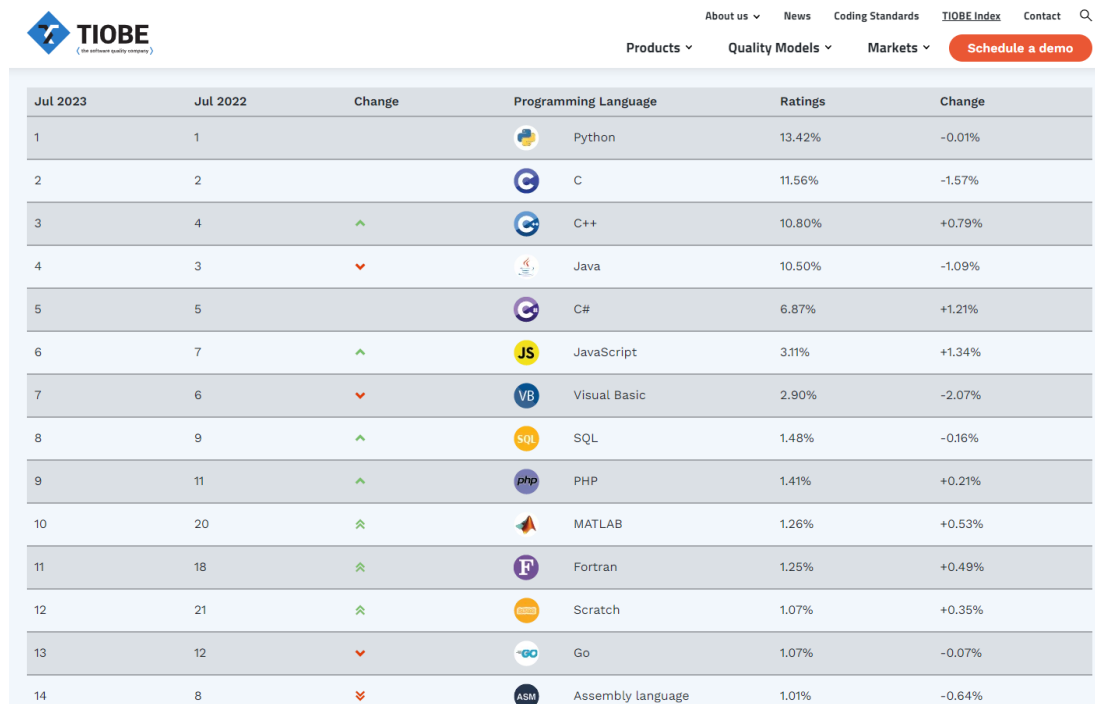
















1. 파이썬 소개

파이썬 소개

1. 소개

- 1991년에 귀도 반 로섬(Guido van Rossum)이 발표한 프로그래밍 언어
- 오픈소스이며, 인공지능, 머신러닝, 빅데이터 등을 다룰 수 있는 수많은 라이브러리 등이 제공되고 있어 다양한 활용이 가능
- 비교적 문법이 쉽고 간편하며, 인터프리터 언어로 실행이 빠름
- 현재 전세계적으로 가장 인기 있는 프로그래밍 언어 중 하나 (출처: <https://www.tiobe.com/tiobe-index/>)



Jul 2023	Jul 2022	Change	Programming Language	Ratings	Change
1	1		 Python	13.42%	-0.01%
2	2		 C	11.56%	-1.57%
3	4	▲	 C++	10.80%	+0.79%
4	3	▼	 Java	10.50%	-1.09%
5	5		 C#	6.87%	+1.21%
6	7	▲	 JavaScript	3.11%	+1.34%
7	6	▼	 Visual Basic	2.90%	-2.07%
8	9	▲	 SQL	1.48%	-0.16%
9	11	▲	 PHP	1.41%	+0.21%
10	20	▲	 MATLAB	1.26%	+0.53%
11	18	▲	 Fortran	1.25%	+0.49%
12	21	▲	 Scratch	1.07%	+0.35%
13	12	▼	 Go	1.07%	-0.07%
14	8	▼	 Assembly language	1.01%	-0.64%

2. 공식홈페이지

- <https://www.python.org/>
- 2023년 7월 현재 3.11.4 버전
 - 매년 10월경 버전 업데이트를 진행함.

