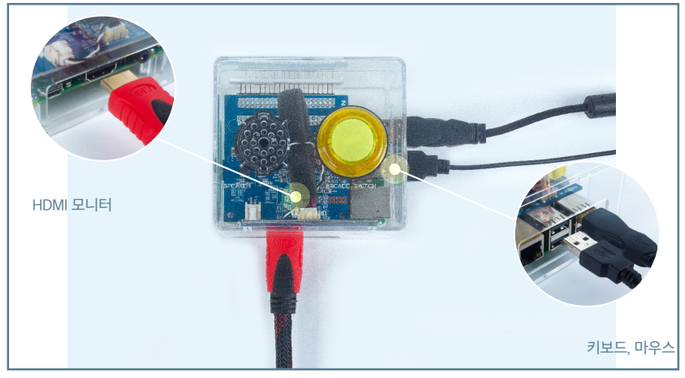


1에 OS가 설치된 마이크로 SD 카드를 삽입합니다.



2 케이블 연결

라즈베리파이를 사용하기 위해서는 HDMI 모니터, 키보드, 마우스와 같은 추가 악세사리가 필요합니다. 우측 사진과 같이 연결해줍니다.



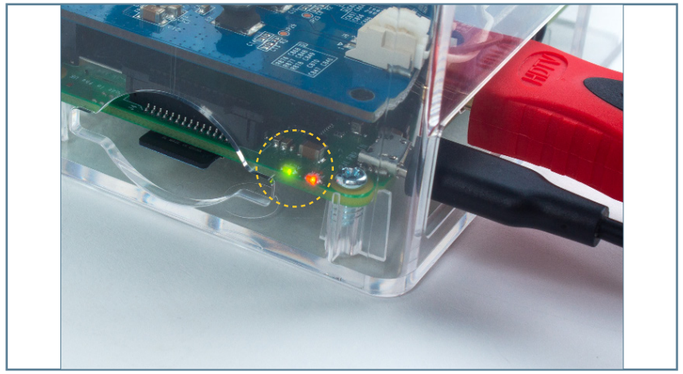
TIP

키보드와 마우스를 연결할 때 어떤 USB 포트를 사용해도 무관합니다.

3 전원 연결

⚠ 꼭 HDMI와 마이크로 sd카드를 연결 후 전원공급을 해주세요.

표준 5핀 어댑터를 라즈베리파이에 꼽아 전원공급을 해주세요. 전원이 제대로 들어가고 올바른 부팅을 할 때에는 라즈베리파이의 빨간색 LED는 계속 켜져 있고, 초록색 LED는 깜빡거리며 불이 들어옵니다.



TIP

라즈베리파이가 정상적으로 작동하지 않을 경우, 다음을 확인해주세요.
- 표준 5핀 어댑터를 뽑고 다시 연결합니다.
- SD카드가 제대로 꽂혔는지 확인합니다.
- SD카드에 OS 이미지가 제대로 레코딩 되었는지 확인합니다.

02
파이썬(Python) 이란?

파이썬은 1990년 네덜란드의 '귀도 반 로섬'이 개발한 인터프리터 언어로 개발자뿐 아니라 초보자들에게도 많은 사랑을 받고 있습니다. 파이썬이라는 이름을 '귀도 반 로섬이' 좋아하는 코미디 프로그램 "몬티 파이썬의 날아다니는 서커스Monty Python's Flying Circus"에서 따왔다고 합니다. 파이썬의 사전적인 의미는 고대 신화에 나오는 파르나소스산 동굴에 살던 큰 뱀의 이름입니다. 대부분의 파이썬 책 표지와 아이콘이 뱀 모양으로 그려져 있는 이유가 여기에 있습니다.

python Logo
사진출처<Python 홈페이지>

프로그래밍 언어는 파이썬 뿐만 아니라 C, C++, 자바 등 많은 언어가 존재합니다. 각 프로그래밍 언어마다 장단점이 있지만, 파이썬이 가장 널리 사용되고 있는 이유는 다른 프로그래밍 언어에 비해 어렵지 않고, 표현하는 구조도 사람이 대화하는 형식으로 되어 있어 초보자도 쉽게 배울 수 있기 때문입니다. 또한 누구든지 무료로 사용할 수 있도록 공개한 것도 대중화에 큰 역할을 했습니다.

1) 인간의 사고와 닮은 파이썬

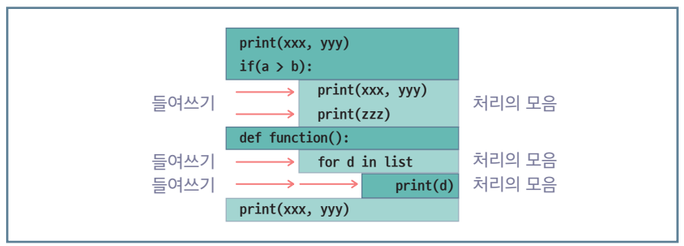
아직 코딩이 익숙하지 않은 초보자들이 C언어, C++, 자바 등의 언어를 사용해 프로그래밍 한 소스 코드를 보면 어떻게 동작하는지, 어떤 내용을 담고 있는지 이해하기 다소 어렵습니다. 하지만 파이썬은 사람이 생각하는 방식을 그대로 표현하기 위해 노력했습니다. 아래 파이썬 소스 코드를 보면 이 말이 쉽게 이해될 것입니다.

```
if 4 in [1, 2, 3, 4]: print("4가 있습니다.")
```

위 소스 코드는 파이썬에서 사용되는 조건문으로, 영어 문장 읽듯이 해석해보면 '만약 4가 1, 2, 3, 4 중에 있다면 "4가 있습니다."를 출력하세요.' 라고 해석됩니다. 이와 같이 파이썬은 사람이 대화하는 형식으로 되어 있어 프로그래밍 문법을 모르더라도 무엇을 뜻하는지 쉽게 알 수 있습니다.

2) 간결하게 표현되는 파이썬

프로그래밍이 간결하다는 것은 소스코드가 잘 정리되어 있다는 것을 뜻합니다. 프로그래밍은 소스 코드를 작성하는 사람에 따라 같은 내용도 다르게 표현되기 때문에 다른 사람이 프로그래밍한 것을 이해하는데 어려움이 있을 수 있습니다. 하지만 파이썬은 소스 코드를 정리해서 작성하지 않으면 제대로 동작하지 않기 때문에 누구나 같은 방법으로 소스 코드를 정리해 가며 프로그래밍 할 수밖에 없습니다.



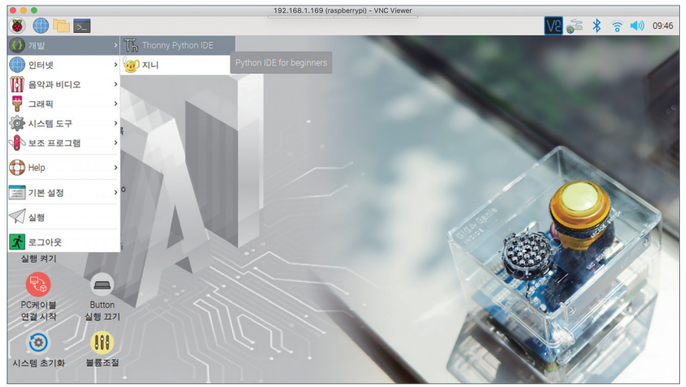
TIP

여기서 소스코드를 정리한다는것은 줄을 맞추어 프로그래밍하는 것을 뜻합니다.

03
통합 개발 환경 알아보기

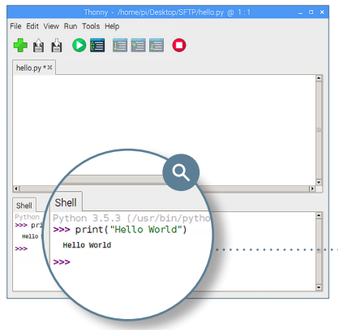
대부분의 프로그래밍 언어는 프로그램 작성과 실행을 돕는 기능을 가진 고유 어플리케이션이 있습니다. 이러한 어플리케이션을 **IDE** Integrated Development Environment, 통합개발환경 라고 부릅니다. 라즈베리파이 운영체제인 라즈비안에도 'Thonny Python IDE'라는 IDE를 사용하여 파이썬 프로그래밍을 할 수 있습니다.

'Thonny Python IDE'는 메인 화면 좌측 상단 라즈베리파이 로고 > '개발' 카테고리 안에서 찾을 수 있습니다.



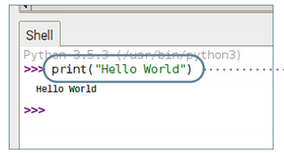
1) 'Hello World' 출력하기

Thonny Python IDE를 실행하면 아래와 같은 창이 열립니다. 창 하단 Shell 영역 안에 있는 >>>(프롬프트) 뒤에 프로그램을 작성하고 Enter 키를 누르면 프로그램이 실행됩니다. 파이썬 에서는 Shell 영역 안에 프로그램을 작성하는 것을 **대화형 세션**이라 부릅니다.



>>>(프롬프트) 뒤에 print('Hello World')를 입력한 후 Enter 키를 누르면 바로 아랫줄에 Hello World가 출력되는 것을 확인할 수 있습니다.

print() 내장 함수를 사용하여 문자를 표시하기 위해서는 print( ) 괄호 안에 표시할 문자열을 작은 따옴표(') 또는 큰 따옴표(")로 감싸주어야 합니다.



print('문자열')
⋮
작은따옴표(') 또는 큰따옴표(")
안에 문자열 입력

TIP

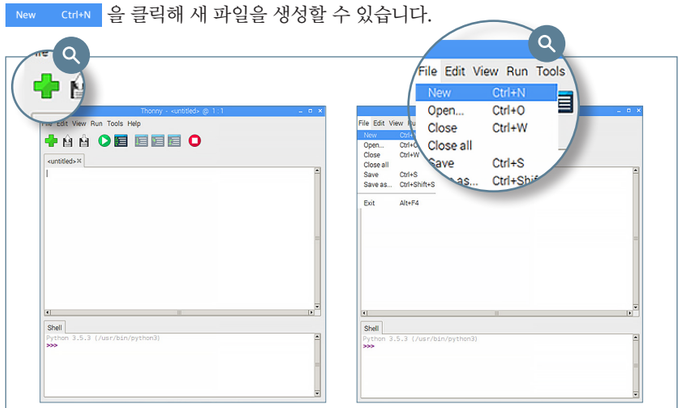
서로 다른 따옴표를 사용할 수 없습니다. (예를 들어 print('Hello World")처럼 앞에는 작은 따옴표 뒤에는 큰 따옴표 사용이 불가능합니다.)

2) 프로그램 파일 생성하고 저장하기

대화형 세션은 프로그램 결과 확인이 빠르다는 장점을 가지고 있으나 복잡한 프로그램을 작성할 때는 적합하지 않습니다. IDE 에디터로 프로그램 파일을 생성하고 저장해두면 프로그램을 반복해서 실행시키거나 나중에 편하게 수정 할 수 있습니다.

1 새 파일 생성하기

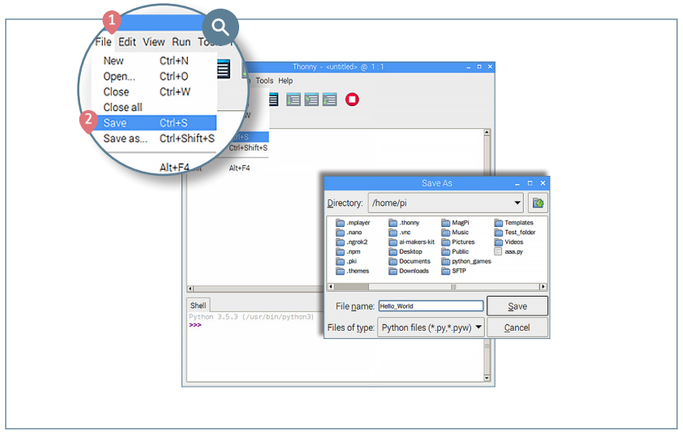
(1) 화면 왼쪽 상단 ✚ 아이콘을 클릭하거나 (2) 화면 상단 옵션창에서 File > New Ctrl+N 을 클릭해 새 파일을 생성할 수 있습니다.



(1) (2)

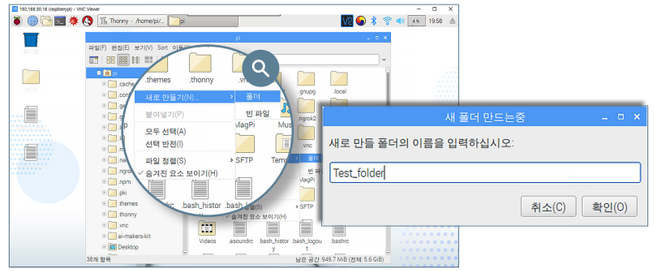
2 파일 저장하기

화면 상단 옵션 창에서 1 File > 2 Save Ctrl+S (저장) 혹은 Save as.. Ctrl+. (다른 이름으로 저장)를 클릭해 파일을 저장합니다. 이때 파일은 .py 확장자로 저장됩니다.



TIP

IDE에서 Save(저장)를 누른 후 새 폴더를 만들 수 없기 때문에 파일을 저장할 폴더를 미리 저장합니다.
※ 프로그램 파일명을 작성할 때는 영문, 숫자, 기호만 쓰는 것이 좋습니다. 만약 영문 이외의 다른 언어나 특수문자를 사용할 경우 오류가 발생할 수 있습니다.

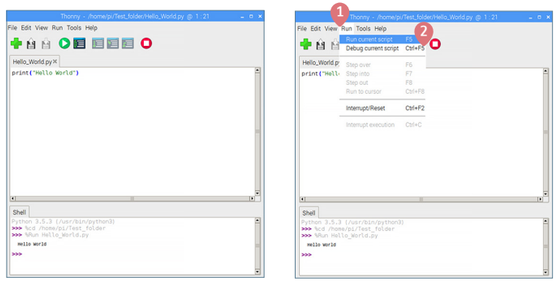


3) 프로그램 실행하기

작성한 프로그램을 실행시키기 위해서는 아래 세 가지 방법 중 하나를 선택해서 사용할 수 있습니다. 결과는 Shell 창에서 확인 할 수 있습니다.

1 IDE 상단 ▶ 아이콘을 클릭하여 프로그램을 실행시킵니다.

2 IDE 상단 옵션중 1 Run > 2 Run current script F5 을 클릭하여 프로그램을 실행시킵니다.



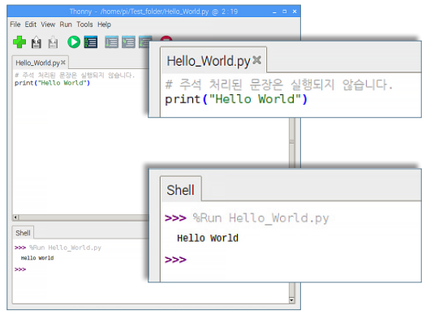
3 프로그램 실행 단축키 'F5'를 눌러 프로그램을 실행시킵니다.

TIP

※ 프로그램 실행시 작성한 프로그램에 오류가 있다면 오류가 발생한 줄과 틀린 부분을 알려줍니다.

4) 주석 처리하기

주석이란 코드에 대한 이해를 돕는 설명을 적거나 디버깅을 위해 작성하는 일종의 메모입니다. 주석 처리가 된 문장은 프로그램에서 실행되지 않습니다. 파이썬에서 주석은 #을 사용합니다. #에서부터 다음 줄의 앞까지 주석으로 처리합니다.

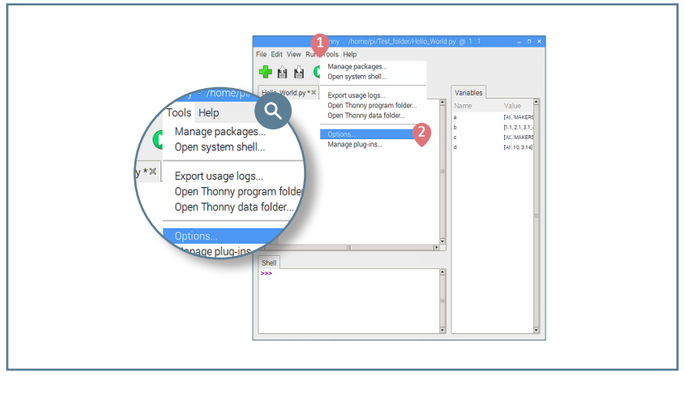


5) IDE 설정하기

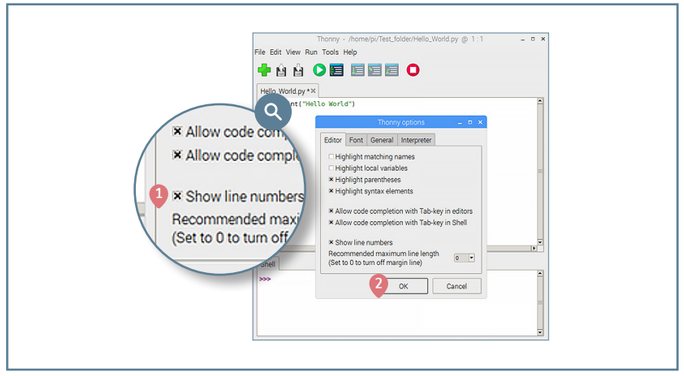
1 줄 번호 표시

프로그램을 작성할 줄 번호를 표시하는 기능입니다.

(1) 화면 상단 옵션 창에서 1 Tools > 2 Options... 를 선택하여 Thonny options 창을 엽니다.



(2) ☐ Show line numbers 를 체크 하고 OK를 클릭합니다.

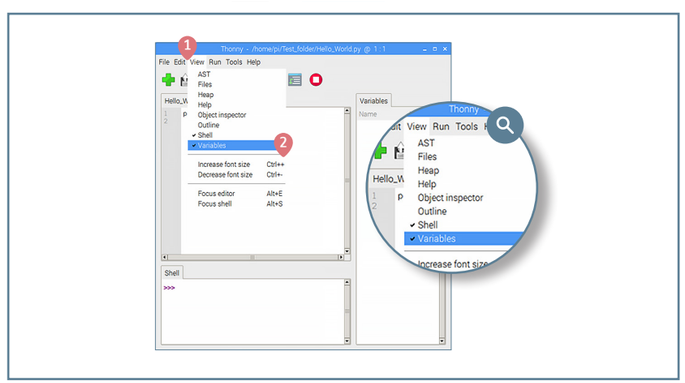


(3) 줄 번호가 표시되는 것을 확인합니다.

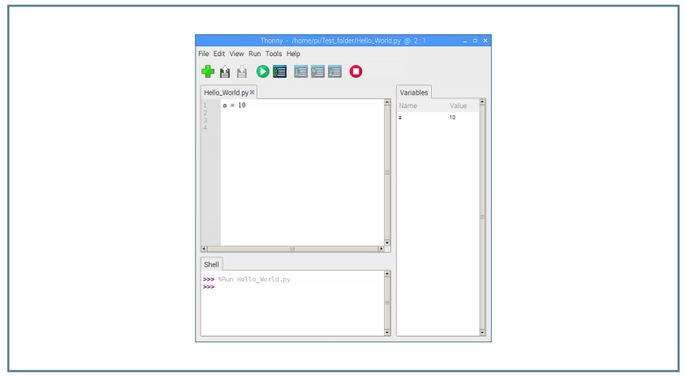
② 변수 표시

변수에 저장된 값을 확인할 수 있는 창을 추가하는 기능입니다.

(1) 화면 상단 옵션 창에서 ① View > ② Variables 를 선택합니다.



(2) 변수를 선언하여 Variables 창에 변수가 추가되는 것을 확인 합니다.



③ 단계별로 디버깅하기

IDE는 화면 상단 네 개의 아이콘을 사용하여 프로그램을 단계별로 실행시킬 수 있습니다.

TIP

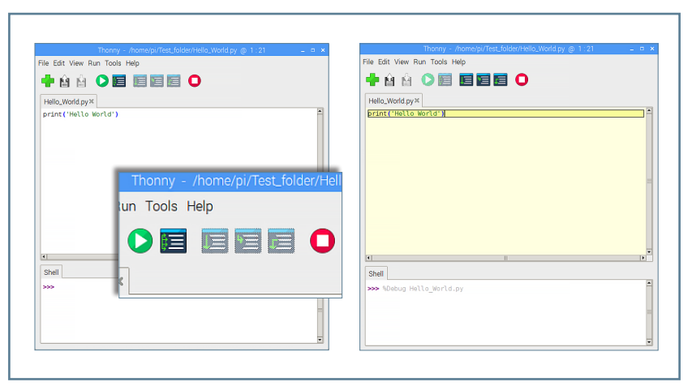
<Debug current script>

<Step over>

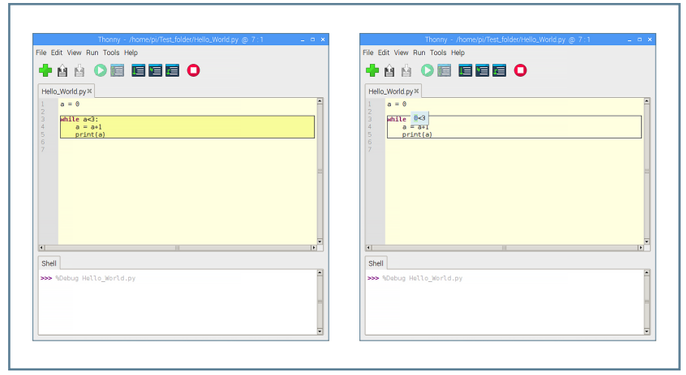
<Step into>

<Step out>

(1) 'Debug current script'을 클릭하면 옆에 있는 세 개의 아이콘이 활성화되면서 단계별로 디버깅을 시작합니다.



(2) 'Step over'는 클릭할 때마다 코드 한 줄씩 실행시킵니다. 이때 'Step over'은 함수, 반복문도 한 덩어리로 보고 결과를 한번에 출력합니다.



(3) 'Step into'는 'Step over'보다 세세하게 결과를 확인시켜줍니다. 'Step into'는 함수, 반복문을 한 덩어리로 보지 않고, 내부로 들어가 실제 프로그램이 실행하듯이 결과를 출력합니다.

(4) 'Step out'는 'Step into'을 사용해 내부로 들어간 함수나 반복문을 빠져 나와 호출한 곳으로 돌아갑니다.

04
변수와 연산

1) 파이썬의 변수

프로그램에는 반드시 데이터를 보관하는 저장소가 필요합니다. 예를 들어 학생들의 수학성적을 입력해 평균을 계산하여 출력하는 프로그램을 만들어 본다고 가정해보겠습니다. 학생들이 컴퓨터에 수학성적을 입력하면 어딘가에 저장해야지만 평균을 계산할 수 있습니다. 이때 '변수'라는 데이터를 담는 상자를 사용해 데이터 값을 저장하고 보관할 수 있습니다.

1 변수 선언하고 사용하기

파이썬은 C, 자바 등과 같은 프로그램 언어보다 변수를 간편하게 사용할 수 있습니다. 다른 프로그램 언어에서 변수를 선언할 때는 변수의 형태가 정수, 부동소수, 문자이냐에 따라 자료형을 앞에 함께 선언을 해주어야 하지만, 파이썬에서는 자료형을 지정해 주지 않아도 변수를 선언할 수 있습니다. 파이썬의 자료형에 대해서는 다음 절에서 자세히 다루도록 하겠습니다.

변수 선언방법

```
변수명 = 값
```

#

연습해보기

1) 변수 a에 3.14를 저장하고 출력하는 프로그램을 작성하시오.

해설

Variable.py ..... 파일이름

```
a = 3.14        ..... 변수 선언
print(a)        ..... 변수 사용
```

출력결과

```
3.14
```

변수 a에 3.14가 저장 되어 있기 때문에 a를 출력하면 3.14가 출력됩니다.

2) 변수 b에 'KT AI 코딩팩'을 저장하고 출력하는 프로그램을 작성하시오.

해설

Variable.py

```
b = 'AI Coding Pack'
print(b)
```

출력결과

```
AI Coding Pack
```

변수 b에 AI Coding Pack이 저장 되어 있기 때문에 b를 출력하면 AI Coding Pack이 출력됩니다.

❷ 변수 이름 짓기

변수명과 뒤에서 배울 함수명은 식별자의 일종입니다. **식별자**identifier란 사람이 '철수', '영희'와 같이 이름으로 식별하듯 변수와 함수를 식별하는 역할을 합니다. 식별자를 지정할 때는 몇 가지 규칙을 따라야 합니다.

1. 식별자는 영문, 숫자, 언더바(_)로 이루어진다.
2. 식별자의 중간에 공백이 들어가서는 안된다.
3. 식별자의 첫 글자는 영문 혹은 언더바(_)로 '시작'해야 된다.
   즉 식별자는 숫자로 시작할 수 없다.
4. 식별자는 대문자와 소문자를 구별한다. 변수 a와 변수 A는 서로 다른 변수이다.
5. 키워드(예약어)와 똑같은 식별자는 사용할 수 없다.

키워드(예약어)는 프로그램에서 제공하는 기능들을 위해 미리 지정해둔 단어들입니다. 파이썬에서는 총 35개의 키워드가 있고 모두 소문자로 이루어져 있습니다. 따라서 키워드를 변수명 혹은 함수명으로 사용할 수 없습니다.

False	raise	lambda	global
await	True	try	not
else	class	as	with
import	finally	def	async
pass	is	from	elif
None	return	nonlocal	if
break	and	while	or
except	continue	assert	yield
in	for	del	

파이썬에서 제공하는 35개의 키워드

TIP

변수명은 사용자 마음대로 정할 수 있지만 저장된 데이터가 어떤 데이터인지를 나타내 주는 단어로 변수명을 정하면 나중에 관리하기가 쉽습니다.

ex)
average = 평균값
sensor_data = 센서측정 값
temperature = 온도값

2) 파이썬의 연산

컴퓨터라는 명칭은 '계산한다'라는 뜻을 가진 라틴어 'Computare'에서 유래되었습니다. 단어에서 알 수 있듯 '컴퓨터는 무언가를 연산하고 그 결과를 저장, 보관했다가 필요할 때 다시 사용할 수 있도록 하는 것'이 컴퓨터의 기본 용도입니다. 프로그램에서는 산술 연산자, 복합 대입 연산자를 사용하여 계산합니다. 프로그램에서 사용되는 연산자들에 대해 알아보도록 하겠습니다.

❶ 산술 연산자 사용하기

연산을 한다고 했을 때 가장 중요한 것은 덧셈, 뺄셈, 곱셈, 나눗셈과 같은 사칙연산일 것입니다. 아마 이 서적의 독자분들은 이러한 사칙연산에 대해 잘 알고 있을 것입니다. 하지만 우리가 알고 있는 기본적인 사칙연산 기호와 파이썬에서 사용하는 산술 연산자의 모양과 의미는 조금 다릅니다. 아래의 표를 참고하여 파이썬의 연산에 대해 알아보도록 하겠습니다.

연산기호	의미	예시	결과
+	더하기	9+7	16
-	빼기	9-7	2
*	곱하기	9*7	49
/	나누기	9/7	1.285…
**	제곱(x^y)	9**7 9(x)의 7(y)승을 의미함	4782969
//	정수로 나누었을 때의 몫	9//7	1
%	정수로 나누었을 때의 나머지	9%7	2
( )	다른 계산보다 괄호 안을 먼저 계산	1*(1+4)	5

파이썬의 연산기호

Calculation.py

```
num1 = 10
num2 = 3

print(num1 + num2)
print(num1 - num2)
print(num1 * num2)
print(num1 / num2)
print(num1 // num2)
print(num1 ** num2)
print(num1 % num2)
```

출력 결과

```
13
7
30
3.3333333333333335
3
1000
1
```

프로그래밍에 대한 이해가 있으신 분들은 num1 / num2의 결괏값을 '3'이라 예상하게 됩니다. 그 이유는 수학적으로 정수형끼리의 결괏값은 정수형으로 계산되기 때문입니다. 하지만 파이썬에서는 일반적인 계산처럼 나머지 해당하는 소수점 자릿수도 함께 계산됩니다. 만약 C언어, 자바 등 다른 프로그램 언어와 같은 결과를 얻고 싶다면 '/' 연산자 대신 '//' 연산자를 사용하면 됩니다.

② 연산 순서 정하기

수학에서는 대괄호[ ], 중괄호{ }, 소괄호( )를 이용하여 연산의 우선순위를 정합니다. 하지만 파이썬에서는 소괄호( )만 이용하여 우선순위를 정합니다.

파이썬 연산 순서 사용 예시

수학	파이썬
40 - [5 x {16 - (65 - 25)}]	40 - (5 x (16 - (65 - 25)))

③ 복합 대입 연산자 사용하기

단순한 연산을 할 때는 산술 연산자 대신 복합 대입 연산자를 사용하기도 합니다. 본인에게 더 편한 스타일을 사용하겠지만 만약 다른 사람이 작성한 프로그램을 분석해야 하는 경우 복합 대입 연산자를 사용할 수 있으니 의미는 알아 두어야 합니다.

파이썬 단순 연산자와 복합 대입 연산자 비교

연산자	복합 대입 연산자
a += b	a = a + b
a -= b	a = a - b
a *= b	a = a * b
a /= b	a = a / b
a //= b	a= a // b
a **= b	a = a ** b
a %= b	a = a % b

Calculation.py

```
# 세미콜론(;)을 사용하면 줄을 바꾸지 않고 이어서 프로그래밍할 수 있습니다.
num1 = 10; num2 = 10; num3 = 10; num4 = 10
num5 = 10; num6 = 10; num7 = 10

num1 += 3      # num1 = num1 + 3과 같은 의미입니다.
num2 -= 3      # num2 = num2 - 3과 같은 의미입니다.
num3 *= 3      # num3 = num3 * 3과 같은 의미입니다.
num4 /= 3      # num4 = num4 / 3과 같은 의미입니다.
num5 //= 3     # num5 = num5 // 3과 같은 의미입니다.
num6 **= 3     # num6 = num6 ** 3과 같은 의미입니다.
num7 %= 3      # num7 = num7 % 3과 같은 의미입니다.

print(num1); print(num2)
print(num3); print(num4)
print(num5); print(num6)
print(num7)
```

출력 결과

```
13
7
30
3.3333333333333335
3
1000
1
```

#

연습해보기

1) 1학년 4반 학생들의 수학성적은 아래와 같습니다. 1학년 4반 학생들의 수학성적 평균을 계산하고 출력하는 프로그램을 작성하시오.

해설

Average.py

```
print((37 + 93 + 59 + 82) / 4)
```

출력결과

```
67.75
```


05 자료형

자료형이란 데이터의 형태를 말합니다. 10과 같은 정수, 3.14와 같은 실수, “Hello World”와 같은 문자 등 프로그래밍에서 사용하는 데이터의 형태는 다양할 수 있습니다. 프로그램에서 데이터의 형태를 정하는 것은 매우 중요한 작업입니다. 이번에는 파이썬에서 사용되는 자료형에 대해 알아보도록 하겠습니다.

파이썬의 주요 내장 자료형과 작성법

내장 자료형	설명	작성법	불/가변
정수	소수가 없는 숫자	a = 10, a = -10	불변
부동소수	소수가 있는 숫자	a = 10.5	불변
불	True 또는 False 값을 가진 자료형	a = True	불변
문자열	문자 데이터를 순서대로 나열한 데이터. 문자가 1개뿐이어도 문자열이라고 부름	a = 'python' 문자열의 0번째(첫 글자)문자는 a[0]으로 지정	불변
리스트	숫자나 문자열 등의 요소를 순서대로 나열한 구조체	a = ['p', 'y', 't', 'h', 'o', 'n'] 리스트 a의 0번째(첫 글자) 요소는 a[0]으로 지정	가변
튜플	리스트와 비슷하지만 불변성 때문에 바꿔 쓸 수 없음	a = ['p', 'y', 't', 'h', 'o', 'n'] 리스트 a의 0번째(첫 글자) 요소는 a[0]으로 지정	불변
딕셔너리	각 데이터에 이름(키)이 있는 데이터 구조	a = {'happy':'(^3^)', 'sad':(*0*)'} 딕셔너리 a의 키 happy에 대응하는 값은 a{'happy'}로 지정	가변

1) 정수형, 부동소수점형, 불형

① 정수형

정수란 소수점 이하를 포함하지 않는 숫자입니다. 정수는 양수와 음수로 나뉘고 숫자 앞에 - 를 붙이면 음수, 아무것도 붙이지 않으면 양수입니다. 다른 프로그래밍 언어에서는 저장 공간 크기가 다른 정수형이 있지만 파이썬에서는 할당된 메모리가 허용하는 한, 큰 숫자도 제한없이 다룰 수 있습니다.

ex) 양수 : 1,2,3,4,5⋯ , 음수 : -1,-2,-3,-4,-5⋯

Integer_Test.py

```
num1 = 123
num2 = -456
```

출력 결과

```
print(num1)
print(num2)
```

② 부동소수점형

부동소수점형은 소수점 이하를 포함하는 숫자로 가수부와 지수부를 조합하여 표시합니다. 부동소수점이란 소수점의 위치가 고정되어 있는 것이 아니라 움직인다는 뜻으로 고정 소수점 방식에 비해 아주 큰 수와 아주 작은 수의 표현이 가능합니다.

123.456을 부동소수형으로 표시하면 가수부가 1.23456이고 지수부가 10^2

Float_Test.py

```
num1 = 3.14
num2 = -24.76
```

출력 결과

```
print(num1)
print(num2)
```

TIP

<반올림 함수>
내장 함수 round()를 사용하면 소수점 이하를 반올림할 수 있습니다.

round() 함수 사용 방법

round(반올림 할 숫자, 자릿수)

```
>>> a = 0.123456
>>> round(a ,4)
0.1235
```

③ 불형

영국의 수학자 겸 논리학자인 조지 불George Boole 의 이름에서 따온 **불리언**boolean (또는 불형)은 **논리 자료형**이라고도 하며 참과 거짓을 나타내는데 쓰입니다. True는 숫자 1, False는 숫자 0으로도 표현할 수 있으며 숫자를 쓰지 않고 참과 거짓을 나타내는 True와 False를 쓰기도 합니다.

TIP

- 불형을 지칭하는 단어는 여러 가지가 있습니다. 불, 불린, 불리언 모두 불형을 뜻합니다.
- True나 False는 파이썬의 예약어로 true, false와 같이 사용하지 말고 첫 문자를 항상 대문자로 사용해야 합니다.

```
>>> a = True
>>> b = False
```

Boolean_Test.py

```
num1 = True
num2 = False

print(num1)
print(num2)
```

출력 결과

```
True
False
```

2) 시퀀스형

파이썬에서는 요소들이 일정한 순서대로 나열된 데이터형을 **시퀀스형**이라 부릅니다. 시퀀스형에 저장된 데이터 하나하나의 값을 '**요소**'라고 하며, 요소를 지정하는 번호를 '**인덱스**'라고 합 니다. 스퀀스형 자료형에는 문자열, 리스트, 튜플이 있습니다.

문자열, 리스트, 튜플의 인덱스

시퀀스형의 인덱스

1 문자열

문자열은 낱개의 문자들을 묶어서 하나로 처리하기 위한 자료형입니다. 파이썬에서 문자열로 선언하기 위해서는 문자열을 작은따옴표(')나 큰따옴표(")로 감싸주어야 합니다. 문자열 변수의 데이터를 통째로 배우는 작업은 가능하지만 일부분만 바꾸는 것은 불가능합니다.

문자열 선언방법

```
문자열 변수 = '문␣자␣열'
```

만약 띄어쓰기가 문자열에 들어갈 경우 띄어쓰기도 하나의 인덱스로 계산됩니다.

(1) 문자열 인덱싱

문자열 인덱싱은 인덱스를 이용해 문자열에서 원하는 인덱스에 저장된 문자를 가져옵니다.

문자열 인덱싱 방법

```
a[x]
```

인덱스 x 요소를 가져옵니다.

String_Test.py

```
Text = 'AI Coding Pack'

print(Text[0]); print(Text[1]); print(Text[2])
print(Text[3]); print(Text[4]); print(Text[5])
print(Text[-8]); print(Text[-7]); print(Text[-6]); print(Text[-5])
print(Text[-4]); print(Text[-3]); print(Text[-2]); print(Text[-1])
```

출력 결과

```
A
I
  ※ 띄어쓰기도 하나의 인덱스로 계산됩니다.
C
o
d
i
n
g
  ※ 띄어쓰기도 하나의 인덱스로 계산됩니다.
P
a
c
k
```

(2) 문자열 슬라이싱

문자열 슬라이싱은 인덱스를 이용해 원하는 범위 안에 있는 모든 문자열을 가져옵니다.

문자열 슬라이싱 방법

```
a[x:y]
```

인덱스 x부터 y-1까지 범위에 있는 요소를 가져옵니다.

String_Test.py

```
Text = ‘AI Coding Pack’

print(Text[0:2])    # 변수 Text의 인덱스 0부터 1까지 범위에 있는 요소를 출력합니다.
print(Text[3:9])    # 변수 Text의 인덱스 3부터 8까지 범위에 있는 요소를 출력합니다.
print(Text[10:])    # 종료 위치가 비어 있으면 마지막 인덱스까지로 지정됩니다.
print(Text[:14])    # 시작 위치가 비어 있으면 있으면 인덱스 0으로 지정됩니다.
```

출력 결과

```
AI
Coding
Pack
AI Coding Pack
```

(3) 문자열 결합, 반복하기

산술 연산자 +와 *를 사용하면 문자열끼리 결합, 반복이 가능합니다.

String_Test.py

```
a = ‘AI’
b = ‘Coding’
c = ‘Pack’

print(a + b + c)
print((a + b + c) * 2)
```

출력 결과

```
AI Coding Pack
AI Coding PackAI Coding Pack
```


(4) 문자열 함수

함수를 사용하면 보다 빠르고 쉽게 작업 할 수 있습니다. 함수에 대해서는 뒤에서 자세하게 다루도록 하겠습니다.

함수명	의미
len()	문자열에 포함된 글자 수를 반환
find(객체, 시작 인덱스, 종료 인덱스)	문자열에 일치하는 문자열이 첫번째로 나온 위치를 반환, 존재하지 않을 경우 -1 반환
in	문자열에 대상이 포함돼 있는지에 대한 여부를 반환, 참일 경우 ‘True’, 거짓일 경우 ‘False’ 반환
upper()	문자열을 대문자로 변환
lower()	문자열을 소문자로 변환
title()	문자열의 첫 글자는 대문자 나머지는 소문자로 변환, 띄어쓰기 이후 첫 글자도 대문자로 변환
count(‘A’)	문자열에서 특정 문자열의 개수를 반환
endswith(‘A’)	마지막 문자가 인자로 끝나는지 검사, 참이면 ‘True’ 거짓이면 ‘False’ 반환
startswith(‘a’)	첫 문자가 인자로 시작하는지 검사, 참이면 ‘True’ 거짓이면 ‘False’ 반환

다양한 문자열 함수의 의미

String_Test.py

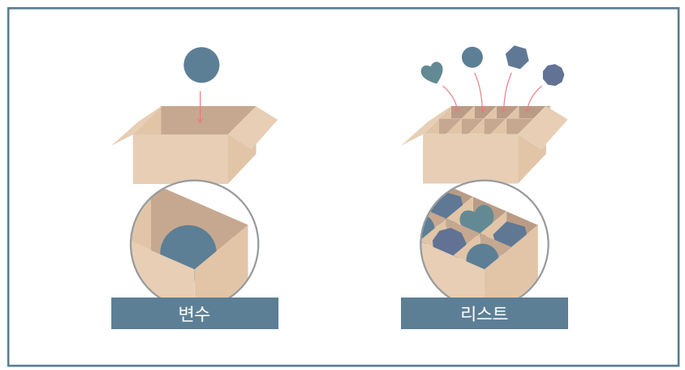
```
Text = ‘AI Coding Pack’

print(len(Text))
print(Text.find(‘AI’))
print(‘Coding’ in Text)
print(Text.upper())
print(Text.lower())
print(Text.title())
print(Text.count(‘A’))
print(Text.endswith(‘T’))
print(Text.startswith(‘a’))
```

출력 결과

```
13
0
True
AI CODING PACK
ai coding pack
Ai oding pack
2
True
False
```

2 리스트

앞서 배웠던 변수에는 데이터 하나를 저장할 수 있습니다. 여러 데이터를 저장하고 관리할 때 변수를 사용하는 것은 매우 복잡하고 혼란스럽기 때문에 바람직하지 않습니다. 데이터 100개가 있다고 가정해보겠습니다. 데이터 100개를 저장하기 위해서는 변수도 100개가 있어야 하고 그에 따른 변수명도 100개가 있어야 합니다. 만약 변수명을 잘 정한다고 하더라도 변수 100개를 관리하는 것은 매우 어려운 일입니다. 이처럼 데이터가 많을 때는 '리스트'를 사용하면 쉽게 데이터를 저장 및 관리할 수 있습니다. 변수가 한 상자에 하나의 데이터를 저장한다고 한다면, 리스트는 변수 상자를 여러 개로 나누어 많은 데이터를 저장합니다. 리스트는 정수, 부동소수, 문자열로 각각 구성할 수 있고 섞어서 구성할 수도 있습니다.



변수와 리스트의 차이

리스트 선언 방법

리스트 변수 = [요소1, 요소2, 요소3, 요소4, 요소5, …]

각 요소를 쉼표(,)로 나누고 대괄호[] 안에 모든 요소를 넣어줍니다.

```
a = [1, 2, 3, 4, 5]           # 정수로 이루어진 리스트
b = [1.1, 2.2, 3.3, 4.4, 5.5] # 부동소수로 이루어진 리스트
c = ['AI', 'Coding', 'Pack']   # 문자열로 이루어진 리스트
d = ['AI', 10, 3.14]           # 정수, 부동소수, 문자열이 섞여 있는 리스트
```

리스트에 또 다른 리스트를 요소로 선언하는 방법

리스트 변수 = [요소1, 요소2, [요소3, 요소4]]

```
a = [1, 2, 3, [1.1, 2.2, 3.3]]
```

(1) 리스트 인덱싱

리스트 인덱싱은 인덱스를 이용하여 리스트로부터 원하는 인덱스에 저장된 요소를 가져옵니다.

리스트 인덱싱 방법

a[x]

인덱스 x 요소를 가져옵니다.

List_Test.py

```
List = [1, 2, 3.14, ['AI', 'Coding', 'Pack']]

print(List[0]); print(List[1])
print(List[-2]); print(List[-1])
print(List[3][0]); print(List[3][-1])
# 자식 리스트의 요소를 호출하기 위해서는 부모 인덱스와 자식 인덱스를 모두 인자로 넣어줍니다.
```

출력 결과

```
1
2
3.14
['AI', 'Coding', 'Pack']
AI
Pack
```

(2) 리스트 슬라이싱

리스트 슬라이싱은 인덱스를 이용해 원하는 범위 안에 있는 모든 요소를 가져옵니다.

리스트 슬라이싱 방법

```
a[x:y]
```

인덱스 x부터 y-1까지 범위에 있는 모든 요소를 가져옵니다.

List_Test.py

```
List = [1, 2, 3.14, ['AI', 'Coding', 'Pack']]

print(List[0:2]) # 변수 List의 인덱스 0부터 2까지 범위에 있는 모든 요소를 출력합니다.
print(List[1:3]) # 변수 List의 인덱스 3부터 5까지 범위에 있는 모든 요소를 출력합니다.
print(List[2:]) # 종료 위치가 비어 있으면 마지막 인덱스까지 지정됩니다.
print(List[:3]) # 시작 위치가 비어 있으면 인덱스 0으로 지정됩니다.
print(List[3][0:2]) # 자식 리스트의 인덱스 0부터 1까지 범위에 있는 모든 요소를 출력합니다.
```

출력 결과

```
[1, 2]
[2, 3.14]
[3.14 ['AI', 'Coding', 'Pack']]
[1, 2, 3.14]
['AI', 'Coding']
```


(3) 리스트 요소 변경하기

리스트는 가변 데이터로 요소를 변경할 수 있습니다.

List_Test.py

```
List = [1, 2, 3, 4, 5, 6]
List[0] = 1.1
List[2] = 3.14
List[3:7] = ['AI', 'Coding', 'Pack'] # 인덱스 3부터 6까지 요소 변환

print(List)
```

출력 결과

```
[1.1, 2, 3.14, 'AI', 'Coding', 'Pack']
```

(4) 리스트 결합, 반복하기

산술 연산자 +와 *를 사용하면 리스트끼리 결합, 반복이 가능합니다.

List_Test.py

```
num = [1, 2, 3]
Text = ['AI', 'Coding', 'Pack']

print(num + Text)
print((num + Text) * 2)
```

출력 결과

```
[1, 2, 3, 'AI', 'Coding', 'Pack']
[1, 2, 3, 'AI', 'Coding', 'Pack', 1, 2, 3, 'AI', 'Coding', 'Pack']
```

(5) 리스트 함수

함수를 사용하면 빠르고 쉽게 원하는 작업을 할 수 있습니다.

함수명	의미
len()	리스트에 저장된 요소의 개수를 반환
max()	리스트에 저장된 요소의 최댓값을 반환
min()	리스트에 저장된 요소의 최솟값을 반환
sum()	리스트에 저장된 요소의 합을 반환
sorted()	리스트에 저장된 요소를 오름차순으로 정렬

List_Test.py

```
num = [100, 50, 354, 23.1, 87.35]

print(len(num))
print(max(num))
print(min(num))
print(sum(num))
print(sorted(num))
```

출력 결과

```
5
354
23.1
641.45
[23.1, 50, 87.35, 100, 354]
```

*append()은 요소 하나만 추가,
extend()은 여러 요소 추가

함수명	의미
append(객체)*	리스트 뒷부분에 지정한 요소를 추가
insert(위치, 객체)	리스트에서 지정한 위치에 요소를 삽입
extend(객체)*	리스트 뒷부분에 지정한 요소를 추가
pop()	리스트의 마지막 요소를 삭제
remove(객체)	리스트에서 지정한 요소를 삭제
sort()	리스트를 알파벳 순서대로 정렬 ※ 아스키코드 순서(대문자 ->소문자)
reverse()	리스트를 역순으로 정렬
index(객체)	리스트에서 지정한 요소가 저장된 인덱스를 반환
count(객체)	리스트에서 지정한 요소의 개수를 반환

List_Test.py

```
num = [1, 2, 3, 4]
fruit = ['banana', 'apple', 'melon']

num.append(5)
print(num)

num.insert(2, 2.5)
print(num)

num.extend([6, 7, 8, 9])
print(num)

num.pop()
print(num)

num.remove(1)
print(num)

fruit.sort()
print(fruit)

fruit.reverse()
print(fruit)

print(fruit.index('apple'))
print(fruit.count('apple'))
```

출력 결과

```
[1, 2, 3, 4, 5]
[1, 2, 2.5, 3, 4, 5]
[1, 2, 2.5, 3, 4, 5, 6, 7, 8, 9]
[1, 2, 2.5, 3, 4, 5, 6, 7, 8]
[2, 2.5, 3, 4, 5, 6, 7, 8]
['apple', 'banana', 'melon']
['melon', 'banana', 'apple']
2
1
```

❸ 튜플

튜플은 위에서 설명한 리스트와 비슷한 자료형입니다. 다만 리스트는 요소를 추가/제거/변경이 가능하지만, 튜플은 한번 선언한 요소를 추가/제거/변경할 수 없다는 차이를 가집니다. 프로그램에서 선언 후, 변경되면 안 되는 데이터일 경우 튜플로 선언하면 데이터의 손실 없이 관리할 수 있습니다. 튜플은 최소 두 개 이상 데이터로 구성되어야 합니다. 만약 데이터 한 개로 튜플을 선언할 경우 문자열로 저장됩니다.

튜플 선언 방법

변수 = (요소1, 요소2, 요소3, 요소4, 요소5, …)

각 요소를 쉼표(,)로 나누고 소괄호() 안에 모든 요소를 넣어줍니다.

```
a = (1, 2, 3, 4, 5)           # 정수로 이루어진 튜플
b = (1.1, 2.1, 3.1, 4.1, 5.1) # 부동소수로 이루어진 튜플
c = ('AI', 'Coding', 'Pack')   # 문자열로 이루어진 튜플
d = ('AI', 10, 3.14)          # 정수, 부동소수, 문자열이 섞여 있는 튜플
```

튜플에 또 다른 튜플을 요소로 선언하는 방법

튜플 변수 = (요소1, 요소2, (요소3, 요소4))

```
a = (1, 2, 3, (1.1, 2.2, 3.3))
```


(1) 튜플 인덱싱

리스트 인덱싱은 인덱스를 이용하여 원하는 인덱스에 저장된 요소를 리스트에서 가져옵니다.

리스트 인덱싱 방법

a[x]

인덱스 x 요소를 가져옵니다.

Tuple_Test.py

```
Tuple = (1, 2, 3.14, ('AI', 'Coding', 'Pack'))

print(Tuple[0]); print(Tuple[1])
print(Tuple[-2]); print(Tuple[-1])
print(Tuple[3][0]); print(Tuple[3][-1])
# 자식 튜플의 요소를 호출하기 위해서는 부모 인덱스와 자식 인덱스를 모두 인자로 넣어줍니다.
```

출력 결과

```
1
2
3.14
('AI', 'Coding', 'Pack')
AI
Pack
```

(2) 튜플 슬라이싱

튜플 슬라이싱은 인덱스를 이용하여 원하는 범위 안에 있는 모든 요소를 가져올 수 있습니다.

튜플 슬라이싱 방법

a[x:y]

인덱스 x부터 y-1까지 범위에 있는 모든 요소를 가져옵니다.

Tuple_Test.py

```
Tuple = (1, 2, 3.14, (‘AI’, ‘Coding’, ‘Pack’))

print(Tuple[0:2]) # 변수 Tuple의 인덱스 0부터 2까지 범위에 있는 모든 요소를 출력합니다.
print(Tuple[1:3]) # 변수 Tuple의 인덱스 3부터 5까지 범위에 있는 모든 요소를 출력합니다.
print(Tuple[2:]) # 종료 위치가 비어 있으면 마지막 인덱스까지로 지정됩니다.
print(Tuple[:3]) # 시작 위치가 비어 있으면 인덱스 0으로 지정됩니다.
print(Tuple[3][0:2]) # 자식 리스트의 인덱스 0부터 1까지 범위에 있는 모든 요소를 출력합니다.
```

출력 결과

```
(1, 2)
(2, 3.14)
(3.14 (‘AI’, ‘Coding’, ‘Pack’))
(1, 2, 3.14)
(‘AI’, ‘Coding’)
```

(3) 튜플 결합, 반복하기

산술 연산자 +와 *를 사용하면 튜플끼리 결합, 반복이 가능합니다.

Tuple_Test.py

```
num = (1, 2, 3)
Text = (‘AI’, ‘Coding’, ‘Pack’)

print(num + Text)
print((num + Text) * 2)
```

출력 결과

```
(1, 2, 3, ‘AI’, ‘Coding’, ‘Pack’)
(1, 2, 3, ‘AI’, ‘Coding’, ‘Pack’, 1, 2, 3, ‘AI’, ‘Coding’, ‘Pack’)
```


(4) 튜플 함수

함수를 사용하면 빠르고 쉽게 원하는 작업을 할 수 있습니다.

함수명	의미
len()	튜플에 저장된 요소의 개수를 반환
max()	튜플에 저장된 요소의 최댓값을 반환
min()	튜플에 저장된 요소의 최솟값을 반환
sum()	튜플에 저장된 요소의 합을 반환
sorted()*	튜플에 저장된 요소를 오름차순으로 정렬하여 리스트 형태로 출력
index(객체)	튜플에서 지정한 객체가 저장된 위치 반환
count(객체)	튜플에서 지정한 객체의 개수 반환

* 함수를 사용할 때 리스트 형태로 변경되어 출력할 뿐, 튜플을 리스트로 변경하는 것은 아님

Tuple_Test.py

```
num = (100, 50, 354, 23.1, 87.35)

print(len(num))
print(max(num))
print(min(num))
print(sum(num))
print(sorted(num))
print(num.index(100))
print(num.count(100))
```

출력 결과

```
5
354
23.1
614.45
[23.1, 50, 87.35, 100, 354]
0
1
```

3) 딕셔너리

키를 지정할 때 문자나 숫자처럼 변하지 않는 자료형으로만 지정할 수 있습니다.

시퀀스형(문자열, 리스트, 튜플)이 인덱스를 통해 요소에 접근하는 것과 달리 딕셔너리는 키를 이용하여 요소에 접근할 수 있습니다. 인터넷에 찾고 싶은 단어(키)를 입력하고 검색하면 그 단어 대한 정보(요소)를 찾을 수 있는 것처럼 딕셔너리도 그와 비슷한 방식입니다. 키(단어)에 요소(단어에 대한 정보)를 지정해주면 그 이후에는 키만 가지고 요소를 찾을 수 있습니다.

딕셔너리 선언 방법

```
변수 = {키1:요소1, 키2:요소2, 키3:요소3, …}
```

콜론(:) 왼쪽에 키를, 오른쪽에 요소를 입력합니다. 각각의 키와 요소는 쉼표(,)로 구분하고 중괄호{ } 안에 넣어줍니다.

```
Dic = {'name':'AI Coding Pack', 'phone':821012345678, 'birth':1021}
```

요소로 리스트와 튜플을 넣을 수도 있습니다.

```
Dic = {'name':'AI Coding Pack', 'List':[1, 2, 3], 'Tuple':(4, 5, 6)}
```

1 키를 사용하여 딕셔너리 요소 가져오기

Dictionary_Test.py

```
Dic = {'name':'AI Coding Pack', 'phone':821012345678, 'birth':1021}

print(Dic['name'])
print(Dic['phone'])
print(Dic['birth'])
```

출력 결과

```
AI Coding Pack
821012345678
1021
```

2 키를 사용하여 딕셔너리 요소 변경하기

Dictionary_Test.py

```
Dic = {'name':'AI Coding Pack', 'phone':821012345678, 'birth':1021}

Dic['phone'] = 821011112222

print(Dic)
```

출력 결과

```
{'phone': 821011112222, 'birth': 1021, 'name': 'AI Coding Pack'}
```

※ 딕셔너리는 순서에 영향을 받지 않습니다

3 딕셔너리 키:요소 추가하기

Dictionary_Test.py

```
Dic = {'name':'AI Coding Pack', 'phone':821012345678, 'birth':1021}

Dic['age'] = 25

print(Dic)
```

출력 결과

```
{'age': 25, 'phone': 821012345678, 'birth': 1021, 'name': 'AI Coding Pack'}
```

4 딕셔너리 요소 제거하기

del 명령어는 키를 이용하여 요소를 제거할 때 사용됩니다.

Dictionary_Test.py

```
Dic = {'name':'AI Coding Pack', 'phone':821012345678, 'birth':1021}
del Dic['birth']
print(Dic)
```

출력 결과

```
{'name':'AI Coding Pack', 'phone':821012345678}
```

5 딕셔너리 함수

파이썬 2.7 버전까지는 keys(), values(), items() 함수를 호출하면 dict 형식이 아닌 리스트를 반환했습니다. 리스트를 반환하는 것은 메모리의 낭비가 발생하는데 파이썬3.0 이후 버전에서는 메모리 낭비를 줄이기 위해 dict 형식을 리턴합니다. 리스트로 만들어서 사용해야 하는 경우 list(객체)를 사용하면 리스트를 생성할 수 있습니다.

딕셔너리 함수는 딕셔너리가 자체적으로 가지고 있는 함수로 다양한 기능을 제공합니다.

함수명	의미
keys()	변수에 저장된 키만 모아서 dict_keys라는 리스트로 반환
values()	변수에 저장된 요소만 모아서 dict_values라는 리스트로 반환
items()	변수에 저장된 키와 요소 쌍을 dict_items라는 튜플로 반환
clear()	변수에 저장된 모든 키와 요소를 삭제하고 빈 딕셔너리로 반환
get(키)	변수에 저장된 키를 사용하여 요소를 반환
in	변수에 키가 저장되어 있는지를 확인, 참이면 True, 거짓이면 false 반환

Dictionary_Test.py

```
Dic = {'name':'AI Coding Pack', 'phone':821012345678, 'birth':1021}

print(Dic.keys())
print(Dic.values())
print(Dic.items())
print(Dic.get('phone'))
print('birth' in Dic)

Dic.clear()
print(Dic)
```

출력 결과

```
dict_keys(['birth', 'name', 'phone'])
dict_values([1021, 'AI Coding Pack', 821012345678])
dict_items([('birth', 1021), ('name', 'AI Coding Pack'), ('phone',
821012345678)]) 821012345678
True
{}
```


4) 형변환(데이터 형태 변환)

데이터는 같은 형태끼리만 처리 할 수 있습니다. 정수는 정수끼리 실수는 실수끼리, 문자는 문자끼리 처리가 가능합니다. 예를 들어, 숫자 10과 문자 10은 서로 더할 수 없습니다. 이때 숫자 10과 문자10을 더하기 위해서는 문자 10을 숫자로 바꿔주어야 합니다. 이러한 경우 형 변환을 통해 문자를 정수 혹은 실수로 변경할 수 있습니다.

함수명	의미
int(문자열)	문자열형을 정수형으로 변환
float(문자열)	문자열형을 부동소수점형으로 변환
str(수치)	수치형이나 부동소수점형을 문자열형으로 변환

Casting.py

```
Text = '3.14'
num = 3

value1 = float(Text) + num
value2 = Text + str(num)

print(value1)
print(value2)
```

출력 결과

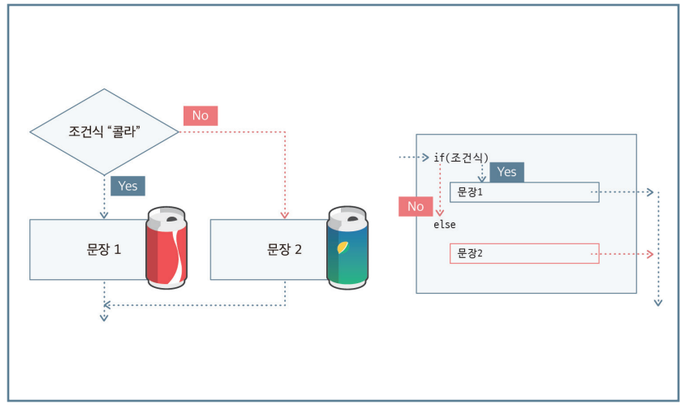
```
6.140000000000001
3.143
```

06 조건문

프로그램은 기본적으로 위에서부터 아래로 내려가면서 각각의 문장이 순차적으로 실행되지만, 특정 조건이 주어지면 다르게 동작해야 하는 경우가 생깁니다. 예를 들어, 인공지능 스피커는 우리가 명령을 내렸을 경우(특정 조건) 그 명령에 따라 행동을 하고, 그렇지 않은 경우 아무런 반응이 없듯이 조건문은 프로그래밍에서 조건을 판단하고, 조건에 맞는 프로그램을 수행하는 데 사용됩니다.

1) if-else문

if문은 주어진 조건식에 만족하는 경우(True) if문 프로그램을 실행시키고, 조건을 만족하지 않을 경우(False) if문을 생략하고 else문으로 넘어갑니다. else문은 if문 조건식에 만족하지 않는 모든 경우에 실행됩니다.



파이썬에서 if-else문을 사용할 때 꼭 지켜야 하는 두 가지 약속

1. 조건식 끝과 else 끝에는 콜론(:)을 붙인다.
2. 처리문(수행할 문장) 앞에는 반드시 들여쓰기를 붙여준다.

if문의 기본 구조

```
if 조건식 :
    수행할 문장1
    수행할 문장2
    …
else :
    수행할 문장A
    수행할 문장B
    …
```

TIP

들여쓰기는 문장 시작 부분에 공백을 넣어 띄어 쓰는 것으로 파이썬에서 들여쓰기는 매우 중요합니다. 다른 프로그래밍 언어의 경우 들여쓰기에 크게 신경을 쓰지 않더라도 들여쓰기로 인한 오류가 발생하지 않고 프로그램이 정상 동작하지만 파이썬의 경우 들여쓰기는 코드가 어디에 속한 것인지 구분해주는 역할을 하기 때문에 들여쓰기를 제대로 하지 않으면 오류가 발생 하거나 프로그램이 제대로 동작하지 않습니다.

1 비교 연산자를 활용하여 조건식 만들기

비교 연산자는 두 데이터의 값을 비교할 때 사용합니다. 두 데이터의 값이 같은지, 어느 쪽이 더 큰지, 작은지를 통해 다양한 조건식을 만들 수 있습니다.

함수명	의미
x == y	x와 y가 같으면 True, 그렇지 않으면 False
x != y	x와 y가 같지 않으면 True, 같으면 False
x < y	x보다 y가 작으면 True, 크거나 같으면 False
x > y	x보다 y가 크면 True, 작거나 같으면 False
x <= y	x보다 y가 작거나 같으면 True, 크면 False
x >= y	x보다 y가 크거나 같으면 True, 작으면 False
x in y	x가 y의 요소(문자열, 리스트, 튜플 등)면 True, 요소가 아니면 False
x not in y	x in y의 반대, x가 y에 포함되어 있으면 False, 포함되어 있지 않으면 True

#

연습해보기

1) 변수 grade에 수학 점수를 저장하고, 만약 수학 점수가 80점 이상이라면 “Excellent”를 80점 미만이면 “Good job”을 출력하는 프로그램을 if 문을 사용하여 작성하시오.

해설

If_Test.py

```
grade = 90

if grade > 80:
    print('Excellent')
else:
    print('Good job')
```

출력결과

```
Excellent
```

2) 변수 Data에 A, B, C, D, E가 담긴 리스트를 저장하고, C가 리스트에 포함되어 있으면 “True”를 포함되어 있지 않으면 ‘False’를 출력하는 프로그램을 작성하시오.

해설

If_Test.py

```
Data = ['A', 'B', 'C', 'D', 'E']

if 'C' in Data:
    print('True')
else:
    print('False')
```

출력결과

```
True
```

② 논리 연산자를 활용하여 여러 조건식 만들기

논리 연산자는 여러 개의 조건을 조합하여 참인지 거짓인지를 판단할 때 사용됩니다. “내일이 휴일이고 비가 오지 않는다면 드라이브를 할거야.”라는 문장에서 드라이브를 하기 위해서는 휴일이면서 비가 오지 않아야 한다는 두 가지 조건을 만족해야 합니다. 이처럼 한 조건식에 여러 조건을 조합해야 될 경우 논리 연산자를 사용합니다.

함수명	의미
x and y	x와 y의 논리곱, x와 y 둘 다 True면 True
x or y	x와 y의 논리합, x와 y 중 하나라도 True면 True
not x	x의 부정, x가 True면 False, False면 True

#

연습해보기

1) 변수 grade에 수학 점수를 저장하고, 만약 수학 점수가 70점 이상 90점 미만이라면 “The grade is B”를 출력하는 프로그램을 작성하시오.

해설

If_Test.py

```
grade = 87

if grade >=70 and grade < 90:
    print('The grade is B')
```

출력결과

```
The grade is B
```

2) 변수 Data에 A, B, C, D, E가 담긴 리스트를 저장하고, A 혹은 a가 리스트에 포함되어 있으면 "True"를 포함되어 있지 않으면 'False'를 출력하는 프로그램을 작성하시오.

해설

If_Test.py

```
Data = ['A', 'B', 'C', 'D', 'E']

if 'A' in Data or 'a' in Data:
    print('True')
else:
    print('False')
```

출력결과

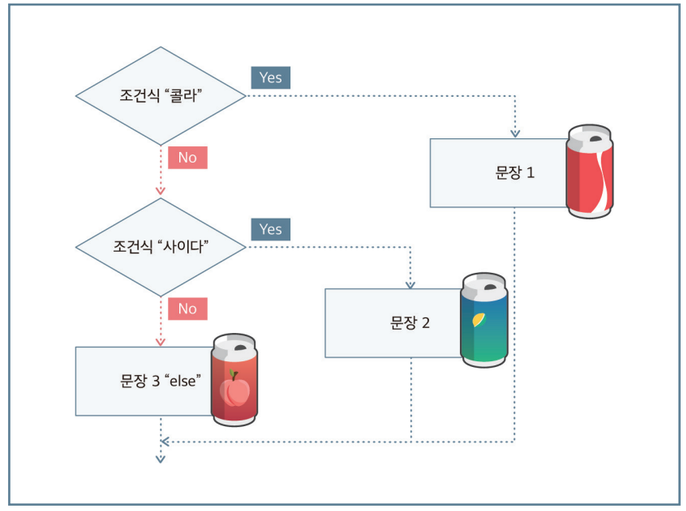
```
True
```


2) elif문

elif문은 if문의 조건식 이외의 새로운 조건식을 추가해야 하는 경우 사용합니다. 자판기에서 다양한 음료를 판매하는 것처럼 주어질 수 있는 조건이 여러 개일 경우 elif문을 추가하여 조건식을 추가합니다.

elif문 사용 방법

```
if 조건식 :
    수행할 문장1
    수행할 문장2
    …
elif 조건식 :
    수행할 문장A
    수행할 문장B
    …
else :
    수행할 문장a
    수행할 문장b
    …
```



#

연습해보기

1) 변수 grade에 수학 점수를 저장하고, 만약 수학 점수가 60점 미만이라면 “Your grade is C”를, 60점 이상 80점 미만이라면 “Your grade is B”를, 80점 이상이라면 “Your grade is A”를 출력하는 프로그램을 작성하시오.

해설

If_Test.py

```
grade = 87

if grade >= 80:
    print('Your grade is A')
elif grade >= 60 and grade < 80:
    print('Your grade is B')
else:
    print('Your grade is C')
```

출력결과

```
Your grade is A
```

07 반복문

반복문은 프로그램 안에서 같은 작업을 여러 번 반복해서 되는 경우 사용합니다. 예를 들어 100개의 데이터가 저장된 리스트가 있다고 가정해봅시다. 리스트의 각 인덱스에서 저장된 데이터를 모두 출력하는 프로그램을 작성한다고 했을 때, 모든 인덱스에 저장된 데이터를 출력하기 위해서는 print 명령어를 100번 사용해야합니다. 하지만 반복문을 사용하면 print 명령어를 100번 사용하지 않고 출력할 수 있습니다. 이처럼 반복문은 프로그램을 간결하게 만들어줍니다.

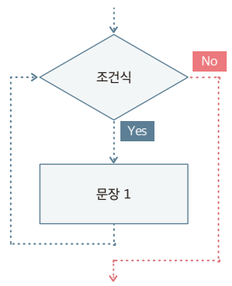
1) while문

while문은 조건식이 참(True)일 동안 아래에 속한 문장을 계속 반복합니다.

while문의 기본 구조

```
while 조건문 :
    수행할 문장1
    수행할 문장2
```

while문 조건식에 True 혹은 숫자 1을 입력하면 조건이 성립함으로 무한 반복됩니다.



#

연습해보기

1) while를 사용하여 리스트의 모든 요소를 출력하는 프로그램을 작성하시오.

해설

While_Test.py

```
List = [1, 2, 3, 4, 5, 6, 7, 8, 9]
count = 0

while count < len(List):
    print(List[count])
    count = count + 1
print('end')
```

출력결과

```
1
2
3
4
5
6
7
8
9
end
```


1 while문 빠져나가기

while문은 조건이 참인 경우 계속해서 작업을 반복합니다. 하지만 반복 도중 빠져나가야 하는 경우가 발생하곤 합니다. 예를 들어, 자판기는 음료수가 다 떨어질 때까지 같은 일을 반복합니다. 하지만 자판기 음료수가 다 떨어진 경우 더 이상 판매를 할 수 없다는 안내를 해야 합니다. 이처럼 음료를 판매하다가 강제로 멈춰야 하는 경우 사용하는 것이 break문입니다.

While_Test.py

```
cola = 10

while True:
    print('콜라가 %d개 남았습니다.' %cola)
    cola = cola - 1
    if cola == 0:
        print('콜라가 없습니다.')
        break

print('콜라를 보충해 주세요.')
```

출력 결과

```
콜라가 10개 남았습니다.
콜라가 9개 남았습니다.
콜라가 8개 남았습니다.
콜라가 7개 남았습니다.
콜라가 6개 남았습니다.
콜라가 5개 남았습니다.
콜라가 4개 남았습니다.
콜라가 3개 남았습니다.
콜라가 2개 남았습니다.
콜라가 1개 남았습니다.
콜라가 없습니다.
콜라를 보충해 주세요.
```

문자열 포맷팅(Formatting)

문자열 포맷팅이란 문자열 내에 값(정수, 실수, 문자)을 삽입하는 것으로 중복되는 문장에서 수치 혹은 단어만 다르게 출력할 때 사용합니다.

형식 지정자	의미	사용법
%d	10진 정수로 출력	print('오늘 날씨는 %d도입니다.' %20)
%f	실수로 출력	print('원주율은 %f입니다.' %3.141592)
%s	문자열로 출력	print('당신의 성적은 %s입니다.' %'A')

② while문 맨 처음으로 돌아가기

"while문 맨 처음으로 돌아가기"라는 제목을 보고, 변수를 초기화 시킨다거나 반복을 맨 처음 부터 다시 시작한다고 오해할 수 있지만 여기서 맨 처음으로 돌아간다는 말은 while문 바로 밑의 줄로 돌아간다는 뜻입니다. 프로그래밍을 하다 보면 아래 문장을 실행시키지 않고 다시 처음 문장으로 돌아가고 싶은 경우 continue 명령어를 사용하면 처음 문장으로 돌아갈 수 있습니다. 아래에서 0에서 10까지 숫자 중 3, 6, 9만 출력하는 예제를 continue 명령어를 사용하여 만들어 보겠습니다.

While_Test.py

```
count = 0

while count <= 10:
    count = count + 1
    if count % 3 != 0: continue   # 3으로 나누었을 때 나머지가 0이 아닐 경우 처음으로 돌아갑니다.
        print(count)
```

출력 결과

```
3
6
9
```

2) for문

for문과 while문의 차이점은 for문은 선언할 때 반복 횟수를 지정해준다는 것입니다. for문은 파이썬이 얼마나 사람의 사고와 비슷한지를 보여주는 가장 대표적인 예입니다.

for문의 기본 구조

리스트 또는 튜플, 문자열에 저장된 요소 수만큼 반복하고, 변수에는 반복 횟수에 따라 요소 값이 할당됩니다.

```
for 변수 in 리스트 또는 튜플, 문자열 :
    """ 수행할 문장1
    """ 수행할 문장2
    """ ...
```

For_test.py

```
data = ['AI', 'Coding', 'Pack']

for a in data:
    print(a)
```

출력 결과

```
AI
Coding
Pack
```

③ range( ) 함수로 리스트를 생성하기

for문과 자주 사용되는 range( ) 함수는 등차수열 리스트를 생성합니다.

range( ) 함수 사용법

```
range(x)          # 0부터 x-1까지 리스트를 생성합니다.
range(5)          # [0, 1, 2, 3, 4, 5]
range(x, y)       # x부터 y-1까지 리스트를 생성합니다.
range(5, 10)      # [5, 6, 7, 8, 9]
range(x, y, z)    # x부터 y-1까지 z 단위로 리스트를 생성합니다.
range(0, 10, 2)   # [0, 2, 4, 6, 8]
```

For_test.py

```
for a in range(10):
    print(a)
print('end')
```

출력 결과

```
0
1
2
3
4
5
6
7
8
9
end
```

#

연습해보기

1) For문을 사용하여 구구단을 출력하는 프로그램을 작성하시오.

해설

For_test.py

```
for a in range(2,10):
    for b in range(1,10):
        print('%d x %d = %d' % (a, b, a * b))
```

출력결과

```
2 x 1 = 2
2 x 2 = 4
2 x 3 = 6
2 x 4 = 8
.
.
.
9 x 6 = 54
9 x 7 = 63
9 x 8 = 72
9 x 9 = 81
```


4 for문 맨 처음으로 돌아가기

for문에서도 continue 명령어를 사용하면 아래 문장을 실행시키지 않고 다시 처음 문장으로 돌아갈 수 있습니다.

아래의 예제는 continue 명령어를 사용하여 0에서 10까지 숫자 중 3, 6, 9만 출력하는 코드입니다.

For_test.py

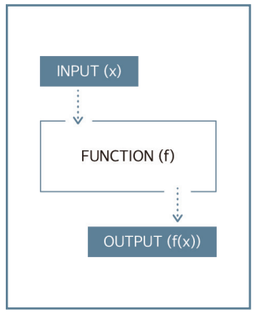
```
for i in range(10):
    if i % 3 != 0: continue
    # 3으로 나누엇을 때 나머지가 0이 아닐 경우 처음으로 돌아갑니다.
    print(i)
```

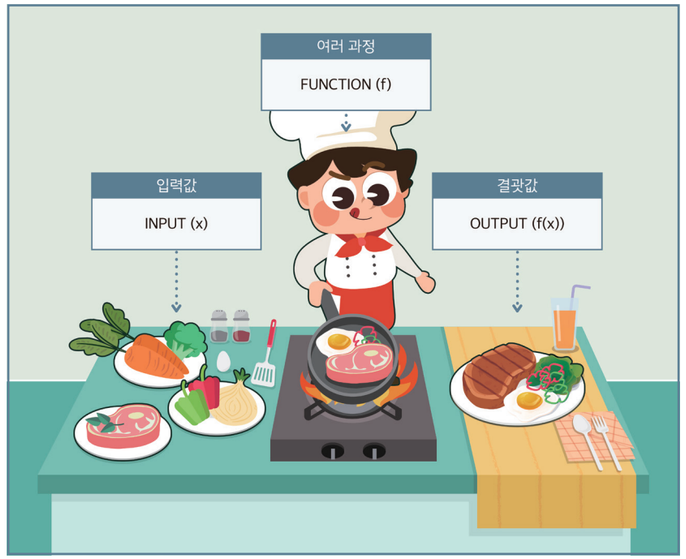
출력 결과

```
3
6
9
```

08 함수

함수는 마치 요리사와 같습니다. 요리사는 계란, 양파, 고기 등 식재료를 굽고, 볶고, 튀겨 하나의 요리를 만들어 내는 것처럼 함수는 입력값(INPUT x)을 가지고 여러 과정(FUNCTION f)을 거쳐 결괏값(OUTPUT f(x))을 반환해줍니다. 이때 입력값 x를 **전달인자**argument, 결괏값을 **반환값**return value 이라 부르고 함수를 사용하는 것을 "함수를 호출한다."라고 말합니다.





옆 페이지 상단에 작성된 두 가지 코드는 상품 무게를 기준으로 택배 요금을 책정하는 프로그램입니다. (1)코드를 보면 요금을 계산하는 문장이 상품 무게가 바뀔 때마다 작성되어 있는 것을 확인할 수 있습니다. 이러한 작업은 매우 비효율적이고, 의미 없는 작업입니다. 그렇다면 (2)코드를 한번 보겠습니다. (2)코드는 함수를 사용하여 택배 요금을 계산하는 프로그램으로, 상품 무게만 입력해주면 택배 요금을 반환해주는 것을 확인할 수 있습니다. 두 프로그램을 비교해보면 (2)프로그램이 훨씬 효율적이고 어떤 작업을 하는지 한 눈에 파악할 수 있다는 것을 알 수 있습니다. 이처럼 함수는 "반복적으로 사용되는 코드"를 모아 놓은 덩어리라 생각할 수 있습니다.

(1)

```
fee = 2500                    # 기본 요금
weight = 20                   # 상품 무게
if weight > 1:
   fee = fee + weight * 100
print(fee)
...
fee = 2500
weight = 12.2
if weight > 1:
   fee = fee + weight * 100
print(fee)
...
fee = 2500
weight = 18.76
if weight > 1:
   fee = fee + weight * 100
print(fee)
...
```

(2)

```
def calculate_fee(weight):        # 상품 무게
    fee = 2500                    # 기본 요금
    if weight > 1:
        fee = fee + weight * 100
    return fee

result = calculate_fee(20)

print(result)
```

1) 함수 호출방법

함수를 호출하기 위해서는 호출하고자 하는 함수명을 선언하고 전달 인자를 소괄호( )로 묶어 줍니다.

함수 호출방법

```
함수명(전달인자1, 전달인자2, …)
```

만약 함수로 보낼 입력 데이터가 없을 경우 전달인자를 입력하지 않아도 무관합니다.

2) 함수 선언방법

함수의 이름 앞에 def 라는 단어를 붙이는데, define(정의)의 약어이고 함수를 만들기 위해 쓰이는 **예약어**입니다.

함수 선언방법

```
def 함수이름(매개변수1, 매개변수2, ...):
    실행할 문장 1
    실행할 문장 2
    ...
    return 반환 값
```

매개변수는 전달 인자를 받아오는 부분으로 함수를 호출할 때 넘겨준 전달인자 개수와 매개변수 개수가 동일해야 합니다. 만약 개수가 다를 경우 오류가 발생합니다.

#

연습해보기

1) 1학년 4반 학생들의 수학성적은 아래와 같습니다. 1학년 4반 학생들의 수학성적 평균을 계산하고 출력할 수 있도록 함수를 이용한 프로그램을 작성하시오.

해설

Function_Test.py

```
scores = [37, 93, 59, 82]

def calculate_average(scores):
    if len(scores) == 0:
        return 0

    sum = 0
    for score in scores:
        sum = sum + score
    return sum / len(scores)

result = calculate_average(scores)
print(result)
```

출력결과

```
67.75
```


09 모듈

함수가 “반복적으로 사용되는 코드”를 모아 놓은 덩어리라면 모듈은 “연관있는 혹은 자주 사용하는 함수와 변수”를 모아 놓은 덩어리라 생각할 수 있습니다. 예를 들어 파이썬에 내장된 ‘math’라는 모듈은 절댓값(abs), 최댓값(max), 평균(avg), 원주율(pi) 등 수학에 관련된 함수를 모아놓은 모듈입니다. 기본적으로 파이썬에 내장되어 있는 모듈도 있지만 모듈을 직접 만들어 사용할 수도 있고, 혹은 다른 사람이 만들어 놓은 모듈을 불러와서 사용할 수도 있습니다.

1) 모듈 생성하기

일전에 만들어둔 Test_folder에 Calculate_Test.py 파일을 만들어 아래와 같이 작성한 후 저장해보겠습니다.

Calculate_Test.py

```
def add(a, b):
    return a + b

def sub(a, b):
    return a – b

def mul(a, b):
    return a * b

def div(a, b):
    return a / b
```

2) 모듈 불러오기

모듈을 사용하기 위해 가장 먼저해야되는 작업은 import 명령어를 사용하여 모듈을 불러오는 것입니다.

모듈 불러오기

```
import 모듈이름
```

Module_Test.py

```
import Calculate_Test
```

3) 모듈 호출방법

import로 불러온 모듈의 이름 뒤에 점(.)을 붙이고 사용할 함수명과 전달인자를 입력하면 모듈을 호출할 수 있습니다.

모듈 호출방법

```
모듈이름.함수(전달인자1, 전달인자2, …)
```

Module_Test.py

```
# Calculate_Test을 사용하기 위해 모듈을 불러옵니다.
import Calculate_Test

# Calculate_Test 모듈에 저장된 함수들을 호출하고, 그 결과를 출력합니다.
print(Calculate_Test.add(2, 24))
print(Calculate_Test.sub(52, 21))
print(Calculate_Test.mul(3, 6))
print(Calculate_Test.div(16, 4))
```

출력 결과

```
26
31
18
4.0
```


4) 모듈에서 지정한 함수 및 변수만 불러오기

만약 모듈 전체를 가져올 필요가 없다면 모듈에서 필요한 함수 및 변수만 불러올 수도 있습니다.

from 모듈명 import 불러올 함수 및 변수

위 문장은 “모듈명으로부터 함수 및 변수를 가져온다.”로 해석됩니다. 말 그대로 모듈에서 필요한 함수 및 변수만 불러와 사용할 수 있다는 의미입니다. 위 방법을 사용해 모듈을 불러왔다면 모듈을 호출할 때 불러온 모듈명을 따로 언급하지 않고 바로 불러온 변수를 사용하거나 함수를 호출할 수 있습니다.

모듈에서 지정한 함수 및 변수만 불러오기

```
불러온 함수(전달인자1, 전달인자2, …)
불러온 변수
```

Module_Test.py

```
# 두 개 이상 함수 혹은 변수를 불러오려면 쉼표(,)를 사용합니다.
# ex : from Calculate_Test import add, sub

from Calculate_Test import add
print(add(2, 24))
```

출력 결과

```
26
```

TIP

import 뒤에 *을 붙이면 모듈에 저장된 모든 함수 및 변수를 불러올 수 있습니다.

```
from Calculate_Test import *
```

5) if __name__ == "__main__": 의 의미

아래와 같이 Calculate_Test.py을 수정한 후 다시 Module_Test.py을 실행시켜 보겠습니다.

Calculate_Test.py

```
def add(a, b):
    return a + b

def sub(a, b):
    return a – b

def mul(a, b):
    return a * b

def div(a, b):
    return a / b

print(‘The end’)
```

Module_Test.py

```
import Calculate_Test
print(Calculate_Test.add(2, 24))
print(Calculate_Test.sub(52, 21))
print(Calculate_Test.mul(3, 6))
print(Calculate_Test.div(16, 4))
```

출력 결과

```
The end
26
31
18
4.0
```

혹시 이상한 점을 눈치채셨나요? 사실 위 프로그램에서는 add, sub, mul, div 함수 실행 결과가 각각 한 번씩만 출력되어야 합니다. 하지만 출력 결과에서 확인할 수 있듯이 출력되지 않아야 되는 ‘The end’를 출력한 것을 확인할 수 있습니다. 이 경우 문제는 import에서 발생했습니다. Module_Test.py을 아래와 같이 수정한 후 실행시켜보겠습니다.

Module_Test.py

```
import Calculate_Test
```

출력 결과

```
The end
```


위에서 확인할 수 있듯이 import를 했는데 Calculate_Test 모듈이 실행되어 ‘The end’를 출력한 것을 확인할 수 있습니다. import는 모듈을 불러올 때 사용하는 명령어입니다. 하지만 여기서 import는 모듈을 불러오는 것이 아니라 모듈을 실행시켜 모듈 내부에 작성된 print 명령어를 실행시켰습니다. 이는 프로그램이 정상적으로 실행된 것이 아니기 때문에 큰 문제가 됩니다. 이러한 문제를 해결하기 위해 if __name__ == "__main__":을 아래와 같이 추가해줍니다.

Calculate_Test.py

```
def add(a, b):
    return a + b

def sub(a, b):
    return a – b

def mul(a, b):
    return a * b

def div(a, b):
    return a / b

if __name__ == "__main__":
    print(‘The end’)
```

if __name__ == "__main__":은 프로그램이 메인으로 실행시키지 않는다면, 즉, 모듈로 사용된다면 조건문에 포함된 문장을 실행시키지 않도록 하는 코드입니다. 이번 예제에서는 ‘Module_Test.py’이 메인 프로그램으로 사용되고, ‘Calculate_Test.py’은 모듈로 사용됩니다. 다시 메인 프로그램을 실행시켜보겠습니다.

Module_Test.py

```
import Calculate_Test
```

출력 결과

이렇듯 if __name__ == "__main__":를 사용하여 코드를 작성하게 되면 출력 결과에서 아무것도 실행되지 않게 됩니다.

10
객체 지향 프로그래밍

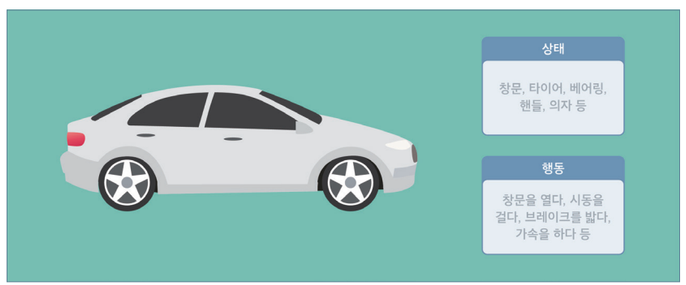
객체 지향 프로그래밍은 현실 속 대상을 소프트웨어적으로 추상화한 여러 객체를 조합하여 하나의 프로그램으로 만드는 프로그래밍 패러다임입니다. 객체는 상태(데이터)와 행동(로직)으로 구성되고, 기존에 따로 관리되던 데이터와 로직들을 하나로 묶어 놓은 것을 의미합니다.

함수를 이용해 절차적으로 진행되던 이전 방식과는 달리 객체들이 서로 소통하며 진행됩니다. “아니, 이게 도대체 무슨 말이야?”라고 생각하는 분들이 많을 것이라 생각합니다. 당연히 그럴 수 있습니다. 천천히 짚어가며 객체 지향 프로그래밍에 대해 알아보도록 하겠습니다.

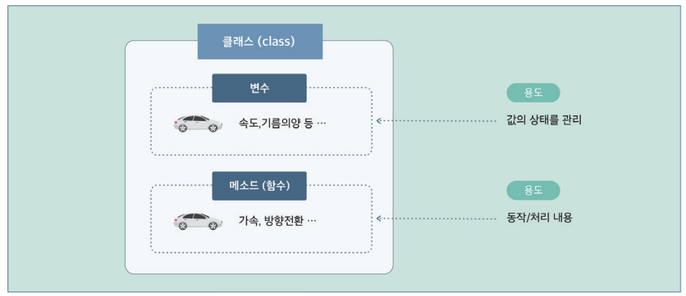
1) 객체

객체 지향 프로그래밍에서의 **객체**Object 는 현실의 모든 것들을 의미합니다. 개, 고양이, 건물, 창문, 커피 머신 등 심지어 여러분들도 객체입니다. 이런 현실의 객체들을 관찰하고 추상화해보면 상태와 행동이라는 두 가지 특징을 찾을 수 있습니다.

객체는 상태와 행동으로 나타낼 수 있습니다. 예를 들어, 자동차의 상태와 행동은 무엇일지 생각해봅시다. 자동차의 상태는 속도, 기름의 양, 문 잠금 상태, 제조사명, 모델명 등이 있을 수 있고, 행동에는 가속, 브레이크, 방향 전환, 트렁크 열기, 주유구 열기 등이 있을 수 있습니다.

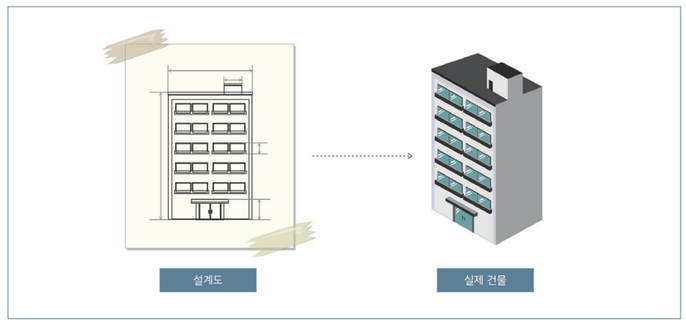


객체 지향 프로그래밍에서는 이러한 현실 속 객체를 소프트웨어적으로 바꾸어 생각할 수 있습니다. 속도, 기름의 양 등을 나타내는 상태는 변수에 저장하고, 가속, 방향 전환과 같은 행동은 함수로 동작시킬 수 있습니다. 객체 지향 프로그래밍에서는 변수와 함수를 체계적으로 관리하기 위해 하나의 **클래스**Class 로 묶어 사용합니다. 아래 그림처럼 덩어리를 만들고, 이것을 조립하여 코딩하는 방식을 객체 지향 프로그래밍이라고 하는 것입니다.

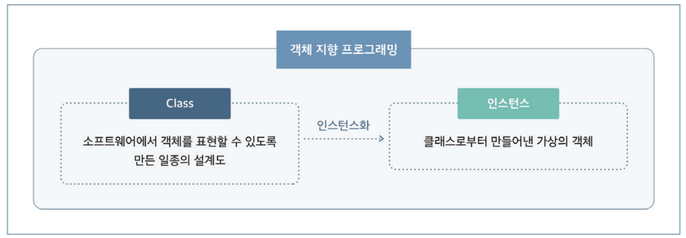


2) 클래스

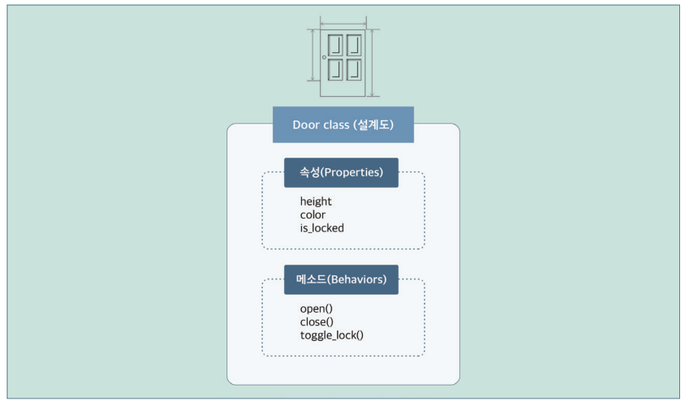
클래스는 현실의 객체가 어떻게 행동하는지, 어떤 상태를 가질지를 보고 소프트웨어에서 객체를 표현할 수 있도록 만든 일종의 설계도입니다. 클래스와 객체의 관계는 설계도와 또는 건축으로 비유 할 수 있습니다. 건물을 짓기 위해서는 설계도가 필요합니다. 이 설계도에는 건물의 크기가 얼마나 되는지, 문이 어디에 배치되는지, 창문은 어디에 얼마나 달릴지 등에 대한 정보를 담고 있습니다. 만약 돈이 충분하다면 설계도를 바탕으로 여러 개의 건물을 지을 수도 있습니다. 여기서 건물의 설계도를 클래스, 설계도를 바탕으로 지어진 건물을 객체로 비유할 수 있습니다.



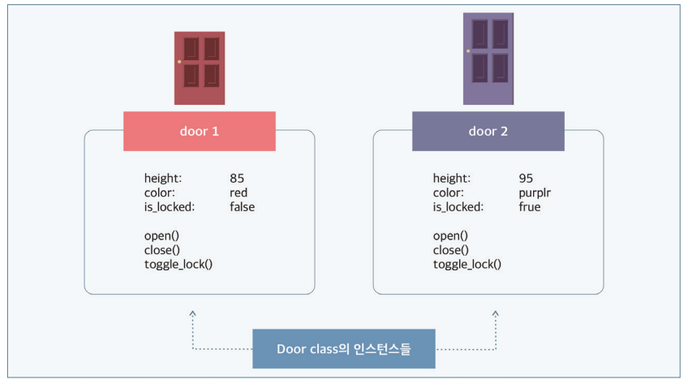
객체 지향 프로그래밍에서는 설계도를 클래스로 표현합니다. 설계도를 보고 건물을 짓듯이 클래스로부터 객체를 만들어 낼 수 있는데, 이 과정을 '**인스턴스화** 한다' 라고 말합니다. 그리고 여러분들이 클래스로부터 만들어낸 가상의 객체를 인스턴스라고 부릅니다. 클래스에서는 **변수(상태)**와 **함수(행동)**을 조금 다르게 표현합니다. 변수는 **속성**, 함수는 **메서드**라고 표현합니다.



클래스에 대한 이해를 돕기 위해 하나의 예를 더 들어보겠습니다. 문을 객체로 나타내어 본다면 상태와 행동을 어떻게 나타낼 수 있을까요? 상태는 문의 크기, 문의 색깔, 문 잠금 상태, 행동은 열기, 닫기, 문 잠금/해제가 있을 수 있겠죠. 이를 클래스로 만들면 아래 사진과 같이 만들 수 있습니다.



문의 설계도를 만들었으면 이제 문을 만들 차례입니다. Door 클래스로 door1 과 door2 인스턴스를 만들어보겠습니다. door1은 높이가 200, 색깔이 흰색 그리고 문이 닫혀 있지만 잠겨 있지 않고 door2는 높이가 220, 색깔이 빨간색 그리고 문이 닫혀있고 잠겨 있습니다. 정리하면 아래와 같습니다.



위 내용을 프로그램으로 작성하면 아마 아래와 같이 작성할 수 있을 것입니다. 각 변수에 알맞은 상태를 저장하고, 실행 결과에 따라 상태 값을 변경해주는 함수를 각각 생성해야 합니다. 문이 두 개라면, 사실 아래와 같은 프로그램을 만들어서 관리를 해도 전혀 문제가 되지 않을 것입니다. 하지만 만약 문이 100개라면 어떻게 될까요? 아마 door1, door2, ..., door100까지 프로그램이 아주 길고 복잡해질 것입니다. 아마 관리도 힘들고, 나중에 수정하기도 힘들 것입니다.

TIP

함수 내부에서는 함수 안에 있는 변수들 밖에 볼 수 없습니다. 그래서 함수 바깥에 있는 변수를 사용하기 위해서는 global 선언을 해주어야 합니다.

```
door_height_1 = 200
door_color_1 = 'White'
door_is_locked_1 = False
door_is_open_1 = False

def open_door_1():
    global door_is_open_1
    door_is_open_1 = True

def close_door_1():
    global door_is_open_1
    door_is_open_1 = False

def lock_door_1():
    global door_is_locked_1
    door_is_locked_1 = True

def unlock_door_1():
    global door_is_unlocked_1
    door_is_locked_1 = False

door_height_2 = 220
door_color_2 = 'Red'
door_is_locked_2 = True
door_is_open_2 = False

def open_door_2():
    global door_is_open_2
    door_is_open_2 = True

...
```

클래스를 사용하면, 문이 몇 개라도 쉽게 관리할 수 있습니다. 아래 프로그램은 클래스를 사용하여 문들을 관리하는 프로그램입니다. Door은 문의 상태와 행동을 정의해 놓은 클래스이고, 이를 인스턴스화하여 만들어진 인스턴스를 각각의 변수 door1과 door2에 대입합니다. 이제부터는 변수 door1과 door2 통해 인스턴스를 사용할 수 있습니다. 모르긴 몰라도 door1의 문을 잠그고, door2의 잠금을 해제하기 위해서는 아래 프로그램처럼 클래스 Door의 lock() 메서드와 unlock() 메서드를 호출하면, 간단하게 해결될 것입니다.

아래 프로그램에는 클래스 Door가 생략되었습니다. 뒤로 가면서 직접 클래스 Door를 만들어봅시다.

```
door1 = Door(200, 'White', False, False)    # Door를 인스턴스화 시켜 door1 변수에 넣습니다.
door2 = Door(220, 'Red', True, False)       # Door를 인스턴스화 시켜 door2 변수에 넣습니다.

door1.lock()              # door1을 잠급니다.
print(door1.is_locked)    # 문의 잠금 상태: true
door2.unlock()            # door2 을 잠금 해제합니다.
print(door2.is_locked)    # 문의 잠금 상태: false
```

이처럼 클래스를 이용하면, 여러 개의 문을 관리하기 위한 코드를 추가할 필요가 없고, 한 개의 클래스를 공유해서 사용할 수 있습니다. 또한 기능을 추가, 수정할 때도 일일이 모든 문을 수정할 필요 없이 클래스의 메서드를 수정하거나 추가하면 되기 때문에 관리가 용이해집니다.

3) Python에서 클래스

지금까지 우리는 객체 지향 프로그램에서 객체, 클래스, 인스턴스가 무엇인지 알아보았습니다. 이제부터 본격적으로 Python에서 클래스를 사용하는 방법에 대해 알아보겠습니다. Python에서 클래스는 아래와 같은 형식을 가지는데, 이전 단원에서 예를 들었던 문 객체를 Python에서 작성해보면서 클래스를 익혀보겠습니다.

```
class 클래스명:
    def __init__(self, 값1, 값2, ...):     # 인스턴스의 초기화를 담당하는 생성자
        self.속성1 = 값1
        self.속성2 = 값2
        ...

    def 메서드명(self, 인자1, 인자2 ...):
        self.속성1 = 인자1
        self.속성2 = 인자2
        ...
```


앞에서 다루었던 문의 속성과 메서드를 클래스로 옮겨 봅시다. 우선 문의 상태 즉, 속성을 정의해야 하는데 **생성자**에 값을 초기화함으로써 정의할 수 있습니다. 생성자는 클래스가 인스턴스화 될 때 호출되며, 인스턴스를 초기화해주는 역할을 합니다. Python에서는 __init__ 메서드가 생성자 역할을 맡고 있습니다. 생성자에서 속성을 초기화할 때 self라는 변수를 사용하는데, 이는 자기 자신을 가리키는 변수입니다. 더 자세한 설명은 메서드 파트에서 알아보도록 하겠습니다.

```
class Door:
    def __init__(self, location, color, is_locked, is_open):
        self.location = location
        self.color = color
        self.is_locked = is_locked
        self.is_open = is_open
        print("인스턴스가 초기화 되었습니다!")
```

우선 클래스의 생성자와 속성을 정의해 보았습니다. 직접 인스턴스를 만들어 속성을 설정해보도록 하겠습니다. 클래스의 인스턴스를 만들기 위해서는 아래와 같이 클래스명(생성자_인자_1, 생성자_인자_2, …) 형식으로 생성할 수 있습니다. 속성을 가져오거나 대입하기 위해서는 아래와 같이 변수.속성 형식을 이용해 사용할 수 있습니다.

인스턴스 생성 형식

```
변수 = 클래스명(생성자_인자_1, 생성자_인자_2, …)
```

인스턴스 사용 형식

```
변수.메서드(인자1, 인자2, …)
변수.속성
```

```
door = Door('부엌', 'Red', False, False)  # 생성자의 인자대로 (위치, 색상, 잠금여부, 열림여부)
print(door.location)    # 출력 결과: 부엌
door.location = '화장실'
print(door.location)    # 출력 결과: 화장실
```

다음으로 메서드를 정의해 보겠습니다. 클래스 Door의 메서드로는 열기(open), 닫기(close), 잠금(lock), 잠금 해제하기(unlock)가 있을 수 있습니다. 각각의 메서드는 열렸는지, 닫혀 있는지 저장하는 속성 is_open와 문이 잠겼는지, 풀렸는지 저장하는 속성 is_locked를 변경해야 합니다. 클래스 Door의 메서드를 작성해본다면 아래와 같이 작성할 수 있습니다.

```
class Door:
    def __init__(self, location, color, is_locked, is_open):
        self.location = location
        self.color = color
        self.is_locked = is_locked
        self.is_open = is_open
        print("인스턴스가 초기화 되었습니다!")

    def open(self):
        print("%s 문을 엽니다" % self.location)
        self.is_open = True

    def close(self):
        print("%s 문을 닫습니다" % self.location)
        self.is_open = False

    def lock(self):
        print("%s 문을 잠금니다" % self.location)
        self.is_locked = True

    def unlock(self):
        print("%s 문을 잠금 해제합니다" % self.location)
        self.is_locked = False
```

이 메서드들을 사용하기 위해서는 속성을 호출했던 방식과 같이 사용할 수 있습니다.

메서드 호출 형식

변수.메서드명(인자1, 인자2, ...)

```
door = Door('부엌', 'Red', False, False)  # 생성자의 인자대로 (위치, 색상, 잠금여부, 열림여부)
print(door.is_locked)    # 출력 결과: False
door.lock()
print(door.is_locked)    # 출력 결과: True
```


메서드를 살펴보면 모든 메서드들의 첫 번째 인자는 self인 것을 확인할 수 있습니다. 여기서 self는 인스턴스 자신을 뜻합니다. 인스턴스에서 메서드를 호출할 때에는 보이지는 않지만 첫 번째 인자로 인스턴스 자신을 넘겨줍니다. 그렇기 때문에 메서드에서 self.is_open에 값을 넣어 수정하면, 자신의 is_open 속성을 수정하게 되는 것입니다. 자기 자신을 담고 있는 변수의 이름을 self라고 지정했는데 다른 이름으로 사용할 수 있지만, 사람들과 암묵적으로 정해 놓은 일종의 약속으로 주로 self를 사용합니다.

4) 상속

흔히 **상속**이라는 단어는 부모가 자식에서 재산을 물려주는 행위를 말합니다. 클래스에서의 상속도 비슷한 개념을 가지고 있습니다. 클래스의 속성과 메서드를 다른 클래스에게 물려줄 때 "클래스를 상속한다."라고 이야기합니다.

공통된 기능을 A 클래스에 작성하고, 나머지 클래스에서는 A 클래스를 상속받기만 하면, A 클래스의 속성과 메서드를 그대로 가져와 사용할 수 있습니다. 또한, A 클래스 기능을 확장할 수도 있고, 반복되는 코드를 작성하지 않아도 됩니다. 수정사항이 생기더라도 여러 클래스를 수정할 필요 없이 A 클래스를 한 번만 수정하면 되기 때문에 관리하기도 훨씬 편리합니다. 이때 A 클래스를 **부모 클래스**라고 하고, A 클래스를 상속받는 클래스를 **자식 클래스**라고 부릅니다.

예를 들어서 여러분이 학교 관리 시스템을 설계한다고 가정해보겠습니다. 학교의 관리 시스템은 선생님, 학생, 학부모의 정보를 저장해 각각의 객체로 관리를 하려 합니다. 그래서 여러분은 선생님 클래스, 학생 클래스, 학부모 클래스를 만들기 위해, 아래와 같이 프로그램을 작성하였습니다.

```
class Student:
    def __init__(self, name, age, address, grade):
        self.name = name        # 이름
        self.age = age          # 나이
        self.address = address  # 주소
        self.grade = grade      # 학년

    def introduce(self):
        print('안녕하세요, 제 이름은 %s이고,' % self.name)
        print('나이는 %d세, %s에서 살고 있습니다.' % (self.age, self.address))
        print('저는 %d학년 입니다.' % self.grade)

class Teacher:
```

```
    def __init__(self, name, age, address, subject):
        self.name = name        # 이름
        self.age = age          # 나이
        self.address = address  # 주소
        self.subject = subject  # 담당 과목

    def introduce(self):
        print('안녕하세요, 제 이름은 %s이고,' % self.name)
        print('나이는 %d세, %s에서 살고 있습니다.' % (self.age, self.address))
        print('담당 과목은 %s 입니다.' % self.subject)

class Parent:
    def __init__(self, name, age, address, child):
        self.name = name        # 이름
        self.age = age          # 나이
        self.address = address  # 주소
        self.child = child      # 자식 인스턴스

    def introduce(self):
        print('안녕하세요, 제 이름은 %s이고,' % self.name)
        print('나이는 %d세, %s에서 살고 있습니다.' % (self.age, self.address))
        print('제 자식은 %s 입니다.' % self.child)
```

선생님, 학생, 학부모 클래스에는 겹치는 부분 즉, 이름, 주소, 나이 속성과 자기소개 메서드를 사람이라는 객체로 합칠 수 있습니다. 선생님, 학생, 학부모 클래스에서 공통으로 정의되어 있던 부분을 사람 클래스로 빼내어 이를 상속받아 사용해봅시다.

```
class Person:
    def __init__(self, name, age, address):
        self.name = name        # 이름
        self.age = age          # 나이
        self.address = address  # 주소

    def introduce(self):
        print('안녕하세요, 제 이름은 %s이고,' % self.name)
        print('나이는 %d세, %s에서 살고 있습니다.' % (self.age, self.address))

class Student(Person):
    def __init__(self, name, age, address, grade):
        super().__init__(name, age, address)
        self.grade = grade      # 학년

    def introduce(self):
        super().introduce()
        print('저는 %d 학년입니다.' % self.grade)

class Teacher(Person):
    def __init__(self, name, age, address, subject):
        super().__init__(name, age, address)
        self.subject = subject  # 담당 과목

    def introduce(self):
        super().introduce()
        print('제 담당 과목은 %s입니다.' % self.subject)

class Parent(Person):
    def __init__(self, name, age, address, child):
        super().__init__(name, age, address)
        self.child = child      # 자식 인스턴스

    def introduce(self):
        super().introduce()
        print('제 자식은 %s 입니다.' % self.child)


teacher = Teacher("홍길동", 34, "대구광역시 달서구", "컴퓨터 과학")
teacher.introduce()    # 자기소개
```

출력 결과

```
안녕하세요, 제 이름은 홍길동이고,
나이는 34세, 대구광역시 달서구에서 살고 있습니다.
제 담당 과목은 컴퓨터 과학입니다.
```

클래스를 상속받으려면 자식 클래스명 옆에 부모 클래스명을 인자로 넣어주어야 합니다. 클래스를 상속받았다면, 부모 클래스의 메서드와 속성들을 불러서 사용할 수 있게 되고, 부모를 뜻하는 super() 명령어를 사용해 부모 클래스의 속성, 메서드를 불러올 수 있습니다.

코딩팩
기본기 익히기

3

</> </>

```
OR_HANDLER_FUNC = CFUNCTYPE(None, c_char_p, c_int, c_char_p, c_int, c_char_p)
 py_error_handler(filename, line, function, err, fmt):
ss
rror_handler = ERROR_HANDLER_FUNC(py_error_handler)

und = cdll.LoadLibrary('libasound.so')
und.snd_lib_error_set_handler(c_error_handler)
 detect_wake_up_word(keyword = '기가지니'):
 not keyword in KWS_KEYWORDS:
return False
```

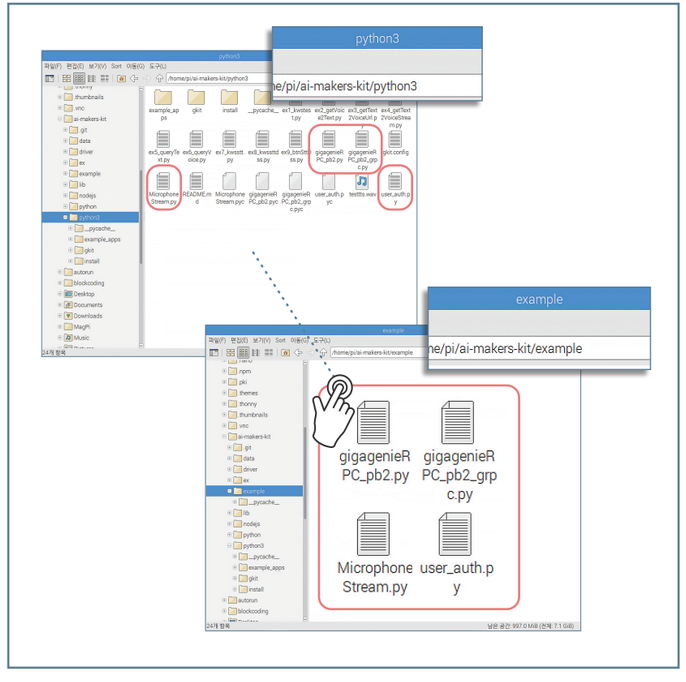
```
ERROR_HANDLER_FUNC = CFUNCTYPE(None, c_char_p, c_int, c_char_p, c_int, c_char_p)
def py_error_handler(filename, line, function, err, fmt):
  pass
c_error_handler = ERROR_HANDLER_FUNC(py_error_handler)

asound = cdll.LoadLibrary('libasound.so')
asound.snd_lib_error_set_handler(c_error_handler)

def detect_wake_up_word(keyword = '기가지니'):
 if not keyword in KWS_KEYWORDS:
   return False
```

01
코딩팩 준비하기

‘코딩팩 기본 예제 탐구하기’에서는 인공지능 스피커가 꼭 갖추어야 되는 호출어 감지, 음성인식, 음성합성, 질의응답에 대해 알아보고 구현해보는 시간을 가집니다. 여러 모듈과 함수를 오가며 복잡한 동작을 하지만 꼭 알아야 되는 부분이기 때문에 잘 따라오시길 부탁드립니다.

기본 예제를 해보길 희망하시는 분들은 코딩팩 설명서를 참고하시길 부탁드립니다.

가장 먼저 해야 되는 일은 사용되는 ‘코딩팩 기본 예제 탐구하기’를 진행하기 위해 사용되는 마이크, 키, 서버 통신 모듈을 ai-makers-kit/python3 폴더에서 ai-makers-kit/example로 옮기는 작업이 필요합니다. ai-makers-kit/python3 폴더에서 ‘gigagenieRPC_pb2_grpc.py’, ‘gigagenieRPC.py’, ‘MicrophoneStream.py’, ‘user_auth.py’ 모듈 네 개를 복사해 ai-makerskit/example 폴더에 붙여 넣습니다.



1) 마이크와 스피커 테스트하기

테스트 프로그램을 통해 오디오와 마이크가 정상적으로 동작하는지를 확인합니다.

① 프로그램 실행

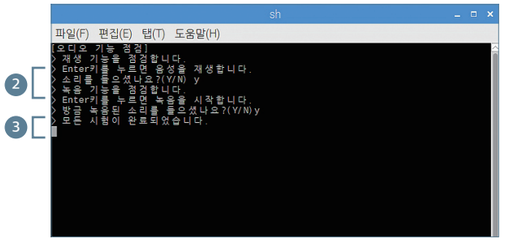
바탕화면  ‘Check Audio’을 더블클릭하여 프로그램을 실행시킵니다.