- 간혹 버전에 따라 호환되지 않는 패키지 등이 있으니 업데이트시 주의

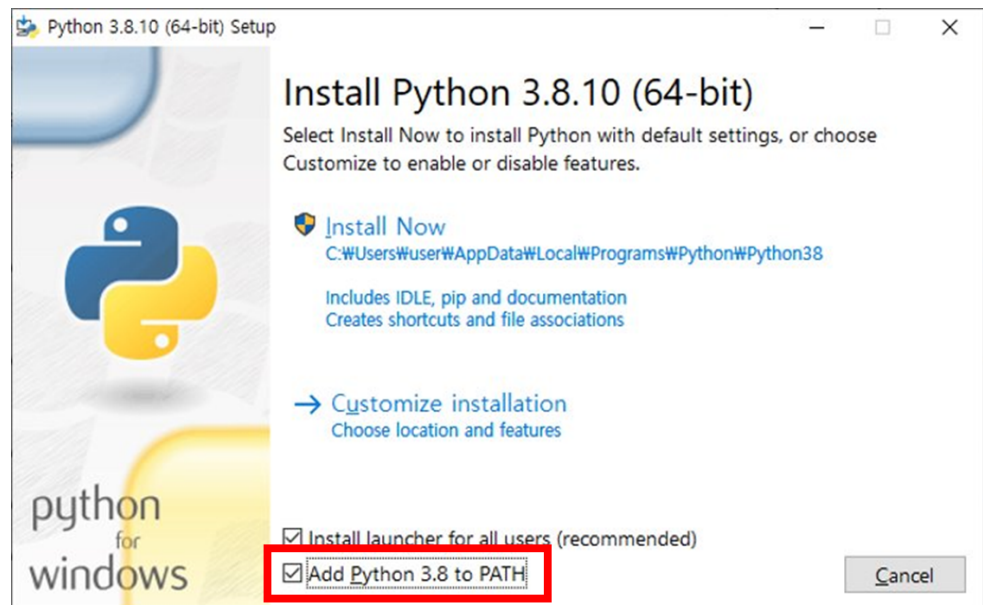
3. 개발환경 구축

- python 설치

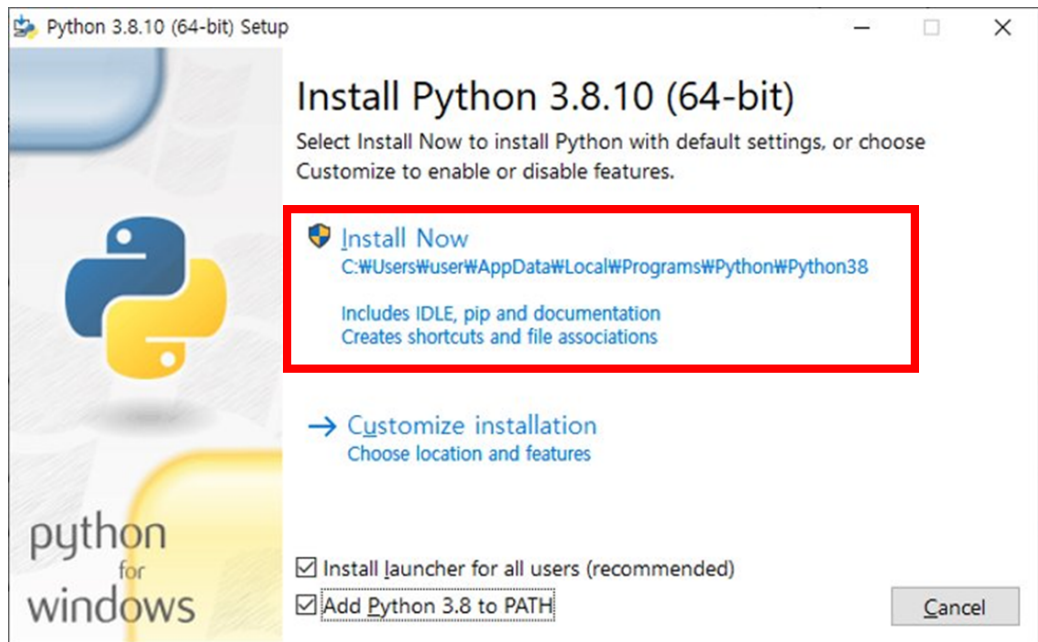
- <https://www.python.org/downloads/release/python-3114/>