② 테스트 진행

엔터키를 눌러 스피커 테스트를 진행합니다. 만약 스피커에서 소리가 출력되었다면 안내에 따라 알파벳 y를 입력 후 엔터키를 눌러 다음으로 넘어갑니다.

③ 녹음 진행

“Enter키를 누르면 녹음을 시작합니다.”라는 메시지를 확인 후 엔터키를 누르면 5초간 녹음을 시작하고, 녹음이 끝난 후 녹음된 소리를 출력합니다. 녹음된 소리가 제대로 출력되었다면 알파벳 y를 입력 후 엔터키를 눌러 테스트를 종료합니다.



TIP

- 음성출력이 되지 않는다면?
  : 스피커가 제대로 연결되었는지 다시 확인합니다.
- 녹음이 되지 않는다면?
  : 마이크가 제대로 연결되었는지 다시 확인합니다.
- 스피커 소리가 작다면?
  : 스피커가 뒤집어서 조립되지 않았는지 확인합니다.

2) 인터넷 연결 및 연결 확인하기

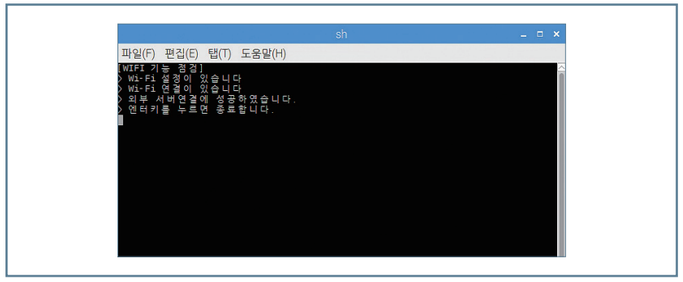
코딩팩을 사용하기 위해서 네트워크 연결은 필수 사항입니다. 라즈베리파이에서는 유선랜과 무선랜 두 가지 방법을 통해 네트워크를 연결할 수 있습니다. 본 교재에서는 무선랜(WiFi)을 사용하여 진행하였습니다.

1 프로그램 실행

바탕화면  'Check WiFi'을 더블클릭하여 프로그램을 실행시킵니다.

2 확인

연결이 되어있다는 메시지를 확인 후 엔터를 눌러 프로그램을 종료시킵니다.



만약 연결이 되어 있지 않다는 메시지를 확인하였다면, 네트워크가 제대로 연결되어 있지 않는 상태입니다. 네트워크를 연결 후 프로그램을 다시 실행시켜 연결이 되었는지 확인이 필요합니다.

3) API 키 발급 및 키 입력하기

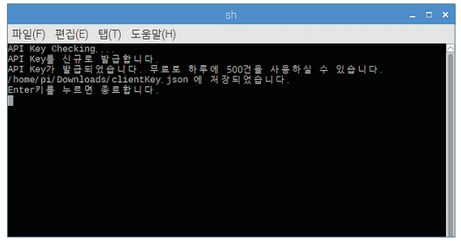
API 키는 'KT AI 서버'를 이용하는데 필요합니다. API 키를 발급받고 입력을 해야만 음성인식, 음성합성, 대화 등 서비스를 이용할 수 있습니다.

1 프로그램 실행

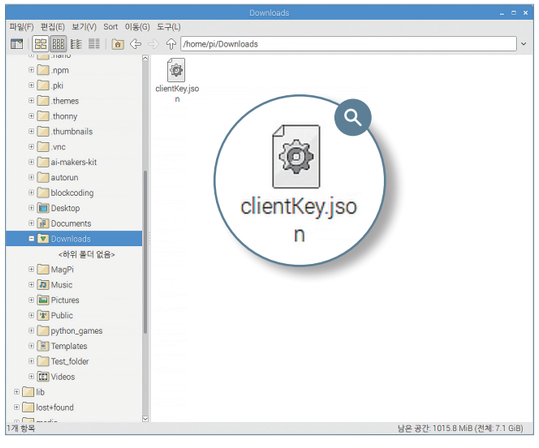
바탕화면  'API 키 발급'을 더블클릭하여 프로그램을 실행시킵니다.

2 확인

프로그램에서 키가 발급되었다는 메시지를 확인 후 엔터키를 눌러 프로그램을 종료합니다. 만약 정상적으로 키가 발급되지 않았다면 프로그램을 다시 실행시켜 키를 발급 받습니다.



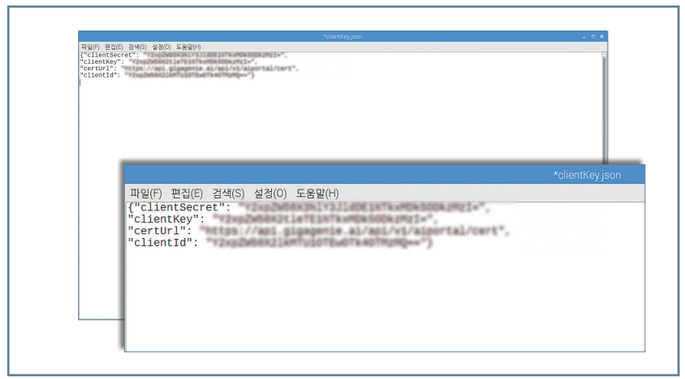
발급받은 키는 'Downloads' 폴더에 'clientKey.json' 파일명으로 저장됩니다.



4) API 키 입력하기

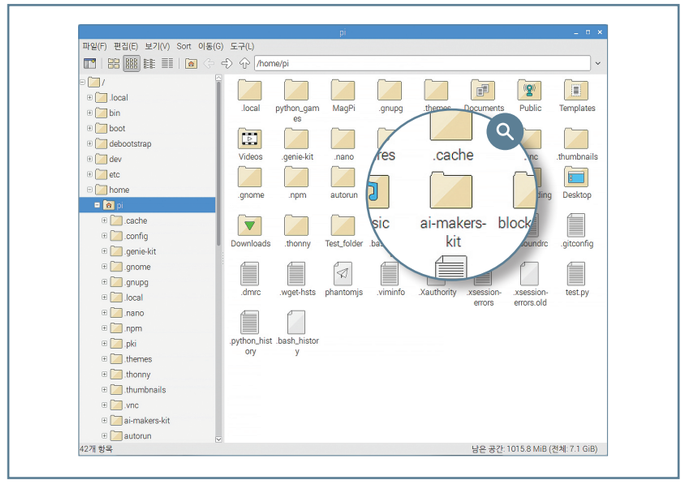
1 프로그램 실행

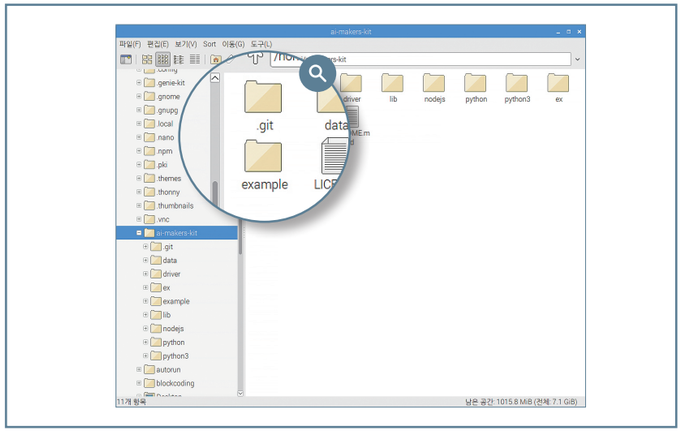
‘clientKey.json’을 열어 아래 사진과 같이 "clientSecret", "clientKey", "clientId"을 확인합니다.

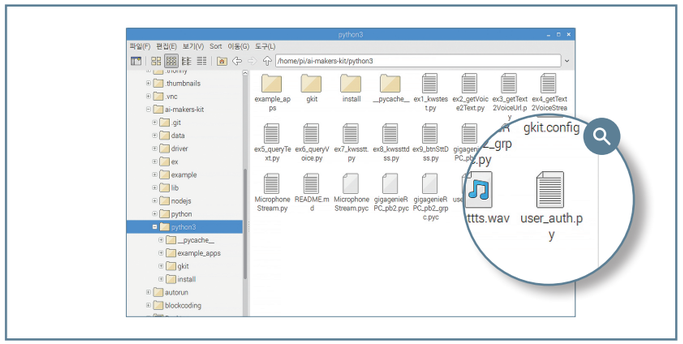


2 파일 열기

키를 확인하였다면, ‘example’ 폴더로 이동하여 ‘user_auth.py’ 파일을 열어줍니다. ‘example’ 폴더는 ‘파일 매니저’ -> ‘ai-makers-kit’ -> ‘example’ 경로에서 찾을 수 있습니다.





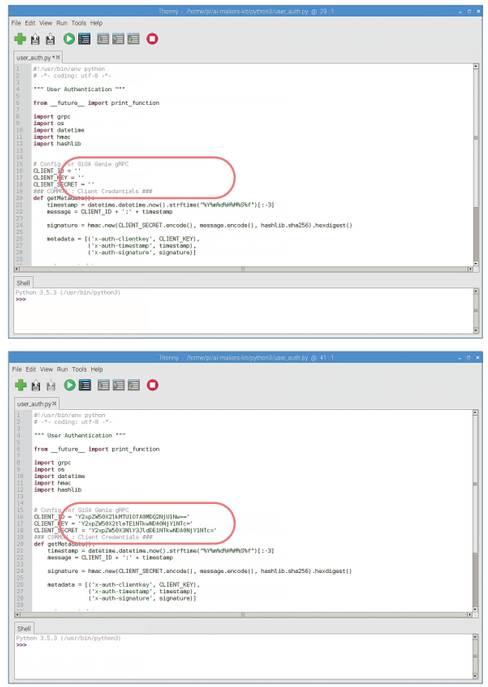


TIP

‘user_auth.py’ 파일은 키 값을 저장해두는 모듈로 사용됩니다. 원래는 모든 예제 파일에 키를 일일이 입력해 주어야 되지만 앞서 배웠던 모듈을 사용하면 필요할 때마다 불러와 사용할 수 있습니다.

❸ 키 작성

발급받은 키를 복사해 아래와 같이 붙여 넣은 후 저장합니다.



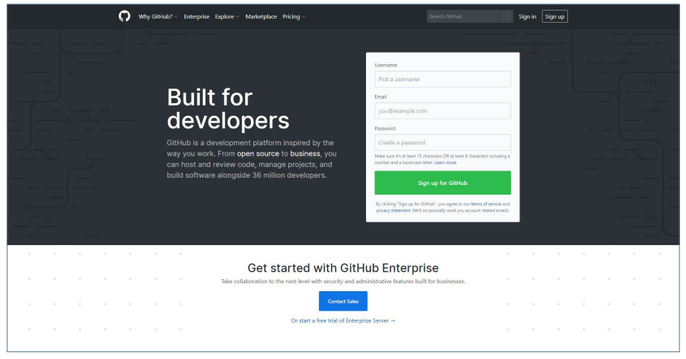
02
호출어 감지

인공지능 스피커에게 무엇인가를 부탁하고 물어볼 때 가장 먼저 하는 일이 무엇일까요? 아마 인공지능 스피커를 부르는 일일 것입니다. 우리는 코딩팩을 "기가지니" 혹은 "지니야", "친구야", "자기야" 중 한 가지를 선택해서 부를 수 있고, 코딩팩을 호출할 때 사용하는 단어라고 해서 **호출어**Wake-up word 라고 부릅니다.

코딩팩은 호출어를 감지하기 위해 **핵심어 검출**Keyword spotting 이라는 기술을 사용합니다. 핵심어 검출은 연속으로 입력되는 음성 데이터에서 미리 정해놓은 핵심어를 찾아내는 기술로, 입력되는 음성 데이터를 소리의 기본 단위인 음소로 나누어 학습된 음성 데이터와 비교함으로 핵심어가 입력되었는지를 판단합니다.

1) GitHub을 이용하여 예제 코드 다운로드하기

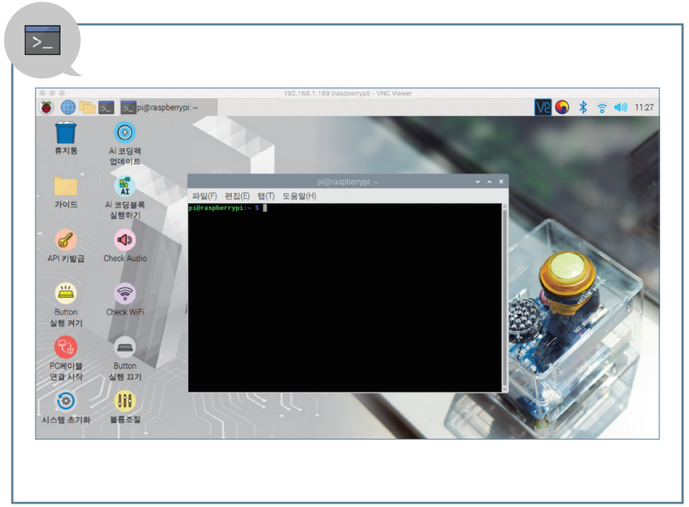
git이란 프로그램을 개발할 때 사용하는 소스코드를 저장소에 저장하여 관리하는 분산 관리 시스템으로, git을 사용하면 여러 사람들이 하나의 소스코드를 수정할 때 체계적으로 관리할 수 있습니다. 하지만 git은 사용자의 컴퓨터에 저장소를 두거나 내부 서버에 저장소를 두고 사용합니다. 즉, 나 또는 내부 사람들끼리만 공유가 가능하다는 아쉬운 점을 가지고 있었습니다. Github는 git의 아쉬운 점을 해결하기 위해 외부 서버에 저장소를 만들어 git을 관리해주는 서비스입니다. Github로 인해 소스코드를 많은 사람과 공유할 수 있게 되었고, 우리가 흔히 알고 있는 오픈소스를 쉽게 가져와 사용할 수 있게 되었습니다. 우리가 만들어 볼 예제와 프로젝트 파일을 Github에 업로드해두었습니다. 라즈베리파이 터미널 창을 사용하여 Github에 저장되어 있는 예제들을 불러와 보도록 하겠습니다.



GitHub sign in page
사진출처<https://github.com>

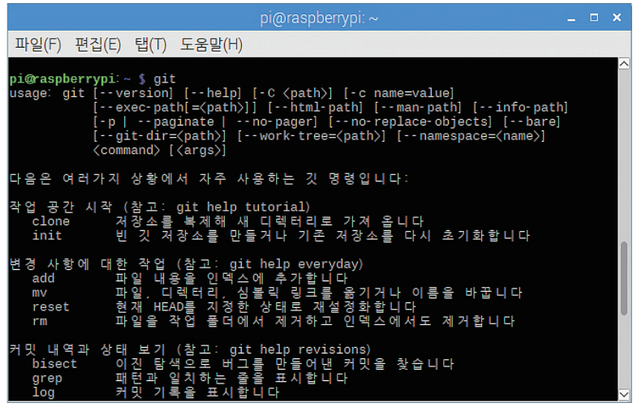
1 프로그램 실행

터미널을 실행합니다.



2 git 설치 확인하기

원래 git을 사용할 때에는 git 패키지를 설치해주어야 하지만 라즈베리파이 OS인 라즈비안에는 git 패키지가 기본적으로 설치되어 있습니다. 터미널에 git이라고 입력하면 사용할 수 있는 패키지 명령어가 출력되는 것을 확인할 수 있습니다.



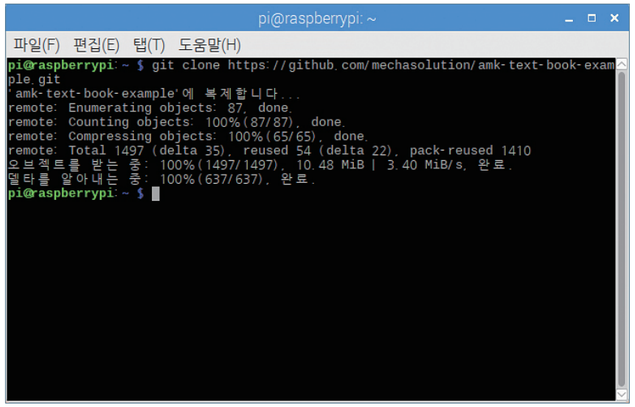
3 예제 다운로드하기

이제 이 git 명령을 사용하여 Github에 저장되어 있는 예제 파일을 가져와 보도록 하겠습니다. 터미널 창에 아래와 같이 입력하여 예제를 다운로드합니다.

다운로드 형식

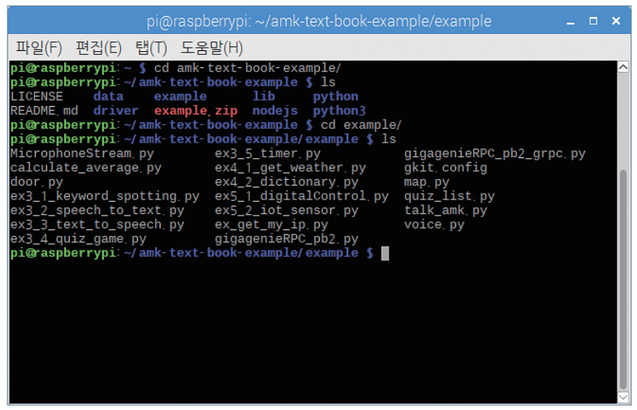
git clone “저장소 위치(URL)”

```
git clone https://github.com/mechasolution/amk-text-book-example.git
```



4 확인하기

이렇게 가져온 예제 소스코드들은 ~/amk-text-book-example 폴더 안에 저장되어 있습니다. 이제 이 폴더 안에 있는 소스코드를 사용하여 예제를 실행시키거나 소스코드를 개발할 수 있습니다.



코딩 전체보기

1 119P
2 119P-120P
3 120P
4 121P
5 121P-124P
6 125P

```
import audioop
from ctypes import *
import ktkws # KWS
import MicrophoneStream as MS
KWS_KEYWORDS = ['기가지니', '지니야', '친구야', '자기야']

RATE = 16000
CHUNK = 512

ERROR_HANDLER_FUNC = CFUNCTYPE(None, c_char_p, c_int, c_char_p, c_int, c_char_p)
def py_error_handler(filename, line, function, err, fmt):
    pass
c_error_handler = ERROR_HANDLER_FUNC(py_error_handler)

asound = cdll.LoadLibrary('libasound.so')
asound.snd_lib_error_set_handler(c_error_handler)

 def detect_wake_up_word(keyword = '기가지니'):
    if not keyword in KWS_KEYWORDS:
        return False

    response_code = ktkws.init("../data/kwsmodel.pack")
    print('response_code on init = %d' % (response_code))
    response_code = ktkws.start()
    print('response_code on start = %d' % (response_code))
    print('\n호출어를 불러보세요~\n')
    ktkws.set_keyword(KWS_KEYWORDS.index(keyword))

    with MS.MicrophoneStream(RATE, CHUNK) as stream:
        audio_generator = stream.generator()

        for content in audio_generator:
            response_code = ktkws.detect(content)

            if response_code == 1:
                MS.play_file("../data/sample_sound.wav")
                print('\n\n호출어가 정상적으로 인식되었습니다.\n\n')
                ktkws.stop()
                return True

if __name__ == "__main__":
    detect_wake_up_word()
```


코딩하기

1 'ex3_1_keyword_spotting.py' 파일을 만들고 모듈과 상수들을 선언합니다

```
import audioop
from ctypes import *
import ktkws # KWS
import MicrophoneStream as MS
KWS_KEYWORDS = ['기가지니', '지니야', '친구야', '자기야']

RATE = 16000
CHUNK = 512
```

부가 설명

audioop
오디오를 조작하는 모듈로 원신호(음성 데이터)를 가공하는데 사용됩니다.

MicrophoneStream
마이크를 조작하는 모듈로 마이크를 통해 음성 신호를 입력 받을 때 사용됩니다.

ctypes
python용 외부 함수(foregin function) 라이브러리 모듈로, ALSA configure에서 발생하는 Error Handler를 조작할 때 사용됩니다.

KWS_KEYWORDS
코딩팩에서 사용할 수 있는 호출어 리스트입니다.

ktkws
키워드 검출 모듈로 호출어 검출 및 사용에 관련된 함수들이 정의되어 있습니다.

※ ALSA는 라즈베리파이 오디오 드라이버입니다.

2 ALSA(Advanced Linux Sound Architecture)의 Configuration으로 인해 발생하는 불필요한 에러 메시지를 삭제하기 위한 Python Error Handler를 정의합니다.

```
ERROR_HANDLER_FUNC = CFUNCTYPE(None, c_char_p, c_int, c_char_p, c_int, c_char_p)
def py_error_handler(filename, line, function, err, fmt):
 pass
c_error_handler = ERROR_HANDLER_FUNC(py_error_handler)

asound = cdll.LoadLibrary('libasound.so')
asound.snd_lib_error_set_handler(c_error_handler)
```

3 120P
4 121P
5 121P-124P

```
def detect_wake_up_word(keyword = '기가지니'):
    if not keyword in KWS_KEYWORDS:
        return False

    response_code = ktkws.init("../data/kwsmodel.pack")
    print ('response_code on init = %d' % (response_code))
    response_code = ktkws.start()
    print ('response_code on start = %d' % (response_code))
    print ('\n호출어를 불러보세요~\n')
    ktkws.set_keyword(KWS_KEYWORDS.index(keyword))

    with MS.MicrophoneStream(RATE, CHUNK) as stream:
        audio_generator = stream.generator()

        for content in audio_generator:
            response_code = ktkws.detect(content)

            if (response_code == 1):
                MS.play_file("../data/sample_sound.wav")
                print ('\n\n호출어가 정상적으로 인식되었습니다.\n\n')
                ktkws.stop();
                return True
```

※ “기가지니”, “지니야”, “친구야”, “자기야” 이외 다른 호출어는 사용할 수 없습니다.

❸ 호출어를 감지하는 함수(detect_wake_up_word)를 생성합니다.

```
def detect_wake_up_word(keyword = '기가지니'):
    if not keyword in KWS_KEYWORDS:
        return False
```

부가 설명

`def detect_wake_up_word(keyword = '기가지니'):`

사용할 호출어를 함수의 인자로 선택합니다. 코딩팩에서 호출어로 사용할 수 있는 키워드를 KWS_KEYWORDS 리스트에 정리해두었습니다. 기본값은 기가지니 입니다.

```
if not keyword in KWS_KEYWORDS:
    return False
```

호출어로 사용할 단어가 KWS_KEYWORDS 리스트에 포함되어있는지를 확인합니다. 만약 호출어로 사용할 단어가 리스트에 포함되지 않았다면, 사용할 수 없는 단어이기 때문에 False를 반환해 함수 종료 시킵니다.

❹ 호출어 감지를 위해 ktkws 모듈 환경을 설정합니다.

```
def detect_wake_up_word(keyword = '기가지니'):
   ...
    response_code = ktkws.init("../data/kwsmodel.pack")
    print ('response_code on init = %d' % (response_code))
    response_code = ktkws.start()
    print ('response_code on start = %d' % (response_code))
    print ('\n호출어를 불러보세요~\n')
    ktkws.set_keyword(KWS_KEYWORDS.index(keyword))
```

부가 설명

`ktkws.init()`

혹시나 ktkws모듈에 다른 데이터가 저장되어있는 것을 방지하기 위해 ktkws 모듈을 초기화 시킵니다. 이때 초기화 데이터로 키워드를 미리 학습시켜 놓은 모델이 저장된 파일을 사용합니다. 초기화가 정상적으로 완료되면 결괏값으로 숫자 1을 반환합니다.

`ktkws.start()`

호출어 대기 상태에서 호출어 인식상태로 변경하는 함수입니다.

`ktkws.set_keyword()`

코딩팩의 호출어로 사용할 단어를 설정하는 함수로 변수 keyword에 저장된 호출어를 함수의 인자로 사용합니다.

❺ 마이크를 통해 입력받은 음성에 호출어가 포함되어 있는지를 확인합니다. 아래 코드에는 generator라는 개념이 등장합니다. 다음장에 따로 설명해 놓은 generator의 개념을 공부한 후 아래 내용을 확인하시길 바랍니다.

```
def detect_wake_up_word(keyword = '기가지니'):
   ...
    with MS.MicrophoneStream(RATE, CHUNK) as stream:
        audio_generator = stream.generator()

        for content in audio_generator:
            response_code = ktkws.detect(content)

            if response_code == 1:
                MS.play_file("../data/sample_sound.wav")
                print('\n\n호출어가 정상적으로 인식되었습니다.\n\n')
                ktkws.stop()
                return True
```

부가 설명

```
with MS.MicrophoneStream(RATE, CHUNK) as stream:
    audio_generator = stream.generator()
```

마이크 사용을 도와주는 MicrophoneStream 클래스를 이용해 음성데이터를 어떻게(RATE, CHUNK) 가져올지를 설정합니다.

```
for content in audio_generator:
```

audio_generator 로부터 음성 데이터를 받아와 content 에 할당합니다.

```
response_code = ktkws.detect(content)
```

입력된 음성 데이터에 호출어가 포함되었는지를 확인합니다. 만약 호출어가 포함되었다면 결괏값으로 숫자 1을 반환하고, 그렇지 않을 경우 숫자 0을 반환합니다.

```
if response_code == 1:
    MS.play_file("../data/sample_sound.wav")
    print('\n\n호출어가 정상적으로 인식되었습니다.\n\n')
```

response_code가 1일 경우 즉, 음성 데이터에 호출어가 포함되어있는 경우, 호출어가 인식되었다는 의미로 “띠리링” 소리가 녹음된 음향 파일을 재생하고 “호출어가 정상적으로 인식되었습니다.”를 출력합니다.

```
ktkws.stop()
```

호출어 인식 상태에서 호출어 대기 상태로 변경합니다. 더 이상 핵심어를 감지하지 않습니다.

```
return True
```

결괏값으로 True를 반환합니다.

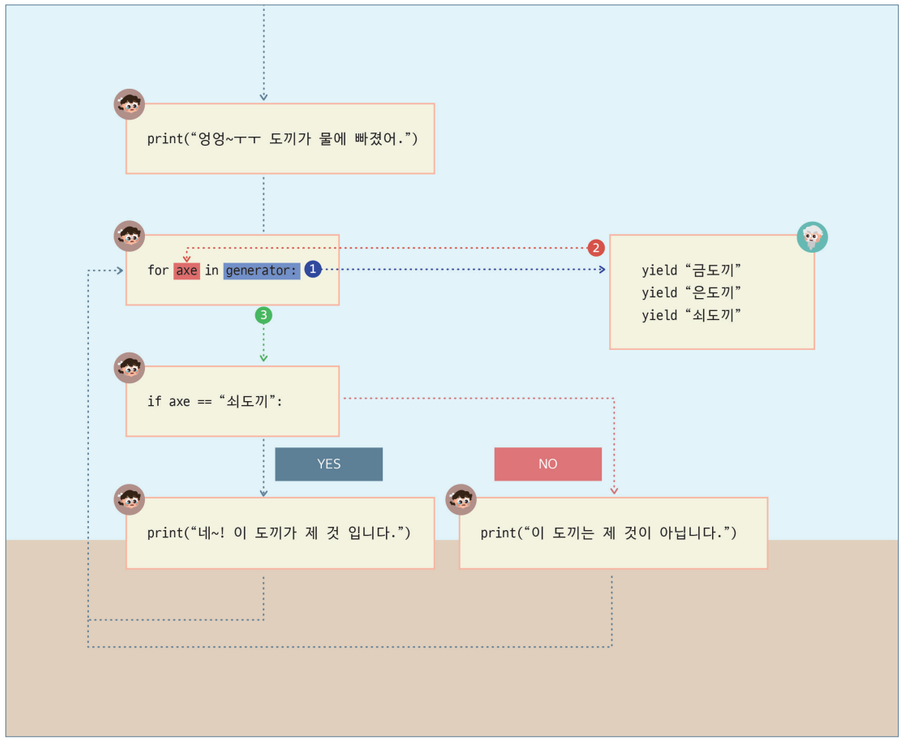
★generator

gnerator는 함수와 비슷하지만 결괏값을 어떤 시점에서 반환해주냐에 따라 다르게 사용됩니다. 함수는 결괏값을 함수가 끝날 때 반환하지만, generator는 yield 명령어를 사용하여 원하는 시점에서 결괏값을 반환하고, 다시 그 시점으로 돌아와 아래 남은 코드를 실행시킬 수도 있습니다. 아래 그림을 통해 자세히 설명하도록 하겠습니다.

출력 결과를 보면 generator가 실행되면서부터 나무꾼과 산신령이 번갈아 가면서 실행되는 것을 확인할 수 있습니다.

프로그램이 시작되면, “엉엉~ㅜㅜ 도끼를 잃어버렸어…”가 출력되고, for axe in generator 문이 시작됩니다. 이때, generator는 산신령_도끼_제너레이터로부터 반환값을 요구합니다. 산신령_도끼_제너레이터는 yield 명령어 차례가 될 때까지 동작하다가 즉, “이 금도끼가 네 도끼냐?”를 출력하고, yield 명령어를 만나면 금도끼를 반환합니다. 반환값(금도끼)은 변수 axe에 할당되어 for문 아래 코드가 순차적으로 실행됩니다. 흐름도를 보면 알 수 있듯이 변수 axe의 값이 쇠도끼가 아니라면 다시 for문 처음으로 돌아가고, generator는 산신령_도끼_제너레이터로부터 다음 반환값을 요구합니다.

산신령_도끼_제너레이터는 두번째 yield 명령어 차례가 될때까지 동작 하다가 즉, “이 은도끼가 네 도끼냐?”를 출력하고, 두 번째 yield 명령어 차례에서 은도끼를 반환합니다. 반환값(은도끼)은 변수 axe에 할당되어 for문 아래 코드가 순차적으로 실행됩니다. 이와 같은 방법으로 산신령_도끼_제너레이터에서 마지막 yield 명령어가 실행될 때까지 혹은 for문의 결과로 return을 반환할 때까지 반복됩니다.



6 메인 함수를 정의합니다.

```
if __name__ == "__main__":
    detect_wake_up_word()
```

출력 결과

```
response_code on init = 1
response_code on start = 10

호출어를 불러보세요~

기가지니~!

connect(2) call to /tmp/jack-1000/default/jack_0 failed (err=No such file or directory)
attempt to connect to server failed

호출어가 정상적으로 인식되었습니다.
```

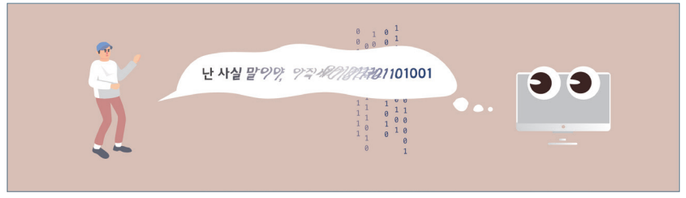
03
음성인식 (STT)

컴퓨터에게 수백 번 같은 말을 반복해도 컴퓨터는 그 말이 무슨 뜻인지 이해하지 못합니다. 마치 외국인이 하는 말을 우리가 알아듣지 못하는 것과 비슷합니다. 그렇다면 어떻게 하면 컴퓨터가 우리 말을 알아들을 수 있을까요? 그건 바로 말을 문자로 바꾸는 것입니다. 우리가 위에서 했던 프로그래밍은 컴퓨터가 찰떡같이 알아듣고 작성된 내용대로 동작합니다. 프로그래밍은 아시다시피 문자로 이루어져 있습니다. 이처럼 코딩팩이 우리 말을 알아듣고 이해하기 위해서는 말을 문자로 바꿔주는 작업이 필요합니다. 이때 사용되는 기술이 **음성인식**Voice Recognition 또는 **STT**Speech-To-Text입니다.

만약 모든 인간이 말하는 방식이 같다면 음성인식이 매우 쉬울 것입니다. 하지만 나이, 성별, 지역 등 여러 조건에 따라 말하는 방식과 목소리는 다를 수밖에 없습니다. 이러한 문제를 컴퓨터는 인간의 뇌를 모방한 **심층 신경망**DNN; Deep Neural Network 기술을 사용하여 수 천만 가지의 데이터를 학습하는 방법으로 해결합니다. 바로 인공지능이 되는 것이죠. 컴퓨터의 데이터 학습으로 만들어진 인공지능은 음성 데이터가 입력되면 학습된 데이터를 바탕으로 최적의 단어를 도출하여 문장으로 만들어줍니다.

음성인식의 성능은 인공지능이 학습한 데이터양에 비례합니다. 학습량이 많으면 많을수록 더 정확한 음성인식 결과를 얻을 수 있습니다. 하지만 우리는 엄청난 양의 데이터를 가지고 있지 않을뿐더러 설령 가지고 있다 하더라도 이 많은 데이터를 학습시키는 일은 매우 어렵습니다. 그래서 우리는 KT가 미리 학습시켜놓은 인공지능이 있는 KT AI 서버로 음성 데이터를 보내 음성인식 기술을 구현합니다.

KT AI 서버를 이용하기 위해서는 서로 통신할 수 있는 수단이 필요한데, 우리는 **gRPC**gRPC Remote Procedure Call 라는 프로토콜을 사용하여 KT AI 서버로 음성 데이터를 보내고, 결과를 다시 받을 것입니다. gRPC 프로토콜은 다른 프로토콜과 달리 복잡한 과정 없이 함수를 사용하듯 쉽게 사용할 수 있습니다.



</> 전체 코드

ex3_2_speech_to_text.py

1 129P

```
import audioop
from ctypes import *
import MicrophoneStream as MS
import grpc
import gigagenieRPC_pb2
import gigagenieRPC_pb2_grpc
import MicrophoneStream as MS
import user_auth as UA

HOST = 'gate.gigagenie.ai'
PORT = 4080

RATE = 16000
CHUNK = 512
```

2 129P

```
ERROR_HANDLER_FUNC = CFUNCTYPE(None, c_char_p, c_int, c_char_p, c_int, c_char_p)
def py_error_handler(filename, line, function, err, fmt):
   pass
c_error_handler = ERROR_HANDLER_FUNC(py_error_handler)

asound = cdll.LoadLibrary('libasound.so')
asound.snd_lib_error_set_handler(c_error_handler)
```

3 130P

```
def generate_request():
   with MS.MicrophoneStream(RATE, CHUNK) as stream:
     audio_generator = stream.generator()
     for content in audio_generator:
       message = gigagenieRPC_pb2.reqVoice()
       message.audioContent = content
       yield message
```

4 131P

```
def get_grpc_stub():
   channel = grpc.secure_channel('{}:{}'.format(HOST, PORT), UA.getCredentials())
   stub = gigagenieRPC_pb2_grpc.GigagenieStub(channel)
   return stub
```

5 132P-133P

```
def get_text_from_voice():
   print ("\n\n 음성인식을 시작합니다.\n\n 종료하시려면 Ctrl+\ 키를 누루세요.\n\n\n")
   stub = get_grpc_stub()
   request = generate_request()
   resultText = ''

  for response in stub.getVoice2Text(request):
     if response.resultCd == 200: # partial
        print('상태 코드=%d ¦ 인식 결과= %s'
              % (response.resultCd, response.recognizedText))
```

```
        resultText = response.recognizedText
      elif response.resultCd == 201: # final
        print('상태 코드=%d ¦ 인식 결과= %s'
              % (response.resultCd, response.recognizedText))
        resultText = response.recognizedText
        break
      else:
        print('상태 코드=%d ¦ 인식 결과= %s'
              % (response.resultCd, response.recognizedText))
        break

  print ("\n\n 최종 인식 결과: %s \n\n" % (resultText))
  return resultText


if __name__ == '__main__':
      print(get_text_from_voice())
```

5 132P-133P

6 133P

</>

코딩하기

1 'ex3_2_speech_to_text.py' 파일을 만들고 필요한 모듈과 상수들을 선언합니다.

```
import audioop
from ctypes import *
import MicrophoneStream as MS
import grpc
import gigagenieRPC_pb2
import gigagenieRPC_pb2_grpc
import MicrophoneStream as MS
import user_auth as UA

HOST = 'gate.gigagenie.ai'
PORT = 4080

RATE = 16000
CHUNK = 512
```

부가 설명

grpc

KT AI 서버와 통신할 때 사용되는 모듈입니다.

gigagenieRPC_pb2, gigagenieRPC_pb2_grpc

KT AI 서버에 제공하는 함수들의 목록과 요청 및 응답 형식이 저장되어 있는 모듈입니다.

user_auth

KT AI 서버를 이용할 수 있는 키를 저장, 관리하는 모듈입니다.

※ ALSA는 라즈베리파이 오디오 드라이버입니다.

2 ALSA(Advanced Linux Sound Architecture)의 Configuration으로 인해 발생하는 불필요한 에러 메시지를 삭제하기 위한 Python Error Handler를 정의합니다.

```
ERROR_HANDLER_FUNC = CFUNCTYPE(None, c_char_p, c_int, c_char_p, c_int, c_char_p)
def py_error_handler(filename, line, function, err, fmt):
 pass
c_error_handler = ERROR_HANDLER_FUNC(py_error_handler)

asound = cdll.LoadLibrary('libasound.so')
asound.snd_lib_error_set_handler(c_error_handler)
```

3 마이크로 입력된 음성을 변수 message에 저장하여 반환하는 제너레이터 generate_request()를 생성합니다.

```
def generate_request():
    with MS.MicrophoneStream(RATE, CHUNK) as stream:
        audio_generator = stream.generator()
        for content in audio_generator:
            message = gigagenieRPC_pb2.reqVoice()
            message.audioContent = content
            yield message
```

부가 설명

```
def generate_request():
  with MS.MicrophoneStream(RATE, CHUNK) as stream:
    audio_generator = stream.generator()
    for content in audio_generator:
      message = gigagenieRPC_pb2.reqVoice()
      message.audioContent = content
```

변수 message에 gigagenieRPC_pb2.reqVoice 클래스의 인스턴스를 생성해 대입하고, 생성된 인스턴스의 audioContent 속성에 음성 데이터를 대입합니다.

```
    yield message
```

음성 데이터가 저장된 변수 message를 반환합니다.

TIP

클래스는 상태와 행동을 나타내기 위해서 사용한다고 배웠습니다. 이번 예제에서는 특별한 행동(메서드)이 없는 클래스(reqText)를 사용하는데요, 그렇다고 딱히 상태가 현실의 것들을 반영하는 것도 아닙니다. 그저 데이터를 담아두기 위한 것으로 사용되는데, 이렇듯 데이터만 담아놓기 위해 사용하는 클래스를 데이터 클래스라고 합니다.

4 KT AI 서버와 통신하기 위한 채널을 설정하고, stub를 생성하는 함수 get_grpc_stub()을 생성합니다. 여기서 stub는 KT AI 서버의 프로그램을 대신 실행시키는 역할을 합니다.

```
def get_grpc_stub():
    channel = grpc.secure_channel('{}:{}'.format(HOST, PORT), UA.getCredentials())
    stub = gigagenieRPC_pb2_grpc.GigagenieStub(channel)
    return stub
```

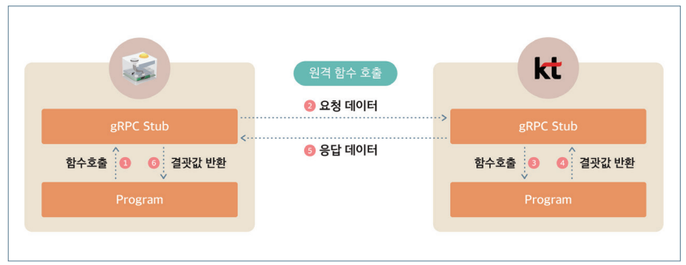
부가 설명

```
channel = grpc.secure_channel('{}:{}'.format(HOST, PORT), UA.getCredentials())
```

코딩팩과 KT AI 서버가 데이터를 주고받는 통로를 설정합니다. 이때 HOST 와 PORT를 설정하고, 앞에서 발급받은 키값을 넣어줍니다.

```
stub = gigagenieRPC_pb2_grpc.GigagenieStub(channel)
```

KT AI 서버의 음성인식, 음성합성, 질의응답 등의 함수를 코딩팩에서 복잡한 절차를 거치지 않고, 내부 함수를 호출하듯이 사용할 수 있게 해줍니다.



5 마이크로 입력된 음성 데이터(request)를 KT AI 서버로 보내 음성인식 결과(문자)를 받아오는 함수 get_text_from_voice()를 선언합니다.

```
def get_text_from_voice():
    print ("\n\n 음성인식을 시작합니다.\n\n 종료하시려면 Ctrl+\ 키를 누루세요.\n\n\n")
    stub = get_grpc_stub()
    request = generate_request()
    resultText = ''

  for response in stub.getVoice2Text(request):
      if response.resultCd == 200: # partial
          print('상태 코드=%d ¦ 인식 결과= %s'
                % (response.resultCd, response.recognizedText))
        resultText = response.recognizedText
      elif response.resultCd == 201: # final
        print('상태 코드=%d ¦ 인식 결과= %s'
              % (response.resultCd, response.recognizedText))
        resultText = response.recognizedText
        break
      else:
        print('상태 코드=%d ¦ 인식 결과= %s'
              % (response.resultCd, response.recognizedText))
        break

  print ("\n\n 최종 인식 결과: %s \n\n" % (resultText))
  return resultText
```

부가 설명

```
    stub = get_grpc_stub()
```

KT AI 서버와 통신할 때 사용하는 stub를 생성합니다.

```
  request = generate_request()
```

마이크로 입력된 음성 데이터를 받아옵니다.

```
  for response in stub.getVoice2Text(request):
      if response.resultCd == 200: # partial
          print('상태 코드=%d ¦ 인식 결과= %s'
                % (response.resultCd, response.recognizedText))
        resultText = response.recognizedText
      elif response.resultCd == 201: # final
        print('상태 코드=%d ¦ 인식 결과= %s'
              % (response.resultCd, response.recognizedText))
        resultText = response.recognizedText
        break
      else:
        print('상태 코드=%d ¦ 인식 결과= %s'
              % (response.resultCd, response.recognizedText))
        break

  print("\n\n 최종 인식 결과: %s \n\n" % (resultText))
  return resultText .
```

실제 KT AI 서버와 통신하는 제너레이터로, 마이크에서 입력된 음성 데이터를 KT AI 서버로 보내 음성인식 결과를 받아옵니다. KT AI 서버에서 반환되는 데이터는 변수 response에 할당되고, 상태 코드를 담은 resultCd, 인식된 문장을 담은 recognizedText를 포함하고 있습니다. 음성인식이 일부만 완성되었을 경우 상태 코드로 숫자 200을 반환하고, 최종 완료되었을 경우 숫자 201을 반환합니다. 변수 resultText에는 음성인식 결과가 저장됩니다.

6 메인 함수를 정의하고 프로그램을 실행한 후 음성인식이 정상적으로 되는지 확인해봅시다.

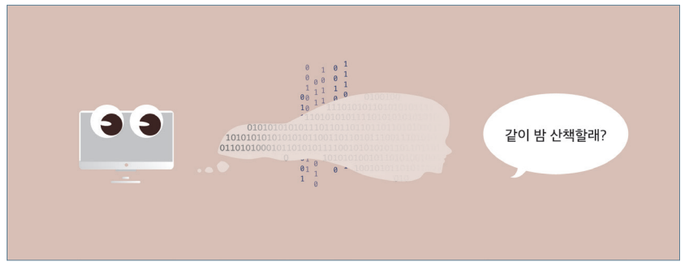
```
if __name__ == '__main__':
    print(get_text_from_voice())
```

출력 결과

```
음성인식을 시작합니다.

종료하시려면 Ctrl+\ 키를 누루세요.


connect(2) call to /tmp/jack-1000/default/jack_0 failed (err=No such file
or directory) attempt to connect to server failed
상태 코드=200 ¦ 인식 결과= 안녕
상태 코드=200 ¦ 인식 결과= 안녕 하세요
상태 코드=201 ¦ 인식 결과= 안녕 하세요


최종 인식 결과: 안녕 하세요


안녕 하세요
```

04
음성합성 (TTS)

음성인식이 음성을 문자로 바꾸는 기술이라면, **음성 합성**Voice Synthesis 은 문자를 음성으로 바꾸는 기술입니다. 앞에서 인공지능을 설명할 때 컴퓨터는 인간의 말을 알아듣지 못하지만, 문자는 찰떡같이 알아듣는다고 이야기했습니다. 그렇다면, 컴퓨터가 결과로 돌려주는 데이터는 과연 어떤 형태로 되어있을까요? 당연히 문자로 되어있습니다. 인공지능 스피커는 결과를 음성으로 출력해야 하는데 결과가 문자로 되어있으니 다시 음성으로 변환을 해 주어야 합니다. 이처럼 사용자의 질문에 대한 답변 혹은 사용자의 요청에 대한 결과를 음성으로 알려주기 위해 음성합성 기술을 사용합니다.



</> 전체 코드

ex3_3_text_to_speech.py

```
import audioop
from ctypes import *
import MicrophoneStream as MS
import grpc
import gigagenieRPC_pb2
import gigagenieRPC_pb2_grpc
import MicrophoneStream as MS
import user_auth as UA

HOST = 'gate.gigagenie.ai'
PORT = 4080

RATE = 16000
CHUNK = 512

ERROR_HANDLER_FUNC = CFUNCTYPE(None, c_char_p, c_int, c_char_p, c_int, c_char_p)
def py_error_handler(filename, line, function, err, fmt):
  pass
c_error_handler = ERROR_HANDLER_FUNC(py_error_handler)

asound = cdll.LoadLibrary('libasound.so')
asound.snd_lib_error_set_handler(c_error_handler)

def generate_request():
  with MS.MicrophoneStream(RATE, CHUNK) as stream:
    audio_generator = stream.generator()
    for content in audio_generator:
      message = gigagenieRPC_pb2.reqVoice()
      message.audioContent = content
      yield message

def get_grpc_stub():
  channel = grpc.secure_channel('{}:{}'.format(HOST, PORT), UA.getCredentials())
  stub = gigagenieRPC_pb2_grpc.GigagenieStub(channel)
  return stub

def get_voice_from_text(text, output_file_name = 'tts.wav'):
  stub = get_grpc_stub()

  message = gigagenieRPC_pb2.reqText()
  message.lang = 0  # 0: 한국어, 1: 영어
  message.text = text

  with open(output_file_name, 'wb') as output:
    for response in stub.getText2VoiceStream(message):
      result_code = response.resOptions.resultCd
```

1 137P
2 137P
3 138P
4 138P
5 138P-139P

```
            print("\n\n 음성합성 응답 상태코드:", result_code)

            if result_code == 200 or result_code == 0:
                output.write(response.audioContent)
            else:
                return False

    return True

def speech(text):
    result_code = get_voice_from_text(text)
    if result_code == True:
        MS.play_file('tts.wav')
        print('음성이 출력되었습니다.')
    else:
        print('에러가 발생하였습니다.')


if __name__ == '__main__':
      speech('안녕하세요? 기가지니입니다.')
```

5 138P-139P

6 140P

7 141P-141P

</> 코딩하기

1 'ex3_3_text_to_speech.py' 파일을 만들고 필요한 모듈과 상수들을 선언합니다.

```
import audioop
from ctypes import *
import MicrophoneStream as MS
import grpc
import gigagenieRPC_pb2
import gigagenieRPC_pb2_grpc
import MicrophoneStream as MS
import user_auth as UA
import os

HOST = 'gate.gigagenie.ai'
PORT = 4080

RATE = 16000
CHUNK = 512
```

부가 설명

os(Operating System)

운영체제에서 제공하는 여러 기능을 파이썬에서 사용할 수 있도록 해주는 모듈입니다. 예를 들어, 파이썬을 이용해 라즈베리파이의 파일을 복사하거나 삭제할 수 있습니다.

※ ALSA는 라즈베리파이 오디오 드라이버입니다.

2 ALSA(Advanced Linux Sound Architecture)의 Configuration으로 인해 발생하는 불필요한 에러 메시지를 삭제하기 위한 Python Error Handler를 정의합니다.

```
ERROR_HANDLER_FUNC = CFUNCTYPE(None, c_char_p, c_int, c_char_p, c_int, c_char_p)
def py_error_handler(filename, line, function, err, fmt):
 pass
c_error_handler = ERROR_HANDLER_FUNC(py_error_handler)

asound = cdll.LoadLibrary('libasound.so')
asound.snd_lib_error_set_handler(c_error_handler)
```

❸ 마이크로 입력된 음성을 변수 message에 저장하여 반환하는 제너레이터 generate_request()를 생성합니다.

```
def generate_request():
    with MS.MicrophoneStream(RATE, CHUNK) as stream:
        audio_generator = stream.generator()
        for content in audio_generator:
            message = gigagenieRPC_pb2.reqVoice()
            message.audioContent = content
            yield message
```

❹ KT AI 서버와 통신하기 위한 채널을 설정하고, stub를 생성하는 함수 get_grpc_stub()을 생성합니다. 여기서 stub는 KT AI 서버의 프로그램을 대신 실행시키는 역할을 합니다.

```
def get_grpc_stub():
    channel = grpc.secure_channel('{}:{}'.format(HOST, PORT), UA.getCredentials())
    stub = gigagenieRPC_pb2_grpc.GigagenieStub(channel)
    return stub
```

❺ 입력한 문자를 KT AI 서버로 보내고, 음성으로 변환된 데이터를 받아오는 함수 get_voice_from_text()를 선언합니다.

```
def get_voice_from_text(text, output_file_name = 'tts.wav'):
    stub = get_grpc_stub()

    message = gigagenieRPC_pb2.reqText()
    message.lang = 0  # 0: 한국어, 1: 영어
    message.text = text

    with open(output_file_name, 'wb') as output:
        for response in stub.getText2VoiceStream(message):
            result_code = response.resOptions.resultCd
            print ("\n\n 음성합성 응답 상태코드:", result_code)

            if result_code == 200 or result_code == 0:
                output.write(response.audioContent)
            else:
                return False

    return True
```

부가 설명

```
stub = get_grpc_stub()
```
KT AI 서버와 통신할 때 사용하는 stub를 생성합니다.

```
message = gigagenieRPC_pb2.reqText()
message.lang = 0  # 0: 한국어, 1: 영어
message.text = text
```
변수 message에 gigagenieRPC_pb2.reqText 클래스의 인스턴스를 생성해 대입하고, 생성된 인스턴스의 lang 속성에 0과 1을 넣어 언어 설정을 하고, text 속성에 음성 합성되기를 원하는 문장을 넣어 줍니다.

```
with open(output_file_name, 'wb') as output:
```
with - as 문을 사용하여 음성합성 결괏값을 받아올 tts.wav(output_file_name) 파일을 열고, 쓰기 원한을 얻기 위해 'wb'를 인자로 사용합니다.

```
for response in stub.getText2VoiceStream(message):
    result_code = response.resOptions.resultCd
    print ("\n\n 음성합성 응답 상태코드:", result_code)

    if result_code == 200 or result_code == 0:
        output.write(response.audioContent)
    else:
        return False

return True
```
for 문과 stub.getText2VoiceStream(제너레이터)를 사용하여 문자를 KT AI 서버로 보내고, 음성합성된 결과를 받아옵니다. 결괏값에는 음성합성이 된 데이터(response.audioContent)와 상태 코드(resonse.resOptions.resultCd)를 포함하고 있습니다. 음성합성이 정상적으로 되었으면, 상태 코드로 숫자 200을 반환하고, 마지막 결괏값이라는 의미로 상태 코드 숫자 0을 반환합니다.

```
output.write(response.audioContent)
```
반환된 음성 데이터를 output.write 함수를 사용하여 tts.wav 파일에 저장합니다.

6 메인 함수를 정의하고 프로그램을 실행한 후 음성인식이 정상적으로 되는지 확인해봅시다.

```
def speech(text):
    result_code = get_voice_from_text(text)
    if result_code == True:
        MS.play_file('tts.wav')
        print('음성이 출력되었습니다.')
    else:
        print('에러가 발생하였습니다.')
```

부가 설명

```
result_code = get_voice_from_text(text)
```

입력한 문자를 get_voice_from_text 함수를 사용하여 KT AI 서버로 보내고, 음성으로 변환된 데이터를 파일에 받아오는 함수입니다.

```
if result_code == True:
  MS.play_file('tts.wav')
 print('음성이 출력되었습니다.')
else:
  print('에러가 발생하였습니다.')
```

함수 get_voice_from_text(text)의 반환값이 True일 경우, 'tts.wav' 파일에 저장된 음성 데이터를 재생하고, "음성이 출력되었습니다."를 출력합니다. 만약 그렇지 않을 경우, "에러가 발생하였습니다."를 출력합니다.

7 메인 함수를 정의하고 프로그램을 실행한 후 음성인식이 정상적으로 되는지 확인해봅시다.

```
if __name__ == '__main__':
      speech('안녕하세요? 기가지니입니다.')
```

출력 결과

```
음성합성 응답 상태코드: 200

음성합성 응답 상태코드: 0 connect(2) call to /tmp/jack-1000/default/
jack_0 failed (err=No such file or directory) attempt to connect to server
failed 음성이 출력되었습니다.
```


05
질의 응답(query)

인공지능 스피커는 "오늘 기분이 어때?"하고 물으면 "오늘 기분 최고예요."와 같이 질문에 대한 답변을 해줍니다. 그렇다면 AI 스피커는 어떻게 질문에 대한 답변을 할 수 있는 것일까요? 컴퓨터는 '자연어 처리 기술'을 사용하여 인간의 질문의 의도를 이해합니다. 자연어 처리 기술은 인간의 언어 현상을 컴퓨터와 같은 기계를 이용하여 묘사할 수 있도록 구현한 기술로, 컴퓨터가 '머신러닝'이라는 학습 방법으로 수 천만 가지의 자연어 데이터를 학습하여 인간의 질문에 대한 의도를 파악해 의도에 맞는 답변을 인간에게 돌려주게 됩니다. 바로 인공지능 스피커가 되는 것입니다.

우리는 KT AI 서버를 이용하기 때문에 다행히 자연어 처리 기술을 컴퓨터에 학습시키지 않아도 질문에 대한 답변을 얻을 수 있습니다. 이제부터 KT AI 서버에서 제공하는 query 모듈을 사용하여 문자로 된 질문을 KT AI 서버로 보내고, 답변을 받아와 출력하는 예제를 같이 해보도록 하겠습니다.

전체 코드

ex3_4_text_query.py

```
import grpc
import user_auth as UA
import gigagenieRPC_pb2
import gigagenieRPC_pb2_grpc
import os

HOST = 'gate.gigagenie.ai'
PORT = 4080

def get_grpc_stub():
    channel = grpc.secure_channel('{}:{}'.format(HOST, PORT), UA.getCredentials())
    stub = gigagenieRPC_pb2_grpc.GigagenieStub(channel)
    return stub

 def query_by_text(text):
    stub = get_grpc_stub()
    message = gigagenieRPC_pb2.reqQueryText()
    message.queryText = text
    message.userSession = "1234"       # 단말기의 관리를 위해 넣어놓는 변수들
    message.deviceId = "yourdevice"    # 현재는 쓰이지 않으나 추후 사용 예정

    response = stub.queryByText(message)

    print("\n\nresultCd: %d" % (response.resultCd))
    if response.resultCd == 200:
        print("\n\n\n 질의한 내용: %s" % (response.uword))
        for action in response.action:
            query_response = action.mesg

        query_response = query_response.replace('<![CDATA[', '').replace(']]>', '')
        return query_response
    else:
        return None

if __name__ == '__main__':
    print("질의에 대한 답변:", query_by_text("오늘 기분이 어때?"))
```

1 143P

2 143P

3 144P-145P

4 145P

코딩하기

1 'ex3_4_text_query.py' 파일을 만들고 필요한 모듈과 상수들을 선언합니다.

```
import grpc
import user_auth as UA
import gigagenieRPC_pb2
import gigagenieRPC_pb2_grpc
import os

HOST = 'gate.gigagenie.ai'
PORT = 4080
```

2 KT AI 서버와 통신하기 위한 채널을 설정하고, stub를 생성하는 함수 get_grpc_stub()을 생성합니다. 여기서 stub는 KT AI 서버의 프로그램을 대신 실행시키는 역할을 합니다.

```
def get_grpc_stub():
    channel = grpc.secure_channel('{}:{}'.format(HOST, PORT), UA.getCredentials())
    stub = gigagenieRPC_pb2_grpc.GigagenieStub(channel)
    return stub
```

3 입력된 문자를 KT AI 서버로 보내고, 답변을 받아오는 함수 query_by_text()를 생성합니다.

```
def query_by_text(text):
    stub = get_grpc_stub()
    message = gigagenieRPC_pb2.reqQueryText()
    message.queryText = text
    message.userSession = "1234"  # 단말기의 관리를 위해 넣어놓는 변수들
    message.deviceId = "yourdevice"# 현재는 쓰이지 않으나 추후 사용 예정

    response = stub.queryByText(message)

    print("\n\nresultCd: %d" % (response.resultCd))
    if response.resultCd == 200:
        print("\n\n\n 질의한 내용: %s" % (response.uword))
        for action in response.action:
            query_response = action.mesg

        query_response = query_response.replace('<![CDATA[', '').
replace(']]>', '')
        return query_response
    else:
        return None
```

부가 설명

```
stub = get_grpc_stub()
  message = gigagenieRPC_pb2.reqQueryText()
  message.queryText = text
  message.userSession = "1234"
  message.deviceId = "yourdevice"
```

변수 message에 gigagenieRPC_pb2. reqQueryText() 클래스의 인스턴스를 생성해 대입하고, 생성된 인스턴스의 queryText 와 userSession, deviceId 속성에 변환할 문자와, 단말기 관리 번호, 디바이스 ID를 대입합니다.

```
  response = stub.queryByText(message)
```

입력된 문자를 KT AI 서버로 보내고, 답변을 받아와 변수 response에 대입합니다. 결괏값에는 질문에 대한 답변(response.action)과 상태 코드(response.resultCd)를 포함하고 있습니다. 질문에 대한 답변을 가져오는 과정이 제대로 이루어졌으면, 상태 코드로 숫자 200을 반환합니다.

```
    for action in response.action:
        query_response = action.mesg
```

response 클래스에 여러 개의 action 객체가 포함되는 경우가 발생할 수 있기 때문에 response. action 제너레이터를 확인하고, 변수 query_response에 대입합니다.

```
    query_response = query_response.replace('<![CDATA[', '').
```

KT AI 서버로부터 받은 데이터에서 필요 없는 부분을 제거합니다.

4 메인 함수를 정의하고 프로그램을 실행한 후 음성인식이 정상적으로 되는지 확인해봅시다.

```
if __name__ == '__main__':
    print("질의에 대한 답변:", query_by_text("오늘 기분이 어때?"))
```

출력 결과

```
resultCd: 200

질의한 내용: <![CDATA[오늘 기분이 어때?]]>

질의에 대한 답변: 주말에 여행을 다녀와서 너무 기분이 좋아요.
```

06
KT API를 하나의 모듈로 만들기

지금까지 만들었던 호출어 감지, 음성인식, 음성합성, 질의응답, 네 가지 기능은 앞으로 진행하게 될 프로젝트에서 계속 사용됩니다. 그래서 하나의 모듈로 만들어 놓으면 어떨까 싶습니다. 이제 다 아시겠지만, 모듈로 만들어 놓으면 필요할 때 불러와 쉽게 사용할 수 있습니다. 지금부터 호출어 감지, 음성 인식, 음성 합성, 질의응답을 하나로 묶어놓은 voice 라는 모듈을 만들어 효율적으로 관리하고 사용하도록 하겠습니다.

</> 전체 코드

Voice.py

1 150P

```
import audioop
from ctypes import *
import ktkws # KWS
import MicrophoneStream as MS
import grpc
import gigagenieRPC_pb2
import gigagenieRPC_pb2_grpc
import MicrophoneStream as MS
import user_auth as UA
import os

HOST = 'gate.gigagenie.ai'
PORT = 4080

KWS_KEYWORDS = ['기가지니', '지니야', '친구야', '자기야']

RATE = 16000
CHUNK = 512
```

2 150P-153P

```
ERROR_HANDLER_FUNC = CFUNCTYPE(None, c_char_p, c_int, c_char_p, c_int, c_char_p)
def py_error_handler(filename, line, function, err, fmt):
    pass
c_error_handler = ERROR_HANDLER_FUNC(py_error_handler)

asound = cdll.LoadLibrary('libasound.so')
asound.snd_lib_error_set_handler(c_error_handler)

def detect_wake_up_word(keyword = '기가지니'):
    if not keyword in KWS_KEYWORDS:
        return False

    response_code = ktkws.init("../data/kwsmodel.pack")
    print('response_code on init = %d' % (response_code))
    response_code = ktkws.start()
    print('response_code on start = %d' % (response_code))
    print('\n호출어를 불러보세요~\n')
    ktkws.set_keyword(KWS_KEYWORDS.index(keyword))

    with MS.MicrophoneStream(RATE, CHUNK) as stream:
        audio_generator = stream.generator()

        for content in audio_generator:
            response_code = ktkws.detect(content)

            if response_code == 1:
                MS.play_file("../data/sample_sound.wav")
```

2 150P-153P

```
            print('\n\n호출어가 정상적으로 인식되었습니다.\n\n')
            ktkws.stop()
            return True

def generate_request():
    with MS.MicrophoneStream(RATE, CHUNK) as stream:
        audio_generator = stream.generator()
        for content in audio_generator:
            message = gigagenieRPC_pb2.reqVoice()
            message.audioContent = content
            yield message

def get_grpc_stub():
    channel = grpc.secure_channel('{}:{}'.format(HOST, PORT), UA.getCredentials())
    stub = gigagenieRPC_pb2_grpc.GigagenieStub(channel)
    return stub

def get_text_from_voice():
    print("\n\n 음성인식을 시작합니다.\n\n 종료하시려면 Ctrl+\ 키를 누루세요.\n\n\n")
    stub = get_grpc_stub()
    request = generate_request()
    resultText = ''

    for response in stub.getVoice2Text(request):
        if response.resultCd == 200: # partial
            print('상태 코드=%d ¦ 인식 결과= %s'
                  % (response.resultCd, response.recognizedText))
            resultText = response.recognizedText
        elif response.resultCd == 201: # final
            print('상태 코드=%d ¦ 인식 결과= %s'
                  % (response.resultCd, response.recognizedText))
            resultText = response.recognizedText
            break
        else:
            print('상태 코드=%d ¦ 인식 결과= %s'
                  % (response.resultCd, response.recognizedText))
            break

    print("\n\n 최종 인식 결과: %s \n\n" % (resultText))
    return resultText

def get_voice_from_text(text, output_file_name = 'tts.wav'):
    stub = get_grpc_stub()

    message = gigagenieRPC_pb2.reqText()
```

2 150P-153P

```
    message.lang = 0  # 0: 한국어, 1: 영어
    message.text = text

    with open(output_file_name, 'wb') as output:
        for response in stub.getText2VoiceStream(message):
            result_code = response.resOptions.resultCd
            print("\n\n 음성합성 응답 상태코드:", result_code)

            if result_code == 200 or result_code == 0:
                output.write(response.audioContent)
            else:
                return False

    return True

def speech(text):
    result_code = get_voice_from_text(text)
    if result_code == True:
        MS.play_file('tts.wav')
        print('음성이 출력되었습니다.')
    else:
        print('에러가 발생하였습니다.')

def query_by_text(text):
    stub = get_grpc_stub()
    message = gigagenieRPC_pb2.reqQueryText()
    message.queryText = text
    message.userSession = "1234"      # 단말기의 관리를 위해 넣어놓는 변수들
    message.deviceId = "yourdevice"   # 현재는 쓰이지 않으나 추후 사용 예정

    response = stub.queryByText(message)

    print("\n\nresultCd: %d" % (response.resultCd))
    if response.resultCd == 200:
        print("\n\n\n 질의한 내용: %s" % (response.uword))
        for action in response.action:
            query_response = action.mesg

        query_response = query_response.replace('<![CDATA[', '').replace(']]>', '')
        return query_response
    else:
        return None
```

3 154P

```
if __name__ == '__main__':
    if detect_wake_up_word():
        recognized_text = get_text_from_voice()
        result = query_by_text(recognized_text)
        print(result)
        speech(result)
```

코딩하기

1 'voice.py' 파일을 만들어 필요한 모듈과 상수를 선언합니다.

```
import audioop
from ctypes import *
import ktkws # KWS
import MicrophoneStream as MS
import grpc
import gigagenieRPC_pb2
import gigagenieRPC_pb2_grpc
import MicrophoneStream as MS
import user_auth as UA
import os

HOST = 'gate.gigagenie.ai'
PORT = 4080

KWS_KEYWORDS = ['기가지니', '지니야', '친구야', '자기야']

RATE = 16000
CHUNK = 512
```

2 코딩팩 기본 예제 탐구하기에서 만들었던 ALSA 모듈의 에러 메시지 삭제 코드, get_grpc_stub(), detect_wake_up_word(), generate_request(), get_text_from_voice(), get_voice_from_text(), speech, query_by_text() 함수를 복사해 붙여줍니다.

+

코드 설명

ALSA 모듈 에러 메시지 삭제 코드

```
ERROR_HANDLER_FUNC = CFUNCTYPE(None, c_char_p, c_int, c_char_p, c_int, c_char_p)
def py_error_handler(filename, line, function, err, fmt):
    pass
c_error_handler = ERROR_HANDLER_FUNC(py_error_handler)

asound = cdll.LoadLibrary('libasound.so')
asound.snd_lib_error_set_handler(c_error_handler)
```

detect_wake_up_word() 함수

```
def detect_wake_up_word(keyword = '기가지니'):
    if not keyword in KWS_KEYWORDS:
        return False

    response_code = ktkws.init("../data/kwsmodel.pack")
    print('response_code on init = %d' % (response_code))
    response_code = ktkws.start()
    print('response_code on start = %d' % (response_code))
    print('\n호출어를 불러보세요~\n')
    ktkws.set_keyword(KWS_KEYWORDS.index(keyword))

    with MS.MicrophoneStream(RATE, CHUNK) as stream:
        audio_generator = stream.generator()

        for content in audio_generator:
            response_code = ktkws.detect(content)

            if response_code == 1:
                MS.play_file("../data/sample_sound.wav")
                print('\n\n호출어가 정상적으로 인식되었습니다.\n\n')
                ktkws.stop()
                return True
```

generate_request() 함수

```
def generate_request():
    with MS.MicrophoneStream(RATE, CHUNK) as stream:
        audio_generator = stream.generator()

        for content in audio_generator:
            message = gigagenieRPC_pb2.reqVoice()
            message.audioContent = content
            yield message
```

get_grpc_stub() 함수

```
def get_grpc_stub():
    channel = grpc.secure_channel('%s:%s' % (HOST, PORT), UA.getCredentials())
    stub = gigagenieRPC_pb2_grpc.GigagenieStub(channel)

    return stub
```

get_text_from_voice() 함수

```
def get_text_from_voice():
    print("\n\n음성인식을 시작합니다.\n\n종료하시려면 Ctrl+\ 키를 누루세요.\n\n\n")
    stub = get_grpc_stub()
    request = generate_request()
    resultText = ''

    for response in stub.getVoice2Text(request):
        if response.resultCd == 200: # partial
            print('상태 코드=%d ¦ 인식 결과= %s'
                % (response.resultCd, response.recognizedText))
            resultText = response.recognizedText
        elif response.resultCd == 201: # final
            print('상태 코드=%d ¦ 인식 결과= %s'
                % (response.resultCd, response.recognizedText))
            resultText = response.recognizedText
            break
        else:
            print('상태 코드=%d ¦ 인식 결과= %s'
                % (response.resultCd, response.recognizedText))
            break

    print("\n\n최종 인식 결과: %s \n\n\n" % (resultText))
    return resultText
```

get_voice_from_text() 함수

```
def get_voice_from_text(text, output_file_name = 'tts.wav'):
    stub = get_grpc_stub()

    message = gigagenieRPC_pb2.reqText()
    message.lang = 0 # 0: 한국어, 1: 영어
```

```
    message.text = text

    with open(output_file_name, 'wb') as output:
        for response in stub.getText2VoiceStream(message):
            result_code = response.resOptions.resultCd
            print("\n\n음성합성 응답 상태코드:", result_code)

            if result_code == 200 or result_code == 0:
                output.write(response.audioContent)
            else:
                return False

    return True
```

speech() 함수

```
def speech(text):
    result_code = get_voice_from_text(text)
    if result_code:
        MS.play_file('tts.wav')
        print('음성이 출력되었습니다.')
    else:
        print('에러가 발생하였습니다.')
```

query_by_text() 함수

```
def query_by_text(text):
    stub = get_grpc_stub()
    message = gigagenieRPC_pb2.reqQueryText()
    message.queryText = text
    message.userSession = "1234"      # 단말기의 관리를 위해 넣어놓는 변수들
    message.deviceId = "yourdevice"   # 현재는 쓰이지 않으나 추후 사용 예정

    response = stub.queryByText(message)

    print("\n\nresultCd: %d" % (response.resultCd))
    if response.resultCd == 200:
        print("\n\n\n질의한 내용: %s" % (response.uword))
        for action in response.action:
                query_response = action.mesg

        query_response = query_response.replace('<![CDATA[', '').replace(']]>', '')
        return query_response
    else:
        return None
```

3 아래의 코드를 작성하고 실행해봅시다. 실행 결과 코딩팩이 호출어에 반응하고, 질문에 대한 답변을 음성으로 출력해주는 것을 확인할 수 있습니다.

```
if __name__ == '__main__':
    if detect_wake_up_word():
        recognized_text = get_text_from_voice()
        result = query_by_text(recognized_text)
        print(result)
        speech(result)
```


07
코딩팩 프로젝트

우리는 코딩팩을 사용해 호출어 감지, 음성인식, 음성합성, 질의응답에 대해 배워보았습니다. 지금부터는 배운 내용을 바탕으로 다양한 프로젝트를 함께 만들어보도록 하겠습니다. 본 교재에서는 퀴즈게임 만들기, 타이머 만들기 이렇게 두 가지 프로젝트를 준비했습니다. 가장 중요한 내용은 데이터를 어떻게 처리하느냐입니다. 과연 어떻게 데이터를 다루고 있는지에 초점을 두고 유심히 관찰해봅시다. 물론 여기서 준비한 프로젝트뿐만 아니라 여러분의 아이디어로 다양한 프로젝트를 만들 수도 있습니다.

1) 퀴즈 게임 만들기

이번 프로젝트는 코딩팩의 음성 입력, 출력 기능을 활용하여 스무고개 게임을 만들어보도록 하겠습니다.

퀴즈 게임은 아래와 같은 방식으로 진행됩니다.
1. 게임은 스무고개 형식으로 진행됩니다. 시작 점수는 10점으로 오답을 말할 경우 -2점, 답을 말하지 않거나 다음 힌트를 요청할 경우 -1점씩 차감됩니다.
2. 게임이 시작되면, 코딩팩에서 힌트를 알려줍니다. (하나의 문제에는 여러 힌트가 있고, 이를 하나씩 알려줍니다.)
3. 정답으로 생각되는 단어를 말합니다.
4. 정답이면 “정답입니다.”를 출력하고, 현재 점수를 알려주면서 게임이 종료됩니다.
5. 오답이면 다음 힌트를 알려줍니다.
6. 모든 힌트를 들었음에도 문제를 맞히지 못하면, 정답과 점수를 알려주고 게임이 종료됩니다.

게임을 진행하기 위해 사전에 퀴즈 리스트를 만들어 두어야 합니다. 만드는 조건은 아래와 같습니다.
1. 한 문제당 힌트의 수는 상관없으나, 5개를 권장합니다.
2. 힌트는 모호할수록 앞으로, 특정하기 쉬울수록 뒤에 오도록 합니다.
3. 문제 수는 상관없습니다.

위 조건들을 지키면서 게임을 만들어 보도록 하겠습니다.

코딩하기

1 ai-makers-kit/example 폴더 안에 문제와 힌트를 모아둔 'quiz_list.py' 파일을 생성합니다. 문제와 힌트는 리스트와 딕셔너리를 이용해 아래와 같이 만들어줍니다.

quiz_list.py

```
quiz_lists = [
    {
        "answer" : "사과",
        "questions" : [
            "과일입니다.",
            "새콤달콤합니다.",
            "풋 과일로도 먹습니다.",
            "가을에 수확합니다.",
            "빨강색입니다."
        ]
    },
    {
        "answer" : "바나나",
        "questions" : [
            "과일입니다.",
            "나무에서 자랍니다.",
            "부드럽습니다.",
            "열대과일 입니다.",
            "노랑색 입니다."
        ]
    },
    {
        "answer" : "포도",
        "questions" : [
            "과일입니다.",
            "나무에서 자랍니다.",
            "열매가 동그랗습니다.",
            "와인을 만들기 위해서 사용합니다.",
            "보라색 입니다."
        ]
    }
]
```

2 'ex3_4_quiz.py' 파일을 생성합니다. 필요한 모듈을 불러오고, 퀴즈 목록 중 랜덤으로 한 가지 문제를 가져오는 함수 get_random_quiz()를 생성합니다. 앞서 만든 quiz_list를 불러오고, random 모듈의 randint 함수를 사용해 만든 문제 중 하나를 임의로 가져옵니다.

ex3_4_quiz.py

```
import quiz_list
import random
import voice
def get_random_quiz():
    quiz_lists = quiz_list.quiz_lists
    random_quiz_number = random.randint(0, len(quiz_lists)-1)
    quiz = quiz_lists[random_quiz_number]

    quiz_answer = quiz["answer"]
    quiz_hints = quiz["questions"]

    return quiz_answer, quiz_hints
```

3 스피커로 힌트를 출력하고, 마이크로 입력받은 값이 정답인지 오답인지를 판단하는 함수 solve_quiz()를 생성합니다.

```
def solve_quiz(quiz_answer, quiz_hint):
    score = 10

    for hint in quiz_hint:
        voice.speech("%d번 힌트 입니다." % (quiz_hint.index(hint) + 1))
        voice.speech(hint)
        input_text = voice.get_text_from_voice()
        if input_text.find(quiz_answer) != -1:
            voice.speech("정답입니다.")
            return score
        elif input_text == "" or input_text.find("다음") != -1:
            voice.speech("다음 힌트를 잘 들어보세요.")
            score -= 1
            continue
        else:
            voice.speech("땡 오답입니다.")
            score -= 2
            continue
    else:
        answer_text = "정답은 %s입니다." % quiz_answer
        print(answer_text)
        return score
```

부가 설명

```
        voice.speech("%d번 힌트 입니다."%(quiz_hint.index(hint)+1))
        voice.speech(hint)
```

voice 모듈의 speech() 함수를 사용하여 스피커로 힌트를 출력합니다.

```
        input_text = voice.get_text_from_voice()
```

voice 모듈의 get_text_from_voice() 함수를 사용하여 마이크로 입력되는 소리를 인식합니다.

```
        if input_text.find(quiz_answer) != -1 :
            voice.speech("정답입니다.")
            return score
        elif input_text == "" or input_text.find("다음") != -1 :
            voice.speech("다음 힌트를 잘 들어보세요.")
            score -= 1
            continue
```

```
        else :
            voice.speech("땡 오답입니다.")
            score -= 2
            continue
```

입력된 데이터가 정답인지 오답인지를 판단하고, 각 상황에 맞는 코드를 실행시킵니다.

```
    else :
        answer_text = "정답은 %s입니다."%quiz_answer
        return score
        print(answer_text)
```

만약 모든 힌트를 제공하였음에도 문제를 맞히지 못할 경우, 정답을 알려주고 프로그램을 종료시키는 코드를 추가합니다.

TIP

python에서 for문은 else문을 추가하여 사용할 수 있습니다. 입력받은 리스트의 모든 요소가 소진되어 for문이 종료될 경우 else문이 호출됩니다. 하지만 for문에서 break를 통해 종료되었을 경우에는 else문은 호출되지 않습니다.

4 메인 함수를 정의하고, 프로그램을 실행한 후 퀴즈를 풀어봅시다.

```
def main():
    voice.speech("지금부터 퀴즈를 시작하겠습니다. 힌트를 듣고 정답을 말해주세요")
    quiz_answer, quiz_hint = get_random_quiz()
    result_score = solve_quiz(quiz_answer, quiz_hint)
    voice.speech("점수는 %d점 입니다." % result_score)

if __name__ == "__main__":
    main()
```

전체 코드

ex3_5_quiz.py

```
import quiz_list
import random
import voice

def get_random_quiz():
    # import된 퀴즈를 사용합니다.
    quiz_lists = quiz_list.quiz_lists
    # 퀴즈의 개수를 사용하여 랜덤으로 숫자를 하나 선택해줍니다.
    random_quiz_number = random.randint(0, len(quiz_lists) - 1)
    # 랜덤숫자를 사용해 하나의 퀴즈를 가져옵니다.
    quiz = quiz_lists[random_quiz_number]
    # 퀴즈의 정답과 힌트를 가져와 리턴해줍니다.
    quiz_answer = quiz["answer"]
    quiz_hints = quiz["questions"]
    return quiz_answer, quiz_hints

# 선택된 문제를 풀어보는 함수 입니다.
def solve_quiz(quiz_answer, quiz_hint):
    score = 10 # 초기 점수를 10으로 설정합니다.
    # for문을 사용하여 힌트를 하나씩 가져옵니다.
    for hint in quiz_hint:
        # 몇번째 힌트인지 안내합니다.
        voice.speech("%d번 힌트 입니다." % (quiz_hint.index(hint) + 1))
        # 힌트를 음성출력합니다.
        voice.speech(hint)
        # STT를 통해 정답을 인식합니다.
        input_text = voice.get_text_from_voice()
        # 인식된 결과가 정답일 경우 정답이라고 말하고 점수를 리턴합니다.
        if input_text.find(quiz_answer) != -1:
            voice.speech("정답입니다.")
            return score
        # 인식된 결과가 없거나 다음을 인식하면 1점을 감점하고 다음힌트를 출력합니
다.
        elif input_text == "" or input_text.find("다음") != -1:
            voice.speech("다음 힌트입니다.")
            score -= 1
            continue
        # 인식된 답이 틀린 답이라면 2점을 감점합니다.
        else:
            voice.speech("땡 오답입니다.")
            score -= 2
            continue
    # 끝까지 정답을 인식하지 못한 경우에는 정답을 안내하고 점수를 리턴합니다.
    else:
        answer_text = "정답은 %s입니다." % quiz_answer
        print(answer_text)
        return score

def main():
    voice.speech("지금부터 퀴즈를 시작하겠습니다. 힌트를 듣고 정답을 말해주세요")
    quiz_answer, quiz_hint = get_random_quiz() #랜덤 퀴즈를 받아옵니다.
    # 랜덤 퀴즈를 입력해주고 퀴즈 결과 점수를 받아옵니다
    result_score = solve_quiz(quiz_answer, quiz_hint)
    # 점수를 음성출력해줍니다.
    voice.speech("점수는 %d점 입니다." % result_score)
if __name__ == "__main__":
    main()
```

quiz_list.py

```
quiz_lists = [
    {
        "answer" : "사과",
        "questions" : [
            "과일입니다.",
            "새콤달콤합니다.",
            "풋 과일로도 먹습니다.",
            "가을에 수확합니다.",
            "빨강색입니다."
        ]
    },
    {
        "answer" : "바나나",
        "questions" : [
            "과일입니다.",
            "나무에서 자랍니다.",
            "부드럽습니다.",
            "열대과일 입니다.",
            "노랑색 입니다."
        ]
    },
    {
        "answer" : "포도",
        "questions" : [
            "과일입니다.",
            "나무에서 자랍니다.",
            "열매가 동그랗습니다.",
            "와인을 만들기 위해서 사용합니다.",
            "보라색 입니다."
        ]
    }
]
```

2 157P
3 158P
3 158P
4 159P
1 156P

2) 타이머 만들기

타이머는 크게 시, 분, 초 단위로 이루어져 있습니다. 각각 입력을 받을 수도 있고 없는 부분은 0으로 채워주어야 합니다. 예를 들면 “5분 타이머”라는 말은 0시 5분 0초 타이머를 실행합니다. 타이머가 완료되면, 코딩팩은 결과를 알려주고 종료됩니다.

타이머 함수는 아래 기능들을 가지고 있습니다.
1. 입력받은 데이터에서 시, 분, 초를 각각 추출
2. 추출한 값으로 설정된 타이머 시작
3. 타이머가 완료되면 알려주기

타이머 함수를 만들 때 주의할 점이 있습니다.
1. 입력 음성 형태는 N 시간 N 분 N 초 타이머 형태
2. 각 자리는 자연수만 사용
3. 시간은 최대 24, 분 초는 최대 60까지 입력 가능

이를 토대로 코드를 작성해 보도록 하겠습니다.

</>

코딩하기

❶ 필요한 모듈을 불러오고, 메인 함수와 start_timer() 함수를 선언합니다.

```
import voice
import time

def start_timer(input_text) :
    if input_text.find("타이머") != -1 :
        print(input_text)

def main():
    input_text = voice.get_text_from_voice()
    start_timer(input_text)

if __name__ == "__main__" :
    main()
```

부가 설명

```
    input_text = voice.get_text_from_voice()
```

voice 모듈의 get_text_from_voice() 함수를 사용하여 마이크로 입력되는 소리를 인식합니다.

```
def main():
    input_text = voice.get_text_from_voice()
    start_timer(input_text)
```

인식된 음성에 “타이머”라는 단어가 포함되어 있으면, 인식된 음성을 출력합니다.

출력 결과

```
음성인식을 시작합니다.

종료하시려면 Ctrl+\ 키를 누루세요.

connect(2) call to /tmp/jack-1000/default/jack_0 failed (err=No such file
or directory)
attempt to connect to server failed
상태 코드=200 ¦ 인식 결과= 30분
상태 코드=200 ¦ 인식 결과= 30분 타이머
상태 코드=200 ¦ 인식 결과= 30분 타이머 추가
상태 코드=201 ¦ 인식 결과= 30분 타이머 추가

최종 인식 결과: 30분 타이머 추가

30분 타이머 추가
```

2 입력받은 데이터(문자)를 어절 단위로 나눠주는 코드를 추가합니다. split(“ ”) 함수를 사용해 띄어쓰기를 기준으로 단어들을 나눈 뒤 각 항목을 확인해 줍니다.

```
def start_timer(input_text) :
    if input_text.find("타이머") != -1 :
        for word in input_text.split(" "):
            print(word)
```

출력 결과

```
상태 코드=200 ¦ 인식 결과= 1시간
상태 코드=200 ¦ 인식 결과= 1시간 30분
상태 코드=200 ¦ 인식 결과= 1시간 30분 타이머
상태 코드=200 ¦ 인식 결과= 1시간 30분 타이머 추가
상태 코드=201 ¦ 인식 결과= 1시간 30분 타이머 추가

최종 인식 결과: 1시간 30분 타이머 추가

1시간
30분
타이머
추가
```

3 각 어절에서 “시간”, “분”, “초”라는 단어가 있는지를 확인하고, 각 항목마다 다른 변수에 따로 저장하는 코드를 추가합니다. 이때 숫자를 제외한 나머지 글자를 삭제하고, 남은 숫자를 정수형 데이터로 변환하여 저장합니다.

```
def start_timer(input_text):
    if input_text.find("타이머") != -1:
        for word in input_text.split(" "):
            if word.find("시간") != -1:
                hour = int(word.replace("시간", ""))
                print(hour)
            if word.find("분") != -1:
                minute = int(word.replace("분", ""))
                print(minute)
            if word.find("초") != -1:
                second = int(word.replace("초", ""))
                print(second)
```

출력 결과

```
상태 코드=200 ¦ 인식 결과= 1분
상태 코드=200 ¦ 인식 결과= 1분 30초
상태 코드=200 ¦ 인식 결과= 1분 30초 타이머
상태 코드=201 ¦ 인식 결과= 1분 30초 타이머

최종 인식 결과: 1분 30초 타이머

1
30
```

4 시, 분, 초를 관리하는 리스트 hms을 선언하고, 변수 hour, minute, second에 저장된 값을 차례대로 대입합니다.

```
def start_timer(input_text) :
    hms = [0, 0, 0]
    if input_text.find("타이머") != -1 :
        for word in input_text.split(" ") :
            if word.find("시간") != -1 :
                hour = int(word.replace("시간", ""))
                hms[0] = hour
            elif word.find("분") != -1 :
                minute = int(word.replace("분", ""))
                hms[1] = minute
            elif word.find("초") != -1 :
                second = int(word.replace("초", ""))
                hms[2] = second
    print(hms)
```

출력 결과

```
음성인식을 시작합니다.

종료하시려면 Ctrl+\ 키를 누루세요.

connect(2) call to /tmp/jack-1000/default/jack_0 failed (err=No such file
or directory)
attempt to connect to server failed
상태 코드=200 ¦ 인식 결과= 1시간
상태 코드=200 ¦ 인식 결과= 1시간 30분
상태 코드=200 ¦ 인식 결과= 1시간 30분 타이머
상태 코드=200 ¦ 인식 결과= 1시간 30분 타이머 추가
상태 코드=201 ¦ 인식 결과= 1시간 30분 타이머 추가

최종 인식 결과: 1시간 30분 타이머 추가

[1, 30, 0]
```

5 설정한 시간만큼 기다린 후 타이머가 종료되었음을 알리는 코드를 추가합니다.

```
import time

def start_timer(input_text) :
    hms = [0, 0, 0]
    if input_text.find("타이머") != -1 :
        for word in input_text.split(" ") :
            if word.find("시간") != -1 :
                hour = int(word.replace("시간", ""))
                hms[0] = hour
            elif word.find("분") != -1 :
                minute = int(word.replace("분", ""))
                hms[1] = minute
            elif word.find("초") != -1 :
                second = int(word.replace("초", ""))
                hms[2] = second
        print(hms)
        timer_sec = hms[0] * 3600 + hms[1] * 60 + hms[2]
        print(timer_sec)
        time.sleep(timer_sec)
        print("timer end")
        voice.speech("땡땡땡 타이머가 종료되었습니다.")
```

부가 설명

```
        timer_sec = hms[0] * 3600 + hms[1] * 60 + hms[2]
        print(timer_sec)
        time.sleep(timer_sec)
```

리스트에 저장된 시와 분을 초로 바꿔 변수 timer_sec에 대입하고, 변수 timer_sec을 time.sleep() 함수의 인자로 사용하여 저장된 시간만큼 기다립니다.

```
        voice.speech("땡땡땡 타이머가 종료되었습니다.")
```

voice모듈의 speech함수를 불러와 작성된 문장을 음성으로 출력합니다.

출력 결과

```
상태 코드=200 ¦ 인식 결과= 일본식
상태 코드=200 ¦ 인식 결과= 1분 15초
상태 코드=200 ¦ 인식 결과= 1분 15초 타이머
상태 코드=201 ¦ 인식 결과= 1분 15초 타이머

최종 인식 결과: 1분 15초 타이머

[0, 1, 15]
75
timer end
```

6 설정된 시간을 알려주는 코드와 타이머가 제대로 설정되지 않았을 경우를 알려주는 코드를 추가합니다.

```
def start_timer(input_text) :
    hms = [0, 0, 0]
    if input_text.find("타이머") != -1 :
        for word in input_text.split(" ") :
            if word.find("시간") != -1 :
                hour = int(word.replace("시간", ""))
                hms[0] = hour
            elif word.find("분") != -1 :
                minute = int(word.replace("분", ""))
                hms[1] = minute
            elif word.find("초") != -1 :
                second = int(word.replace("초", ""))
                hms[2] = second
        print(hms)
        voice.speech("타이머가 %d시간 %d분 %d초로 설정됩니다."%(hms[0],
hms[1], hms[2]))
        timer_sec = hms[0] * 3600 + hms[1] * 60 + hms[2]
        print(timer_sec)
        time.sleep(timer_sec)
        print("timer end")
        voice.speech("땡땡땡 타이머가 종료되었습니다.")
    else :
        voice.speech("타이머가 설정되지 않았습니다")
```

출력 결과

```
상태 코드=200 ¦ 인식 결과= 60초
상태 코드=201 ¦ 인식 결과= 60초 타이머

최종 인식 결과: 60초 타이머

[0, 0, 60]

음성합성 응답 상태코드: 200

음성합성 응답 상태코드: 0
connect(2) call to /tmp/jack-1000/default/jack_0 failed (err=No such file
or directory)
attempt to connect to server failed
음성이 출력되었습니다.
60
timer end

음성합성 응답 상태코드: 200

음성합성 응답 상태코드: 0
connect(2) call to /tmp/jack-1000/default/jack_0 failed (err=No such file
or directory)
attempt to connect to server failed
음성이 출력되었습니다.
```

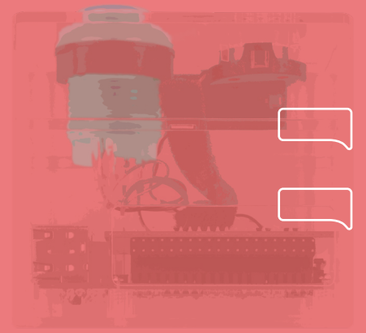
</> 전체 코드

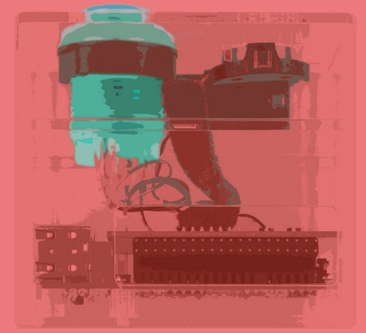
1 163P
4 165P
2 164P
3 4 164P-165P
5 6 166P-168P
1 163P

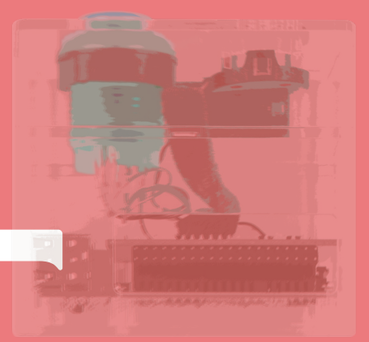
```
import voice
import time

# 입력된 텍스트에서 시간을 찾아 타이머를 동작합니다.
def start_timer(input_text) :
    hms = [0, 0, 0] # 시, 분, 초를 저장하는 리스트 입니다.
    if input_text.find("타이머") != -1 : #텍스트에서 타이머를 찾습니다.
        #띄워쓰기 단위로 문장을 단어로 분리하여 word에 넣어줍니다.
        for word in input_text.split(" ") :
            # 단어에 시, 분, 초가 있다면
            # 단어 앞의 숫자를 분리해 리스트에 저장합니다.
            if word.find("시간") != -1 :
                hour = int(word.replace("시간", ""))
                hms[0] = hour
            elif word.find("분") != -1 :
                minute = int(word.replace("분", ""))
                hms[1] = minute
            elif word.find("초") != -1 :
                second = int(word.replace("초", ""))
                hms[2] = second
        print(hms)
        #설정할 시간을 음성으로 출력합니다.
        voice.speech("타이머가 %d시간 %d분 %d초로 설정됩니다."%(hms[0], hms[1],
hms[2]))
        # 시, 분을 초로 바꾸어서 초에 더해줍니다.
        timer_sec = hms[0] * 3600 + hms[1] * 60 + hms[2]
        print(timer_sec)
        # 설정된 초만큼 프로그램을 일시정지합니다.
        time.sleep(timer_sec)
        print("timer end")
        # 타이머의 카운트가 끝나면 음성 출력합니다.
        voice.speech("땡땡땡 타이머가 종료되었습니다.")
    else :
        voice.speech("타이머가 설정되지 않았습니다")

def main():
    voice.speech("타이머를 시작합니다. 설정할 시간을 말해주세요")
    input_text = voice.get_text_from_voice()
    #타이머를 실행합니다.
    start_timer(input_text)

if __name__ == "__main__" :
    main()
```

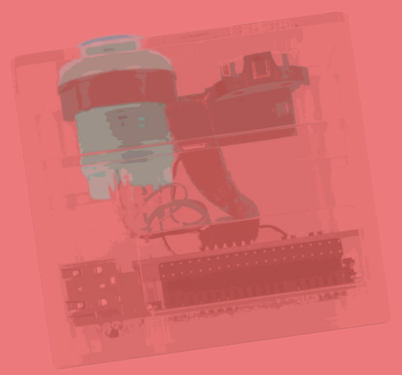
코딩팩에서
웹 데이터 이용하기

4









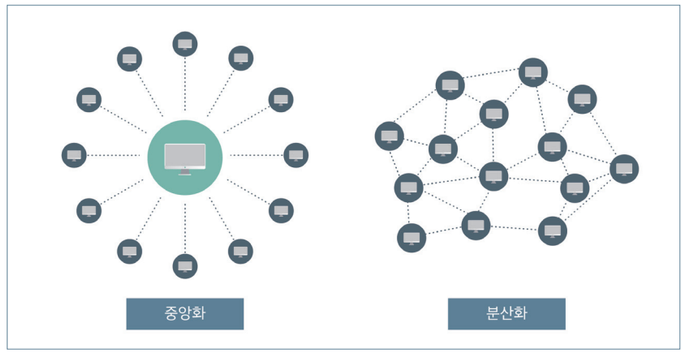
01
웹에 대해 알아보기

하루에 인터넷을 한 번이라도 사용하지 않는 사람이 과연 몇 명이나 될까요? 우리는 컴퓨터와 핸드폰을 통해 너무나 쉽게 인터넷을 사용할 수 있습니다. 또한, 인터넷을 통해 장소와 거리의 구애 없이 서로 채팅을 할 수 있고, 얼굴을 마주 보며 통화를 할 수도 있습니다.

인터넷 검색창에 '인공지능'을 검색하면, 한국어로 된 정보뿐만 아니라 영어, 중국어, 일어 등 전 세계 사람들이 올려놓은 정보를 확인할 수 있습니다. 그렇다면 어떻게 전 세계 사람들이 올려놓은 정보를 우리가 확인할 수 있는 것일까요? 그리고 이렇게 방대한 정보를 가지고 있는 인터넷을 누가 관리하는 것일까요? 본격적으로 웹에 대해 알아보기 전에 웹의 기반이 되는 인터넷에 대해 알아본 후 웹을 다루어 보도록 하겠습니다.

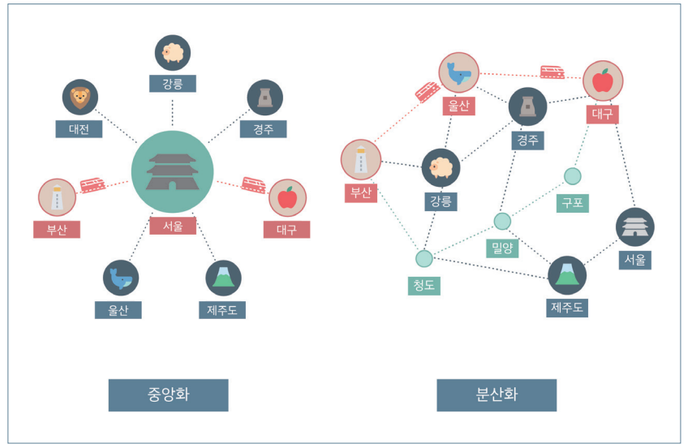
1) 인터넷이란?

인터넷은 이름에서 의미를 찾을 수 있습니다. Internet 은 Inter-Network의 약어로 "여러 개의 네트워크를 묶다." 즉, "여러 통신망을 하나로 연결한다."라는 의미를 가지고 있습니다. 그렇다면 전 세계 컴퓨터들을 어떻게 연결할 수 있을까요? 아래의 그림을 보며 설명을 이어 가보겠습니다.



위 그림에서 왼쪽은 한 컴퓨터에 주변 모든 컴퓨터가 연결되어 있고, 오른쪽은 컴퓨터와 컴퓨터가 서로 연결이 되어있습니다. 이들은 각각 중앙화, 분산화 통신 방식이라 부르고 우리가 사용하고 있는 인터넷은 분산화 통신 방식을 사용하고 있습니다. 즉, 인터넷은 네트워크끼리 서로 연결되어 방대한 네트워크를 이루고 있습니다. 이해를 돕기 위해 예를 들어보겠습니다.

대구에서 부산으로 기차를 타고 여행을 떠나기 위해서는 대구와 부산을 이어주는 기찻길이 연결되어 있어야 합니다. 중앙화, 분산화 통신 방식을 이 기찻길에 비유하여 예를 들어보겠습니다. 중앙화 방식을 이용해 기찻길을 만들면 모든 기차가 모이는 중앙센터를 거쳐 목적지로 갈 수 있습니다. 만약 중앙센터가 서울에 있다면, 대구에서 부산으로 가기 위해 대구->서울행 기차를 타고 서울 중앙센터로 간 다음 다시 서울->부산행 기차로 갈아타야 합니다. 물론 다시 대구로 돌아올 때도 마찬가지입니다. 대구에서 부산을 가는데 서울을 거쳐 가야 하니 시간도 많이 들고 정말 비효율적입니다. 그리고 만약 서울 중앙센터에 사고가 난다면 모든 기차 노선이 마비될 것입니다. 분산화 방식은 현재 기차 노선과 비슷합니다. 대구에서 부산으로 가려면 청도, 밀양, 구포를 거쳐 바로 부산으로 가면 됩니다. 중앙화 방식 보다 시간도 단축되고, 만약 청도역에 사고가 났다고 하더라도 울산을 거쳐 부산으로 갈 수 있습니다. 이때 각 도시를 네트워크 또는 컴퓨터라고 생각할 수 있고, 기찻길로 모든 도시가 서로 이러진 모습을 현재 인터넷이라 생각하면 조금 쉽게 중앙화 통신 방식과 분산화 통신 방식을 이해할 수 있습니다. 사실, 인터넷은 중앙화 통신을 사용할 수 있었지만 분산화 통신을 사용하고 있는 이유에는 사연이 있습니다.

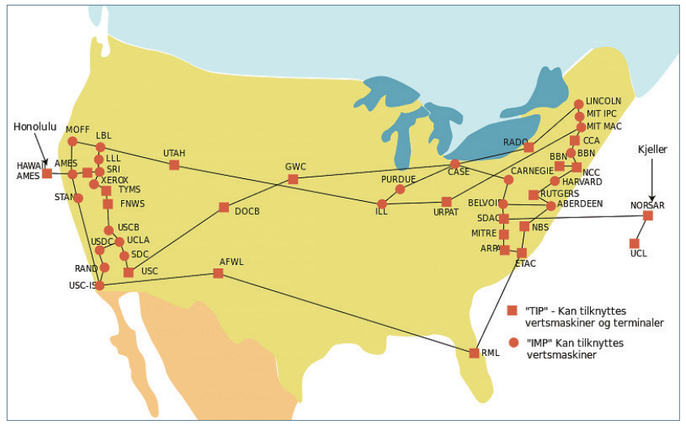


1 인터넷의 역사

인터넷이 만들어진 과정을 이해하기 위해 1960년대의 시대 배경을 알아보도록 하겠습니다. 1960년대에는 미국과 소련은 적대적 관계였습니다. 하지만 서로 무기를 들고 싸우지는 않았고, 대신 간접적으로 여러 부문에서 경쟁하며 갈등과 긴장이 오고 갔던 냉전 상태였습니다.

1960년대에 미국 국방성은 소련과의 핵전쟁 등 극한의 상황이 닥치더라도 각 지역의 컴퓨터들과 통신할 수 있는 방법을 찾고 있었습니다. 기존의 통신 방식은 마치 옛날 전화국과 흡사합니다. 직원이 전화를 거는 사람과 받는 사람을 선으로 이어줬던 것처럼 중앙에서 회선을 교환해주는 통신 방식을 지향했습니다. 하지만 이런 방식은 중앙 컴퓨터에 문제가 생기면 네트워크에 연결된 모든 컴퓨터가 통신 할 수 없다는 문제가 있었습니다.

그러던 와중에 RAND의 엔지니어인 '폴 바란'은 미 공군의 핵 공격에서 살아남을 수 있는 통신 시스템을 개발해달라는 요구에 '분산적 통신에 관해'라는 책자에서 기존의 방식과는 다른 통신 방식을 제안했습니다. 기존의 중앙화 방식과는 다르게, 컴퓨터와 컴퓨터를 서로 연결하여 분산적인 네트워크를 구축하자는 방안을 내놓았습니다. 폴이 제안한 분산화 통신 방식은 연결된 컴퓨터들의 40%나 파괴되어도 서로 통신할 수 있는 엄청난 생존 가능성을 자랑했습니다.



1980년대 ARPA Net지도
사진출처<>

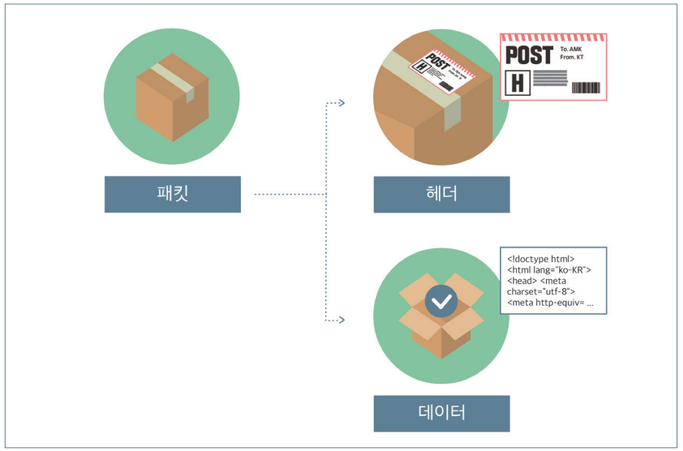
폴은 분산화 통신 방식과 함께 패킷 교환 방식을 제안하였습니다. 패킷 교환 방식은 전송할 데이터를 잘게 쪼개어 패킷으로 만든 다음 패킷에 목적지의 주소를 입력해 전송하는 방식으로, 1대1 통신을 하는 기존 회선 교환 방식은 길목이 하나인 것에 비해, 패킷 교환 방식은 패킷이 여러 컴퓨터를 거쳐 목적지로 도착하는 방식이기 때문에 여러 길목이 존재합니다.

이런 '생존 가능성'이 높은 통신 방식을 고수하던 고등 연구 계획국은 폴 바란의 분산화 통신과 패킷 교환 방식을 채택하기로 결정하였고, 1969년 로버트 테일러를 중심으로 한 고등 연구 계획국의 정보처리기술실(IPTO)에서 패킷 교환 방식의 ARPA Net을 만들었고, 학술 연구를 목적으로 여러 대학교의 컴퓨터가 연결되면서부터 규모가 점점 커지기 시작했습니다. 1984년 국방성이 ARPA Net을 군사용과 민간용으로 분리하는 작업을 마친 후 ARPA Net을 민간에 공개를 하게 됩니다. ARPA Net에 민간의 컴퓨터들이 연결되면서 오늘날의 인터넷이 만들어지게 되었습니다.

2 패킷과 TCP/IP

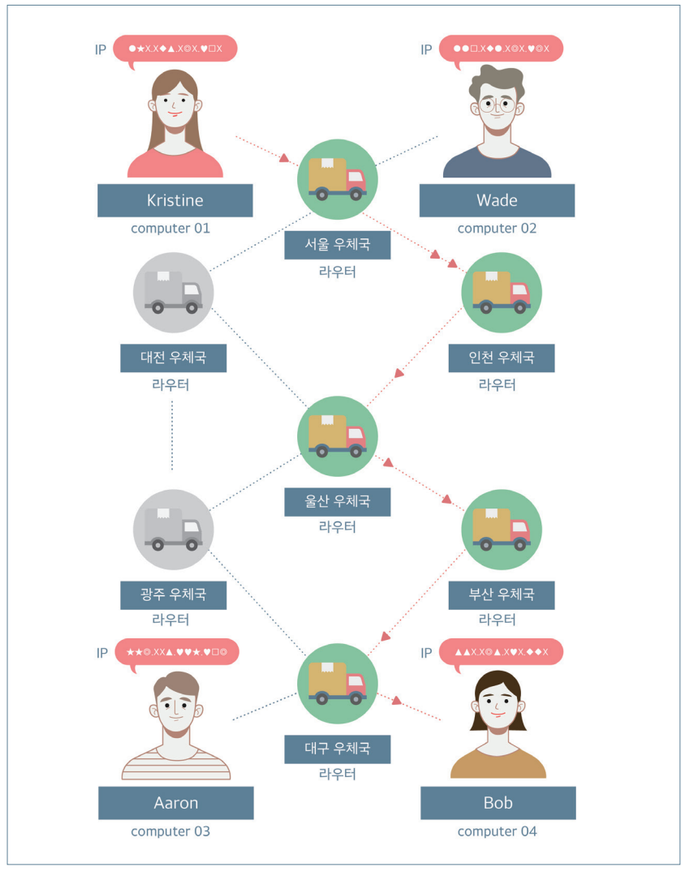
앞서 얘기했듯이 데이터를 쪼갠 일부를 가지는 **패킷**packet 은 통신에서 사용하는 가장 작은 단위입니다. 오늘날 인터넷에는 수없이 많은 양의 패킷이 오고 가고 있습니다. 여러분들이 인터넷으로 하는 모든 행위는 패킷으로 구성되어 전달하고, 전달받고 있습니다.

패킷은 마치 택배와 같습니다. 택배의 겉에는 보내는 사람, 보내는 주소, 받는 사람, 받는 주소 등 정보가 적혀 있고, 택배의 내용물은 상대방에게 보내는 물건이 들어있습니다. 패킷도 택배처럼 두 가지로 나뉩니다. 택배의 겉에 해당하는 **헤더**, 택배의 내용물을 **데이터**라고 부릅니다.



택배를 보낼 때 받는 사람의 주소를 적어줘야 하는 것처럼 패킷에도 받는 컴퓨터의 주소를 적어주어야 합니다. 컴퓨터의 주소는 그 유명한 **IP** Internet Protocol 주소를 사용합니다. IP주소는 네트워크에서 장치를 구분할 수 있도록 도와주는 특수한 번호입니다.

패킷을 전송하면 여러 컴퓨터를 거쳐서 목적지에 도착 합니다. 각 컴퓨터에서는 목적지까지의 적절한 경로를 판단하여 다음 컴퓨터에 전달하고 다음 컴퓨터도 동일한 과정을 거쳐 목적지까지 도착하게 됩니다. 이 과정을 **라우팅**이라 하고, 패킷이 가장 빠른 길로 지나갈 수 있도록 길을 찾아주고 패킷을 다음 컴퓨터로 넘겨주는 특수한 컴퓨터를 **라우터**라고 합니다.



우리가 물건을 주문하면, 배송 도중에 물건이 손상되거나, 분실이 되는 경우가 발생할 수 있습니다. 이럴 경우 우리는 구매처로부터 교환 신청을 해서 새로운 물건을 받습니다. 라우팅 할 때도 마찬가지입니다. 여러 라우터와 전송 매체를 거쳐 목적지에 도착했을 때, 패킷이 손상되거나 분실되는 경우가 발생할 수 있습니다. 당연히 사용할 수 없는 패킷이기 때문에 교환 신청을 해야 합니다. 그렇다면 여기서 누구한테 어떻게 교환 신청을 해야 할까요? 그리고 이 패킷이 쪼갠 데이터 중 어느 부분인지 어떻게 알 수 있을까요? 그리고 패킷이 라우터를 거쳐 목적지에 도착했다면, 이 패킷들이 과연 출발한 순서대로 들어왔다는 보장을 할 수 있을까요? 당연히 보장할 수 없습니다. 가장 먼저 출발했지만 더 많은 라우터를 거쳐 목적지에 도착했을 수도 있기 때문입니다. 이런 문제를 해결하기 위해 인터넷은 **TCP** Transmission Control Protocol 라는 프로토콜을 사용합니다. TCP 프로토콜은 패킷에 일련번호를 부여하여 패킷의 순서를 알아낼 수 있게 도와줍니다. 그리고 받아오는 패킷을 검증하여 손상되었는지, 분실되었는지 확인하여 직접 재전송을 요청합니다. 인터넷은 이전 TCP/IP의 도움으로 안정적이고 정확하게 데이터를 주고 받을 수 있게 됩니다.