1. 파일 다운로드 후 첫 실행 화면에서 Add Python 3.8 to PATH 반드시 체크

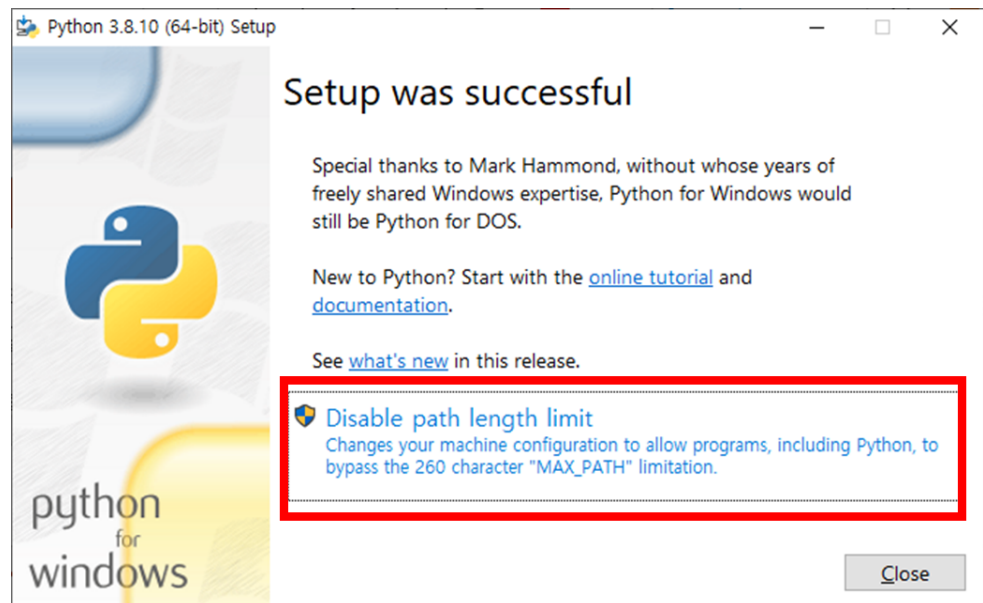
- a. 설치시에 환경변수를 추가하는 과정이며, 체크를 해야 PC 어디서든 파이썬 명령어 사용이 가능



2. 체크 후 Install Now 클릭



3. 설치완료 후 마지막 화면에서 Disable path length limit
 - a. 파일 이름이나 경로길이 제한(260자)을 해제하는 옵션



4. 설치 완료 후 cmd에서 python --version 입력 후 아래 그림과 같이 버전 확인

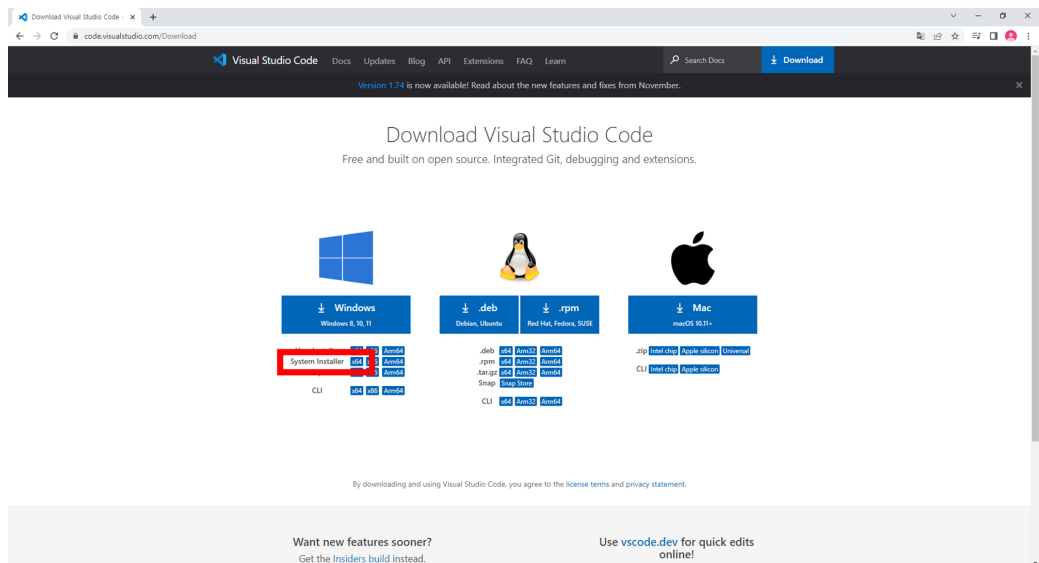
```
명령 프롬프트
Microsoft Windows [Version 10.0.19045.2486]
(c) Microsoft Corporation. All rights reserved.

C:\Users\user>python --version
Python 3.8.10

C:\Users\user>
```

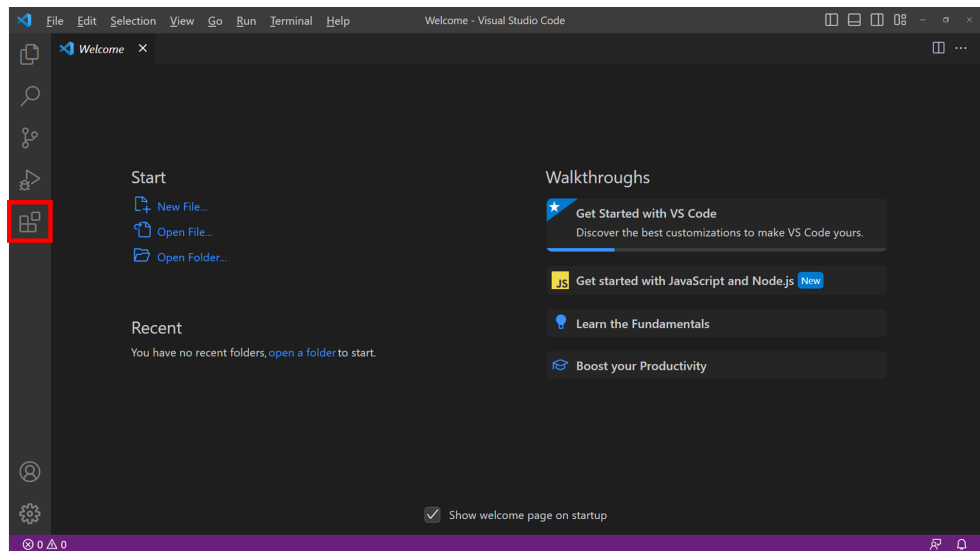
- Visual Studio Code 설치

- <https://code.visualstudio.com/Download>
- 해당 주소 접속 후 아래 그림과 같이 OS에 맞는 파일로 설치 (*예시 화면은 Windows 64-bit)