2) 월드 와이드 웹(WWW)의 등장

1 웹에 대하여 알아보기

1960년대에 인터넷이 등장하면서 다양한 방법을 통해 컴퓨터를 통한 통신을 하기 시작했습니다. 인터넷을 사용하면 먼 거리에 있는 사람과 편지를 주고받을 수 있고, 동영상 및 사진 등을 서로 공유 할 수 있습니다. 초기의 인터넷은 지금과 전혀 다른 검은 창에 명령어를 입력하는 방식을 사용했습니다. 그리고 운영체제와 프로그램에 따라서 다른 명령어를 사용했기 때문에 인터넷을 통해 정보를 주고받을 수는 있었지만 높은 진입장벽이 있었습니다. 1989년에 유럽의 물리연구소의 '팀 버너스-리'는 이와 같은 불편함을 해결하기 위해 WWW(월드 와이드 웹)이라는 새로운 방법을 개발하였습니다.

월드 와이드 웹은 하이퍼텍스트 형식으로 표현된 인터넷상의 정보를 검색할 때 사용하는 정보검색 시스템으로, 음성, 문자, 그림, 동영상 등 다양한 정보를 효과적으로 검색할 수 있습니다. 월드 와이드 웹은 아래와 같이 세 가지 기능으로 요약할 수 있습니다.

1. 통일된 위치 지정 방법 URL
2. 웹과 정보를 주고받을 때 사용하는 HTTP 프로토콜
3. 웹페이지가 어떻게 구성되는지를 정의하는 HTML

위에서 보여주는 월드 와이드 웹의 세 가지 기능은 우리가 앞으로 함께 해볼 웹으로부터 정보를 받기 위해서 꼭 사용되는 기능입니다. 그렇기 때문에 기능이 어떠한 역할을 하고, 또 어떻게 사용하는지에 대해 알아보도록 하겠습니다.

2 웹에서 일어나는 일

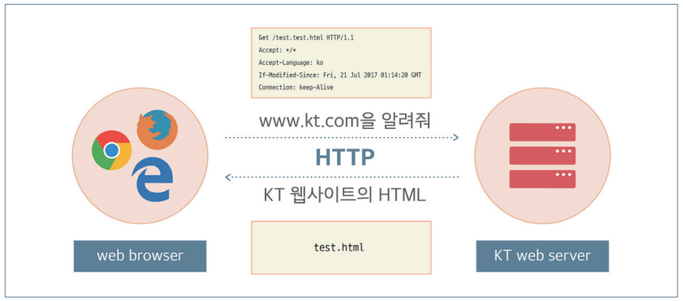
우리가 웹 서핑을 할 때, 웹 브라우저의 주소창에 URL을 입력하면 원하는 웹 사이트가 화면에 보이게 됩니다. 이렇게 간단해 보이는 과정 속에는 사실 많은 일이 일어납니다. 지금부터 웹 페이지가 우리에게 보이기까지의 과정에 대해 알아보도록 하겠습니다.

웹 브라우저의 주소창에 'www.google.com'을 입력하고 엔터를 누르는 순간 컴퓨터는 수백 킬로미터 이상 떨어져 있는 구글의 서버와 통신을 합니다. 그리고 1초도 걸리지 않아 구글 홈페이지를 보여줍니다.

이 과정은 아래의 단계를 거쳐 일어납니다.

1. 웹 브라우저의 주소창에 URL을 입력하고 엔터를 누른다.
2. 웹 브라우저는 웹 서버에게 웹 페이지를 요청한다.
3. 웹 브라우저의 요청으로 웹 서버가 돌려주는 웹 페이지를 받아온다.
4. 웹 브라우저는 받아온 웹 페이지를 모니터 혹은 화면 출력장치를 통해 출력한다.

이렇게 매우 많은 일이 일어나지만, 위 네 단계는 사람이 인식하지 못할 정도로 순식간에 일어납니다. 웹 브라우저와 웹 서버는 서로 데이터를 주고받을 때 **HTTP**HyperText Transfer Protocol이라는 통신 방법을 사용하여 통신합니다. 어디서 많이 보지 않았나요? 우리가 URL을 주소창에 입력할 때 www 앞에 http://가 자동으로 붙어지는 것을 한 번쯤은 보았을 것입니다. HTTP를 사용하여 다른 컴퓨터에 정보를 요청할 때는 HTTP GET이라는 요청 방법을 사용합니다. 내용에 대해서는 뒤에서 자세하게 다루도록 하겠습니다.



다른 컴퓨터 혹은 서버로부터 정보를 요청할 때는 원하는 정보의 위치와 종류를 정확하게 알려 주어야 합니다. 원하는 정보가 무엇인지를 알려주지도 않고 그저 달라고만 하는 것은 당연히 불가능한 일입니다. 이때 사용되는 것이 URL입니다. URL은 너무나도 고맙게 일련 규칙을 가지고 있고, URL에는 네트워크상에 퍼져있는 특정 정보의 종류와 위치가 포함되어 있습니다. 웹 서버에게 HTTP GET 요청을 할 때, 다른 정보와 URL을 함께 웹 서버로 보내 원하는 정보의 위치와 종류를 웹 서버에게 알려줍니다. 일반적인 URL의 형태는 아래와 같습니다. 아마 우리가 많이 보기도 했고, 사용하기도 했을 것입니다.

다른 컴퓨터 혹은 서버로부터 정보를 요청할 때는 원하는 정보의 위치와 종류를 정확하게 알려 주어야 합니다. 원하는 정보가 무엇인지를 알려주지도 않고 그저 달라고만 하는 것은 당연히 불가능한 일입니다. 이때 사용되는 것이 URL입니다. URL은 일련 규칙을 가지고 있고, URL에는 네트워크상에 퍼져있는 특정 정보의 종류와 위치가 포함되어 있습니다. 웹 서버에게 HTTP GET 요청을 할 때, 원하는 정보의 위치와 종류가 담긴 URL을 기반으로 웹서버에게 알려줍니다.

HTTP GET 요청을 받은 웹 서버는 웹 브라우저로부터 요청받은 정보 즉, 웹 페이지를 다시 웹 브라우저에 돌려줍니다. 그렇다면 웹 서버로부터 돌려받은 웹 페이지는 어떻게 구성되어 있을까? 대부분의 웹 페이지는 **HTML**Hyper Text Markup Language과 **CSS**Cascading Style Sheets 로 구성됩니다. HTML은 월드 와이드 웹을 통해 볼 수 있는 문서를 만들 때 사용하는 마크업 언어로 홈페이지의 뼈대를 구축하고 있습니다. HTML로 작성된 문서를 HTML 문서라고 하며 웹 서버로부터 받은 HTML 문서를 웹 브라우저가 해석하여 우리에게 보여주게 되는 것입니다. CSS는 HTML을 화려하게 꾸며주는 역할을 맡고 있습니다. 웹 페이지를 유심히 관찰해보면 다양한 색이 들어 있고, 다른 크기의 글자들이 나열되어 있습니다. 이처럼 CSS는 글꼴이나, 배경색, 너비, 높이, 위치 등을 지정합니다.

지금까지 웹 브라우저와 웹 서버가 어떻게 정보를 교환하는지 그리고 그 정보가 어떻게 이루어져 있는지에 대해 간략하게 알아보았습니다. 뒤에서는 각 내용에 대해 세세하게 알아보는 시간을 가지도록 하겠습니다.

❸ HTTP를 사용하여 데이터를 요청하는 과정

구글에 '맛집'을 검색하면, 구글은 우리에게 맛집 정보를 알려줍니다. 검색창에 궁금한 내용을 입력하면 바로 답변을 주기 때문에 그 과정이 어떻게 진행되는지에 관해서는 관심을 가지지 않았습니다. 지금부터는 웹 서버로 데이터를 어떻게 요청하고, 전달받는지에 대해 알아보도록 하겠습니다.

앞에서 웹 서버로 데이터를 요청할 때 HTTP GET이라는 요청 방법을 사용한다고 말했습니다. 그리고 데이터의 위치와 종류를 알려주기 위해 URL을 포함시킨다고 했습니다. URL은 크게 두 가지 정보를 담고 있습니다.

(1) 누구에게 어떻게 요청을 할 것인가?

URL에는 필요한 데이터를 누구에게 요청해야 할지에 대한 정보를 포함하고 있습니다. 이 정보는 보통 IP 주소로 나타내지만 사람들이 보고 힘들다는 단점을 가지고 있습니다. 그래서 사람이 보기 편하게 IP 주소를 'www.google.com'과 같은 **도메인**으로 변환하여 사용합니다. 그런데 이 도메인은 사람들이 보기에는 좋지만, 컴퓨터는 이해를 할 수 없는 방식입니다. 그래서 DNSDomain Name System 를 사용하여 도메인을 IP 주소로 변환하여 전달합니다.

(2) 어디에 있는 무엇을 요청할 것인가?

URL에는 필요한 데이터가 무엇인지 그리고 그 데이터가 어디에 있는지에 대한 정보를 담고 있습니다. 아래와 같이 URL에 구글에게 물어볼 내용의 위치와 종류를 담아 구글 웹 서버에게 정보를 요청해볼까 합니다.

https://www.google.com/search?q=맛집

위 URL을 웹 브라우저 검색창에 입력한 후 검색해보면, 구글이 맛집 정보를 알려주는 것을 확인할 수 있습니다. URL의 마지막을 보면, 물음표(?)와 함께 "q=맛집"이 적혀 있습니다. 무슨 내용인지 모르긴 몰라도 우리에게 필요한 정보인 맛집을 구글 서버에게 요청한다는 것을 알 수 있습니다. 이처럼 HTTP GET 요청을 보낼 때 URL에 누구에게, 어디에 있는, 어떤 정보를 요청할 것인지를 담아 웹 서버로 보내고, HTTP GET 요청을 받은 웹 서버는 URL에 포함된 인자를 추출하여 분석합니다. 그 결과 우리에게 필요한 정보를 담은 웹 페이지를 돌려주게 됩니다.

3) HTML에 대해서 알아보기

앞에서 HTML은 홈페이지 뼈대를 구축한다고 이야기했습니다. 그렇다면 어떻게 뼈대를 구축하는지에 대해 알아보겠습니다. HTML을 쉽게 설명해보면, 글자, 사진, 동영상 등 다양한 콘텐츠들을 어떻게 보여줄 것인지를 구조화하는 언어입니다. 그리고 웹 브라우저는 이 언어를 해석하여 우리가 볼 수 있는 웹 페이지로 출력해줍니다. HTML을 직접 작성해보고 결과를 확인하면서 HTML에 대해 알아보도록 하겠습니다.

1 HTML의 기본구조

HTML의 구조를 알아보도록 하겠습니다. HTML은 프로그래밍 언어가 아닌 마크업 언어입니다. 마크업 언어는 콘텐츠의 구조를 정의하는 언어로, 콘텐츠를 구조화하고 구분하기 위해 아래와 같이 콘텐츠를 태그로 감쌉니다.

여는 태그부터 닫는 태그까지의 모든 부분을 통틀어 **요소**Element라고 표현합니다. 요소는 여는 태그와 닫는 태그, 그리고 콘텐츠로 이루어집니다. 여는 태그는 요소의 시작을 알리고, 닫는 태그는 요소의 끝을 알립니다. 간혹 초보자들이 닫는 태그를 누락하는 경우가 있는데, 잘못된 구성으로 결과가 이상하게 나타날 수 있으니 유의하시길 바랍니다. 콘텐츠는 요소의 내용인데, 위 예제에서는 문자이지만, 사진, 동영상 등도 콘텐츠에 추가할 수 있습니다. 요소에는 속성이라는 값을 가지는데, 아래와 같이 사용할 수 있습니다.

속성에는 요소의 추가 정보를 포함합니다. 예를 들어 요소를 꾸미는 스타일이나 요소를 구별할 수 있는 식별자 등을 지정할 수 있습니다. 속성은 속성 이름=" 속성값" 형식을 사용합니다.

가장 기본적인 HTML을 살펴보면서 웹 페이지를 구성하는 태그를 살펴봅시다.

```
<!DOCTYPE html>
<html>
<head>
    <title>Example</title>
    <meta charset="utf-8"/>
</head>
<body>
<p style="color: red;">Hello, HTML!</p>
<p>My Name is Hong-gil-dong</p>
</body>
</html>
```

(1) <!DOCTYPE html>

HTML의 버전 중에서 HTML5를 사용하겠다고 선언해주는 코드입니다. </ >태그를 사용하지 않아도 됩니다.

(2) <html></html>

HTML의 문서의 처음과 끝을 나타내는 태그입니다. 웹 페이지 모든 콘텐츠는 html 태그 안에 작성되어야 합니다.

(3) <head></head>

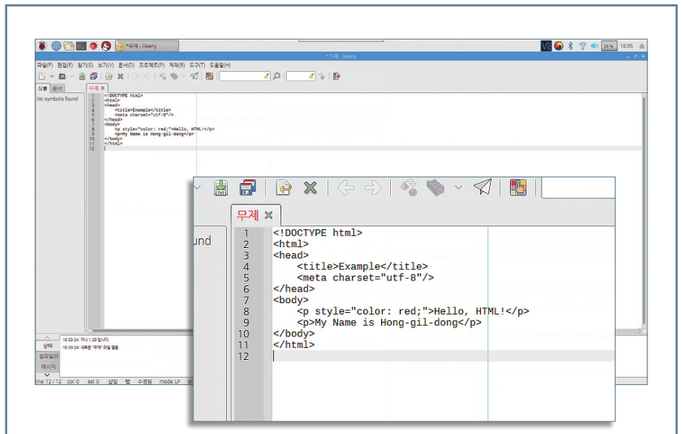
웹 페이지에 출력이 되는 부분은 아니지만, 사용자들에게 보이는 콘텐츠 이외의 웹 페이지 정보가 작성되는 태그입니다. 예를 들어 웹 페이지의 제목, 인코딩 방법 등이 작성됩니다.

(4) <body></body>

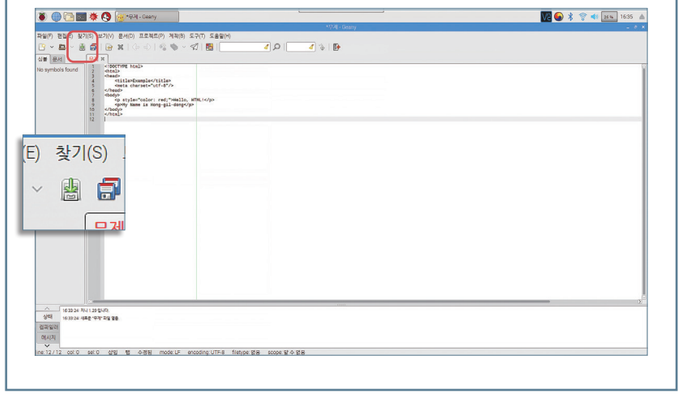
우리가 보는 웹 페이지의 화면 데이터가 있는 부분입니다. body안에 있는 부분에 문자와 같은 콘텐츠를 입력하면 화면으로 출력됩니다.

4) 라즈베리파이를 사용해 HTML코드를 작성해보기

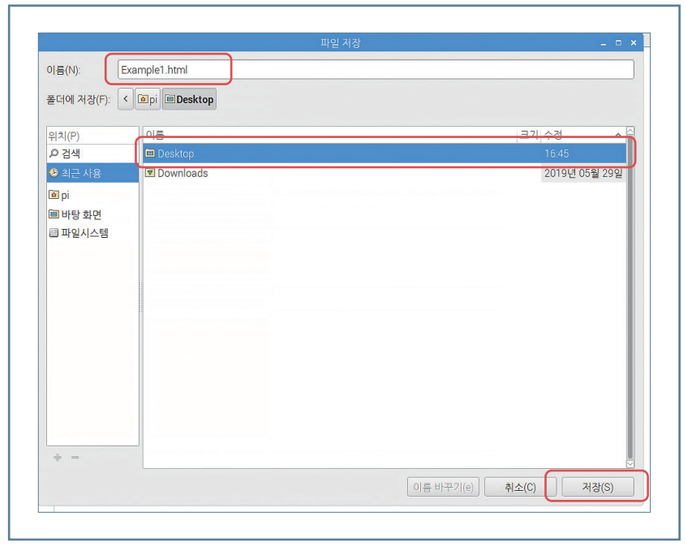
라즈베리파이의 지니에디터를 사용해 HTML 코드를 작성해보도록 하겠습니다. 위에서 알아보았던 HTML 코드를 아래 사진과 같이 입력합니다.



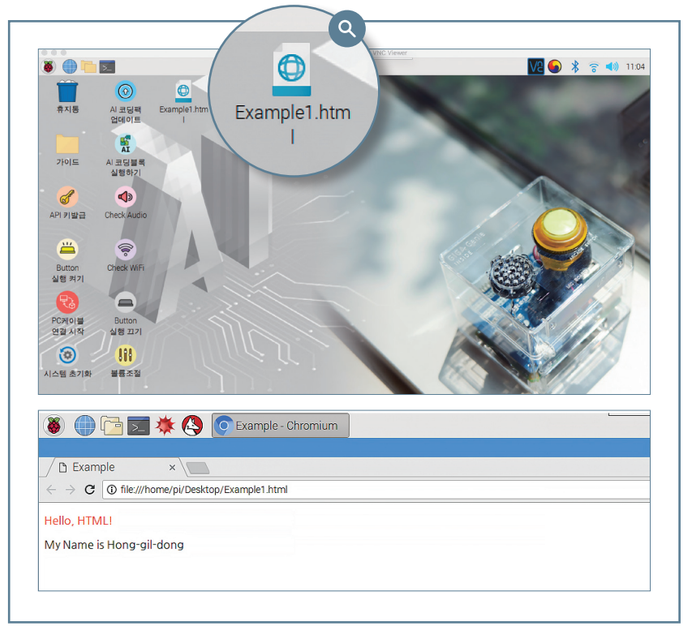
타이핑이 완료되면, 저장 아이콘 또는 단축키(ctrl+s)를 눌러 저장을 해줍니다.



저장 경로는 Desktop으로 지정하고, Example1.html로 저장합니다. 이때, 확장자로는 html을 사용합니다.



파일이 저장되면, 바탕화면의 Example1.html 파일을 실행하여 결과를 확인합니다.



축하합니다, 성공적으로 여러분의 첫 번째 웹 페이지를 만들어보았습니다.

5) 자주 사용하는 HTML태그에 대해 알아보기

1.
 태그
닫는 태그인 </ >태그를 사용하지 않으며, 문장 마지막에 넣으면 강제로 줄 바꿈이 되어 출력됩니다.

2. <h1></h1>, <h2></h2>, … <h6></h6> 태그
제목 태그입니다. 숫자에 따라서 다른 크기의 텍스트를 출력해주고 자동 줄 바꿈을 해줍니다.

3. <img src=”이미지의 위치”> 태그
이미지를 가져오는 태그입니다. src에 인터넷의 이미지 위치를 넣어주거나 컴퓨터 내부에 있는 이미지의 경로를 넣어 이미지를 삽입 할 수 있습니다.

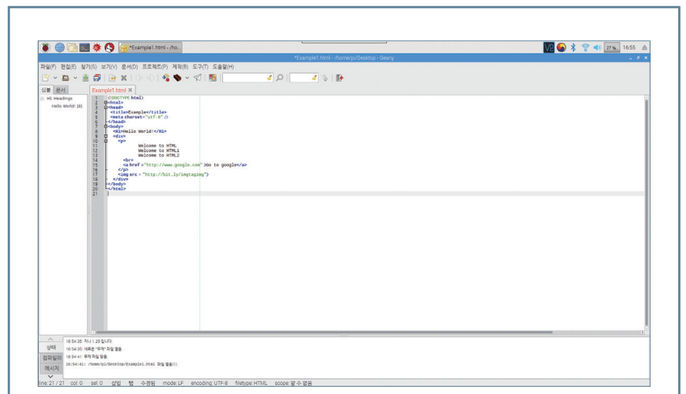
4. <a> 태그
HTML에서 가장 강력한 기능인 하이퍼 링크를 연결할 때 사용하는 태그입니다.
와 같이 사용합니다.

5. 태그
웹 사이트의 구역을 나누는데 사용됩니다. 예를 들어 메뉴 부분, 본문 부분, 마지막 소개 부분과 같이 페이지의 구역을 나누는데 사용됩니다.

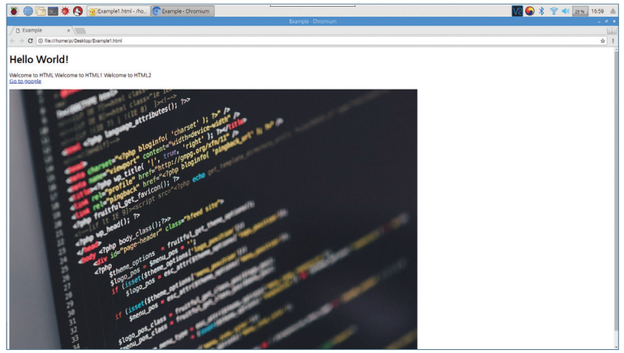
이 위의 태그 이외에도 HTML에는 굉장히 많은 태그들이 있습니다.
이와 같은 태그들은 ‘https://www.w3schools.com/tags/default.asp’의 사이트에서 확인할 수 있습니다.

설명했던 태그들을 사용하여 웹 페이지를 만들어보도록 하겠습니다. 지니 에디터를 사용하여 Example1.html의 HTML 코드를 아래와 같이 수정합니다.

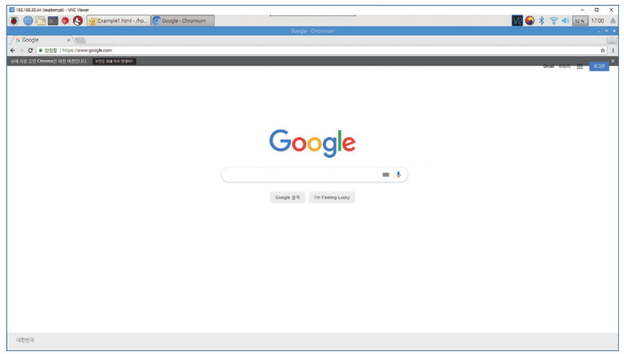
```
<!DOCTYPE html>
<html>
<head>
  <title>Example</title>
  <meta charset="utf-8"/>
</head>
<body>
    <H1>Hello World!</H1>
    <div>
        <p>
            Welcome to HTML
            Welcome to HTML1
            Welcome to HTML2
            <br>
            <a href ="http://www.google.com">Go to google</a>
        </p>
        <img src = "http://bit.ly/imgtagimg">
    </div>
</body>
</html>
```



파일을 저장하고, 바탕화면의 Example1.html 파일을 실행하여 결과를 확인합니다.

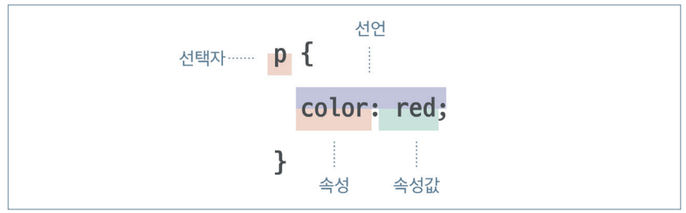


Go to google 하이퍼링크를 눌러 구글 웹 페이지로 이동해봅시다.

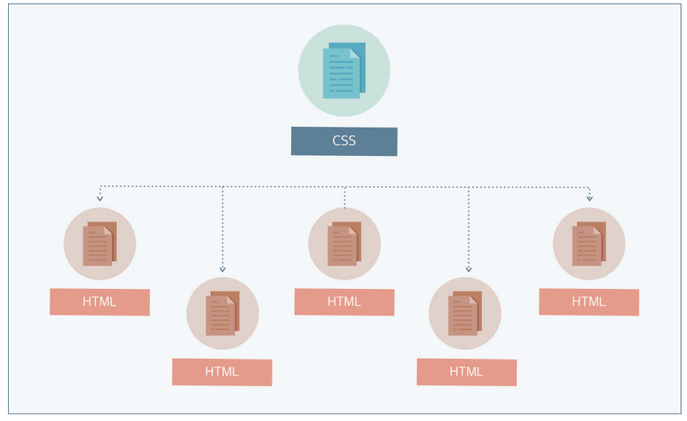


6) 웹 페이지를 꾸며주는 CSS

우리는 HTML을 사용하여 간단한 웹페이지를 만들어 보았습니다. 하지만 우리가 자주 보던 웹 페이지와는 사뭇 다른 하얀 바탕에 검은 글씨뿐인 삭막한 웹 페이지를 보았습니다. HTML은 정보를 보여주는 목적으로 만들어져 있기 때문에 아무 설정을 해주지 않는다면 하얀 바탕에 검은 글씨밖에 보여주지 않습니다. 이런 웹 페이지를 멋지게 꾸미기 위해 **CSS**Cascading Style Sheets 를 사용합니다. CSS는 HTML, XHTML, XML과 같은 문서의 스타일을 꾸밀 때 사용하는 스타일 시트 언어로, HTML이 글자, 사진, 동영상 등 웹 페이지에 포함되어야 하는 구성의 뼈대를 잡았다면, CSS는 뼈대에 살을 붙이고, 옷을 입혀 보기 좋게 꾸미는 역할을 맡고 있습니다. 예를 들어 글자 모양과 크기를 정하고, 사진과 동영상의 위치를 지정하는 등 HTML 태그 중 화면에 출력되는 부분을 꾸밀 수 있습니다. CSS는 아래 사진과 같이 선택자와 속성, 속성값으로 이루어져 있습니다. 선택자는 어떤 요소 즉, 무엇을 꾸밀지를 지정해주고, 속성과 속성값으로 어떻게 꾸밀지를 지정합니다.



CSS가 도입되기 전에는 뼈대를 구성하고, 꾸미는 작업을 HTML에서 모두 진행하였습니다. 과연 어땠을까요? 총 100 페이지로 이루어진 웹 페이지가 있다고 가정해보겠습니다. 이러한 웹 페이지의 모습을 수정하기 위해서는 100페이지나 되는 HTML 문서를 일일이 수정해야 된다는 번거로움이 있습니다. 이러한 문제를 해결하기 위해 CSS가 개발되었고, 그 결과 CSS 파일 하나만 수정하면 스타일을 참조하고 있던 모든 HTML 문서를 한 번에 수정할 수 있었습니다. 또 HTML에서는 지원하지 않는 다양한 글자와 줄 간격 기능 등 여러가지 속성을 제공하여 웹 페이지를 훨씬 쉽고, 다양하게 구성할 수 있습니다.



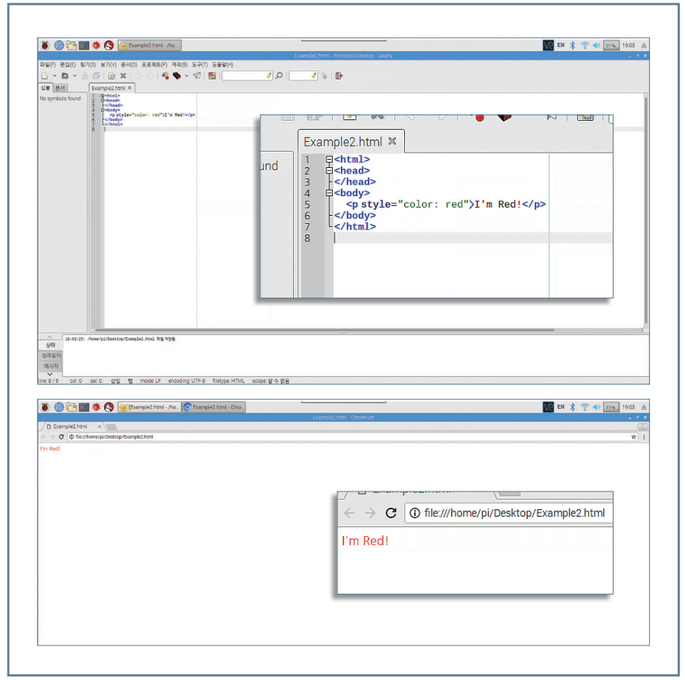
1 HTML 문서에 CSS를 사용하는 세 가지 방법

(1) 이미 작성되어 있는 HTML 요소 내부에 style 속성을 추가하는 방법

이미 작성되어 있는 HTML 요소 내부에 style 속성을 추가함으로, 제일 간단하게 스타일을 지정할 수 있는 방법입니다. 예를 들어 I'm Red! 요소에서 문자 요소를 빨간색으로 변경시키기 위해서는 아래와 같이 CSS 속성을 추가하면 됩니다. 간단하게 사용할 수 있지만, 하나의 요소에만 적용된다는 단점이 있습니다.

Example2.html

```
<html>
<head>
</head>
<body>
    <p style="color: red">I'm Red!</p>
</body>
</html>
```

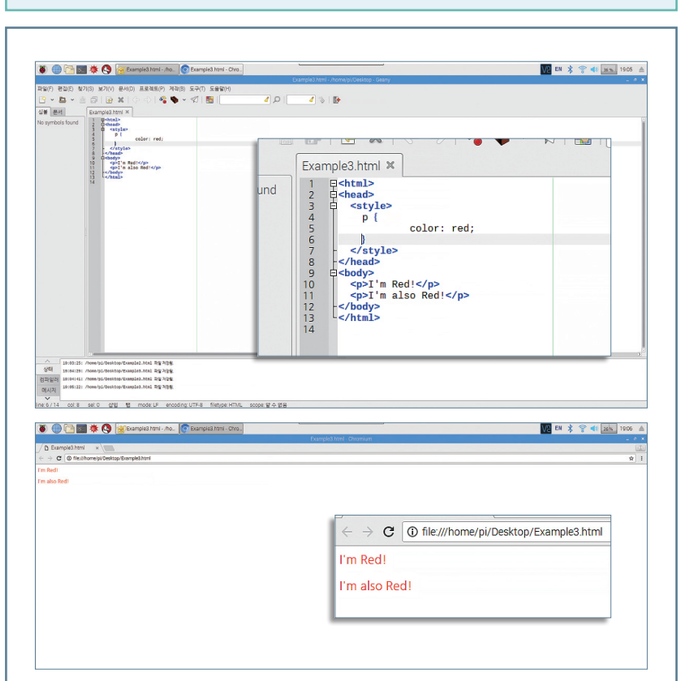


(2) head 태그 내부에 style 태그를 포함시키는 방법

<head>와 </head> 사이에 style 태그를 포함시키는 방법으로 한 번에 여러 태그에 스타일을 지정할 수 있습니다. 아래와 같이 head 태그 내부에 style 태그를 만들고, 앞에서 배운 CSS 형식에 맞춰 선택자, 속성, 속성값을 넣어줍니다. 선택자 p의 스타일은 HTML 문서의 모든 p 태그에 적용됩니다. 이 방법은 HTML 문서마다 스타일을 매번 지정해 주어야 하는 번거로움은 있지만, 한 문서에만 해당되는 스타일을 지정할 때 용이하게 사용할 수 있습니다.

Example3.html

```
<html>
<head>
    <style>
        p {
            color: red;
        }
    </style>
</head>
<body>
    <p>I'm Red!</p>
    <p>I'm also Red!</p>
</body>
</html>
```

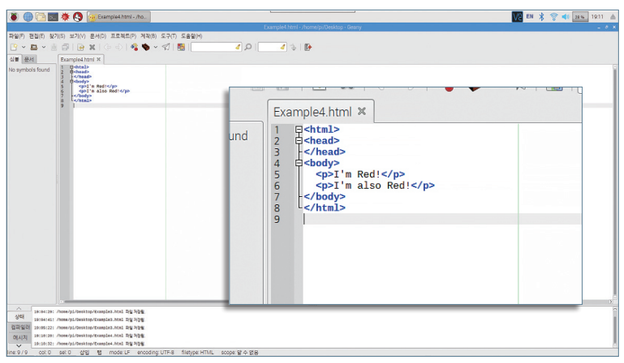


(3) 외부에 CSS 파일을 별도로 만들어 HTML 파일에 적용시키는 방법

외부에 CSS 파일을 별도로 만들어서 HTML 파일에 적용시키는 방법으로, 이 방법을 사용하면 여러 웹페이지의 스타일을 일관성 있게 유지할 수 있고, 수정과 변경을 쉽게 할 수 있습니다. 한번 사용해보겠습니다. 우선 style 태그가 빠진 html 파일을 만들어줍니다.

Example4.html

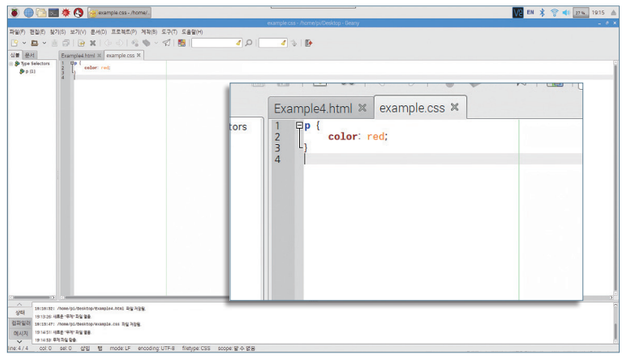
```
<html>
<head>
</head>
<body>
    <p>I'm Red!</p>
    <p>I'm also Red!</p>
</body>
</html>
```



파일-> 새 파일을 눌러 HTML 파일에 적용할 CSS 파일을 만든 후, 아래와 같이 입력하고, 저장합니다. 이때, 파일명은 example.css, 확장자는 css로 설정합니다. 여기서 사용하는 선택자 p는 앞에서 언급한 태그 선택자로, HTML 파일의 모든 p태그에 적용됩니다.

example.css

```
p {
    color: red;
}
```

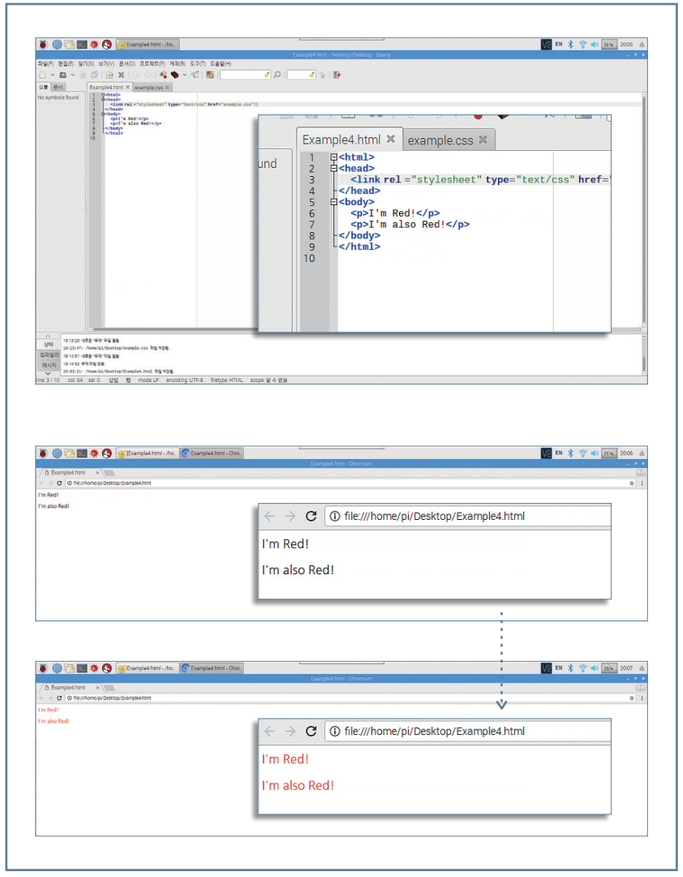


그리고 head 태그 안에 link 태그를 포함시켜 example.css 파일을 가져와 연결해줍니다.

```
<head>
    <link rel ="stylesheet" type="text/css" href="example.css"/>
</head>
```

TIP

link 태그는 외부의 문서를 연결해주기 위해 사용하는 태그입니다. 태그의 콘텐츠는 없지만, 어떤 문서를 가져올지에 대한 속성을 지정해주어야 합니다. rel 속성은 외부 문서와의 연결 관계를 뜻하며, stylesheet, alternate, author, help 등 여러 가지가 있으나 주로 stylesheet를 사용합니다. type 속성은 연결 문서의 파일 유형을 나타냅니다. 본 예제에서는 CSS 파일을 받기 때문에 text/css를 사용합니다. href는 경로를 설정해주는 속성입니다.



② CSS 선택자

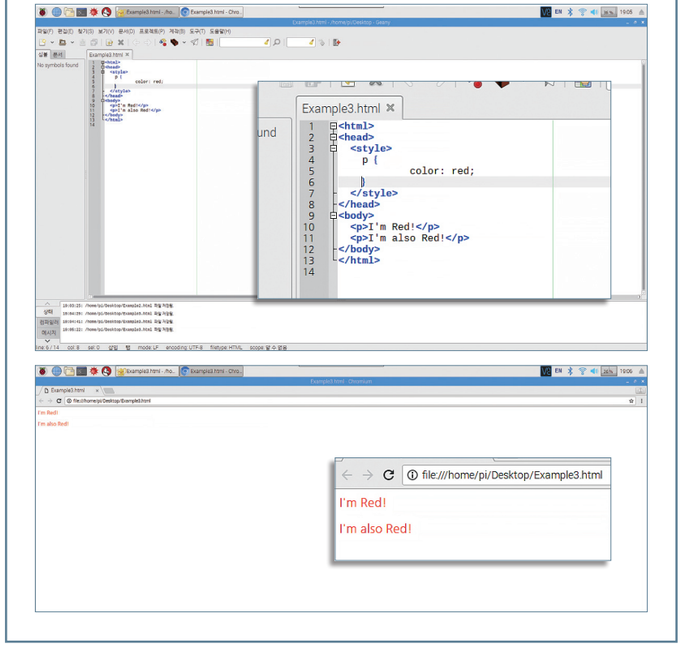
선택자를 사용하면 HTML 문서 안의 특정 요소를 선택해 스타일을 지정할 수 있습니다.

(1) 태그 선택자

태그 선택자는 태그명이 동일한 HTML 태그를 선택하여 스타일을 지정합니다.

Example3.html

```
<html>
<head>
    <style>
        p {
            color: red;
        }
    </style>
</head>
<body>
    <p>I'm Red!</p>
    <p>I'm also Red!</p>
</body>
</html>
```

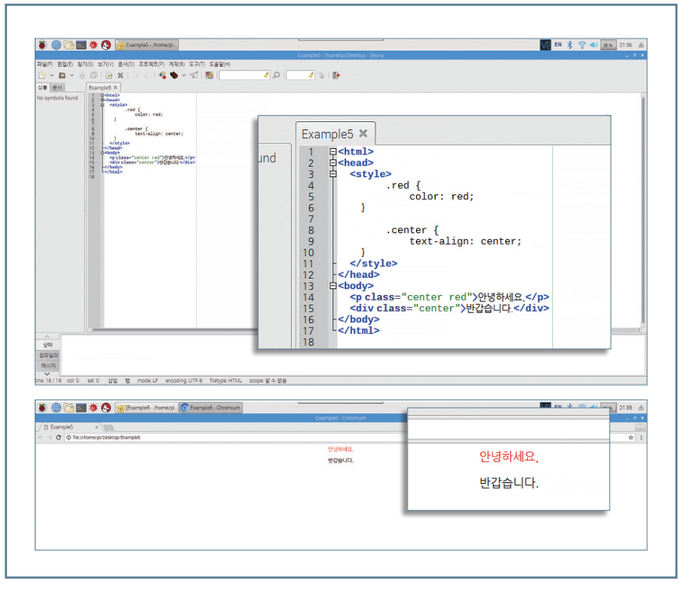


(2) 클래스 선택자

클래스의 속성값이 클래스 선택자명으로 지정된 요소를 선택하여 스타일을 지정합니다. 다른 선택자와 구분하기 위해 클래스 선택자명 앞에 마침표(.)를 붙여 사용합니다.

Example5.html

```
<html>
<head>
    <style>
        .red {
            color: red;
        }

        .center {
            text-align: center;
        }
    </style>
</head>
<body>
    <p class="center red">안녕하세요.</p>
    <div class="center">반갑습니다.</div>
</body>
</html>
```



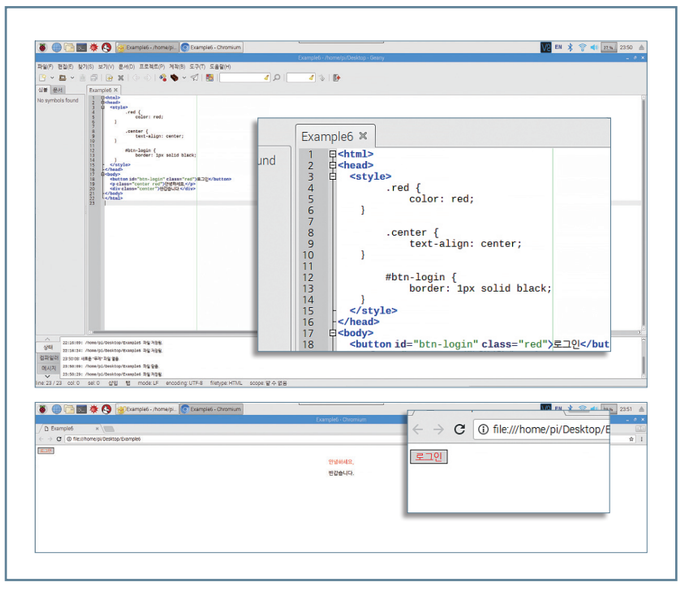
(3) 아이디 선택자

아이디의 속성값이 아이디 선택자명으로 지정된 요소를 선택하여 스타일을 적용합니다. 다른 선택자와 구분하기 위해 아이디 선택자명 앞에 샵(#)를 붙여 사용합니다.

Example6.html

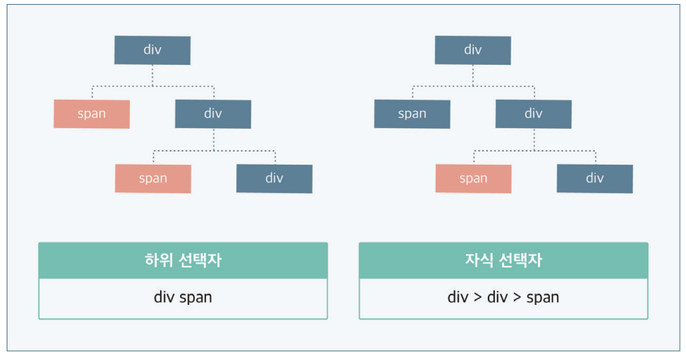
```
<html>
<head>
    <style>
        .red {
            color: red;
        }

        .center {
            text-align: center;
        }

        #btn-login {
            border: 1px solid black;
        }
    </style>
</head>
<body>
    <button id="btn-login" class="red">로그인</button>
    <p class="center red">안녕하세요.</p>
    <div class="center">반갑습니다.</div>
</body>
</html>
```

클래스 선택자와 아이디 선택자는 매우 비슷합니다. 두 선택자의 차이점은 몇 개의 요소에게 스타일을 지정하느냐입니다. 클래스 선택자는 한 페이지 내에서 여러 요소에게 한 번에 스타일을 지정할 수 있지만, 아이디 선택자는 단 한 번 유일하게 적용될 스타일을 지정할 때 사용합니다.



(4) 복합 선택자

복합 선택자는 태그 선택자, 클래스 선택자, 아이디 선택자를 조합해서 사용하는 선택자로, 조금 더 세부적으로 CSS 선택자를 지정할 수 있습니다. 선택자를 조합하는 방법은 크게 하위 선택자, 자식 선택자로 나눌 수 있습니다. 두 복합 선택자의 차이점은 한 번에 여러 요소의 스타일을 적용하느냐 아니면, 한 번에 요소 한 개의 스타일을 적용하느냐의 차이입니다.



위 그림을 한번 보도록 하겠습니다. 왼쪽은 하위 선택자, 오른쪽은 자식 선택자를 각각 나타냅니다.

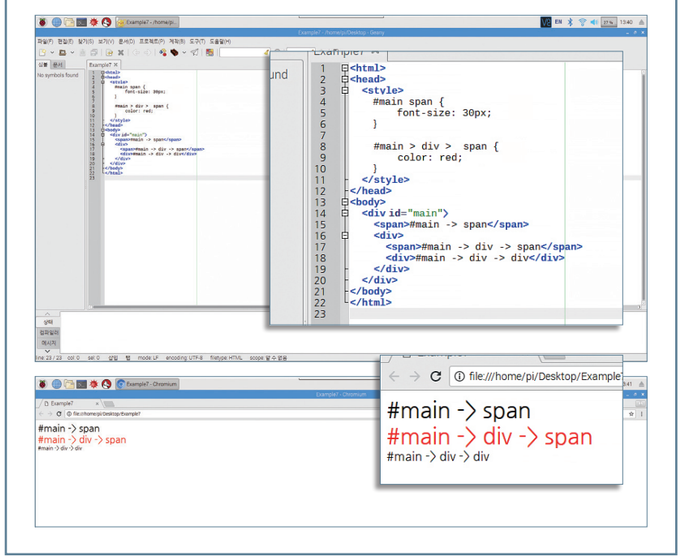
1)하위 선택자 (A B)
- 하위 선택자는 A 요소의 자손인 모든 B 요소를 선택합니다.

2)자식 선택자 (A>B)
- 자식 선택자는 A 요소의 자식인 B 요소만 선택합니다.

※ A와 B 에는 태그 선택자 외에 클래스, 아이디 선택자도 포함될 수 있습니다.

복합 선택자를 사용하여 간단한 예제를 만들어 보겠습니다.
(예제에는 글꼴의 크기를 지정해주는 font-size 속성과 글꼴색을 지정해주는 color 속성이 사용됩니다.)

```
<html>
<head>
    <style>
        #main span {
            font-size: 30px;
        }

        #main > div > span {
            color: red;
        }
    </style>
</head>
<body>
    <div id="main">
        <span>#main -> span</span>
        <div>
            <span>#main -> div -> span</span>
            <div>#main -> div -> div</div>
        </div>
    </div>
</body>
</html>
```



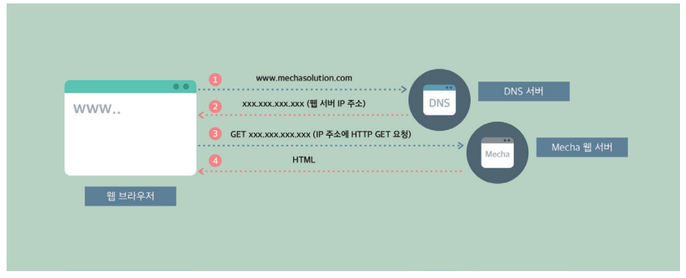
위 HTML에는 ❶하위 선택자와 ❷자식 선택자를 각각 한 개씩 포함하고 있습니다. 하위 선택자는 main이라는 id를 가진 div 태그에 포함된 모든 span 태그❸,❹의 글자 크기를 30px로 변경하고, 자식 선택자는 main이라는 id를 가진 div 태그 바로 아래 그룹에 있는 ❹span 태그의 글자 색을 빨간색으로 변경합니다. 이처럼 복합 계산자는 여러 종류의 선택자를 조합해 보다 세부적으로 스타일을 지정할 수 있습니다.

02 Requests와 웹의 정보

1) Python에서 웹에 접근해보기

우리는 앞에서 웹 페이지 화면이 보이는 원리에 대해 배웠습니다. 웹 브라우저와 웹 서버 간의 통신 방식을 Python에서도 따라 해볼 수 있습니다. 이를 위해 기본적으로 주어지는 웹 관련 모듈인 urllib을 사용할 수 있지만, Python 버전에 따라 사용법이 다르고, 한국어 및 다국어 url 지원이 미흡하기 때문에 본 교재에서는 보다 간편하게 사용할 수 있는 requests 모듈을 사용합니다.

주소창에 주소를 입력하면 DNS을 통해 웹 서버의 IP 주소를 받아와 알아낸 IP 주소에 HTTP의 GET 요청을 보냅니다. 웹 서버에서는 여러 가지 과정을 통해 사용자에게 보여줘야 할 정보가 담긴 HTML을 만들어 되돌려 보내고, 웹 브라우저는 이를 받아 화면에 보여주게 된다고 배웠습니다.

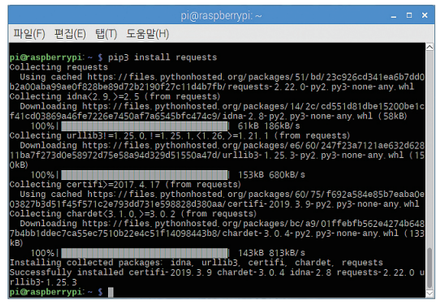


Python과 웹 통신을 사용하면 어떤 점이 좋을까요? Python에서 requests 모듈을 사용하여 웹 통신 과정을 사용하면 사람 대신 기차 예약 사이트에 접속하여 자동으로 로그인해 기차표를 예약할 수 있고, 게시판에 새로운 글이 올라왔는지 확인을 해주는 등의 작업이 가능합니다. 이처럼 인터넷과 Python의 연결은 강력한 자동화 기능을 만들어 낼 수 있습니다.

1 requests 모듈 설치하기

아래 명령어를 터미널 창에 입력하여 requests 모듈을 다운로드합니다. 이때 pip3 명령어를 사용하는 이유는 코딩팩은 python2, 3버전이 모두 설치되어 있고 그에 따라 패키지 관리자도 두 개가 존재합니다. 본 교재에서는 python3 버전을 사용하기 때문에 pip 명령어 뒤에 3을 붙입니다.

```
$ pip3 install requests
```



pip 명령어

pip 명령어는 파이썬 라이브러리의 설치 및 관리해주는 패키지 관리자입니다. 필요한 라이브러리 이름을 안다면 터미널 창에서 pip명령어를 사용해 쉽게 설치할 수 있습니다.

TIP

모듈이 함수와 변수의 집합이라면 **패키지는 모듈의 집합**입니다. 모듈들을 폴더 안의 폴더처럼 계층적 구조로 관리할 수 있습니다.

2 requests 모듈 사용하기

웹 브라우저의 주소창에 URL을 입력하면 해당 주소로 HTTP GET 요청을 보낸 후 서버로부터 HTML 코드를 받아옵니다. 이 과정처럼 Python에서 HTML 코드를 받아오기 위해 requests 모듈의 get 함수를 이용하여 HTML 코드를 받아옵니다. 다음 예제는 KT 의 웹 페이지로부터 HTML 을 불러오는 코드입니다.

request.get_Test.py

```
import requests

response = requests.get('http://www.kt.com')    # 해당 웹페이지로 GET 요청을 보냅니다.
print(response.status_code)                     # 요청 결과의 상태코드
print(response.text)                            # 요청 결과에 담겨있는 HTML 코드
```

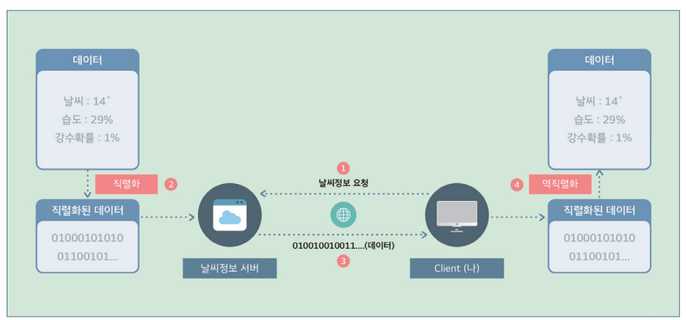
출력 결과

```
200                                             ※ 상태코드
<!doctype html>
<html lang="ko-KR">                             ※ HTML 코드
… [생략] …
</html>
```

해당 웹페이지에서 올바르게 HTML을 가져왔다면 response.status_code 로 200을 반환하고, response.text 에는 HTML 코드가 반환됩니다.

2) Python 변수를 압축해 웹으로 보내기 – 직렬화

HTML 코드는 웹 페이지로 변환되어 사람들에게 컨텐츠를 보여주기 위한 목적을 가지고 있습니다. 그래서 사람은 웹 페이지의 화면만 보고 어떤 내용을 담고 있는지 금방 알아차리지만 컴퓨터가 웹 페이지에서 정보를 찾아내기에는 한계가 있습니다. HTML 코드는 글, 사진, 동영상 등으로 변환되어 사람들에게 콘텐츠를 보여주기 위한 목적을 담고 있습니다. 그래서 컴퓨터끼리 웹에서 통신을 하기 위해 **직렬화**Serialize라는 방법을 이용해 컴퓨터가 이해하는 형태로 데이터를 변환해 주고받습니다.



직렬화는 사람들이 이해할 수 있는 방법과, 이해하지 못하는 방법이 있습니다. 사람이 이해하기 쉽다는 것은 컴퓨터가 이해하기 어렵다는 것이고, 컴퓨터가 이해하기 쉽다는 것은 사람이 이해하기 어렵다는 것입니다. 실제로 사용되는 Python의 두 가지 직렬화 방식을 예로 들어 설명해보도록 하겠습니다. Python에서 기본적으로 제공하는 직렬화 기법인 Pickle과 가장 유명한 범용 직렬화 방식 JSON을 비교해보겠습니다.

1 이진 직렬화 - Python Pickle

Pickle은 Python에서만 사용 가능한 직렬화 방식입니다.

pickle_test1.py

```
import pickle
data = {'hello': 'world'}

with open('pickled.txt', 'wb') as file:
    pickle.dump(data, file)

with open('pickled.txt', 'rb') as file:
    print(pickle.load(file))
```

출력 결과

```
$ python3 pickle_test1.py
{'hello': 'world'}
```

Python에서 직렬화는 dump 와 load 두 함수만 알면 매우 쉽습니다. dump 함수를 이용해 pickled.txt에 data 변수를 직렬화한 데이터를 저장합니다. 이후 pickled.txt 파일을 읽기 전용으로 열어 load 함수를 사용해 역직렬화 합니다. 이렇게 직렬화된 데이터를 해석하는 과정을 **역직렬화** 또는 **파싱**이라 합니다.