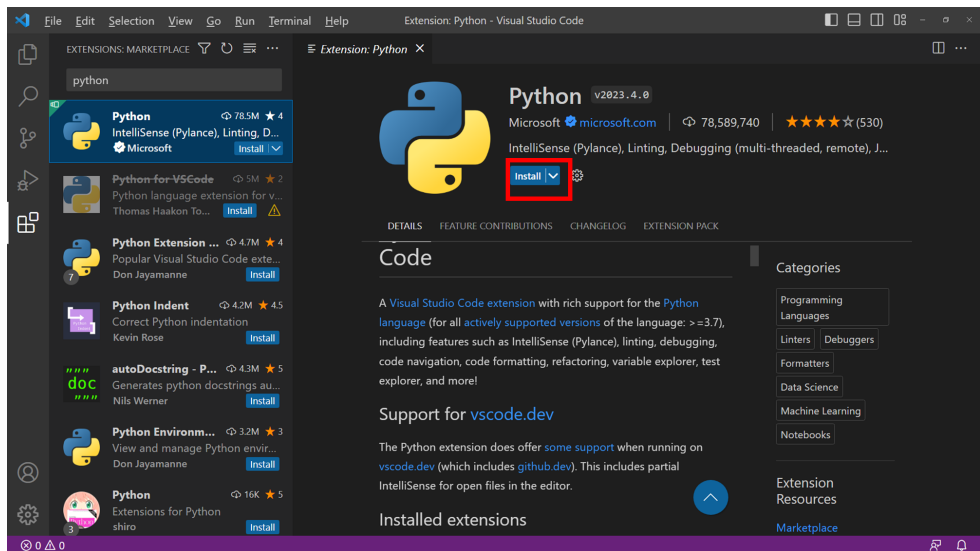


1. 파이썬 관련 Extension(확장프로그램) 설치하기

- a. 첫 실행 후 Extensions 아이콘 선택(아래 화면 빨간색 네모)



b. 검색창에 파이썬 검색 후 Python 선택하여 Install



2. 기본 문법 - 변수와 자료형

기본 문법 - 변수와 자료형

1. 변수와 자료형

- a. 프로그램은 수 많은 데이터를 다루게 됨
 - i. 메신저의 메시지 또는 이미지, 은행앱의 입출금 내역 또는 잔고, 쇼핑몰의 상품 정보, 나의 구매내역, 장바구니 내역 등
- b. 이러한 데이터들을 다루기 위해선 데이터를 저장하거나 전달할 수 있는 공간 또는 매개체가 필요함. 이를 변수로 활용할 수 있으며, 변수에는 데이터의 종류를 표현하는 자료형이 필수
- c. 자료형을 다룰 때 파이썬 특징
 - i. 다른 프로그래밍언어(Java, C 등)와는 다르게 변수 선언시 별도의 자료형을 함께 선언하지 않고 변수이름에 값을 대입하는 형태로 사용

```
# Python에서 정수형 변수를 선언
number = 10
# Java에서 정수형 변수를 선언
int number = 10;
```

- d. 자료형 종류(참고:<https://docs.python.org/3.11/library/index.html>)
 - i. 숫자형(Numeric Types): int(정수), float(실수), complex(복소수)

```
ex01_numeric.py

# int
int_var = 10
# float
float_var = 1.2345
# complex
complex_var = 1 + 2j
```

- ii. 문자나열형(Text Sequence Type): str
 - 1. 문자, 단어, 문장 등을 다룰 수 있는 자료형

2. 선언시 값은 ‘(싱글쿼터) 또는 “(더블쿼터) 사용
3. 각각의 문자는 인덱스로 지정하여 사용할 수 있음
4. ‘+’를 사용하여 2개 이상의 변수를 연결할 수 있음

```
ex02_str.py

str_var1 = "python"
print(str_var1) # 출력: python
print(str_var1[0]) # 출력: p
print(str_var1[3]) # 출력: h

str_var2 = "파이썬"
print(str_var1 + str_var2)

str_var3 = str_var1 + str_var2
print(str_var3)
```

iii. 논리형(Boolean Types): bool

1. True, False 두 가지 값만 가질 수 있음.

iv. 나열형(Sequence Types): list, tuple, range

1. 리스트(list)

- a. 순서대로 데이터를 나타내는데 사용함
- b. [] (대괄호)를 사용하여 리스트 요소를 나열하며, 다양한 자료형을 포함시킬 수 있음

```
ex03_list.py

# 숫자 리스트
int_list = [1, 2, 3, 4]
# 문자열 리스트
str_list = ["안녕", "하세요", "python"]
# 다양한 자료형 리스트
mix_list = [10, 20, "반갑습니다"]
# 리스트 내에 리스트
list_in_list = [100, 200, ["내부", "리스트"], 10.567]
```

- c. 각 요소는 인덱스로 접근할 수 있으며, 수정이 가능함

```
# 2번 인덱스 값 출력
print(int_list[2]) # 출력: 3
# 2번 인덱스에 10 대입
int_list[2] = 10
```

```
# 2번 인덱스 값 출력
print(int_list[2]) # 출력: 10
```

2. 튜플(tuple)

- a. 리스트와 비슷하게 순서대로 데이터를 나타내는데 사용함
- b. ()(소괄호)를 사용하여 튜플 요소를 나열하고, 리스트와의 차이점은 요소 수정 불가능

```
ex04_tuple.py

# 숫자 튜플
int_tuple = (1, 2, 3, 4)
# 문자열 튜플
str_tuple = ("안녕", "하세요", "python")
# 다양한 튜플 리스트
mix_tuple = (10, 20, "반갑습니다")
# 튜플 내에 튜플
tuple_in_tuple = (100, 200, ("내부", "리스트"), 10.567)
```

- c. 리스트처럼 요소는 인덱스로 접근하며, 수정은 불가능함

```
# 2번 인덱스 값 출력
print(int_tuple[2]) # 출력: 3
# 2번 인덱스에 10 대입
int_tuple[2] = 10 # 에러 발생
```

3. range

- a. 지정한 범위 내에 숫자를 만들어낼 때 사용하며, for문과 같은 반복문에서 많이 사용함

```
ex05_range.py

# 0부터 10까지 만들어서 출력하기
print(list(range(11)))
# 1부터 10까지 만들어서 출력하기
print(list(range(1, 11)))
# 2, 4, 6, 8, 10 만들어서 출력하기
print(list(range(2, 10, 2)))
```

v. 매핑형(Mapping Types): dict

- 1. 딕셔너리 자료형은 key:value 형태의 요소를 가지는 그룹형 자료형. 순서가 없으며, key를 이용하여 접근함


```

ex06_dic.py

word_dic = {
    "dog": "강아지",
    "cat": "고양이",
    "tiger": "호랑이",
    "lion": "사자"
}
# lion 키값 출력
print(word_dic["lion"])
# tiger 키값 수정
word_dic["tiger"] = "호랭이"
# tiger 키값 출력
print(word_dic["tiger"])
# word_dic 전체 값 출력
print(word_dic)
# bear 추가
word_dic["bear"] = "곰"
# word_dic 전체 값 출력
print(word_dic)

```

e. 실습내용

- i. 손흥민 선수의 이름(name), 생년월일(birth), 키(height), 소속팀(team_name) 정보를 변수에 각각 담아서 출력해보기

```

name = "손흥민"
birth = "1990년 7월 8일"
height = 183.0
team_name = "토트넘 홋스퍼"
print(name)
print(birth)
print(height)
print(team_name)

```

- ii. 위의 정보를 딕셔너리에 담아서 출력해보기. 등번호(number)값을 추가해보기

```

son_info = {
    "name": "손흥민",
    "birth": "1990년 7월 8일",
    "height": 183.0,
    "team_name": "토트넘 홋스퍼"
}

print(son_info)

# 등번호 값 추가
son_info["number"] = 7
print(son_info)