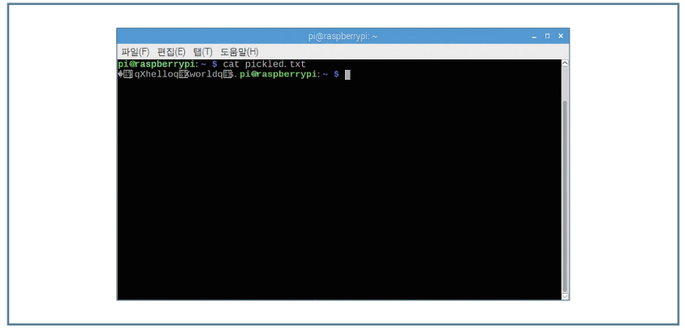
pickled.txt 파일에는 직렬화된 데이터가 쓰이고 읽힙니다. 과연 어떤 내용이 들어있는지 cat 명령어를 사용하여 pickled.txt 파일을 열어보도록 하겠습니다. 터미널 창에 아래와 같이 입력 후 엔터 키를 눌러 파일에 저장된 내용을 확인합니다.

```
$ cat pickled.txt
```

출력 결과

```
♦}qXhelloqXworldqs
※ cat 명령어는 명령어 뒤에 입력된 파일 내용을 그대로 출력하는 리눅스 명령어입니다.
```

출력 결과에서 확인할 수 있듯이 pickled.txt 파일은 알 수 없는 문자로 이루어져 있습니다. 이처럼 Pickle 직렬화 방식은 사람이 알아볼 수 없는 0과 1로 이루어진 이진 형식을 사용합니다. 다른 직렬화 방식에 비해 성능은 떨어지지만, 변수나 리스트뿐만 아니라, 심지어는 제너레이터와 함수도 직렬화를 시킬 수 있어 Python에 특정된 프로그램끼리 소통할 때 유용하게 쓰입니다.



사실 위에서 사용한 방법은 파일에 어떤 내용이 저장되는지를 보여주기 위해서 사용했습니다. 파일을 거치지 않고 사용하는 방법은 아래와 같습니다.

pickle_test2.py

```
import pickle
data = {'hello': 'world'}

serialized = pickle.dumps(data)
print(serialized)
print(pickle.loads(serialized))
```


② 문자열 형식 직렬화 - JSON

문자열 형식 직렬화는 이전 형태로 직렬화되던 방법과는 달리 사람들이 알아볼 수 있게끔 변환합니다. 문자열 형식 직렬화에는 CSV, YAML, XML 등 여러 가지 방식이 있으나 본 교재에서는 웹에서 통신할 때 보편적으로 쓰이는 **JSON** Java Script Object Notation 를 사용하도록 하겠습니다.

JSON은 단순함과 보편성을 중시하도록 설계되었습니다. 표현할 수 있는 데이터의 종류를 한정시키고, 기존에 쓰이던 XML보다 매우 간단한 문법으로 다른 직렬화 기법보다 작은 크기를 자랑합니다.

TIP

null은 Python의 None과 같은 개념입니다.

아래 예는 '홍길동'이라는 사람의 정보를 나열한 **JSON 객체**Object 입니다. 중괄호{ } 안에 감싸져 있는 한 덩어리를 JSON 객체라 합니다. JSON은 Key-Value 형태의 문법을 사용하는데, 이는 Python의 딕셔너리와 비슷합니다. Key는 문자열, Value에는 숫자, 문자열, 불형, JSON 객체, 배열 그리고 null을 가질 수 있습니다.

```
{
    "name": "홍길동",
    "age": 60,
    "address": "서울특별시 강서구 ",
    "company": {
        "name": "메카솔루션",
        "address": "대구광역시 달서구",
        "call": "053-123-4567",
        "emails": [
            "aaaaa@aaaaa.com",
            "bbbbb@bbbbb.com",
            "ccccc@ccccc.com"
        ]
    },
    "isMarried": false
}
```

위 JSON 문자열에서 이름, 나이, 주소, 회사, 결혼 상태를 알 수 있습니다. 이때 회사의 정보는 단순히 이름만 있는 것이 아니라 여러 정보가 있을 수 있습니다. 그래서 "company" 키에는 회사를 나타내는 이름, 주소, 전화번호 등 다양한 JSON 객체가 들어갈 수 있습니다. 그리고 "company"의 "emails"에는 대괄호가 들어가 있습니다. 이는 여러 값을 나열하는 Python의 리스트와 동일한 역할을 합니다.

JSON 모듈도 Pickled 모듈처럼 dump(s)와 load(s) 함수를 사용하여 쉽게 사용 가능합니다.

json_test.py

```
import json
data = {'hello': 'world'}

with open('test.json', 'w') as file:
    json.dump(data, file)

with open('test.json', 'r') as file:
    print(json.load(file))

serialized = json.dumps(data)
print(serialized)
print(json.loads(serialized))
```

출력 결과

```
{'hello': 'world'}
{"hello": "world"}
{'hello': 'world'}
```

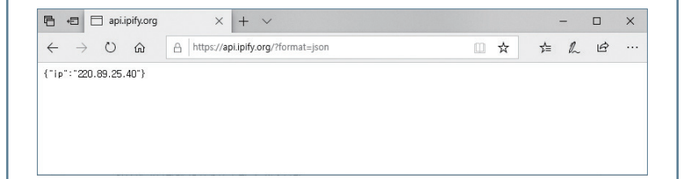
출력 결과를 통해 확인할 수 있듯이 문자열 형식 직렬화(JSON)는 사람과 컴퓨터 모두가 알아볼 수 있게끔 변환됩니다. 이진 직렬화 방식과 확연히 다른 점을 확인할 수 있습니다.

❸ JSON 과 requests을 이용해 내 IP주소 가져오기

웹에는 날씨, 뉴스, 우체국, 미세먼지, 버스 정보 등 여러 가지 기능 및 정보를 제공하는 여러 API가 있습니다. 그중에서 가장 간단하게 사용할 수 있는 API 중 하나인 '내가 사용하고 있는 IP주소'를 알아보는 웹 API을 사용해보겠습니다.

웹 API는 웹 브라우저에서도 결괏값을 확인해볼 수 있습니다. HTML 코드를 가져오는 것 대신에 JSON 문자열을 받아와 화면에 출력합니다. 아래 링크로 접속하면, JSON 문자열 형태로 사용하고 있는 IP주소가 출력됩니다.

```
https://api.ipify.org?format=json
```



python에서 requests 모듈을 사용하여 IP주소를 가져오는 코드를 만들어 보겠습니다. requests 모듈의 get 함수를 통해 http://api.ipify.org?format=json 의 HTML 코드를 받아오고, response의 json 함수를 이용해 간단하게 역직렬화를 할 수 있습니다.

requests_test.py

```
import requests
response = requests.get('http://api.ipify.org?format=json')
print(response.json())
```

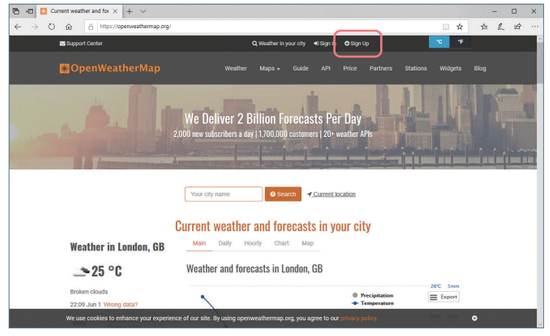
출력 결과

```
{'ip': '123.45.65.89'}
```

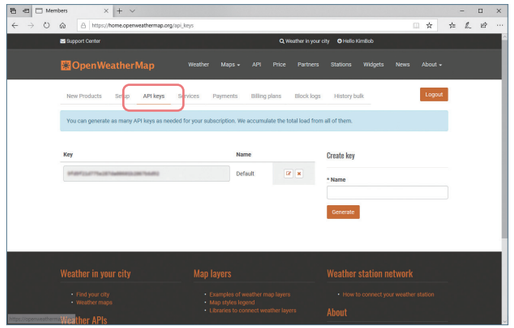
03 날씨 API 사용하기

세상에는 여러 정보가 수집이 됩니다. 세계의 날씨 정보, 미세먼지 정보, 뉴스 정보, 지역의 도서관 정보 등 많은 정보가 있을 수 있습니다. 대부분 이런 정보들은 개인이 직접 관리하고 배포하기가 힘듭니다. 그래서 정부, 기업, 기관에서 정보들을 수집, 관리하는데, 이 정보들을 공개하고 누구나 사용할 수 있도록 API의 형태로 배포하는 것을 Open API라고 합니다.

세계의 날씨 정보를 무료로 제공해주는 Open API와 코딩팩을 연동해 날씨 정보를 가져오는 예제를 만들어 보겠습니다. 날씨 정보를 제공하는 많은 Open API 중 'OpenWeatherMap'이라는 날씨 서비스를 사용해보도록 하겠습니다. OpenWeatherMap은 현재 날씨 정보(무료), 3시간 간격으로 5일간 예보(무료), 날씨 지도(무료), 시간별 예보(유료), 16일간 예보(유료) 등 여러 가지 날씨 API를 제공합니다. OpenWeatherMap에서 정보를 받아오기 위해서는 OpenWeatherMap 회원가입이 필요합니다. 'https://openweathermap.org/' 링크에 접속한 후 상단 Sign Up 'Sign up' 버튼을 눌러 회원가입을 진행합니다.



아래 사진과 같이 로그인 후 API keys 'API Keys' 카테고리로 들어가 발급된 Key를 확인합니다.



TIP

API을 제공하는 회사들은 API를 소개하는 설명서를 함께 제공합니다. 설명서에는 API 소개, 구성, 사용법 등 다양한 정보를 제공합니다. 뒤에서 사용하게 될 API 형식도 아래 설명서에서 확인할 수 있습니다.
참고링크 : https://openweathermap.org/current

1) Python을 사용하여 날씨 정보 가져오기

OpenWeatherMap에서 무료로 제공하는 현재 날씨 정보 API는 도시의 이름, 도시의 코드, 위도와 경도, 우편번호 네 가지 방법을 이용하여 해당 지역의 날씨를 가져올 수 있습니다. 우선 도시의 이름을 이용하여 현재 날씨를 가져와 보겠습니다. 도시의 이름을 이용해 날씨를 가져오기 위해서는 날씨 정보를 요청하는 URL에 도시의 이름을 적어주어야 합니다. 아래의 형식을 지켜 날씨 정보를 요청할 수 있습니다.

```
https://api.openweathermap.org/data/2.5/weather?q={도시명},{국가코드}&appid={API 키}&units={빈칸 또는 metric 또는 imperial}
```

units 뒤에 오는 인자로 온도와 풍속의 단위를 설정할 수 있습니다.

도량형 단위법	온도단위	풍속단위
-	켈빈온도	m/s
metric(미터법)	섭씨온도	m/s
imperial(야드파운드법)	화씨온도	mph(시간당마일)

도시명과 국가코드를 인자로 받아 그 지역의 현재 날씨 정보를 돌려주는 함수를 만들어봅시다.

weather.py

```
import requests

def get_weather_by_code(code):
    api_url = "https://api.openweathermap.org/data/2.5/weather?q=%s&appid=%s&units=%s"
    app_id = "여러분에게 발급된 API 키"
    unit = "metric"    # 미터법을 이용하겠다 명시

    response = requests.get(api_url % (code, app_id, unit))
    if response.status_code == 200:
        return response.json()
    else:
        return None

if __name__ == "__main__":
    print(get_weather_by_code('Seoul,KR'))
    print(get_weather_by_code('Daegu,KR'))
```

weather.py 파일은 실행 결과로 아래와 같이 딕셔너리 자료형으로 변환된 날씨 데이터를 출력합니다.

출력 결과

```
{'visibility': 10000, 'base': 'stations', 'weather': [{'icon': '50d', 'description': 'haze', 'main':
'Haze', 'id': 721}], 'wind': {'gust': 5.7, 'deg': 200, 'speed': 3.1}, 'timezone': 32400, 'cod': 200,
'name': 'Seoul', 'main': {'temp_min': 22, 'temp': 24.82, 'temp_max': 26, 'humidity': 56, 'pressure':
1010}, 'id': 1835848, 'coord': {'lon': 126.98, 'lat': 37.57}, 'clouds': {'all': 20}, 'sys': {'message':
0.0058, 'type': 1, 'id': 5501, 'country': 'KR', 'sunset': 1559213139, 'sunrise': 1559160824}, 'dt':
1559197978}  ※ 서울날씨
{'visibility': 10000, 'base': 'stations', 'weather': [{'icon': '04d', 'description': 'broken clouds',
'main': 'Clouds', 'id': 803}], 'wind': {'deg': 260, 'speed': 2.6}, 'timezone': 32400, 'cod': 200, 'name':
'Daegu', 'main': {'temp_min': 25, 'temp': 28.22, 'temp_max': 30, 'humidity': 23, 'pressure': 1007},
'id': 1835329, 'coord': {'lon': 128.56, 'lat': 35.85}, 'clouds': {'all': 75}, 'sys': {'message': 0.0082,
'type': 1, 'id': 5503, 'country': 'KR', 'sunset': 1559212485, 'sunrise': 1559160720}, 'dt': 1559198069}
※ 대구날씨
```

출력 결과를 보고 "이게 뭐야!" 라고 생각하시는 분들이 많을 것으로 예상합니다. 출력 결과에는 현재 날씨 상태, 온도, 습도 등 다양한 날씨 정보가 들어있습니다. 아래 내용은 'OpenWeatherMap' 홈페이지에서 제공하는 날씨 정보 API에 대한 해석을 한국어로 번역한 것입니다. 출력 결과와 비교하면서 확인하시길 바랍니다.

```
{
    "coord": {
        "lon": 경도,
        "lat": 위도
    },
    "weather": [
        {
            "id": 날씨코드,
            "main": 날씨상태,
            "description": 날씨설명,
            "icon": 아이콘코드
        }
    ],
    "main": {
        "temp": 온도(기본 단위: K, Metric: °C, Imperial: °F),
        "pressure": 대기압(hPa),
        "humidity": 습도(%),
        "temp_min": 최저온도(기본 단위: K, Metric: °C, Imperial: °F),
        "temp_max": 최고온도(기본 단위: K, Metric: °C, Imperial: °F),
        "sea_level": 해면 대기압(hPa),
        "grnd_level": 지면 대기압(hPa)
    },
    "wind": {
        "speed": 풍속(기본 단위: m/s, Metric: m/s, Imperial: mph),
        "deg": 풍향(°)
    },
    "clouds": {
        "all": 구름의양(%)
    },
    "rain": {
        "1h": 1 시간 동안의 강우량(mm)
        "3h": 3 시간 동안의 강우량(mm)
    },
    "snow": {
       "1h": 1 시간 동안의 강수량(mm)
       "3h": 3 시간 동안의 강수량(mm)
    },
    "dt": 요청 UTC 기준 UNIX 시간,
    "sys": {
        "type": 내부 인자,
        "id": 내부 인자,
        "message": 내부 인자,
        "country": 국가 코드(KR, US 등),
        "sunrise": 일출시간(UTC 기준 UNIX 시간)
        "sunset": 일몰시간(UTC 기준 UNIX 시간)
    },
    "timezone": UNIX 시간대,
    "id": 도시코드,
    "name": 도시이름,
    "cod": 내부 인자
}
```

UNIX 시간이란 1970년 1월 1일 정오 이후로 지난 초를 기준으로 세는 시간 법입니다. 예를 들어 1559128398은 1970년 1월 1일 정오 이후로 1547436540초 만큼 지난 UTC 기준 2019년 1월 14일 11시 29분 00초를 뜻합니다.

2) 날씨 API와 코딩팩 연동하기

코딩팩에게 특정 지역의 날씨를 물어보면 자동으로 날씨 정보를 불러와 브리핑 해주는 애플리케이션을 만들어 보겠습니다. “서울특별시 강남구 날씨 알려줘” 혹은 “강남구 날씨 알려줘”라는 문장에서 위치 정보는 ‘서울특별시’, ‘강남구’가 됩니다. 서울특별시에 있는 강남구 날씨를 가져오면 정말 좋겠지만 OpenWeatherMap은 영국 회사에서 만든 API이기 때문에 서울특별시와 강남구를 알아듣지 못합니다. 그래서 우리는 주소를 위도와 경도로 변형해주는 API를 추가로 사용해 위도와 경도를 이용하여 해당 지역의 날씨 정보를 OpenWeatherMap에서 가져오도록 하겠습니다.

Weather 파일에 있는 code (지역명) 함수와 coord(위도, 경도) 함수를 혼동하기 쉬우니 주의하시길 바랍니다.

기능을 정리해보면, 코딩팩은 아래의 과정을 거쳐 날씨 정보를 알려주게 됩니다.

1. 사용자에게 날씨를 알고 싶은 지역 입력 받기
2. 입력받은 지역의 위도, 경도를 검색하기
3. 위, 경도를 기반으로 현재 날씨 가져오기
4. 날씨 정보를 문장으로 바꿔 음성으로 출력하기

① 위도, 경도를 사용하여 날씨 정보를 가져오는 API 함수 만들기

위도, 경도를 사용하는 API는 아래 형식과 같이 사용할 수 있습니다.

```
https://api.openweathermap.org/data/2.5/weather?lat={위도}&lon={경도}
&appid={API 키}
```

특정 지역의 위도와 경도 입력해 날씨 정보를 받아오는 함수를 만들어보겠습니다.

Weather.py

```
def get_weather_by_code(code):
    api_url = "https://api.openweathermap.org/data/2.5/weather?q=%s&appid=%s&units=%s"
    app_id = "여러분에게 발급된 API 키"
    unit = "metric" # 미터법을 이용하겠다 명시

    response = requests.get(api_url % (code, app_id, unit))
    if response.status_code == 200:
        return response.json()
    else:
        return None

if __name__ == "__main__":
     print(get_weather_by_code('Seoul,KR'))
     print(get_weather_by_coord(37.50059, 127.050775)) # 서울특별시 강남구의 위도와 경도
```


출력 결과

```
{'visibility': 10000, 'weather': [{'main': 'Clear', 'icon': '01d', 'description': 'clear sky', 'id':
800}], 'sys': {'sunset': 1559472463, 'sunrise': 1559419952, 'type': 1, 'country': 'KR', 'message':
0.0055, 'id': 5509}, 'name': 'Seoul', 'clouds': {'all': 1}, 'wind': {'speed': 1, 'deg': 280}, 'main':
{'temp': 27.13, 'humidity': 28, 'pressure': 1006, 'temp_max': 28, 'temp_min': 26}, 'cod': 200,
'timezone': 32400, 'base': 'stations', 'coord': {'lat': 37.57, 'lon': 126.98}, 'id': 1835848, 'dt':
1559455570}
{'visibility': 10000, 'weather': [{'main': 'Clear', 'icon': '01d', 'description': 'clear sky', 'id':
800}], 'sys': {'sunset': 1559472435, 'sunrise': 1559419947, 'type': 1, 'country': 'KR', 'message':
0.0052, 'id': 5509}, 'name': 'Sinch'ŏn-dong', 'clouds': {'all': 1}, 'wind': {'speed': 1, 'deg': 280},
'main': {'temp': 27.13, 'humidity': 28, 'pressure': 1006, 'temp_max': 28, 'temp_min': 26}, 'cod': 200,
'timezone': 32400, 'base': 'stations', 'coord': {'lat': 37.5, 'lon': 127.05}, 'id': 1837217, 'dt':
1559455933}
```

입력한 지역 혹은 위도, 경도 근처의 측정소에서 측정된 데이터를 바탕으로 날씨 정보를 가져옵니다. 출력 결과에서 표시해 놓은 두 부분은 서울특별시의 온도와 습도, 기압, 최고온도, 최저온도를 나타냅니다.

② 주소를 위도, 경도로 변환하기

주소를 위도와 경도로 바꾸기 위해 OpenStreetMap이라는 무료 지도 사이트에서 제공하는 API를 사용하도록 하겠습니다. API는 아래와 같은 형식으로 사용할 수 있으며, format 인자를 이용해 HTML, XML, JSON 등 여러 가지 형태로 받을 수 있습니다. 그리고 countrycodes 인자를 지정하면 해당국가의 결과만 받아오도록 제한할 수도 있습니다.

```
https://nominatim.openstreetmap.org/search?q={서울특별시 강남구}&format=
json&countrycodes=kr
```

인터넷 주소 입력창에 위와 같이 API를 입력 후 검색하면 아래와 같이 위도와 경도가 포함된 JSON 문자열을 얻을 수 있습니다.

결과로 받아온 JSON 문자열을 펼쳐 보겠습니다. JSON 문자열에는 다른 여러 정보를 포함하고 있지만, 우리에게 필요한 것은 위도와 경도입니다. 아래 표시된 부분이 서울특별시의 위도와 경도입니다.


```
[
        {
                "place_id":198678373,
                "licence":"Data © OpenStreetMap contributors, ODbL 1.0. https://osm.org/copyright",
                "osm_type":"relation",
                "osm_id":2410520,
                "boundingbox":[
                        "37.4564808",
                        "37.5399807",
                        "127.0083762",
                        "127.1227254"
                ],
                "lat":"37.50059",                    # 서울특별시 강남구 위도
                "lon":"127.050775",                  # 서울특별시 강남구 경도
                "display_name":"강남구, 서울특별시, 대한민국",
                "class":"boundary",
                "type":"administrative",
                "importance":0.694794797841327,
        },
        {
                "place_id":195333306,
                "licence": "Data © OpenStreetMap contributors, ODbL 1.0. https://osm.org/copyright",
                "osm_type":"way",
                "osm_id":601312744,
                "boundingbox":[
                        "37.5069361",
                        "37.5099481",
                        "127.0675681",
                        "127.0684735"
                ],
                "lat":"37.5088558",
                "lon":"127.068001155645",
                "display_name":"강남구 탄천주차장, 동부간선도로, 대치2 동, 강남구, 서울특별시, 06175,
대한민국",
                "class":"amenity",
                "type":"parking","importance":0.22100000000000003
        }
 ]
```


OpenStreetMap을 이용해 위도와 경도를 찾는 과정을 Python 코드를 이용해서 자동화해보도록 하겠습니다. 주소를 텍스트로 입력받아 첫 번째 결과의 위도와 경도를 가져오는 함수를 만들어보겠습니다. map.py 파일을 새로 만들어 아래와 같이 입력합니다.

map.py

```python
import requests

def get_coord_by_address(address):
    api_url = 'https://nominatim.openstreetmap.org/search?q=%s&format=json&countrycodes=kr'
    response = requests.get(api_url % address)
    if response.status_code == 200:
        json = response.json()
        if len(json) > 0:
            first = json[0]
            return (first['lat'], first['lon'])
        else:
            return None
    else:
        return None

if __name__ == "__main__":
    print(get_coord_by_address('서울특별시 강남구'))
```

만약 검색 결과가 있다면, 검색 결과의 첫 번째를 가져와 'lat'(위도)과 'lon'(경도)을 튜플로 만들어 값을 되돌려 줍니다.

출력 결과

```
('37.50059', '127.050775')
```

3 음성으로 결과를 출력하는 함수 만들기

날씨 데이터를 문장으로 적절히 바꾸어주는 코드가 필요합니다. create_weather_text 함수를 만들어 문장으로 바꾸어 봅시다. create_weather_text 함수는 서버로부터 받아온 딕셔너리 형태의 날씨 정보를 인자로 받습니다. 만약 None이 인자로 넘어왔다면 “날씨를 알 수 없습니다.”를 출력합니다.

weather.py

```
def create_weather_text(weather_info, address):
    response = "현재 %s 의 온도는 %d 도이고, 습도는 %d 퍼센트입니다."
    if weather_info != None:
        return response % (address, weather_info['main']['temp'],
weather_info['main']['humidity'])
    else:
        return "날씨를 알 수 없습니다."
```

4 날씨 API와 코딩팩 연동하는 함수 만들기

main 함수는 날씨 API와 코딩팩을 연동하는 부분으로 앞에서 만들어 두었던 함수를 단계별로 실행시켜 날씨 정보를 가져와 음성으로 출력합니다.

```
def main():
    voice.speech("어디의 날씨를 알려드릴까요?")
    address = voice.get_text_from_voice()

    coord = map.get_coord_by_address(address)
    if coord == None:
        voice.speech("%s 을 찾지 못했습니다." % address)
    else:
        weather_info = get_weather_by_coord(coord[0], coord[1])

        response_text = create_weather_text(weather_info, address)
        voice.speech(response_text)


if __name__ == "__main__":
    main()
```

사용자로부터 받은 입력을 address에 넣고 map 모듈의 get_coord_by_address 함수를 이용해 위도와 경도를 튜플로 받아옵니다. 만약에 좌표를 찾지 못했다면 찾지 못했다고 말해주고, 아니라면 좌표를 이용해 날씨를 가져옵니다. 이 날씨 정보를 문장으로 바꾸어 음성으로 출력합니다.

코딩팩이 “어디의 날씨를 알려드릴까요?” 라고 질문하면 원하는 지역명을 이야기합니다. 예를 들어 “서울”, “서울 강남구” 등으로 대답합니다. 여기서 뒤에 “날씨 알려줘.”는 생략합니다. 즉, “서울특별시 날씨 알려줘.” 라고 대답하면 올바른 결과를 가져오지 못합니다.

코딩 전체보기

weather.py

```
import requests
import voice
import map

def get_weather_by_code(code):
    api_url = "https://api.openweathermap.org/data/2.5/weather?q=%s&appid=%s&units=%s"
    app_id = "여러분에게 발급된 API 키"
    unit = "metric" # 미터법을 이용하겠다 명시

    response = requests.get(api_url % (code, app_id, unit))
    if response.status_code == 200:
        return response.json()
    else:
        return None

def get_weather_by_coord(lat, lon):
    api_url = "https://api.openweathermap.org/data/2.5/weather?lat=%s&lon=%s&appid=%s&units=%s"
    app_id = "여러분에게 발급된 API 키"
    unit = "metric" # 미터법을 이용하겠다 명시

    response = requests.get(api_url % (lat, lon, app_id, unit))
    if response.status_code == 200:
        return response.json()
    else:
        return None

def create_weather_text(weather_info, address):
    response = "현재 %s 의 온도는 %d 도이고, 습도는 %d 퍼센트입니다."
    if weather_info != None:
        return response % (address, weather_info['main']['temp'], weather_info['main']['humidity'])
    else:
        return "날씨를 알 수 없습니다."
def main():
    voice.speech("어디의 날씨를 알려드릴까요?")
    address = voice.get_text_from_voice()

    coord = map.get_coord_by_address(address)
    if coord == None:
        voice.speech("%s 을 찾지 못했습니다." % address)
    else:
        weather_info = get_weather_by_coord(coord[0], coord[1])

        response_text = create_weather_text(weather_info, address)
        voice.speech(response_text)


if __name__ == "__main__":
    main()
```

1 212P-213P

2 213P-215P

3 215P

4 216P

map.py

```
import requests

def get_coord_by_address(address):
    api_url = 'https://nominatim.openstreetmap.org/search?q=%s&format=json&countrycodes=kr'
    response = requests.get(api_url % address)
    if response.status_code == 200:
        json = response.json()
        if len(json) > 0:
            first = json[0]
            return (first['lat'], first['lon'])
        else:
            return None
    else:
        return None
```

04
웹 크롤링과 HTML코드

1) 웹 크롤링을 사용하여 HTML 코드에서 필요한 정보 가져오기

이때까지는 API를 사용하여 웹에서 정보를 가져올 수 있었습니다. 하지만 아쉽게도 웹에서는 모든 정보를 API로 제공하지는 않습니다. 음원 차트의 순위 및 제목, 포털 사이트의 실시간 검색어 순위 등의 정보를 웹에서 가져와 파이썬에서 사용하고 싶지만 requests 모듈과 API만으로는 해결할 수 없는 한계에 부딪히고 맙니다. 지금부터는 requests 모듈과 API를 사용하지 않고 웹에 있는 정보를 수집하는 방법과 수집된 정보 중 필요한 것들만 추출해내는 방법에 대해 알아보도록 하겠습니다.

> 크롤링은 웹의 정보를 수집하는 행위입니다. 하지만 웹의 정보들은 모두 저작권이 있습니다. 이를 무작정 수집하여 도용하는 행위는 불법이며, 큰 이슈가 되고 있습니다. 그래서 크롤링을 하기 전, 저작권을 확인 후 시작해야 합니다.

앞서 설명한 것과 같이 웹 페이지는 웹 브라우저가 서버로부터 HTML코드를 받아와 화면에 출력하는 것입니다. 결국 웹 페이지는 HTML 코드로 구성되어 있는 것입니다. 너무나 고맙게도 HTML 코드를 작성할 때 개발자들은 다른 개발자들을 위해서 정형화된 규칙을 지키고 있습니다. 우리는 정형화된 규칙속에서 필요한 패턴을 찾아 그 패턴이 어떤 정보를 가지고 있는지 확인만 하면 우리에게 필요한 정보를 얻을 수 있습니다. 이렇게 웹 페이지에 나열되어 있는 정보를 수집하고, 추출하는 행위를 "웹 크롤링"이라 하고, 실제 크롤링 하는 로봇을 "웹 크롤러"라 부릅니다. 방대한 인터넷 상에서 마치 웹이라는 거미줄을 돌아다니는 모습과 비슷하다 하여 "스파이더"라고 부르기도 합니다. 지금부터 크롤러를 이용해 웹을 돌아다니며 정보를 수집해보고, 이 정보를 코딩팩과 연동시키는 작업을 해보도록 하겠습니다.

2) 파이썬 BeautifulSoup 모듈 사용하기

BeautifulSoup 모듈은 웹에서 HTML 코드를 불러오지는 못하지만, 다른 방법을 이용해 불러온 HTML 코드를 탐색하는 작업을 도와주는 모듈입니다. 그래서 이전에 사용했던 requests 모듈을 이용해 HTML 코드를 가져와 BeautifulSoup에 전달해주는 방식으로 진행해보겠습니다.

우선 requests 모듈을 설치할 때처럼 pip 명령어를 사용하여 BeautifulSoup 모듈을 설치해보겠습니다. 아래의 명령어를 터미널 창에 입력하여 BeautifulSoup 모듈을 설치합니다.

```
$ pip3 install beautifulsoup4
```

설치가 완료되었다면, BeautifulSoup 모듈을 사용하는 간단한 예제를 따라 해보겠습니다. 우선 BeautifulSoup 모듈을 import를 한 후, BeautifulSoup 모듈에서 사용할 HTML 코드를 아래와 같이 만들어 변수에 넣어줍니다.

simple_crawler.py

```
from bs4 import BeautifulSoup

html = '<div id="content"><p class="text">Hello, World!</p><p class="text">Hello, Giga Genie!</p></div>'
```

HTML 문자열 변수를 BeautifulSoup 모듈로 넘겨 인스턴스를 만들어줌과 동시에 이번에는 HTML을 파싱 하겠다는 의미로 'html.parser'를 인자로 함께 넘겨줍니다.

```
bs = BeautifulSoup(html, 'html.parser')
```

BeautifulSoup 모듈로 HTML 코드를 탐색하는 가장 단순한 방법은 아래와 같습니다. 태그의 이름을 마침표(.)로 구분하여 탐색할 수 있습니다.

```
print(bs.div)
print(bs.div.p)
```

출력 결과

```
<div id="content"><p class="text world">Hello, World!</p><p class="text giga">Hello, Giga Genie!</p></div>
<p class="text">Hello, World!</p>
```

이 방법을 이용할 경우 가장 첫 번째에 위치한 태그만 가져온다는 문제점이 있습니다. 예를 들어 두 번째 출력 결과를 보면 p 태그가 현재 두 개가 있음에도 위의 방법을 이용하면 맨 처음에 위치한 p 태그만 가져옵니다. 그리고 태그로 구분하는 방법 이외, 태그 id 와 class를 사용해 구분하기 위해 **선택자**라는 개념이 등장합니다.

1 선택자를 이용해 HTML 코드 탐색

CSS에서 스타일을 적용시키고 싶은 HTML 코드의 태그를 찾기 위해서 사용하던 문법이 바로 선택자였습니다. 아래의 예제 CSS에서 div와 #content .text와 #content .giga가 바로 선택자입니다. 예를 들어 #content .text은 HTML 코드에서 아이디가 'content'인 태그 안에 클래스가 'text'인 태그를 찾으라는 뜻을 가지고 있습니다.

```
div {
    margin: 50px;
}
#content .text {
    color: red;
}

#content .giga {
    color: blue;
}
```

그렇다면 BeautifulSoup 모듈에서 HTML 코드를 탐색할 때 위와 같은 선택자를 이용하면 원하는 태그들을 지정하여 검색할 수 있지 않을까요? select 메서드를 사용하면 입력한 선택자에 부합하는 태그들을 찾아내 리스트로 돌려줍니다.

```
# p 태그들을 가져옵니다.
print(bs.select('p'))
# 클래스가 giga 인 p 태그들을 가져옵니다.
print(bs.select('p.giga'))
# 아이디가 content 인 태그 내에 클래스가 world 인 태그들을 가져옵니다.
print(bs.select('#content .world'))
```

출력 결과

```
[<p class="text world">Hello, World!</p>, <p class="text giga">Hello, Giga Genie!</p>]
[<p class="text giga">Hello, Giga Genie!</p>]
[<p class="text world">Hello, World!</p>]
```


만약 첫 번째 결과만 가져오고 싶다면, select_one 메서드를 사용하면 됩니다.

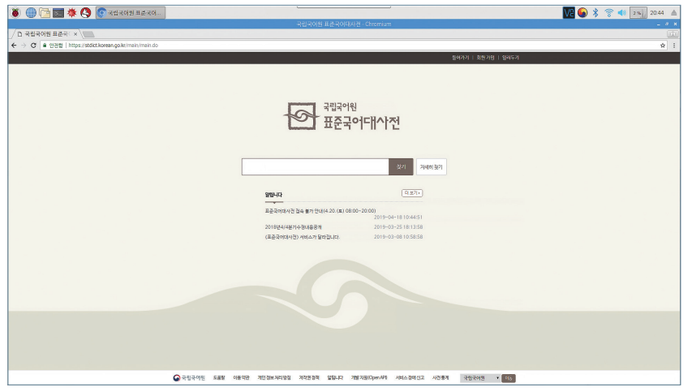
```
# p 태그들 중 첫 번째 결과를 가져옵니다.
print(bs.select_one('p'))
```

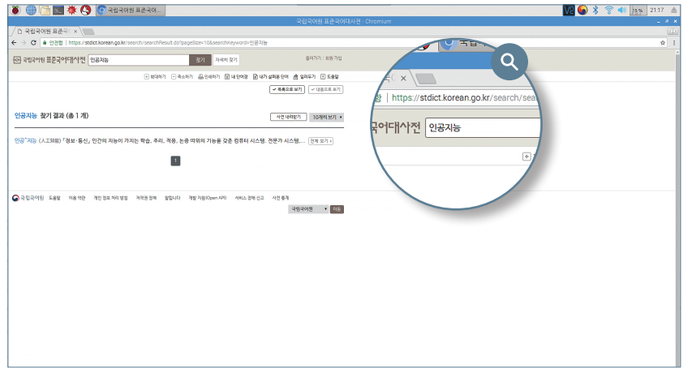
출력 결과

```
<p class="text world">Hello, World!</p>
```

3) 크롤링으로 국어 사전 만들기

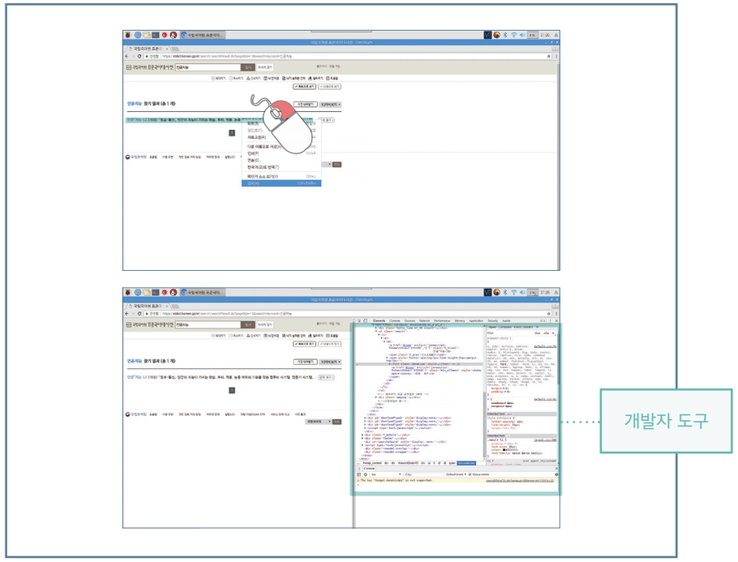
검색 결과의 HTML 코드를 받아와 파싱으로 원하는 단어의 뜻을 가져오는 프로그램을 만들어봅시다. 이 과정을 통해 어떤 식으로 파싱을 해야 하는지에 대한 전반적인 과정을 알아보고자 합니다. 우선 브라우저를 열어 'https://stdict.korean.go.kr'에 접속한 후 '인공지능'을 검색합니다.



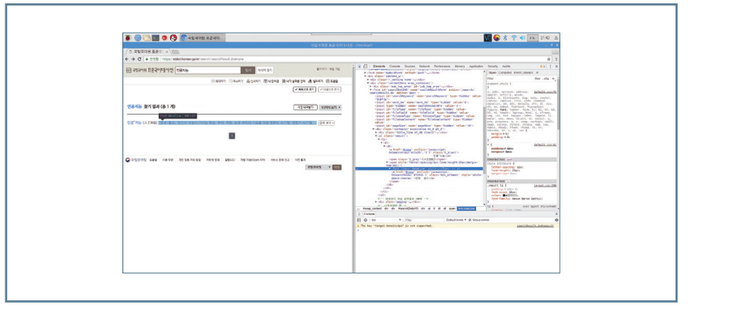


1 개발자 도구

검색 결과의 상세 설명에 마우스를 올려 오른쪽 클릭하면 나오는 옵션 중 검사를 클릭하면, 아래와 같은 화면을 확인할 수 있습니다. 화면에서 오른쪽 패널은 개발자 도구라고 불리며 HTML 코드의 태그 분석, 실시간 네트워크 캡처, 성능 측정, 메모리 분석 등 여러 가지 기능을 제공하지만, 우리는 태그 분석 기능을 사용해보겠습니다.

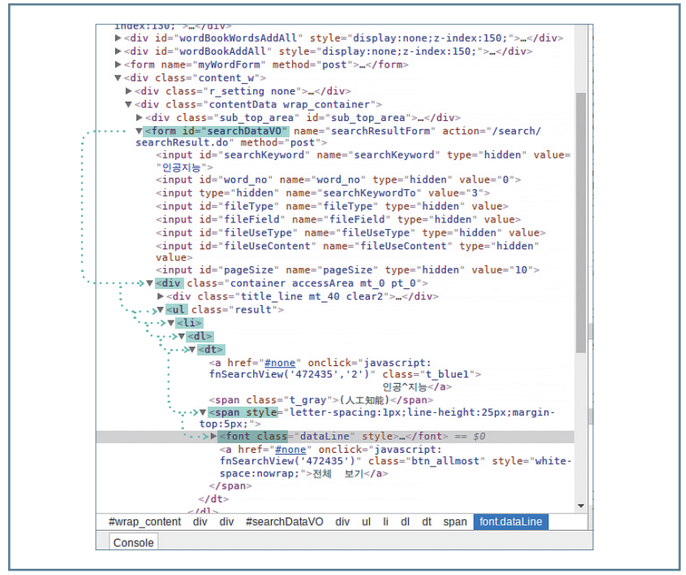


단축키 Ctrl + Shift + C를 함께 누르거나 개발자 도구의 왼쪽 상단의 마우스 아이콘을 누르고 분석하고 싶은 영역을 클릭하면 해당 영역의 HTML 코드를 확인할 수 있습니다. 우리는 '인공지능'이란 단어의 뜻을 가지고 있는 HTML 코드를 분석해 보겠습니다.



※ 코드 옆에 붙어있는 화살표의 줄 맞춤을 통해 태그가 어떤 태그(상위 태그)에 속해 있는지 알 수 있습니다.

2 HTML 코드 분석



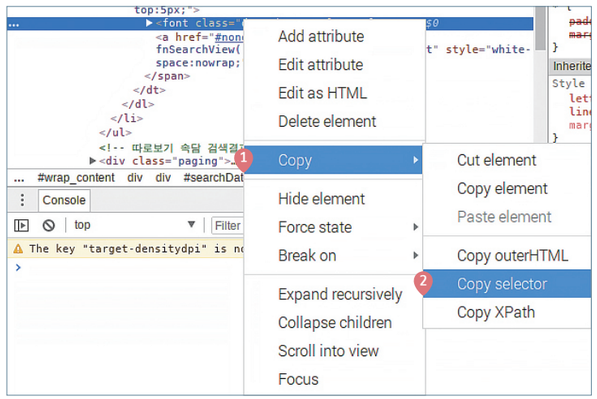
우리에게 최종적으로 필요한 것은 위 사진에 표시해 놓은 것처럼 font 태그입니다. font 태그를 찾아가는 수많은 방법 중 필자가 가장 많이 사용하는 방법으로 설명해보겠습니다. 저는 가장 먼저 id 태그를 찾습니다. 위 코드를 보면 id 태그는 form 태그에 속해 있는 searchDataV0입니다. 그다음은 div -> ul -> li -> dl -> dt -> span -> font 태그 순으로 이어지는 것을 확인할 수 있습니다. 이를 선택자로 표현하면 아래와 같이 표현할 수 있습니다.

```
#searchDataVO > div > ul > li > dl > dt > span > font
```

물론 앞에서 말했듯 이 방법은 필자가 사용하는 방법입니다. class, result 등 필자가 사용한 태그 이외 다른 태그를 사용하여 선택자를 구성하는 경우도 있습니다.

TIP

브라우저의 개발자 도구에서 원하는 태그의 선택자를 바로 알아낼 수 있는 방법도 있습니다. 원하는 부분에서 오른쪽 클릭을 한 후 1 Copy > 2 Copy selector 를 클릭하면 자동으로 선택자가 생성이 되어 복사가 됩니다.



생성된 선택자: #searchDataVO > div > ul > li:nth-child(1) > dl > dt > span:nth-child(3) > font

③ URL 패턴 확인

URL 창을 유심히 관찰하면서 '삼겹살'과 '꽃등심'을 검색해 봅시다. 메인화면에서는 별 다른 인자가 붙지 않았지만, 검색을 했을 경우에는 'pageSize'와 'searchKeyword' 두 개의 인자가 나타나는 것을 확인할 수 있습니다.

```
https://stdict.korean.go.kr/search/searchResult.do?pageSize=10&searchKeyword=삼겹살
https://stdict.korean.go.kr/search/searchResult.do?pageSize=10&searchKeyword=꽃등심
```

'pageSize'와 'searchKeyword'의 뜻을 가지고 우리는 각 인자가 어떤 역할을 하는지 대충 알 수 있습니다. pageSize는 한 페이지에 몇 개의 결과를 표시할 것인지를 결정하는 인자이고, searchKeyword은 검색 키워드를 결정하는 인자입니다.

```
https://stdict.korean.go.kr/search/searchResult.do?searchKeyword={검색키워드}
```

searchKeyword를 다른 단어로 변경해 보면 변경된 단어를 검색하는 것을 확인할 수 있습니다. 우리는 검색을 할 것이기 때문에 아래의 형식과 같이 URL을 선택할 수 있습니다.

4) Python으로 국어 사전 크롤링하기

선택자와 URL의 패턴을 찾았으니 BeautifulSoup 모듈을 이용해 실제로 크롤링을 해보도록 하겠습니다.

① 모듈 불러오기

필요한 모듈을 불러오는 작업으로 시작해보겠습니다. HTML을 불러오기 위한 requests 모듈과 HTML 코드를 탐색하기 위한 BeautifulSoup 모듈을 불러옵니다.

dictionary.py

```
import requests
from bs4 import BeautifulSoup
```

② GET 요청하기

검색 URL의 형식을 아래와 같이 입력하고 requests로 GET 요청을 보냅니다.

dictionary.py

```
url = "https://stdict.korean.go.kr/search/searchResult.do?searchKeyword=%s"
response = requests.get(url % '인공지능')
```

③ 태그 찾기

응답 코드가 200이라면 받아온 HTML 코드를 BeautifulSoup 모듈로 넘겨주고, 앞에서 만들어 놓은 선택자를 이용해 태그들을 찾아냅니다.

dictionary.py

```
if response.status_code == 200:
    html = response.text
    bs = BeautifulSoup(html, 'html.parser')
    dataLines = bs.select('#searchDataVO > div > ul> li > dl > dt > span > font'))
```

출력 결과

```
[<font class="dataLine">「정보·통신」 인간의 지능이 가지는 학습, 추리, 적응, 논증 따위의 기능을 갖춘 컴퓨터 시스템. 전문가 시스템, 자연 언어의 이해, 음성 번역, 로봇 공학, 인공 시각, 문제 해결, 학습과 지식 획득, 인지 과학 따위에 응용한다.≒에이아이.</font>]
```

위 코드를 실행해보면 '인공지능'이라는 단어를 검색하면 나오는 결과들을 성공적으로 받아온 것을 확인할 수 있습니다.

4 함수 만들기

다른 단어를 검색하기 위해 위 코드를 반복해서 적는 것보다 함수로 만들면 코드가 더 간결해지고 관리하기가 편해집니다. 이때까지 작성한 코드를 바탕으로 단어의 뜻들을 돌려주는 함수를 만들어 보겠습니다. Import를 제외한 이전 코드를 지우고 아래와 같이 함수로 만듭니다.

dictionary.py

```
def search_korean(keyword):
    url = "https://stdict.korean.go.kr/search/searchResult.do?searchKeyword=%s"
    response = requests.get(url % keyword)

    if response.status_code == 200:
        html = response.text
        bs = BeautifulSoup(html, 'html.parser')
        dataLines = bs.select('#searchDataVO > div > ul> li > dl > dt > span > font')
    else:
        return None

if __name__ == "__main__":
    print('인공지능의 뜻: ', search_korean('인공지능'))
    print('인터넷의 뜻: ', search_korean('인터넷'))
```

출력 결과

```
인공지능의 뜻:  [<font class="dataLine">『정보·통신』 인간의 지능이 가지는 학습, 추리, 적응, 논증 따위의 기능을 갖춘 컴퓨터 시스템. 전문가 시스템, 자연 언어의 이해, 음성 번역, 로봇 공학, 인공 시각, 문제 해결, 학습과 지식 획득, 인지 과학 따위에 응용한다.≒에이아이.</font>]
인터넷의 뜻:  [<font class="dataLine">『정보·통신』 전 세계의 컴퓨터가 서로 연결되어 정보를 교환할 수 있는, 하나의 거대한 컴퓨터 통신망.</font>]
```


5 태그 내부의 문자만 추출하기

출력 결과를 보면 <font class="dataLine">와 같이 우리에게 필요 없는 정보도 포함하고 있습니다. 태그 부분을 제외한 텍스트를 새로운 리스트 meanings에 넣어 돌려주는 부분을 추가하였습니다. BeautifulSoup 모듈의 select 메서드를 통해 검색된 태그는 Tag라는 클래스에 담겨 돌아옵니다. Tag라는 클래스 안에는 태그의 이름, 클래스, 아이디 등 여러 가지 속성이 있는데, 이 중에서 text라는 속성을 사용하면 태그 내부의 문자만 추출할 수 있습니다.

dictionary.py

```
def search_korean(keyword):
    url = "https://stdict.korean.go.kr/search/searchResult.do?searchKeyword=%s"
    response = requests.get(url % keyword)

    if response.status_code == 200:
        html = response.text
        bs = BeautifulSoup(html, 'html.parser')
        dataLines = bs.select('#searchDataVO > div > ul> li > dl > dt > span > font')
        meanings = list()
        for data in dataLines:
            meanings.append(data.text)
        return meanings
    else:
        return None
```

출력 결과

```
인공지능의 뜻:  ['『정보·통신』 인간의 지능이 가지는 학습, 추리, 적응, 논증 따위의 기능을 갖춘 컴퓨터 시스템. 전문가 시스템, 자연 언어의 이해, 음성 번역, 로봇 공학, 인공 시각, 문제 해결, 학습과 지식 획득, 인지 과학 따위에 응용한다.≒에이아이.']
인터넷의 뜻:  ['『정보·통신』 전 세계의 컴퓨터가 서로 연결되어 정보를 교환할 수 있는, 하나의 거대한 컴퓨터 통신망.']
```

</>

전체 코드

dictionary.py

```
import requests
from bs4 import BeautifulSoup

def search_korean(keyword):
    url = "https://stdict.korean.go.kr/search/searchResult.do?searchKeyword=%s"
    response = requests.get(url % keyword)

    if response.status_code == 200:
        html = response.text
        bs = BeautifulSoup(html, 'html.parser')
        dataLines = bs.select('#searchDataVO > div > ul> li > dl > dt > span > font')
        meanings = list()
        for data in dataLines:
            meanings.append(data.text)
        return meanings
    else:
        return None

if __name__ == "__main__":
    print('인공지능의 뜻: ', search_korean('인공지능'))
    print('인터넷의 뜻: ', search_korean('인터넷'))
```

1 227P
2 227P
3 227P-228P
5 229P
4 228P

05 사전과 코딩팩 연동하기

앞에서 우리는 국립국어원의 표준국어 대사전을 크롤링하여 단어의 뜻들을 돌려주는 함수(dictionary.py)를 만들었습니다. 이 함수를 이용하여 사용자가 코딩팩에게 단어의 뜻을 물으면 대답해주는 프로젝트를 진행해보겠습니다.

프로젝트는 아래 순서대로 진행됩니다.

1. 사용자가 궁금한 단어를 코딩팩에게 질문을 하고, 코딩팩은 사용자가 말하는 단어를 인식합니다
2. 국어 대사전에서 입력된 단어를 검색합니다.
3. 검색 결과가 없다면 뜻을 찾지 못했다 말하고, 검색 결과가 있다면 첫 번째 결과를 말합니다.

dictionary.py 파일에 음성출력을 위한 voice 모듈을 import 하고, main()이라는 이름을 가진 함수를 만들고 프로젝트 순서대로 코드를 작성해봅시다.

dictionary.py

```
import voice

def main():
    voice.speech('궁금하신 단어가 무엇인가요?')
    word = voice.get_text_from_voice()
    meanings = search_korean(word)
    if meanings == None or len(meanings) == 0:
        voice.speech('%s 에 관한 뜻을 찾지 못했어요.' % word)
    else:
        voice.speech('사전에서 찾은 검색결과입니다. %s' % meanings[0])


if __name__ == "__main__":
    main()
```

맨 처음 프로그램이 실행되면서 '궁금하신 단어가 무엇인가요?' 질문을 한 뒤 사용자로부터 단어를 인식해 입력받습니다. 그리고 searchKorean 함수를 이용하여 검색 결과 리스트를 돌려받습니다. 이때 검색 결과 리스트가 None 이거나 길이가 0일 경우(단어의 의미가 없는 경우)에는 '뜻을 찾지 못했어요.'라고 말하고, 검색 결과가 있다면 첫 번째 검색 결과를 말해줍니다.

</>

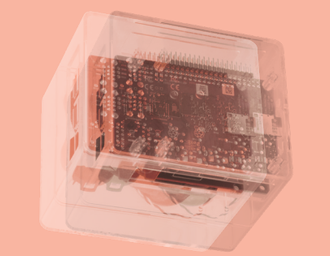
전체 코드

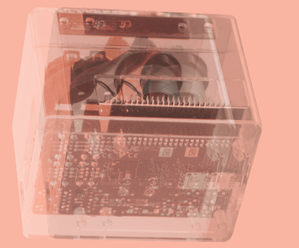
dictionary.py

```
import requests
from bs4 import BeautifulSoup

import voice

def search_korean(keyword):
     url = "https://stdict.korean.go.kr/search/searchResult.do?searchKeyword=%s"
     response = requests.get(url % keyword)

    if response.status_code == 200:
         html = response.text
         bs = BeautifulSoup(html, 'html.parser')
         dataLines = bs.select('#searchDataVO > div > ul> li > dl > dt > span >
font')
         meanings = list()
         for data in dataLines:
             meanings.append(data.text)
         return meanings
     else:
         return None


def main():
     voice.speech('궁금하신 단어가 무엇인가요?')
     word = voice.get_text_from_voice()
     meanings = search_korean(word)
     if meanings == None or len(meanings) == 0:
         voice.speech('%s 에 관한 뜻을 찾지 못했어요.' % word)
     else:
         voice.speech('사전에서 찾은 검색결과입니다. %s' % meanings[0])


if __name__ == "__main__":
     main()
```

1 231P

코딩팩으로
사물인터넷 구성하기

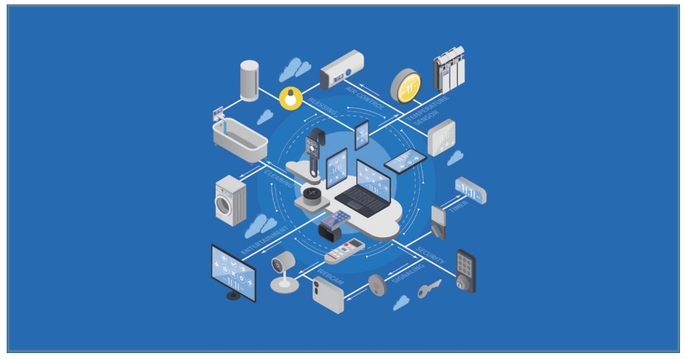




5

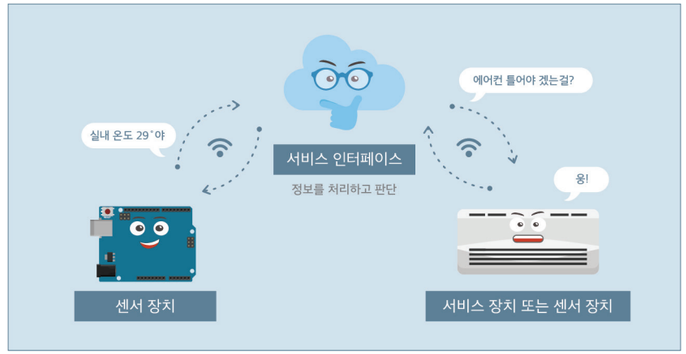
01
사물인터넷이란 무엇인가?

사물인터넷internet of Things 이란 사물 + 인터넷 즉, 각종 사물에 인터넷 통신 기능을 추가하여 사람과 사물, 사물과 사물 간의 정보를 상호 소통할 수 있게 하는 기술입니다. TV 광고를 보면 스마트폰으로 보일러, 에어컨을 조정하고 AI 스피커를 통해서 TV의 채널을 바꾸거나 물건을 구입하기도 하며 집 전등을 꺼달라고 요청하는 모습을 종종 볼 수 있습니다. 이처럼 사물인터넷을 이용해 기존에 사람이 직접 조작했던 사물(장치)에 인터넷 연결을 통하여 장소의 구애 없이 제어할 수 있습니다. 더 나아가 사물에 센서를 장착하여 센서에서 수집된 데이터로 빅데이터를 구축할 수 있고, 인공지능은 구축된 빅데이터를 기반으로 사용자가 신경 쓰지 않더라도 필요한 서비스를 알아서 제공해줍니다.



Internet of Things
사진출처<freepik 제공>

사물인터넷의 구성요소는 크게 센서장치, 서비스 인터페이스, 통신 네트워크, 보안으로 이루어져 있습니다. 센서 장치는 실시간으로 온도, 습도, 미세먼지, 위치, 모션, 생체신호 등과 같은 주위 환경의 정보를 측정합니다. 측정된 데이터는 통신 네트워크(이더넷, WI FI, 블루투스, 이동 통신)을 사용해 서비스 인터페이스로 전송하게 됩니다. 서비스 인터페이스는 전달받은 정보를 처리하고 판단하여 알맞은 서비스를 제공합니다. 이 모든 과정에서 센서의 데이터와 사용자의 정보는 인터넷을 사용하여 전달되기 때문에 보안에 취약할 수 있습니다. 그렇기 때문에 사물인터넷에서는 보안 역시 매우 중요합니다.



02
MQTT 프로토콜

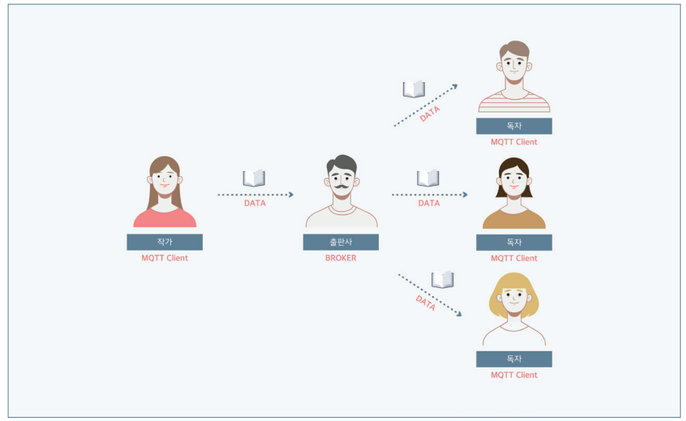
MQTTmessage Queuing Telemetry transport 프로토콜은 **M2M**machine to machine 통신이나 사물인터넷 통신에서 사용되는 메시지 프로토콜입니다. 주로 무선 환경에서 사용하기 때문에 배터리를 사용하고, 인터넷 연결의 신뢰성이 높지 않은 곳에서 사용되는 경우가 많기 때문에 MQTT 프로토콜은 이러한 환경을 고려하여 패킷의 크기를 줄여 가볍고 낮은 전력, 낮은 대역폭 환경에서 사용할 수 있도록 설계되었습니다.

1) MQTT 프로토콜 동작 구조

MQTT 프로토콜은 유튜브를 생각하면 쉽게 이해할 수 있습니다. 유튜브를 시청하다 마음에 드는 채널의 구독과 알림을 신청하면 영상이 업로드될 때 채널에서 업로드 알림을 받을 수 있습니다. MQTT 프로토콜도 유사한 동작 구조를 갖습니다. MQTT 프로토콜에 참여하는 클라이언트끼리는 **메시지 발행**Publish 과 **구독**Subscribe 을 할 수 있습니다. 클라이언트가 메시지를 **특정 채널**Topic 에 발행하면 해당 채널을 구독한 모든 클라이언트에게 메시지가 전달됩니다.

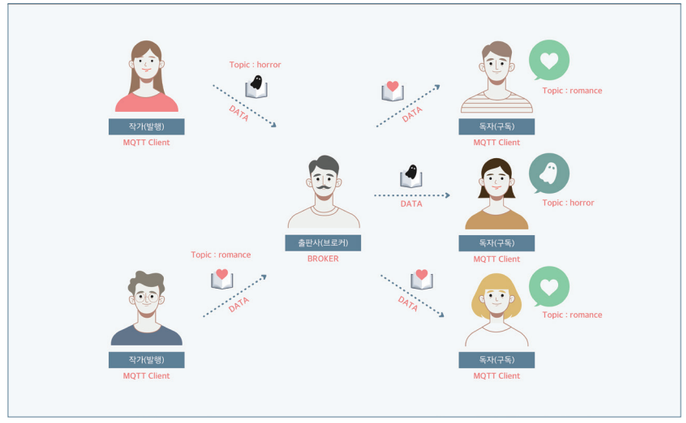
1 Broker

MQTT 프로토콜 자체는 메시지를 어떻게 보낼 것인지를 정의하는 규약일 뿐입니다. 실제 MQTT 프로토콜을 동작시키기 위해서는 클라이언트 간에 전달되는 데이터를 관리해줄 장치와 프로그램이 필요한데 이를 MQTT **브로커**Broker라고 합니다. 클라이언트 간에 통신할 때 바로 클라이언트로 데이터를 보내는 것이 아니라 데이터를 발신하는 클라이언트가 브로커로 데이터를 보내고 브로커는 각종 장치들이 보내주는 데이터를 수집하고 이걸 다시 필요한 클라이언트에게 재분배합니다.



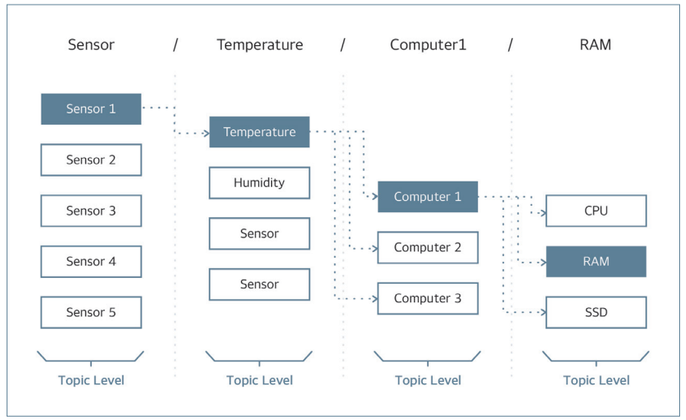
2 Publish/Subscribe

클라이언트와 브로커가 데이터를 주고받을 때는 **발행**Publish과 **구독**Subscribe 방식을 사용합니다. 클라이언트가 메시지를 특정 채널에 발행하면 해당 채널을 구독한 모든 클라이언트에게 메시지가 전달됩니다. 이때 사용하는 채널을 **토픽**Topic이라고 합니다.



3 Topic

토픽은 클라이언트가 메시지를 발행/구독할 때 사용되는 채널 혹은 경로입니다. 토픽은 문자열을 사용해서 만들며 컴퓨터의 폴더처럼 ' / '(슬래시)를 사용해서 하위 토픽을 추가할 수 있습니다. ' / '를 통해 나누어지는 토픽의 단위는 'Topic level'이라 합니다.



위 표에서 클라이언트가 Computor1의 RAM의 온도 값을 받아오기 위해서는 Sensor1/Temperature/Computer1/RAM와 같이 토픽을 구독해주면 토픽에서 **발행**publish 되는 데이터를 받아올 수 있습니다.

4 Topic wildcard

토픽은 두 가지 와일드카드라 불리는 문자를 사용해 동일한 Topic level을 한 번에 구독하거니 하위 Topic level을 한 번에 구독할 수 있습니다.

(1) 와일드카드 + (Single-level wildcard)

- + 문자는 단일 레벨 와일드카드를 구성하는데 사용됩니다. '+' 문자가 사용된 Topic level의 토픽을 전부 구독합니다.
- 'Sensor1/Temperature/+/RAM' - Sensor1/Temperature에서 Computer1, Computer2, Computer3에 있는 모든 RAM 토픽을 한 번에 구독할 수 있습니다.
- 'Sensor1/Temperature/Computer1/+' - Sensor1/Temperature/Computer1에서 CPU, RAM, SDD 토픽을 한 번에 구독할 수 있습니다.

(2) 와일드카드 # (Multi-level wildcard)

- # 문자는 멀티 레벨 와일드카드를 구성하는데 사용됩니다. '#' 뒤에 있는 하위 토픽을 전부 구독합니다.
- 'Sensor/Temperature/#' - Sensor/Temperature에 있는 각 Computer의 모든 토픽을 구독할 수 있습니다.

5 QoS

MQTT는 메세지를 구독/발행할 때 레벨에 따라서 데이터 전송의 중요도를 설정하는 QoS 기능을 제공하고 있습니다. 각각의 레벨에 따라서 전송하는 방법이 달라짐으로 필요한 서비스에 맞게 설정하여 사용할 수 있습니다.

QoS 0
데이터가 한 번만 전달되며, 전달 여부를 확인하지 않습니다.

QoS 1
데이터를 한 번 이상 전달합니다. 전달 여부를 확인하지 않기 때문에 중복 전송될 수도 있습니다.

QoS 2
데이터를 한 번만 전달하고, 전달 여부를 확인하기 때문에 높은 품질을 보장하지만 전달 속도가 느립니다.

03 MQTT 통신해보기

MQTT 프로토콜을 사용하기 위해서는 클라이언트가 최소 두 개 이상이 필요합니다. 우리는 현재 코딩팩, Python, PC 세 가지 클라이언트를 사용할 수 있습니다. 코딩팩에 브로커를 설치하고 각 클라이언트를 사용하여 MQTT 통신을 해보도록 하겠습니다. 이번 예제에서는 클라이언트 간에 동일한 네트워크를 사용하여 코딩팩, Python, PC 간 MQTT 통신을 사용합니다.

1) Mosquitto(MQTT 브로커) 설치

MQTT 통신을 사용하기 위해서는 앞서 설명했듯 브로커가 필요합니다. 다양한 방법으로 브로커를 만들고 사용할 수 있지만 본 교재에서는 라즈베리파이(코딩팩)에 Mosquitto라는 오픈소스 브로커를 설치하여 사용하도록 하겠습니다.

※ Mosquito 브로커 설치는 라즈베리파이(코딩팩)의 터미널에서 진행합니다.

1 패키지 인증키 다운로드 및 설치

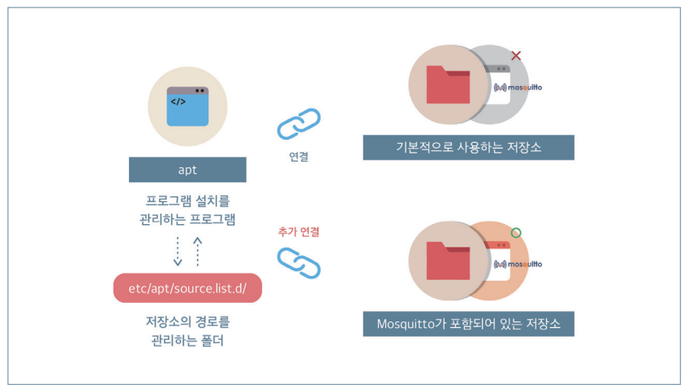
※ wget 명령어는 웹 서버로부터 파일을 다운로드할 때 사용하는 명령어입니다.
※ apt-key 명령어는 apt가 패키지를 인증하는데 사용하는 키를 관리하는 명령어로 키를 추가, 삭제, 업데이트할 수 있습니다.

Mosquitto 브로커 패키지는 외부에서 패키지를 설치하는 중에 해킹하는 것을 방지하기 위해 설치 과정에서 인증키를 같이 배포합니다. wget 명령어를 사용하여 인증키를 다운로드하고, apt-key 명령어를 사용하여 인증키를 등록합니다

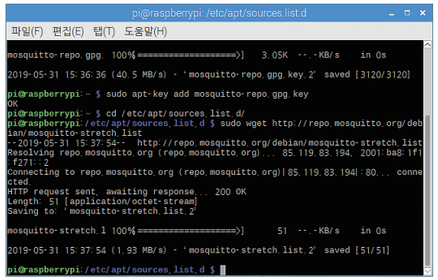
```
$ sudo wget http://repo.mosquitto.org/debian/mosquitto-repo.gpg.key
$ sudo apt-key add mosquitto-repo.gpg.key
```

2 Mosquitto 브로커 저장소에 패키지 등록

라즈베리파이가 사용하는 데비안 버전 Mosquitto 패키지를 사용하기 위해서는 apt가 기본적으로 사용하는 패키지 저장소 이외의 새로운 패키지 저장소가 필요합니다. apt 패키지 저장소를 관리하는 etc/apt/source.list.d/ 폴더로 이동하여 Mosquitto 패키지 저장소의 경로를 다운로드합니다.



```
$ cd /etc/apt/source.list.d/
$ sudo wget http://repo.mosquitto.org/debian/mosquitto-stretch.list
```

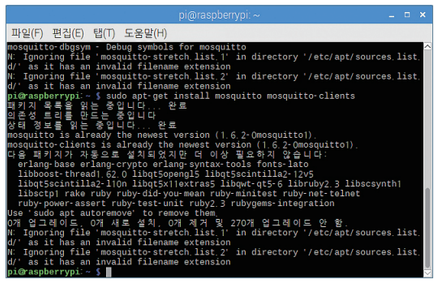


❸ Mosquitto 브로커 설치

※ 4장에서 공부했던 pip 명령어가 파이썬 패키지 설치 명령어라면, apt-get 명령어는 데비안 리눅스 계열에서 사용되는 패키지 설치 명령어입니다.

apt-get 명령어를 사용하여 Mosquitto 브로커를 설치합니다.

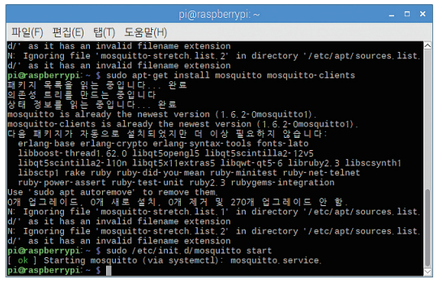
```
$ cd ~
$ sudo apt-get update
$ sudo apt-get install mosquitto mosquitto-clients
```



❹ Mosquitto 브로커 서버를 활성화

다운받은 mosquitto 브로커 서버를 사용하기 위해 서버를 활성화합니다.

```
$ sudo /etc/init.d/mosquitto start
```



2) 라즈베리파이(코딩팩) 터미널을 사용하여 MQTT 통신하기

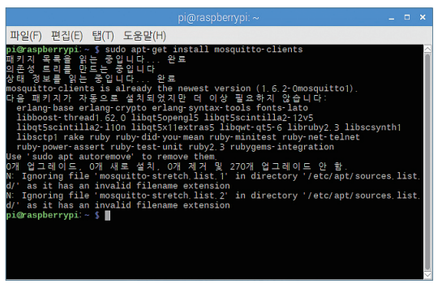
라즈베리파이(코딩팩) 터미널에서 MQTT 통신을 해보도록 하겠습니다. 주의할 점은 터미널 창을 구독 클라이언트, 발행 클라이언트 각각 한 개씩을 사용한다는 것입니다. 각 터미널 창은 다른 프로그램으로 다른 역할을 수행합니다.

❶ Mosquitto 클라이언트 패키지를 다운로드

※ 앞에서 이미 Mosquitto 브로커를 설치하면서 클라이언트도 설치했기 때문에 바로 명령을 입력해 사용할 수 있습니다.

터미널에서는 apt-get 명령어를 사용하여 간단하게 Mosquitto 클라이언트 패키지를 다운로드 할 수 있습니다.

```
$ sudo apt-get install mosquitto-clients
```



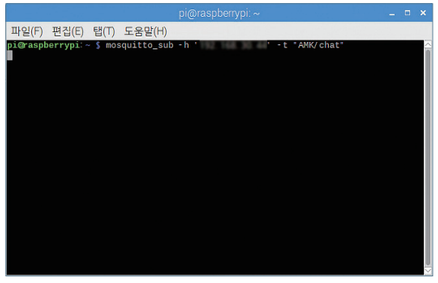
② 메시지 발행, 구독

터미널에서는 발행(Pub), 구독(Sub) 명령어를 이용하여 MQTT 통신을 할 수 있습니다. 이때 발행, 구독은 서로 다른 클라이언트에서 실행됩니다. 즉, 다른 터미널 창을 사용합니다.

구독 명령어
브로커를 라즈베리파이(코딩팩)에 설치했기 때문에 브로커 IP주소는 라즈베리파이(코딩팩)의 IP주소와 같습니다. 토픽은 문자와 '/'를 사용하여 원하는 문자열을 구독해줍니다.

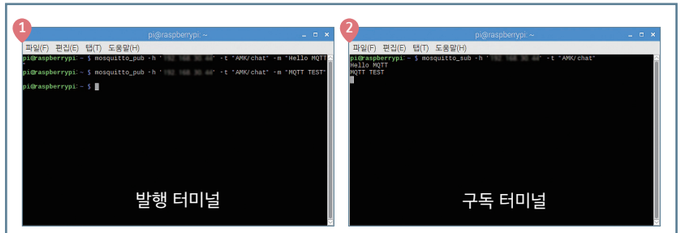
```
$ mosquitto_sub -h '브로커 IP주소' -t “토픽”
  예) mosquitto_sub -h 'xxx.xxx.xx.xx' -t "AMK/chat"
                       : 사용자 IP 주소
```

※ 구독 명령이 실행되면 사용자가 종료할 때 까지 메세지를 수신합니다. 종료할 때에는 Ctrl + c를 눌러줍니다.



발행 명령어
메시지를 발행할 때 구독에서 사용한 토픽과 같은지 꼭 확인합니다.

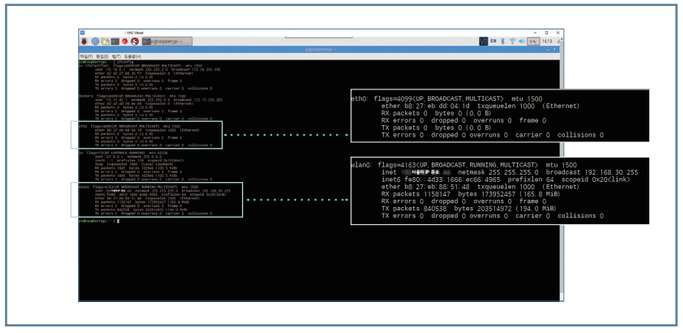
```
$ mosquitto_pub -h '브로커 IP주소' -t "토픽" -m "전달할 메시지"
예) mosquitto_pub -h 'xxx.xxx.xx.xx' -t "AMK/chat" -m "Hello MQTT"
                     : 사용자 IP 주소
```



③ IP주소 확인 방법

IP주소를 확인하기 위해서 터미널 창에 ifconfig 명령어를 입력합니다. 이 명령어는 연결된 네트워크의 구성 정보를 알려주는 명령어입니다.

코딩팩이 와이파이를 사용해 네트워크에 연결되었다면 wlan0, 랜선을 사용해 네트워크에 연결되었다면 eth0에서 IP주소를 확인합니다.



와이파이를 사용해 네트워크에 연결된 코딩팩

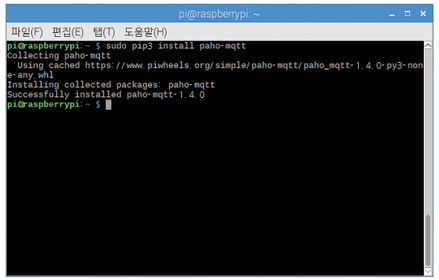
3) python을 사용하여 MQTT 통신하기

python에서는 paho-mqtt 라이브러리를 사용하여 MQTT 클라이언트를 사용할 수 있습니다.

① paho-mqtt 라이브러리 다운로드

pip 명령어를 사용하여 paho-mqtt 라이브러리를 다운로드합니다.

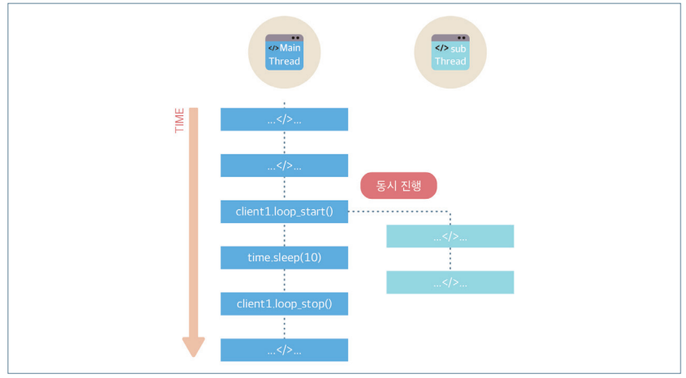
```
$ sudo pip3 install paho-mqtt
```



② MQTT 클라이언트 구독

Mqtt_sub.py 파일에 작성된 소스코드는 지금까지 우리가 알던 방식과 조금 다른 방식으로 동작합니다. 프로그램은 아래 그림과 같은 흐름으로 동작합니다. 여기서 **스레드**Thread 라는 개념이 새롭게 등장합니다.

스레드는 프로그램들이 실행되는 흐름의 단위를 뜻합니다. 일반적으로 프로그램은 하나의 메인 스레드로 동작하지만, 환경에 따라 하나 이상의 스레드가 동시에 실행될 수 있습니다. 아래에서 사용할 MQTT 클라이언트 구독 프로그램도 두 개의 스레드가 동작합니다. 다음 그림과 소스코드를 함께 보면서 설명하도록 하겠습니다.



왼쪽은 main 스레드이고 오른쪽은 구독 스레드라고 할 때 프로그램을 처음 실행시키면 main 스레드가 실행되면서 MQTT 통신에 필요한 모듈을 사용하여 브로커에 연결합니다. 그다음 구독 스레드에서 전달하는 메시지를 출력해주는 콜백 함수(on_message)를 설정합니다. Main 스레드에서 client1.loop_start()가 실행되면서부터 구독 스레드는 다른 클라이언트에서 발행되는 메시지가 전달되는 것을 기다리게 됩니다. 하지만 Main 스레드가 동작이 끝나게 되면 프로그램이 종료되면서 다른 스레드 또한 동작이 종료됩니다. 구독 스레드에서 메시지를 받아올 수 있는 시간이 필요하기 때문에 Main 스레드는 동작을 지연시켜주는 time 모듈의 sleep 함수를 사용하여 기다려준 후에 구독 스레드를 종료 후 브로커와의 연결을 끊고 프로그램을 종료합니다.

Mqtt_sub.py

```
import paho.mqtt.client as mqtt
import time

broker_address="xxx.xxx.xx.xx"

# subscriber 콜백함수
def on_message( client , userdata , message ):
    # 콜백함수에 들어오는 데이터를 변환하여 토픽과 메세지를 출력합니다.
    print("message received " ,str(message.payload.decode("utf-8")))
    print("message topic=",message.topic)

def main():
    client1 = mqtt.Client() # MQTT 라이브러리의 Client 인스턴스를 생성합니다.
    client1.connect(broker_address, 1883) # connect() 메서드를 사용해서 브로커에 연결해줍니다.
    client1.on_message = on_message # 구독한 토픽에 발행되는 메시지를 처리해주는 콜백함수를 설정합니다.

    client1.subscribe("AMK/chat") # 원하는 토픽을 구독해줍니다.
    print("Topic Subscribe")

    client1.loop_start() # 구독한 토픽에서 데이터를 받기 위해서 무한루프를 시작합니다.
    time.sleep(10) # 10 초 딜레이를 사용합니다.
    client1.loop_stop() # 무한루프를 종료합니다.

    client1.disconnect() # 브로커와 연결을 끊습니다.

if __name__ == "__main__":
    main()
```

❸ MQTT 클라이언트 발행

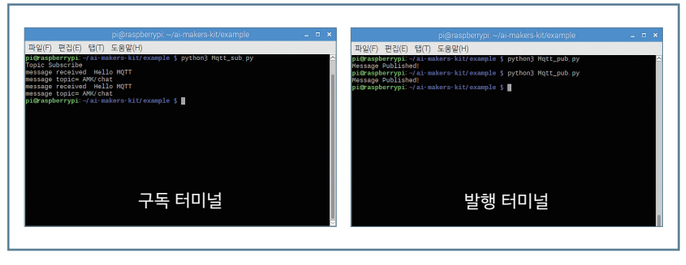
라이브러리를 사용하여 브로커의 IP주소를 통해 클라이언트로 접속하고 publish() 함수를 통해 “Hello MQTT” 메시지를 토픽에 발행합니다.

Mqtt_sub.py

```
import paho.mqtt.client as mqtt

broker_address = "xxx.xxx.xx.xx" # 브로커 IP 상수를 선언하여 IP 주소를 입력합니다.

def main():
    client1 = mqtt.Client()  # MQTT 라이브러리의 Client 인스턴스를 생성합니다.
    client1.connect(broker_address, 1883)
    # MQTT의 Client 인스턴스의 connect() 메서드를 사용해서 브로커에 연결합니다.
    # 이때 1883은 MQTT 브로커의 기본 포트 번호입니다.
    client1.publish("AMK/chat", "Hello MQTT") # 토픽에 메시지 발행합니다.
    print("Message Published!")
    client1.disconnect() # MQTT 브로커와 연결을 끊기위해 disconnect() 메서드를 사용합니다.

if __name__ == "__main__":
    main()
```

❹ 프로그램 실행

MQTT 클라이언트 발행과 구독은 아래 사진과 같이 서로 다른 터미널 창에서 실행됩니다. 그 이유는 앞에서도 설명한 것처럼 서로 다른 클라이언트에서 발행과 구독이 실행되어야 하기 때문입니다.

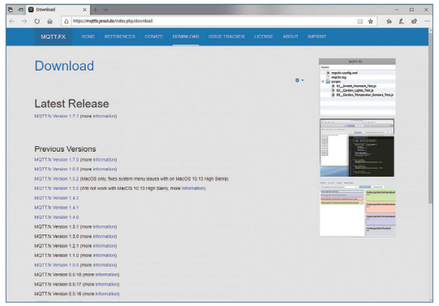


4) PC를 사용하여 MQTT 통신하기

PC에서는 ‘MQTT.fx’라는 클라이언트를 사용하여 MQTT 통신을 해볼 수 있습니다.

❶ ‘MQTT.fx’ 다운로드

링크‘https://mqttfx.jensd.de/index.php/download’로 접속해 최신 버전 ‘MQTT.fx’ 클라이언트를 사용하고 있는 운영체제에 맞춰 다운로드 합니다. (현재 1.7.1)



❷ 브로커 정보 설정

프로그램 실행 후 설정 화면으로 들어가 브로커 정보를 아래 내용과 같이 변경합니다.

❶ 클라이언트에서 사용할 브로커 프로파일명(Profile Name)을 정합니다. 이때 이름은 원하는 것으로 사용해도 무관합니다.

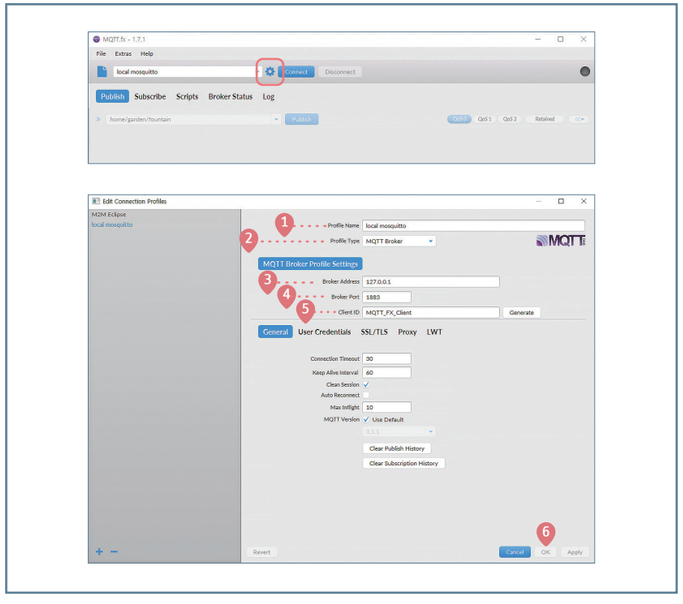
❷ Profile Type은 MQTT Broker로 설정합니다.

❸ Broker Address에 **IP주소** 또는 **도메인 주소**를 입력합니다. 이번 예제에서는 라즈베리파이(코딩팩)에 브로커를 설치했기 때문에 라즈베리파이(코딩팩)의 IP주소를 사용합니다.

❹ Broker Port는 기본값인 1883으로 설정합니다.

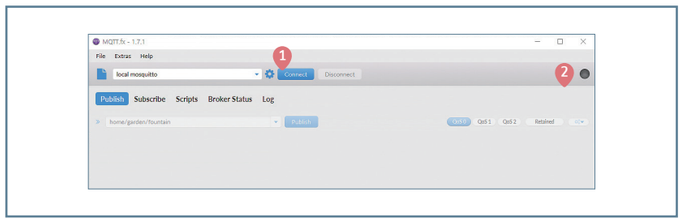
❺ Client ID는 사용처에 맞게 설정합니다. 다만 같은 브로커를 사용하는 클라이언트끼리는 Client ID를 중복해서 사용할 수 없습니다.

❻ 모든 설정을 수정이 끝나면 OK 버튼을 눌러 변경된 내용을 적용합니다.



3 브로커 연결

Connect 를 눌러 클라이언트와 브로커를 연결합니다. 클라이언트가 정상적으로 브로커와 연결되면 오른쪽 초록 신호(⚫ > 🟢)가 활성화됩니다.

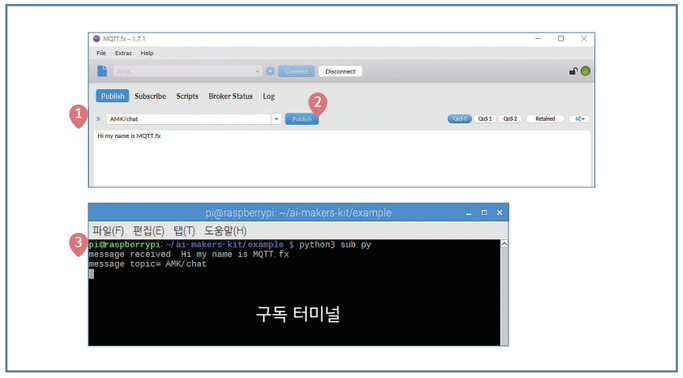


⚠ 연결이 원활하게 되지 않는 경우 다음과 같은 문제를 확인하세요.

1. 브로커와 클라이언트가 서로 다른 공유기를 사용하고 있진 않나요?
2. 브로커가 실행되고 있나요?
3. 동일한 Client ID가 브로커에 이미 접속되어 있나요?

4 메시지 발행

1 토픽과 메시지를 입력하고 2 Publish (발행)버튼을 누르면 메시지를 발행할 수 있습니다. 이때 동일 한 토픽을 구독한 다른 클라이언트는 3 발행된 메시지를 확인할 수 있습니다.

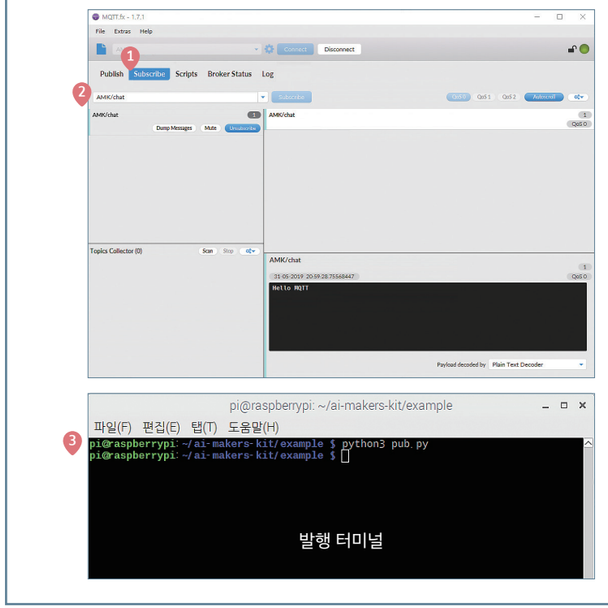


PC에서 MQTT 발행
(Hi my name is MQTT.fx)

파이썬에서 MQTT 구독

5 메시지 구독

'MQTT.fx' 프로그램 카테고리 중 1 Subscribe 를 선택해 2 원하는 토픽을 구독하면 다른 클라이언트가 발행한 3 메시지를 확인할 수 있습니다.



PC에서 MQTT 구독

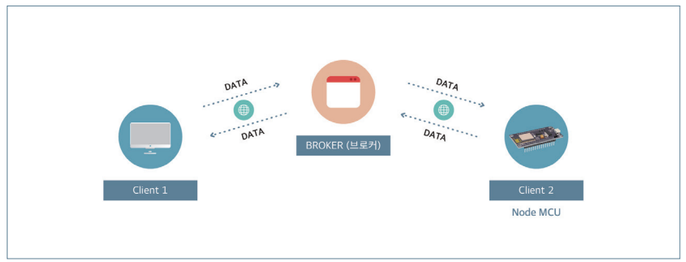
파이썬에서 MQTT 발행

04 와이파이 개발보드와 MQTT 통신

'NodeMCU'라는 와이파이 개발 보드를 사용하여 사물인터넷 구성요소 중 '센서장치'를 구현해 보도록 하겠습니다. 센서장치는 센서에서 측정되는 데이터를 네트워크를 통해 MQTT 브로커로 보내고, 브로커가 보내는 데이터를 받는 하나의 클라이언트로도 사용됩니다.

1) NodeMCU에 대해 알아보기

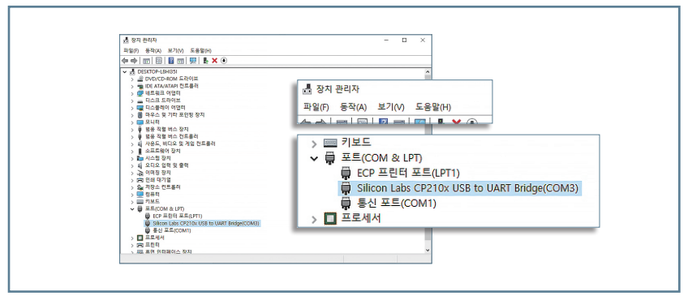
NodeMCU는 아두이노에 WiFi 기능이 추가되었다고 생각하면 조금 쉽게 이해할 수 있습니다. 아두이노를 사용하여 블루투스나 WiFi와 같은 통신 기능을 사용하기 위해서는 추가 부품이 필요합니다. 하지만 NodeMCU는 추가 부품 없이 WiFi를 사용할 수 있다는 장점을 가지고 있습니다. 또한 아두이노와 동일한 IDE를 사용하기 때문에 많은 오픈소스와 라이브러리를 쉽게 사용할 수 있습니다.



2) IDE 설치하기

❶ USB 드라이버 설치

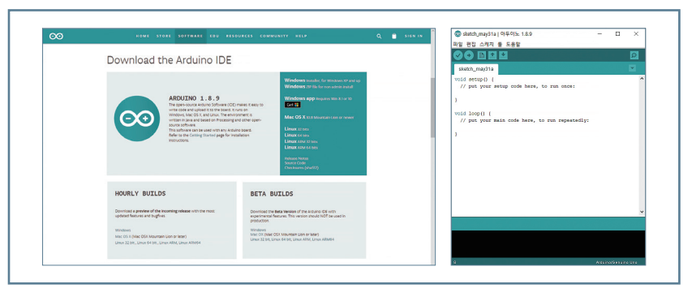
PC가 NodeMCU를 인식하기 위해서는 USB 드라이버를 설치해주어야 합니다. 아두이노 IDE를 설치할 때 FT232 드라이버를 설치해주지만 NodeMCU에 장착되어있는 CP2102칩을 사용하기 위해서는 추가로 드라이버를 설치해주어야 합니다. NodeMCU를 PC에 연결하면 자동으로 드라이버 설치를 시작합니다. 하지만 만약 그렇지 않다면 수동으로 드라이버를 설치해주어야 합니다. 수동으로 드라이버를 설치해야 하는 경우 'https://www.silabs.com/products/development-tools/software/usb-to-uart-bridge-vcp-drivers'사이트에서 드라이버를 다운로드 후 설치합니다.



드라이버가 정상적으로 설치되었다면 위 사진과 같이 장치 관리자에서 연결된 포트를 확인할 수 있습니다.

❷ 아두이노 IDE 설치

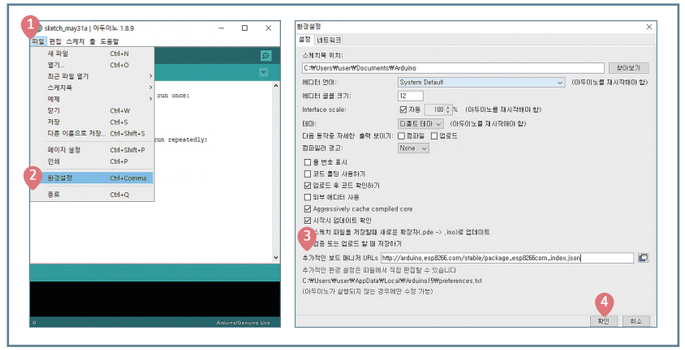
NodeMCU는 원래 Lua라는 프로그램 언어를 사용하도록 개발되었지만, 아두이노 IDE 개발 환경이 지원되면서부터 많은 사람이 아두이노 IDE를 사용해 프로그램을 개발하고 있습니다. 본 교재에서도 아두이노 IDE를 사용하도록 하겠습니다. 'https://www.arduino.cc/en/Main/Software' 사이트에서 아두이노 IDE를 운영체제에 맞춰 다운로드 후 설치합니다.



(좌) 아두이노 IDE 다운로드 페이지
(우) IDE 실행 화면

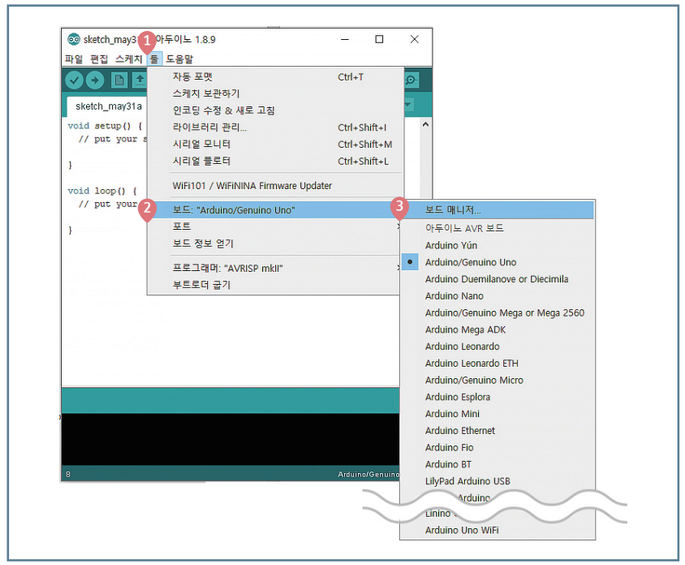
3 NodeMCU 개발환경 설정

상단 탭에서 ① 파일 > ② 환경설정 으로 이동합니다. 환경설정 에서 ③ '추가적인 보드 매니저 URLs'에 NodeMCU 보드 설정 URL을 입력 후 ④ 확인 을 눌러서 변경된 내용을 저장합니다.

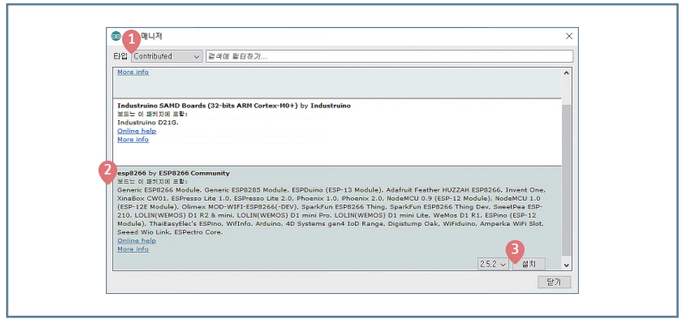


추가적인 보드 매니저 URLs http://arduino.esp8266.com/stable/package_esp8266com_index.json

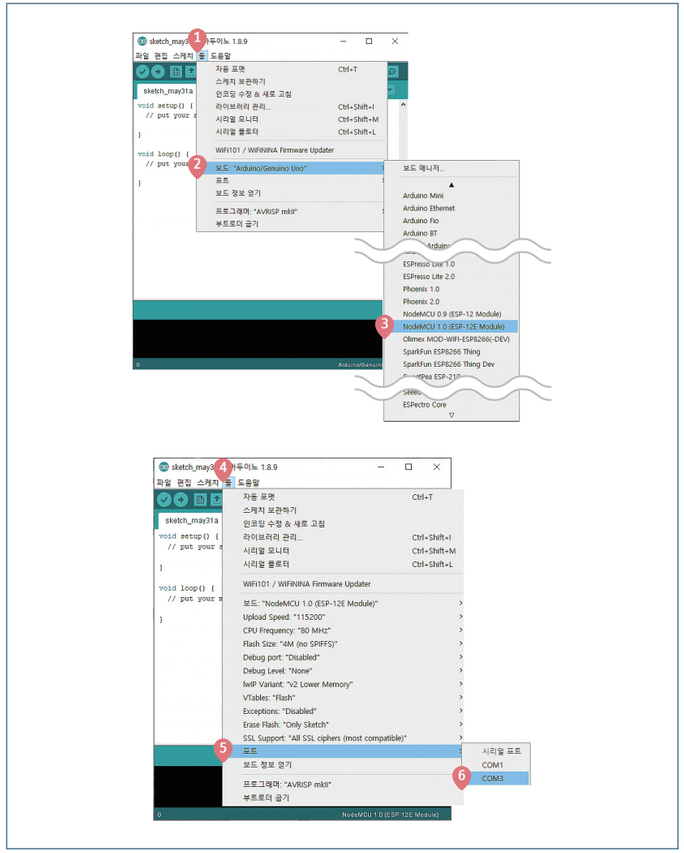
IDE 상단 탭에서 ① 툴 > ② 보드 > ③ 보드 매니저 로 이동합니다.



보드 매니저 타입을 ① 'Contributed'로 변경하고, 아래 항목 중 ② 'esp8266 by ESP8266 Community' 항목을 찾아 ③ 설치 버튼을 눌러 설치합니다.



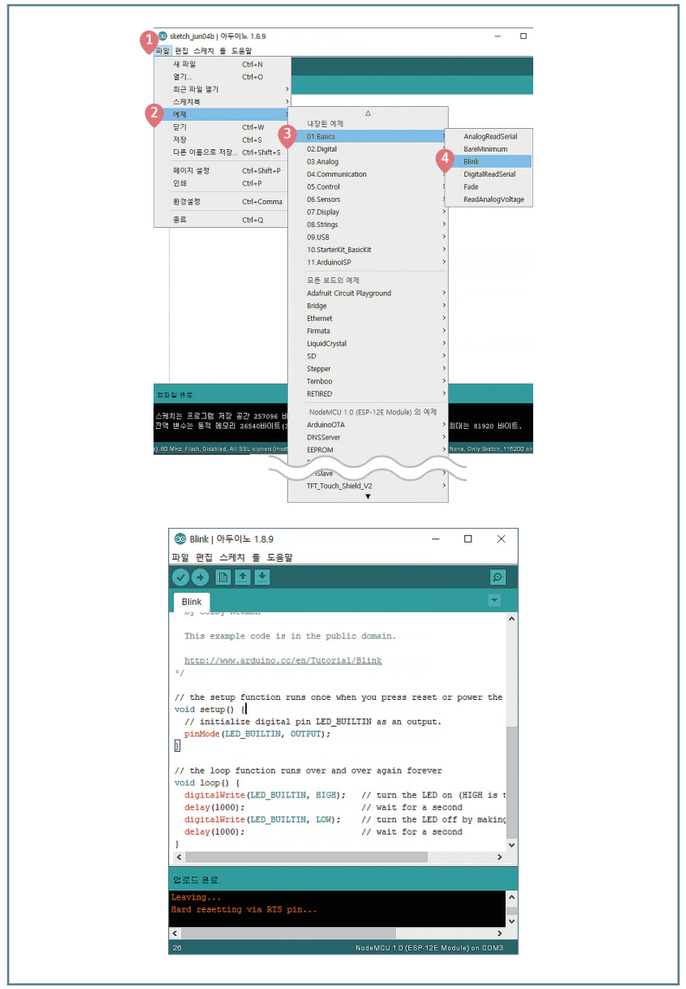
설치가 완료되면 ① 툴 > ② 보드 에서 보드 를 ③ NodeMCU 1.0으로 변경하고, ④ 툴 > ⑤ 포트 에서 포트 를 ⑥ NodeMCU가 잡혀있는 포트 번호로 변경합니다.



3 Blink 예제를 이용해 업로드 테스트하기

아두이노에 내장된 예제 blink를 이용해 프로그램이 NodeMCU에 업로드되는지를 확인해 보도록 하겠습니다. IDE 상단 옵션 중 1 파일 > 2 예제 > 3 01.Basic > 4 Blink 를 열어 NodeMCU에 소스코드를 업로드합니다. 업로드가 완료된 후 NodeMCU의 LED가 1초 주기로 점멸하는지 확인합니다. 만약 정상적으로 동작한다면 위에서 해주었던 NodeMCU 설정 작업이 완벽하게 진행된 것입니다.

※ 소스코드를 업로드 할 때에는 버튼을 눌러줍니다.

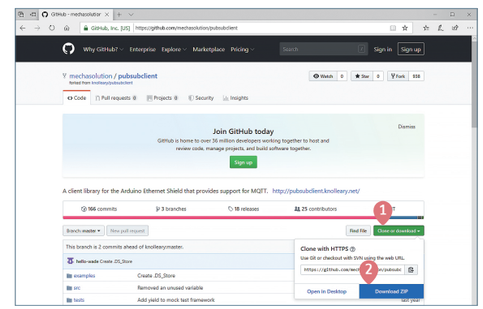


4 NodeMCU를 사용하여 MQTT 통신하기

NodeMCU 설정과 업로드 테스트가 완료되었다면 아두이노 IDE에서 사용할 수 있는 MQTT 라이브러리인 'PubSubClient'를 사용하여 라즈베리파이의 MQTT 브로커와 통신을 해보겠습니다.

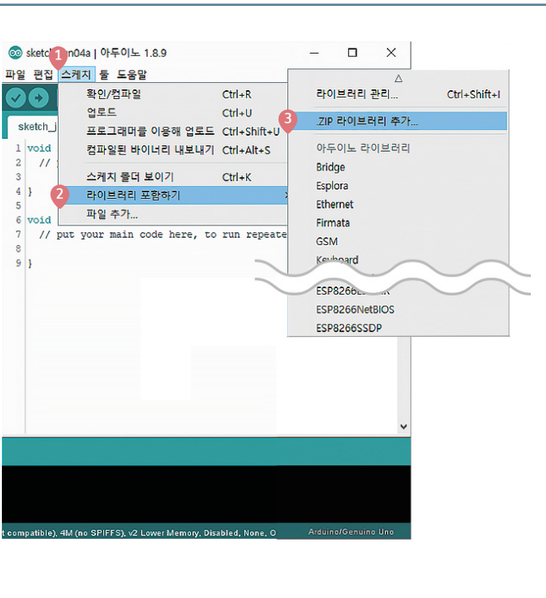
(1) PubSubClient 라이브러리 설치

'https://github.com/mechasolution/pubsubclient' 링크(GitHub)로 접속해 PubSubClient 라이브러리 ZIP파일을 다운로드 합니다.



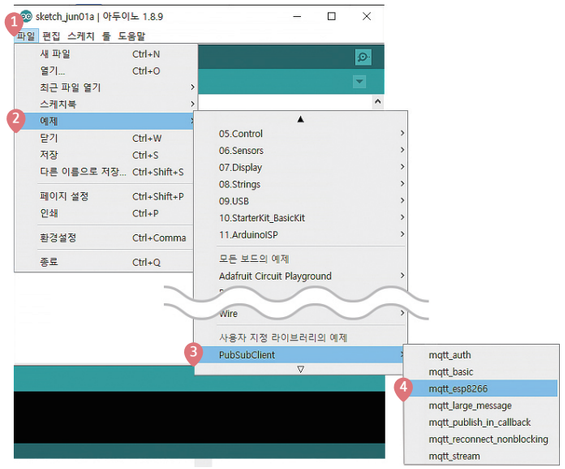
다운로드 받은 ZIP파일을 아두이노 라이브러리에 추가합니다. 라이브러리 추가는 1 스케치 > 2 라이브러리 포함하기 > 3 .ZIP 라이브러리 추가 에서 할 수 있습니다.

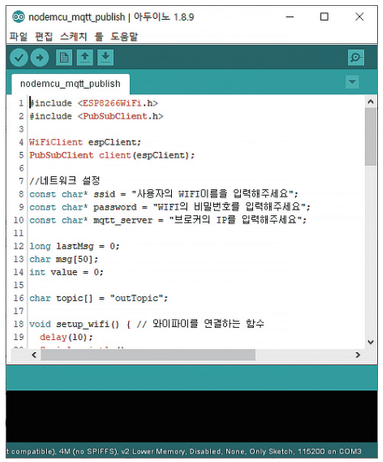
※ 다운로드한 ZIP 파일 압축을 풀지 않은 상태로 아두이노에서 라이브러리를 추가합니다.



(2) NodeMCU로 메시지 발행(Pub)

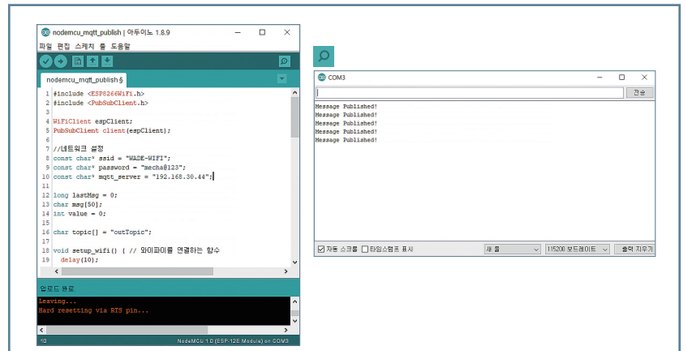
PubSubClient 라이브러리를 사용하여 라즈베리파이의 MQTT 브로커에게 메시지를 발행하기 위해 PubSubClient 라이브러리 예제 중 nodemcu_mqtt_publish를 불러옵니다. nodemcu_mqtt_publish 예제는 ① 파일 > ② 예제 > ③ PubSubClient > ④ nodemcu_mqtt_publish 경로를 통해 불러올 수 있습니다.





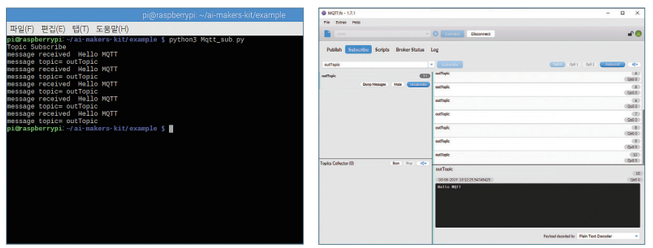
MQTT 통신을 하기위해 'nodemcu_mqtt_publish' 예제의 8, 9, 10, 16번 줄에 WiFi명, WiFi 비밀번호, 브로커 IP주소, 토픽을 입력 후 소스코드를 업로드합니다.

업로드가 완료되면 시리얼 모니터를 이용해 진행 상황을 확인할 수 있습니다. 제대로 메시지가 발행되고 있다면 Message Published!가 시리얼 모니터 ()에 출력됩니다. 또한 다른 클라이언트에서 동일한 토픽을 구독하고 있다면 NodeMCU에서 발행되는 메시지를 확인할 수 있습니다.



NodeMCU에서 MQTT발행

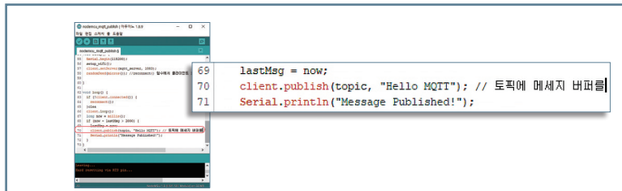
Publish Client



다른 클라이언트에서 MQTT구독

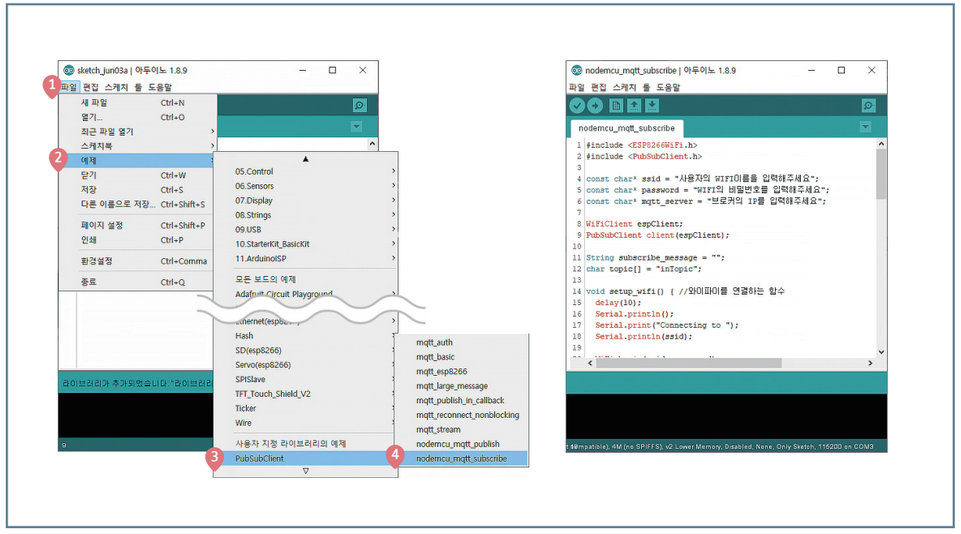
Subscribe Client

'nodemcu_mqtt_publish' 예제의 소스코드 70번째줄 client.publish( ) 함수의 인자 값을 변경 하면 발행되는 메시지를 변경할 수 있습니다.

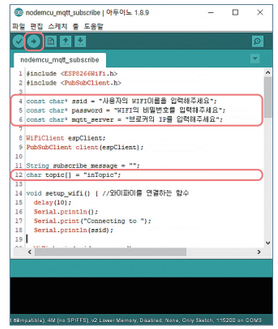


(3) NodeMCU로 메시지 구독(Sub)

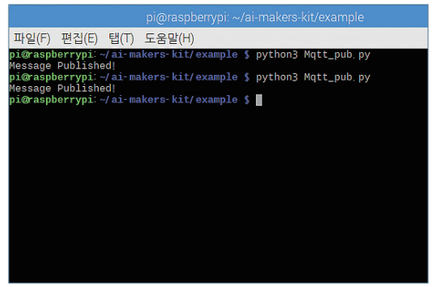
PubSubClient 라이브러리를 사용하여 라즈베리파이의 MQTT 브로커에게 메시지를 구독하기 위해 PubSubClient 라이브러리 예제 중 nodemcu_mqtt_subscribe를 불러옵니다. nodemcu_mqtt_subscribe 예제는 1 파일 > 2 예제 > 3 PubSubClient > 4 nodemcu_mqtt_subscribe 경로를 통해 불러올 수 있습니다.