```

3. 기본 문법 - 조건문

기본 문법 - 조건문

1. 제어문이란

a. 코드의 흐름을 제어하기 위해 사용하는 문법

i. 제어: 조건, 반복 등과 같이 특정 환경에 따라 코드의 흐름을 바꿀 수 있도록 하는 것

b. 다른 프로그래밍 언어들과 마찬가지로 파이썬도 조건문, 반복문을 위한 문법 제공

c. 조건문

i. 어떠한 조건을 만족하는지 만족하지 않는 지에 따라 코드의 흐름을 제어함

ii. ex) 어떤 숫자가 양수인지 음수인지 판단, 어떤 숫자가 홀수인지 짝수인지 판단 등

iii. 조건문에는 if 문을 사용

d. 반복문

i. 어떠한 조건을 만족할 때 까지 또는 조건을 만족하는 동안 동일한 코드를 반복하여 실행

ii. ex) 1부터 10까지의 숫자를 순차적으로 만들기, 숫자의 누적 합계를 계산하면서 100이 넘으면 계산을 중단, 리스트 또는 튜플 내에 있는 요소를 순차적으로 접근하여 출력

iii. 반복문에는 for, while문을 사용

2. 조건문

a. 조건을 따질 문장이 반드시 필요하며, 조건식은 bool 타입의 값, 변수, 식 등으로 작성할 수 있음.

b. 조건문 문법

i. if 문

1. 조건을 만족하는 경우 특정 문장을 실행할 때

```
if [조건]:  
    조건을 만족하는 경우 실행될 문장
```

ii. if else 문

1. 조건을 만족하는 경우, 만족하지 않는 경우를 구분하여 실행할 때
2. else 부분에는 조건을 쓰지 않음.

```
if [조건]:  
    조건을 만족하는 경우 실행될 문장  
else:  
    조건을 만족하지 않는 경우 실행될 문장
```

iii. if elif 문

1. 조건이 두 개 이상인 경우

```
if [조건1]:  
    조건1을 만족하는 경우 실행될 문장  
elif [조건2]:  
    조건2을 만족하는 경우 실행될 문장
```

2. 조건의 수에 따라 elif는 계속 추가 가능

```
if [조건1]:  
    조건1을 만족하는 경우 실행될 문장  
elif [조건2]:  
    조건2을 만족하는 경우 실행될 문장  
elif [조건3]:  
    조건3을 만족하는 경우 실행될 문장  
....  
elif [조건N]:  
    조건N을 만족하는 경우 실행될 문장
```

iv. if elif else 문

1. 조건이 여러 개일 때 어떠한 조건도 만족하지 않는 경우 실행할 부분을 정의할 때

```
if [조건1]:  
    조건1을 만족하는 경우 실행될 문장  
elif [조건2]:  
    조건2을 만족하는 경우 실행될 문장
```

```
elif [조건3]:
    조건3을 만족하는 경우 실행될 문장
else:
    조건1, 2, 3을 모두 만족하지 않는 경우 실행될 문장
```

c. 실습 - 조건식 작성 방법(코드박스를 확인해주세요)

i. 조건 예(bool 타입 값)

```
ex01_if_bool.py

if True:
    print("True 일 때 출력됩니다")
elif False:
    print("False 일 때 출력됩니다")
```

ii. 조건 예(bool 타입 변수)

```
ex02_if_variable.py

answer = True
if answer:
    print("answer는 True 입니다")
elif not answer:
    print("answer는 False 입니다, elif부분")
else:
    print("answer는 False 입니다")
```

iii. 조건 예(조건식)

```
ex_03_if_condition.py

num = 10
if num > 0:
    print("양수입니다")
elif num < 0:
    print("음수입니다")
else:
    print("0입니다")
```

3. 실습 예제

- a. 어떠한 정수가 홀수인지 짝수인지 판별하여 홀수입니다. 짝수입니다. 홀수/짝수를 판단할 수 없습니다. 출력해보기

ex04_even_odd.py

```
num = 1.2323
print(num%2)
if num%2 == 0:
    print("짝수입니다.")
elif num%2 == 1:
    print("홀수입니다.")
else:
    print("홀수/짝수를 판단할 수 없습니다.")
```