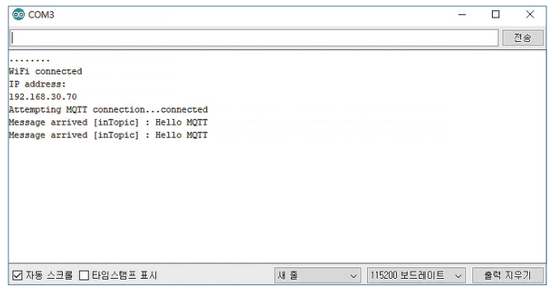
MQTT 통신을 하기위해 'nodemcu_mqtt_subscribe' 예제의 4, 5, 6, 12번 줄에 WiFi명, WiFi 비밀번호, 브로커 IP주소, 토픽을 입력 후 소스코드를 업로드합니다.



업로드가 완료되면 시리얼 모니터를 이용해 진행상황을 확인할 수 있습니다. 같은 토픽을 이용하고 있는 다른 클라이언트가 브로커로 발행한 메시지를 구독할 수 있습니다.



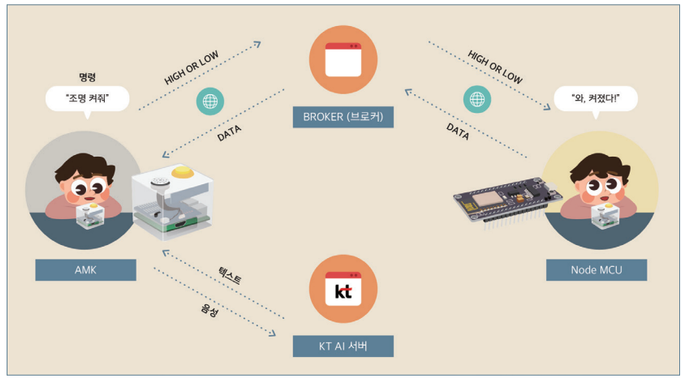
코딩팩 Client (발행)



NodeMCU Client (구독)

05 코딩팩과 NodeMCU 연동하기

우리는 지금까지 굉장히 많은 내용을 해보았습니다. 이번에는 앞서 배운 내용을 토대로 코딩팩과 센서장치(NodeMCU)가 데이터를 주고받는 실제 사물인터넷 환경을 구축해보는 프로젝트를 함께 해보도록 하겠습니다. 우리가 구축할 사물인터넷 환경에서 코딩팩과 NodeMCU의 역할은 아래와 같습니다.



본격적인 내용에 들어가기 앞서 아래 내용을 꼭 확인하고 진행하시길 부탁드립니다.

1. 각 장치(코딩팩, NodeMCU)는 동일한 공유기에 연결되어야 합니다.
2. 라즈베리파이에 Mosquitto 브로커가 설치되어 있어야 합니다.
3. NodeMCU 사전 준비(개발환경 설정)가 되어 있어야하고, 동작 테스트가 완료되어야 합니다.

1) 코딩팩으로 NodeMCU 디지털 출력 제어하기

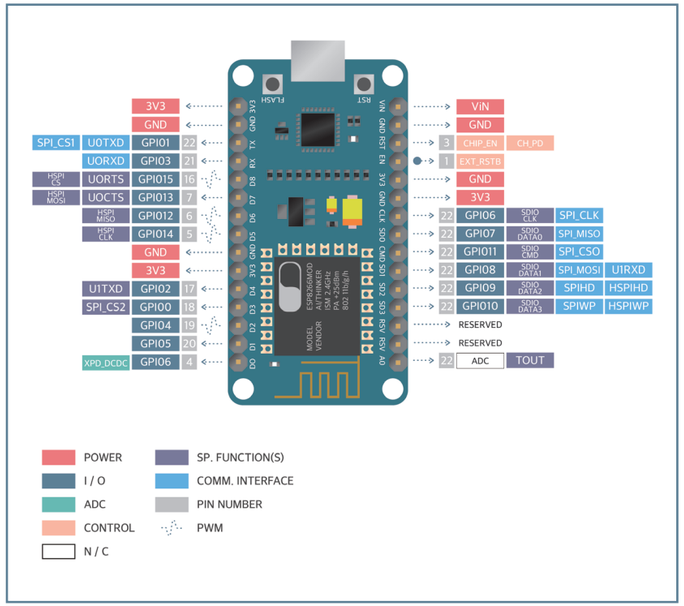
코딩팩을 호출해 “조명 켜줘” 혹은 “조명 꺼줘”라는 음성명령에 따라 NodeMCU 디지털 핀에 연결된 LED의 ON/OFF를 제어하는 프로젝트를 해보도록 하겠습니다.

프로젝트는 아래 순서로 진행됩니다.
1. NodeMCU에서 토픽을 지정하여 구독합니다.(토픽 - AMK/Control/node1)
2. 코딩팩에서는 음성 명령에 따라서 토픽에 약속된 메세지를 발행합니다.
예)“조명 켜줘”라고 인식되면 ‘AMK/Control/node1’토픽에 “HIGH”라는 메시지를 발행합니다.
3. NodeMCU는 지정한 토픽에서 약속한 메시지가 들어오면 디지털 핀을 제어합니다.

2) NodeMCU

1 아날로그 회로 구성

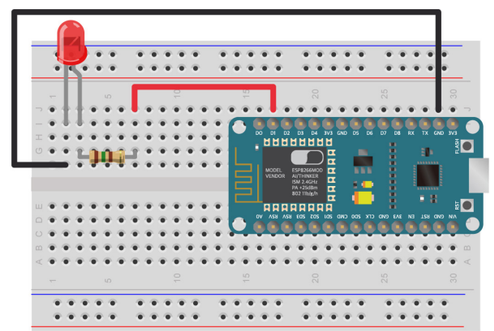
NodeMCU 각각 핀들은 역할이 정해져 있습니다. 만약 정해진 역할이 아닌 다른 일을 시켰다면, 핀은 일을 수행할 수 없기 때문에 올바른 결과를 가져오지 못할 것입니다. 아날로그 회로를 구성하기 전 꼭 NodeMCU 핀 맵을 확인 후 진행하시길 바랍니다. 아래 사진은 NodeMCU의 핀 맵입니다.



아래 예제는 NodeMCU의 D1 핀을 사용하여 1초 주기로 LED를 깜빡이게 하는 예제입니다. 아래 사진과 같이 회로를 구성한 후 소스코드를 작성하여 NodeMCU에 업로드합니다.

준비물

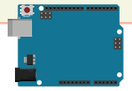
NodeMCU, LED, 150Ω 저항, 브레드보드, M-M 케이블



```
void setup(){
  pinMode(D1, OUTPUT); // D1번핀 출력설정
}
void loop(){
  digitalWrite(D1, HIGH); //D1번핀 HIGH신호 출력(3V 출력)
  delay(1000);
  digitalWrite(D1, LOW); //D1번핀 LOW 신호 출력(0V 출력)
  delay(1000);
}
```

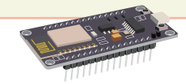
NodeMCU에서 핀 번호를 사용하는 경우 아두이노처럼 앞 알파벳을 생략할 수 없습니다.

```
void setup(){
    pinMode(1, OUTPUT);
}
```



아두이노 ('D' 생략)

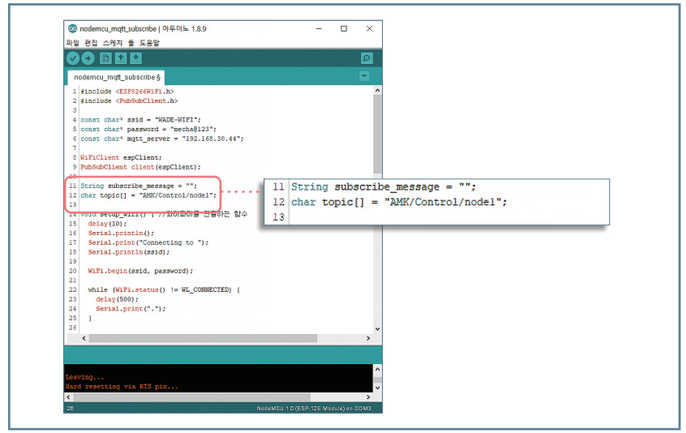
```
void setup(){
    pinMode(D1, OUTPUT);
}
```



NodeMCU ('D' 작성)

② 토픽 구독

코딩팩이 브로커로 보내는 메시지를 받아오기 위해 앞서 다루었던 구독 예제 'nodemcu_mqtt_subscribe'에서 topic 변수를 'AMK/Control/node1'으로 변경하여 'AMK/Control/node1' 토픽을 구독합니다.



3 LED 제어 핀 설정

NodeMCU의 D1핀을 사용하여 LED를 제어하기 위해 D1핀을 출력으로 설정합니다. 'nodemcu_mqtt_subscribe' 예제 void setup() 함수에 아래와 같이 pinMode(D1, OUTPUT)을 추가합니다.

4 콜백함수 수정

브로커에서 보내는 결과를 받아 LED를 ON/OFF 하는 조건문을 콜백 함수에 추가합니다. 이때 브로커가 NodeMCU로 보내는 메시지는 "HIGH" 또는 "LOW" 두 가지입니다. 만약 브로커가 "HIGH" 메시지를 보내면 LED가 켜지고, "LOW" 메시지를 보내면 LED가 꺼지도록 아래와 같이 void callback() 함수에 조건문을 추가합니다.

코딩 전체보기

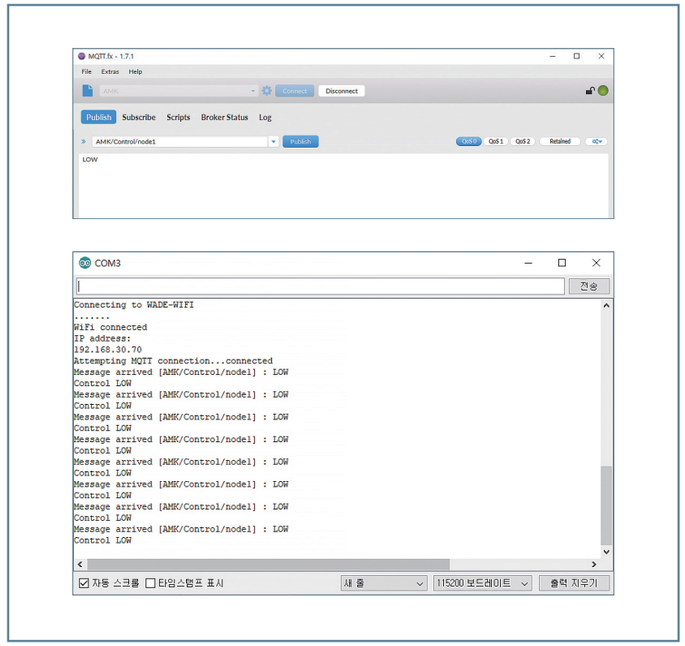
```
#include <ESP8266WiFi.h>
#include <PubSubClient.h>
const char* ssid = "사용자의 WIFI이름을 입력해주세요";
const char* password = "WIFI의 비밀번호를 입력해주세요";
const char* mqtt_server = "브로커의 IP를 입력해주세요";
WiFiClient espClient;
PubSubClient client(espClient);
void setup_wifi() {
 delay(10);
 Serial.println();
 Serial.print("Connecting to ");
 Serial.println(ssid);
 WiFi.begin(ssid, password);
 while (WiFi.status() != WL_CONNECTED) {
   delay(500);
   Serial.print(".");
 }
 randomSeed(micros());
 Serial.println("");
 Serial.println("WiFi connected");
 Serial.println("IP address: ");
 Serial.println(WiFi.localIP());
}
String subscribe_message = "";
void callback(char* topic, byte* payload, unsigned int length) {
 subscribe_message = "";
 Serial.print("Message arrived [");
 Serial.print(topic);
 Serial.print("] : ");
 for (int i = 0; i < length; i++) {
   subscribe_message += (char)payload[i]; //버퍼에 페이로드의 데이터를 담아줍니다.
 }
 Serial.println(subscribe_message);
 if (subscribe_message == "HIGH") {
   digitalWrite(D1, HIGH);
   Serial.println("Control HIGH");
 }
 else if (subscribe_message == "LOW") {
   digitalWrite(D1, LOW);
   Serial.println("Control LOW");
 } else {
   Serial.print("Wrong data");
 }
 }
}
void reconnect() {
 while (!client.connected()) {
   Serial.print("Attempting MQTT connection...");
   String clientId = "ESP8266Client-";
   clientId += String(random(0xffff), HEX);
   if (client.connect(clientId.c_str())) {
     Serial.println("connected");
     client.publish("outTopic", "hello world");
     client.subscribe("inTopic");
   } else {
     Serial.print("failed, rc=");
     Serial.print(client.state());
     Serial.println(" try again in 5 seconds");
     delay(5000);
   }
 }
}
void setup() {
 pinMode(D1, OUTPUT); // D1핀을 출력모드로 설정
 Serial.begin(115200);
 setup_wifi();
 client.setServer(mqtt_server, 1883);
 client.setCallback(callback);
}
void loop() {
 if (!client.connected()) {
   reconnect();
 }
 client.loop();
 client.subscribe("AMK/Control/node1");
}
```

예제 'nodemcu_mqtt_subscribe'를 다음에 사용할 수도 있기 때문에 'Mqtt_Client_node'라는 파일명을 가지도록 위에서 수정한 코드를 '다른 이름으로 저장' 합니다.

2 267P
3 268P
4 269P

5 동작 테스트

PC에서 MQTT.fx 클라이언트를 사용하여 토픽(AMK/Control/node1)으로 메시지를 어떻게 발행하느냐에 따른 LED 제어가 정상적으로 동작하는지를 확인합니다. 이때 MQTT.fx 클라이언트에서 발행하여 인식할 수 있는 메시지는 'HIGH'와 'LOW'입니다.



3) 코딩팩

1 음성인식

코딩팩이 우리 말을 인식해 음성을 문자로 변환하는 음성인식 기능은 필수입니다. 우리는 앞에서 voice라는 음성 모듈을 만들었습니다. voice 모듈을 그대로 가져와 사용하도록 하겠습니다. Mqtt_Client_AMK 파일에 아래와 같이 작성한 후 음성인식이 제대로 되는지 확인합니다.

Mqtt_Client_AMK.py

```
import voice

def main():
    input_text = voice.get_text_from_voice()
    print(input_text)

if __name__ == "__main__":
     main()
```

2 get_target_status() 함수 생성

get_target_status() 함수는 음성인식 결과에 따라 브로커로 발행하는 메시지를 생성하는 함수입니다. "켜줘"가 인식되었다면 'HIGH'를, "꺼줘"가 인식되었다면 'LOW'를 결괏값으로 반환합니다. 만약 "켜줘", "꺼줘" 이외 다른 음성이 인식되었다면 "잘못된 명령입니다."를 출력하고 'None'를 반환합니다. 위에서 작성한 Mqtt_Client_AMK 파일의 'import voice' 아래에 아래와 같이 get_target_status() 함수를 생성합니다.

get_target_status() 함수는 브로커로 보낼 메시지를 생성할 뿐 브로커로 메시지를 발행하지는 않습니다.

Mqtt_Client_AMK.py

```
# STT 로 받아온 음성인식 텍스트에서 켜줘와 꺼줘를 구분해주는 함수
def get_target_status( input_text ):
    if(input_text.find("조명 켜줘") != -1): # 켜줘로 인식되면 HIGH 를 리턴합니다.
        print("조명 켜줘를 인식했습니다.")
        return "HIGH"
    elif(input_text.find("조명 꺼줘") != -1): # 꺼줘로 인식되면 LOW 를 리턴합니다.
        print("조명 꺼줘를 인식했습니다.")
        return "LOW"
    else: # 입력된 텍스트에 명령이 없다면 None 을 리턴합니다.
        print("잘못된 명령입니다.")
        return None
```

③ publish_target_status() 함수 생성

publish_target_status() 함수는 브로커로 메시지를 발행하는 함수입니다. get_target_status() 함수 결괏값(target_status)을 publish_target_status() 함수의 인자로 사용합니다. 이때 인자가 "HIGH" 또는 "LOW"가 아닐 경우 False를 반환하면서 함수는 종료되고 그렇지 않을 경우 AMK/Control/node1' 토픽(AMK/Control/node1)으로 메시지(target_status)를 발행한 후 True를 반환합니다. publish_target_status() 함수는 'import voice' 아래, get_target_status() 함수 위에 작성합니다.

Mqtt_Client_AMK.py

```
import paho.mqtt.client as mqtt
broker_address = "xxx.xxx.xxx.xxx" # 브로커의 IP 를 입력합니다.

# get_target_status() 에서 리턴된 값을 토픽에 발행해주는 함수
def publish_target_status(target_status):
    # 리턴된 값에 HIGH 또는 LOW 가 없다면 False 를 리턴합니다.
    if target_status != "HIGH" and target_status != "LOW":
        return False
    client1 = mqtt.Client()
    # Mqtt 라이브러리의 Client 인스턴스를 생성합니다.
    client1.connect(broker_address, 1883)
    # connect() 메서드를 사용해서 브로커에 연결해줍니다.
    # NodeMCU 에서 구독하고 있는 AMK/Control/node1 에 처리된 데이터를 발행합니다.
    client1.publish("AMK/Control/node1", target_status)
    # 발행처리가 완료되면 브로커와 연결을 끊어줍니다.
    client1.disconnect()
    # 모든 과정이 끝나면 True 를 리턴합니다.
    return True
```

④ main() 함수 생성

main() 함수는 각 함수를 연결하는 역할을 합니다. 또 실행 결과를 음성으로 변환하여 출력합니다. main() 함수는 publish_target_status() 함수 아래에 작성합니다.

Mqtt_Client_AMK.py

```
def main():
    voice.speech("조명을 제어합니다. 켜줘 또는 꺼줘로 제어해보세요")
    input_text = voice.get_text_from_voice()
   # STT 를 실행하여 텍스트를 받아옵니다.
    result_target = get_target_status(input_text)
   # 입력된 텍스트를 입력해 명령을 구분합니다.
   # 구분된 명령을 발행하고 결과를 받아옵니다.
    publish_result = publish_target_status(result_target)

    # 구분된 명령과 발행된 결과값을 처리하여 음성으로 출력합니다.
    if publish_result == True:
        if result_target == "HIGH":
            voice.speech("조명을 켭니다.")
        elif result_target == "LOW":
            voice.speech("조명을 끕니다.")
    else:
        voice.speech("잘못된 명령이 인식되었습니다.")
```

코딩 전체보기

Mqtt_Client_AMK.py

```
import voice
import paho.mqtt.client as mqtt

broker_address = "xxx.xxx.xxx.xxx" # 브로커의 IP 를 입력합니다.

# get_target_status() 에서 리턴된 값을 토픽에 발행해주는 함수
def publish_target_status( target_status ):
    # 리턴된 값에 HIGH 또는 LOW 가 없다면 False 를 리턴합니다.
    if target_status != "HIGH" and target_status != "LOW":
        return False
    client1 = mqtt.Client() # Mqtt 라이브러리의 Client 인스턴스를 생성합니다.
    client1.connect(broker_address, 1883)
    # connect() 메서드를 사용해서 브로커에 연결해줍니다.
    # NodeMCU 에서 구독하고 있는 AMk/Control/node1 에 처리된 데이터를 발행합니다.
    client1.publish("AMK/Control/node1", target_status)
    # 발행처리가 완료되면 브로커와 연결을 끊어줍니다.
    client1.disconnect()
    # 모든 과정이 끝나면 True 를 리턴합니다.
    return True

# STT 로 받아온 음성인식 텍스트에서 조명 켜줘와 조명 꺼줘를 구분해주는 함수
def get_target_status( input_text ):
    if(input_text.find("조명 켜줘") != -1): # 켜줘로 인식되면 HIGH 를 리턴합니다.
        print("조명 켜줘를 인식했습니다.")
        return "HIGH"
    elif(input_text.find("조명 꺼줘") != -1): # 꺼줘로 인식되면 LOW 를 리턴합니다.
        print("조명 꺼줘를 인식했습니다.")
        return "LOW"
    else: # 입력된 텍스트에 명령이 없다면 None 을 리턴합니다.
        print("잘못된 명령입니다.")
        return None

def main() :
    voice.speech("조명을 제어합니다. 조명 켜줘 또는 조명 꺼줘로 제어해보세요")
    input_text = voice.get_text_from_voice() # STT 를 실행하여 텍스트를 받아옵니다.
    result_target = get_target_status(input_text)
    # 입력된 텍스트를 입력해 명령을 구분합니다.
    # 구분된 명령을 발행하고 결과를 받아옵니다.
    publish_result = publish_target_status(result_target)

    # 구분된 명령과 발행된 결과값을 처리하여 음성으로 출력합니다.
    if publish_result == True:
        if result_target == "HIGH":
            voice.speech("조명을 켭니다.")
        elif result_target == "LOW":
            voice.speech("조명을 끕니다.")
    else
       voice.speech("잘못된 명령이 인식되었습니다.")

if __name__ == "__main__" :
    main()
```

3 274P

2 273P

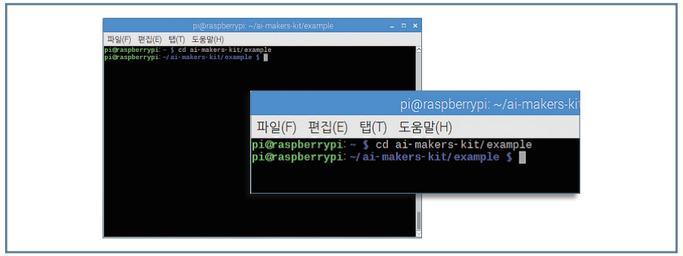
4 275P

4 275P

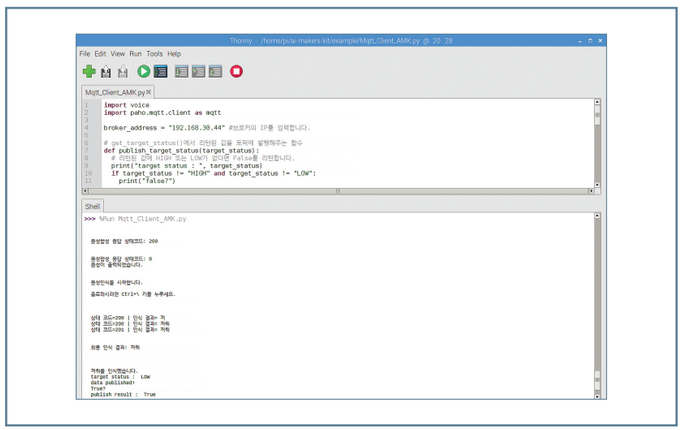
5 프로그램 실행

터미널 창에서 아래 명령어를 입력하여 Mqtt_Client_AMK.py 파일이 있는 경로로 이동합니다.

```
$ cd ai-makers-kit/example
```

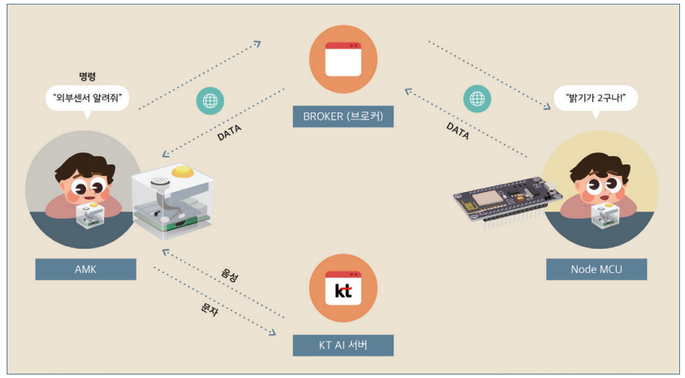


파이썬3으로 Mqtt_Client_AMK.py 파일을 실행시켜 NodeMCU에 연결된 LED를 제어해봅시다. 이때 “조명 켜줘” 또는 “조명 꺼줘” 이외의 명령은 사용할 수 없습니다. 아래 사진은 프로그램을 실행한 후 “조명 꺼줘” 명령에 따른 처리 결과를 보여줍니다.



06 코딩팩으로 조도센서 값 받아오기

앞에서 해봤던 프로젝트는 코딩팩에서 브로커를 거쳐 NodeMCU로 메시지를 발행하는 것이었다면, 이번에는 반대로 NodeMCU에서 발행된 메시지를 코딩팩에서 구독해 그 결과를 음성으로 출력하는 프로젝트를 한번 해보도록 하겠습니다. 이번 프로젝트에서 NodeMCU는 조도센서가 측정한 데이터를 메시지로 발행합니다. 프로그램이 실행되면 조도센서에서 측정된 데이터를 음성으로 출력합니다.



1) NodeMCU의 역할

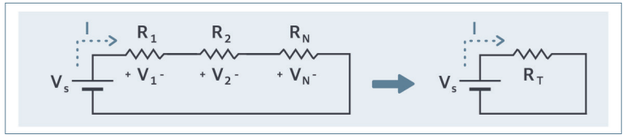
1 아날로그 회로 구성

조도센서는 밝기를 측정하는 아날로그 센서로 밝을수록 저항값이 낮아지고, 어두울수록 저항값이 높아집니다. 센서에서 출력되는 아날로그 신호를 개발 보드가 입력받기 위해서는 아날로그 신호를 디지털로 변환해주어야 합니다. 이때 사용되는 것이 ADC(Analog to Digital Converter)입니다. 라즈베리파이(코딩팩)에는 ADC가 내장되어 있지 않기 때문에 아날로그 신호를 입력받기 위해 추가로 ADC 모듈을 사용해야 되지만 NodeMCU에는 다행히 10bit ADC 가 내장되어 있습니다. ADC는 0~3.3V 사이로 측정되는 전압을 0~1023(10bit) 범위를 가지는 디지털 신호로 변환합니다.

조도 센서는 조도에 따라서 약 10KΩ~100KΩ사이의 저항을 출력합니다. 조도 센서와 10KΩ 저항과 같이 사용하면 전압 분배 법칙에 의해 조도센서와 저항 사이의 전압이 출력됩니다. 이 전압 값 이용하면 NodeMCU의 아날로그 입력으로 사용할 수 있습니다.

전압 분배 법칙

전압 분배 법칙은 한 회로 안에 다수의 저항이 있을 경우 각각의 저항이 저항값에 비례하는 전압을 분배 받는다는 법칙입니다. 아래 직렬 회로를 보면서 설명해보도록 하겠습니다. 저항이 오른쪽 회로와 같이 직렬로 있으면 모든 저항값을 더해 하나의 저항으로 나타낼 수 있기 때문에 왼쪽 회로는 같은 상태라고 할 수 있습니다. 그렇게 하면 이 회로의 전류 'I'를 옴의 법칙에 적용하여 계산할 수 있습니다.



$$I = \frac{V_1 + V_2 + V_3 + \cdots + V_N}{R_1 + R_2 + R_3 + \cdots + R_N} = \frac{V_S}{R_T}$$

직렬회로에서는 각 저항에 전류가 동일하게 적용됩니다. 그러면 저항과 전류를 통해 각 저항에 분배된 전압을 계산할 수 있습니다. 그리고 각 저항의 전압을 모두 더하게 되면 전체 전압을 계산할 수 있습니다.

$$V_1 = I \cdot R_1$$
$$V_2 = I \cdot R_2$$
$$V_3 = I \cdot R_3$$
$$V_S = V_1 + V_2 + V_3$$

직렬 회로에서 전체 전압을 계산하는 식을 정리하면 아래와 같은 식을 구할 수 있습니다.

$$V_1 = I \cdot R_1 + I \cdot R_2 + I \cdot R_3 + \cdots + I \cdot R_N$$
$$V_S = I \cdot (R_1 + R_2 + R_3 + \cdots + R_N) \cdots R_T$$

위의 공식들을 사용하면 각 저항에 대한 전압을 계산할 수 있는 공식을 도출해낼 수 있습니다. 이 공식을 사용하면 전압 분배된 저항의 전압값을 쉽게 계산할 수 있습니다.

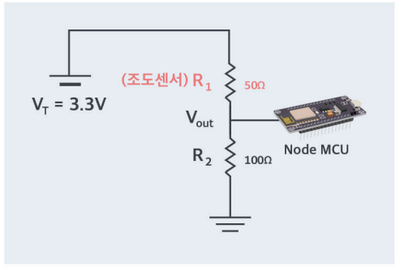
$$V_N = \frac{V_S}{R_T} \cdot R_N$$

$$V_1 = \frac{V_S}{R_T} \cdot R_1$$

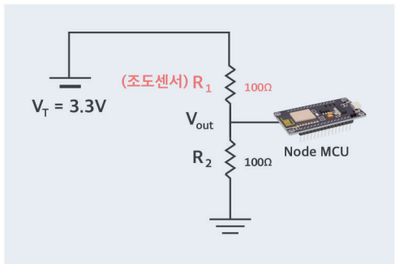
$$V_2 = \frac{V_S}{R_T} \cdot R_2$$

따라서 각 저항에서의 전압강하는 전체 저항에 대한 각 저항의 비에 전원 전압을 곱한 것과 같습니다. 전압 분배 법칙을 사용하면 직렬회로의 저항 비율만으로 각 저항에서의 전압강하를 쉽게 구할 수 있습니다. 아래 회로는 조도센서 값을 NodeMCU에서 측정하기 위해 사용할 회로로, 조도센서의 저항값에 따라 NodeMCU에 입력되는 전압값이 어떻게 변하는지 한번 보도록 하겠습니다.

※ 쉽게 설명하기 위해 고정 저항값을 100Ω으로 고정합니다. 실제 회로를 구성할 때는 100Ω 저항 대신 10KΩ 저항을 사용합니다.



$$V_{out} = \frac{3.3}{150} \times 50 = \frac{3.3}{3} = 1.1V$$

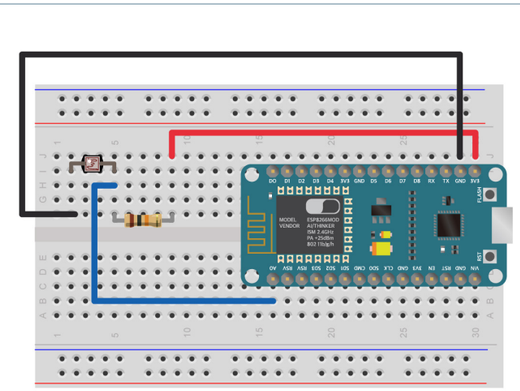


$$V_{out} = \frac{3.3}{200} \times 100 = \frac{3.3}{2} = 1.65V$$

위에서 알 수 있듯 조도센서의 저항값에 따라 NodeMCU로 입력되는 전압값이 변하는 것을 확인할 수 있습니다.

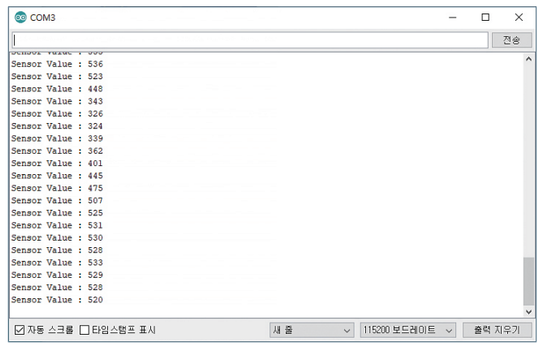
회로 구성

준비물 : NodeMCU, 10KΩ 저항, 브레드보드, M-M 케이블



```
void setup() {
  Serial.begin(115200);
}
void loop() {
  Serial.print("Sensor Value : ");
  Serial.println(analogRead(A0));
  delay(100);
}
```

출력 화면



② 브로커에게 메시지 발행

MQTT 통신을 사용하여 지정한 토픽에 5초에 한 번 메시지(조도센서 값)를 발행하는 프로그램을 만들어 보겠습니다. 앞서 사용해보았던 NodeMCU 발행 예제인 'nodemcu_mqtt_publush'의 void loop() 부분을 아래와 같이 수정하여 사용합니다.

'nodemcu_mqtt_publush' 예제의 WiFI 명, 비밀번호, IP주소와 topic("AMK/Sensor/node") 변수가 올바르게 입력되었는지 꼭 확인합니다.

```
void loop() {
 if (!client.connected()) {
   reconnect();
 }
 client.loop();
 long now = millis();
 if (now - lastMsg > 2000) {
   lastMsg = now;
   client.publish(topic, "Hello MQTT");
   Serial.println("message Puslished!");
 }
}
```

⋮

```
void loop() {
 if (!client.connected()) {
   reconnect();
 }  client.loop();
 long now = millis();
 if (now - lastMsg > 5000) {
 //마지막에 보낸 메세지 기준으로 5초를 지연합니다.
   lastMsg = now;
   int sensor_value = analogRead(A0);
   Serial.print("Sensor Value : ");
   Serial.println(sensor_value);
   itoa(sensor_value, msg, 10);
   client.publish("AMK/Sensor/node1", msg);
   Serial.println("message Puslished");
 }
}
```

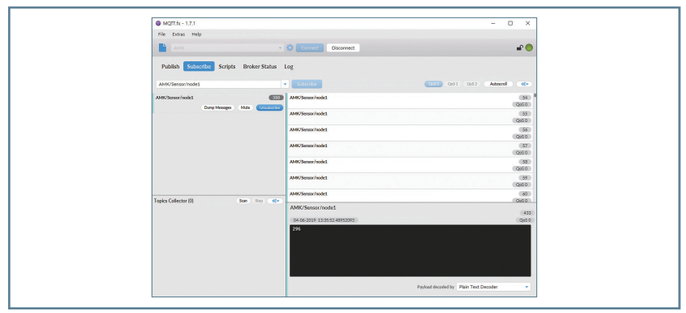
itoa 함수

itoa 함수는 정수형을 문자열로 변환하는 함수입니다. 메시지 발행은 문자열 데이터만 전송할 수 있기 때문에 측정된 센서 데이터를 문자열로 변경해 메시지로 발행합니다. itoa 함수는 아래와 같은 방법으로 사용할 수 있습니다.

itoa(정수형 변수, 문자형 변수, 변환할 진수)

프로그램을 업로드한 후 'MQTT.fx' 클라이언트를 사용하여 NodeMCU가 브로커로 발행하는 메시지를 구독해보도록 하겠습니다.

'MQTT.fx' 클라이언트 토픽이 'AMK/Sensor/node1'인지 꼭 확인합니다.



코딩 전체보기

nodemcu_mqtt_publish.ino

```
#include <ESP8266WiFi.h>
#include <PubSubClient.h>
WiFiClient espClient;
PubSubClient client(espClient);
//네트워크 설정
const char* ssid = "사용자의 WIFI 이름을 입력해주세요";
const char* password = "WIFI의 비밀번호를 입력해주세요";
const char* mqtt_server = "브로커의 IP를 입력해주세요";
long lastMsg = 0;
char msg[50];
int value = 0;
char topic[] = "outTopic";
void setup_wifi() { // 와이파이를 연결하는 함수
  delay(10);
  Serial.println();
  Serial.print("Connecting to ");
  Serial.println(ssid);
  WiFi.begin(ssid, password);
 while (WiFi.status() != WL_CONNECTED) {
   delay(500);
   Serial.print(".");
 }
 Serial.println("");
 Serial.println("WiFi connected");
 Serial.println("IP address: ");
 Serial.println(WiFi.localIP());
}
void reconnect() { //브로커와 연결이 끊기면 재연결해주는 함수
 while (!client.connected()) {
   Serial.print("Attempting MQTT connection...");
   String clientId = "ESP8266Client-";
   clientId += String(random(0xffff), HEX);
   if (client.connect(clientId.c_str())) {
     Serial.println("connected");
     client.publish(topic, "hello world");
   } else {
     Serial.print("failed, rc=");
     Serial.print(client.state());
     Serial.println(" try again in 5 seconds");
     delay(5000);
   }
 }
}
void setup() {
 Serial.begin(115200);
 setup_wifi();
 client.setServer(mqtt_server, 1883);
 randomSeed(micros());
//reconnect() 함수에서 클라이언트 ID를 생성할 때 사용되는 랜덤시드를 설정
}
void loop() {
 if (!client.connected()) {
   reconnect();
 }  client.loop();
 long now = millis();
 if (now - lastMsg > 5000) { //마지막에 보낸 메세지 기준으로 5초를 지연합니다.
   lastMsg = now;
   int sensor_value = analogRead(A0);
   Serial.print("Sensor Value : ");
   Serial.println(sensor_value);
   itoa(sensor_value, msg, 10);
   client.publish("AMK/Sensor/node1", msg);
   Serial.println("message Puslished");
 }
}
```

1 281P

2 282P

'nodemcu_mqtt_publish' 예제의 WiFI 명, 비밀번호, IP주소와 topic("AMK/Sensor/node1") 변수를 사용자의 환경에 맞게 꼭 변경합니다.

3) 코딩팩

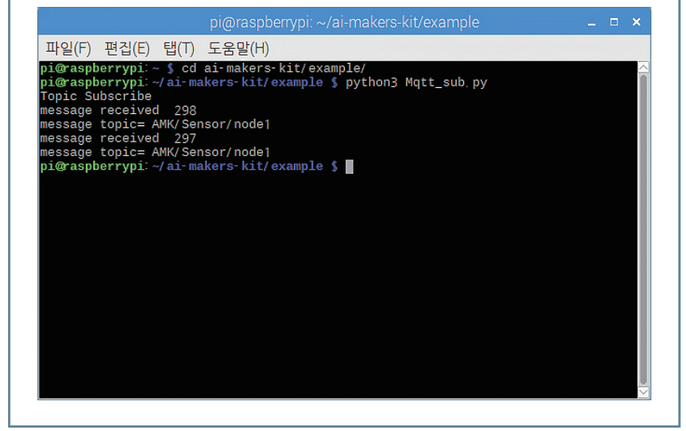
1 토픽(AMK/Sensor/node1) 구독

앞서 사용했던 파이썬 MQTT 구독 예제인 Mqtt_sub.py을 복사하여 파일명 AMK_mqtt_sub.py파일을 새롭게 생성합니다. AMK_mqtt_sub.py 파일을 열어 브로커 IP주소를 입력하고 토픽을 AMK/Sensor/node1으로 변경합니다.

AMK_mqtt_sub.py

```
import paho.mqtt.client as mqtt
import time

broker_address = "xxx.xxx.xxx.xxx"

# subscriber 콜백함수
def on_message(client, userdata, message):
    # 콜백함수에 들어오는 데이터를 변환하여 변수에 저장하고 토픽과 메세지를 출력합니다.
    print("message received ", str(message.payload.decode("utf-8")))
    print("message topic=", message.topic)

def subscribe_message():
    client1 = mqtt.Client()
    # MQTT 라이브러리의 Client 인스턴스를 생성합니다.
    client1.connect(broker_address, 1883) # 브로커에 연결합니다.
    client1.on_message = on_message # 콜백함수를 설정해줍니다.

    client1.subscribe("AMK/Sensor/node1") # 지정한 토픽을 구독해줍니다.
    print("Topic Subscribe")

    client1.loop_start()
    # 구독한 토픽에서 데이터를 받기 위해서 무한루프를 시작합니다.
    time.sleep(10) # 10 초 딜레이를 사용합니다.
    client1.loop_stop() # 무한루프를 종료합니다.

    client1.disconnect() # 브로커와 연결을 끊습니다.
def main():
    subscribe_message()

if __name__ == "__main__":
    main()
```

아래의 이미지는 프로그램 실행 결과입니다. 구독한 토픽으로부터 조도센서 값을 잘 받아오는 것을 확인할 수 있습니다. 앞서 소스코드에서 설명했듯이 NodeMCU는 5초에 한 번씩 데이터를 발행하고, 코딩팩은 10초 동안 데이터를 받아 오기 때문에 결과를 두 번 출력한 후 종료되는 것을 확인할 수 있습니다.



② on_message() 함수, subscribe_message() 함수 수정

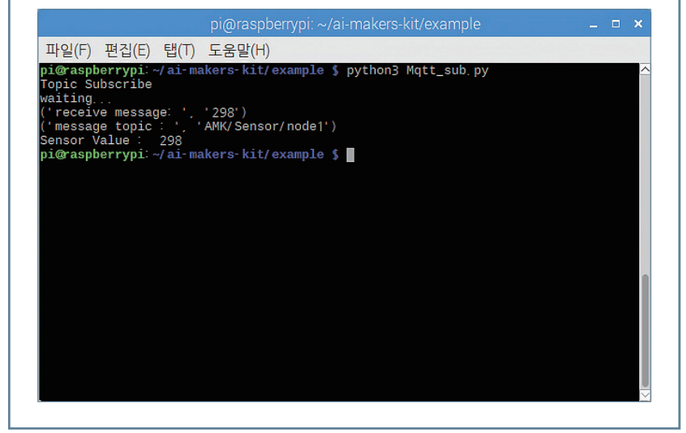
위에서 작성한 구독 프로그램에서 10초 동안 발행된 모든 메시지를 출력하는 부분을 메시지가 발행될 때까지 기다렸다가 메시지가 한번 발행되면 출력하고 종료되도록 수정하도록 하겠습니다.

· 변수 broker_address 아래 전역변수 recv_message를 선언한 후 아래와 같이 on_message() 함수와 subscribe_message() 함수를 수정합니다.

AMK_mqtt_sub.py

```
recv_message = None

# subscriber 콜백함수
def on_message(client, userdata, message):
    global recv_message # 전역변수를 가져옵니다.
    # 콜백함수에 들어오는 데이터를 변환하여 변수에 저장하고 토픽과 메세지를 출력합니다.
    recv_message = str(message.payload.decode("utf-8"))
    print(("receive message: ",recv_message))
    print(("message topic : ",message.topic))

def subscribe_message():
  client1 = mqtt.Client() # mqtt Client 인스턴스를 생성합니다.
  client1.connect(broker_address, 1883) # mqtt 브로커에 연결합니다.
  client1.on_message = on_message # 원하는 토픽을 구독해줍니다.

  client1.subscribe("AMK/Sensor/node1") # 원하는 토픽을 구독해줍니다.
  print("Topic Subscribe")

  client1.loop_start()
  # 구독한 토픽에서 데이터를 받기 위해서 무한루프를 시작합니다.
  global recv_message #전역변수를 가져옵니다.

  # 콜백함수에서 메시지가 발행될 때까지 대기하며 무한반복합니다.
  while recv_message == None:
    print("Wating.....")
    time.sleep(1)

  # 발행된 메시지를 출력하고 mqtt클라이언트를 종료합니다.
  print("Sensor Value : ", recv_message)
  recv_message = None
  client1.loop_stop()
  client1.disconnect()
```

아래의 이미지는 프로그램 실행 결과입니다. 구독한 토픽에서 메시지를 발행할 때까지 기다린 후 메시지가 발행되면 출력하고 종료되는 것을 확인할 수 있습니다.



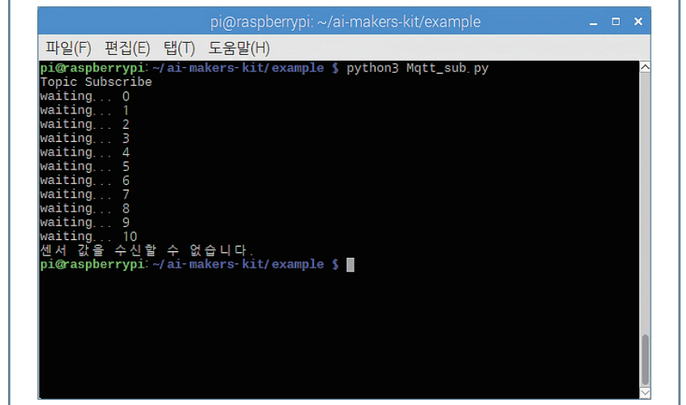
3 subscribe_message() 함수 수정

subscribe_message() 함수는 메시지가 발행된다면 전혀 문제가 되지 않지만, 만약 메시지가 발행되지 않는다면 데이터를 받아올 때까지 프로그램이 종료되지 않는다는 문제가 발생합니다. 그래서 10초 동안 메시지가 발행되지 않는다면 프로그램을 종료하는 부분을 추가하도록 하겠습니다.

AMK_mqtt_sub.py

```
def subscribe_message():
    client1 = mqtt.Client() # mqtt Client 인스턴스를 생성합니다.
    client1.connect(broker_address, 1883) # 브로커에 연결합니다.
    client1.on_message = on_message # 콜백함수를 설정해줍니다.

    client1.subscribe("AMK/Sensor/node1") # 원하는 토픽을 구독해줍니다.
    print("Topic Subscribe")

    client1.loop_start()
    # 구독한 토픽에서 데이터를 받기 위해서 무한루프를 시작합니다.
    global recv_message # 전역변수를 가져옵니다.

    # 콜백함수에서 메세지가 발행될 때까지 대기하며
    # 메세지가 들어오거나 time_out_value 가 10 이되면 다음코드를 실행합니다.
    time_out_value = 0
    return_message = ""
    while recv_message == None and time_out_value < 11:
        print("waiting... %d" % time_out_value)
        time_out_value += 1
        time.sleep(1)
    # 조건문을 사용해 결과에 따라 출력해줍니다.
    if recv_message == None:
        print("센서 값을 수신할 수 없습니다.")
    else:
        print(recv_message)
        recv_message = None # 변수를 초기화 합니다.

    client1.loop_stop()
    client1.disconnect() # 클라이언트를 종료합니다.
```

아래의 이미지는 프로그램 실행 결과입니다. NodeMCU에서 MQTT 통신을 끊은 상태에서 코딩팩에서 구독을 했을 때 즉, 메시지가 발행되지 않을 경우, 아래 결과처럼 10초 동안 기다린 후 프로그램을 종료합니다.



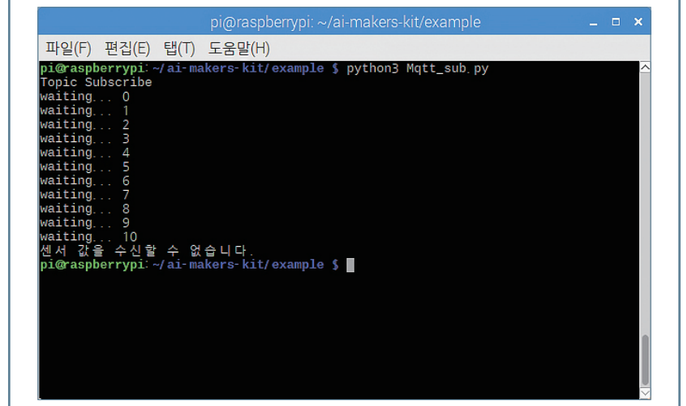
4 음성합성

NodeMCU에서 발행하는 메시지(조도센서 값)를 음성으로 출력하는 부분을 subscribe_message() 함수와 main() 함수에 추가합니다.

AMK_mqtt_sub.py

```
def subscribe_message():
    client1 = mqtt.Client() # mqtt Client 인스턴스를 생성합니다.
    client1.connect(broker_address, 1883) # 브로커에 연결합니다.
    client1.on_message = on_message # 콜백함수를 설정해줍니다.

    client1.subscribe("AMK/Sensor/node1") # 원하는 토픽을 구독해줍니다.
    print("Topic Subscribe")

    client1.loop_start()
    # 구독한 토픽에서 데이터를 받기 위해서 무한루프를 시작합니다.
    global recv_message # 전역변수를 가져옵니다.

    # 콜백함수에서 메세지가 발행될 때까지 대기하며
    # 메세지가 들어오거나 time_out_value 가 10 이되면 다음코드를 실행합니다.
    time_out_value = 0
    return_message = ""
    while recv_message == None and time_out_value < 11:
        print("waiting... %d"%time_out_value)
        time_out_value += 1
        time.sleep(1)
    # 조건문을 사용해 결과에 따라 출력해줍니다.
    if recv_message == None:
        print("센서 값을 수신할 수 없습니다.")
    else:
        print("입력된 센서 값은 %s 입니다." % recv_message)
        return_message =  "입력된 센서 값은 %s 입니다." % recv_message

    recv_message = None # 변수를 초기화 합니다.
    client1.loop_stop()
    client1.disconnect() # 클라이언트를 종료합니다.

    return return_message # 결과를 리턴해줍니다.

def main():
    voice.speech("연결된 센서의 측정 값을 알려드리겠습니다.")
    result_message = subscribe_message()
    voice.speech(result_message)
```

아래의 이미지는 프로그램 실행 결과입니다. NodeMCU에서 발행한 메시지를 코딩팩에서 구독해 그 결과를 음성으로 출력합니다.



코딩 전체보기

AMK_mqtt_sub.py

```
import paho.mqtt.client as mqtt
import time
import voice

broker_address = "xxx.xxx.xxx.xxx"

recv_message = None

# subscriber 콜백함수
def on_message(client, userdata, message ):
    global recv_message # 전역변수를 가져옵니다.
    # 콜백함수에 들어오는 데이터를 변환하여 변수에 저장하고 토픽과 메세지를 출력합
니다.
    recv_message = str(message.payload.decode("utf-8"))
    print(recv_message)
    print("message topic=", message.topic)

def subscribe_message():
    client1 = mqtt.Client() # mqtt Client 인스턴스를 생성합니다.
    client1.connect(broker_address, 1883) # 브로커에 연결합니다.
    client1.on_message = on_message # 콜백함수를 설정해줍니다.

    client1.subscribe("AMK/Sensor/node1") # 원하는 토픽을 구독해줍니다.
    print("Topic Subscribe")

    client1.loop_start()
    # 구독한 토픽에서 데이터를 받기 위해서 무한루프를 시작합니다.
    global recv_message # 전역변수를 가져옵니다.

    # 콜백함수에서 메세지가 발행될 때까지 대기하며
    # 메세지가 들어오거나 time_out_value 가 10 이되면 다음코드를 실행합니다.
    time_out_value = 0
    return_message = ""
    while recv_message == None and time_out_value < 11:
        print("waiting... %d" % time_out_value)
        time_out_value += 1
        time.sleep(1)
    # 조건문을 사용해 결과에 따라 출력해줍니다.
    if recv_message == None:
        print("센서 값을 수신할 수 없습니다.")
        return_message =  "센서 값을 수신할 수 없습니다."
    else:
        print("입력된 센서 값은 %s 입니다."%recv_message)
        return_message =  "입력된 센서 값은 %s 입니다."%recv_message

    recv_message = None # 변수를 초기화 합니다.
    client1.loop_stop()
    client1.disconnect() # 클라이언트를 종료합니다.

    return return_message # 결과를 리턴해줍니다.
def main():
    voice.speech("연결된 센서의 측정 값을 알려드리겠습니다.")
    result_message = subscribe_message()
    voice.speech(result_message)

if __name__ == "__main__":
     main()
```

1 286P
2 288P
3 290P
4 292P

IP주소와 토픽을 확인한 후 프로그램을 실행합니다.

07
프로젝트 코드 어플리케이션으로 사용하기

우리가 지금까지 만든 날씨 정보 가져오기, 사전 만들기, 센서 제어 등 모든 프로젝트는 모듈로 제작되었습니다. 사실 마지막에 모든 프로젝트를 합치기 위해 일부로 모듈로 만들었습니다. 모듈을 이용하면 모든 프로젝트와 코딩팩을 쉽게 연결할 수 있습니다. 프로그램은 아래 순서에 맞춰 동작합니다.

프로그램 동작 순서

1. 호출어 감지
- voice 모듈의 detect_wake_up_word() 함수를 사용하여 호출어를 감지합니다.
2. 음성인식
- voice 모듈의 voice.get_text_from_voice() 함수를 사용하여 마이크로 입력되는 음성을 인식합니다.
3. 프로젝트 실행 명령어 감지
- 입력된 음성 데이터에 각 프로젝트를 실행시킬 명령어가 포함되었는지를 확인합니다.
4. 프로젝트 또는 질의 응답실행
- 만약 음성 데이터에 프로젝트 실행 명령어가 포함되었다면, 해당 프로젝트를 실행시키고, 그렇지 않을 경우 음성 데이터에 맞는 답변을 스피커로 출력합니다.

각 프로젝트를 실행시킬 명령어는 아래와 같습니다.

1. 퀴즈 게임 만들기(XX 페이지) : “퀴즈 게임 시작해줘”
2. 타이머 만들기(XX 페이지) : “타이머 시작해줘”
3. 날씨 API 사용하기(XX 페이지) : “날씨 알려줘”
4. 크롤링으로 국어사전 만들어보기(XX 페이지) : “국어사전 실행해줘”
5. 코딩팩으로 NodeMCU 디지털 출력해보기(XX 페이지) : “외부 제어 실행해줘”
6. 코딩팩으로 NodeMCU 센서 값 받아오기(XX 페이지) : “외부 센서 실행해줘”

talk_amk.py

```
import voice

# 각각의 프로젝트 모듈을 포함합니다.
import ex3_4_quiz_game as quiz_game
import ex3_5_timer as timer
import ex4_1_get_weather as weather
import ex4_2_dictionary as dictionary
import ex5_1_digitalControl as control
import ex5_2_iot_sensor as sensor

def main():
   while True:
       if voice.detect_wake_up_word():
           input_text = voice.get_text_from_voice()
          # 텍스트로 받아온 음성에서 호출 구문을 찾으면 프로젝트를 시작합니다.
           if input_text.find("퀴즈 게임 시작해 줘") != -1:
               quiz_game.main()
           elif input_text.find("타이머 시작해 줘") != -1:
               timer.main()
           elif input_text.find("날씨 알려줘") != -1:
               weather.main()
           elif input_text.find("사전 시작해 줘") != -1:
               dictionary.main()
           elif input_text.find("외부 제어 시작해 줘") != -1:
               control.main()
           elif input_text.find("외부 센서 실행해 줘") != -1:
               sensor.main()
           # 호출 구문이 인식되지 않으면 질의응답을 사용하여 답변을 받습니다.
           else:
               result_answer = voice.query_by_text(input_text)
               voice.speech(result_answer)

if __name__ == "__main__":
    main()
```