4. 기본 문법 - 반복문

기본 문법 - 반복문

1. for문

- a. 반복 대상에 순차적으로 접근하여 순차적인 값을 만들어 내거나 리스트, 튜플과 같은 자료형에 저장된 데이터를 접근할 때 많이 사용

b. for문 문법

```
for [반복변수] in [반복대상]:  
    반복하여 실행할 문장
```

c. for문 사용 방법

i. 순차적으로 숫자 출력하기

```
ex01_for_numbers.py  
  
# 0부터 9까지 출력하기  
for i in range(10):  
    print(i)  
# 1부터 10까지 출력하기  
for i in range(1,11):  
    print(i)
```

ii. 리스트 요소 접근하여 출력하기

```
ex02_for_list.py  
  
list1 = [10, 20, 30, 40, 50]  
for num in list1:  
    print(num)  
  
list2 = ["빨강", "주황", "노랑", "초록", "파랑"]  
for color in list2:  
    print(color)  
  
# 같은 줄에 표현할 때(end 파라미터)  
# print 사용법 간단하게  
list3 = ["빨강", "주황", "노랑", "초록", "파랑", ["분홍", "연한분홍", "진한분홍"]]  
for color in list3:  
    print(color, end=" ")  
    print(color, end="\t")  
    print()
```

iii. 튜플 요소 접근하여 출력하기

```
ex03_for_tuple.py

tuple1 = ("apple", "banana", "melon", "cherry", "tomato")
for fruit in tuple1:
    print(fruit)

tuple2 = ("apple", "banana", "melon", "cherry", "tomato", ("strawberry", "blueberry"))
for fruit in tuple2:
    print(fruit)

tuple3 = ((1, "python"), (2, "java"), (3, "c"), (4, "javascript"))
for index, lang in tuple3:
    print(index, lang)
```

2. while문

a. 무한 반복문을 실행하거나 특정 조건을 만족할 때 까지 반복문을 실행할 때 많이 사용

b. while문 문법

```
while [조건식]:
    반복하여 실행할 문장
```

c. while문 사용 방법

i. 무한 반복하기(종료시 터미널에서 ctrl+C)

```
ex04_while_loop.py

while True:
    print("안녕하세요")
```

ii. 반복을 10회만 하고 종료하기

1. while문 조건식으로 횟수 지정

```
ex05_while_condition_01.py

count = 1
while count <= 10:
    print("안녕하세요", count)
    count = count + 1
```

2. 무한루프에서 while문 내부에 if문과 break로 횟수 지정

```
ex05_while_condition_02.py

count = 1
while True:
```

```
print("안녕하세요", count)
if count == 10:
    break
count = count + 1
```

3. while문 조건식에 bool 타입 변수를 사용하고 if문으로 횟수 지정

```
ex05_while_condition_03.py

count = 1
bool_var = True
while bool_var:
    print("안녕하세요", count)
    if count == 10:
        bool_var = False
    count = count + 1
```

3. 실습 예제

a. for문을 이용하여 구구단 출력하기 (세로로 출력하기, 가로로 출력하기)

i. 출력예시

ii. 세로로 출력하기

```
ex06_numbers_table.py

for i in range(2,10):
    for j in range(1,10):
        print(i,"x",j,"=",i*j)
```

iii. 가로로 출력하기

```
ex06_numbers_table.py

for i in range(2,10):
    print(i,"단")
    for j in range(1,10):
        print(i,"x",j,"=",i*j,end="\t")
    print()
```


5. 파이썬 함수

파이썬 함수

1. 함수

- a. 특정 기능을 수행하기 위한 코드블록
- b. def 키워드를 이용하여 함수를 정의할 수 있으며, 매개변수, 리턴 등을 정의할 수 있음.
- c. 함수 내에 작성한 코드는 항상 실행되는 것이 아니라 호출을 해야 실행됨
- d. 함수의 기본 구조

```
# 매개변수 x, 리턴 x
def [함수이름]:
    함수호출시 실행내용

# 매개변수 x, 리턴 o
def [함수이름]:
    함수호출시 실행내용
    return 리턴문장

# 매개변수 o, 리턴 x
def [함수이름(매개변수)]:
    함수호출시 매개변수를 활용하는 실행내용

# 매개변수 o, 리턴 o
def [함수이름(매개변수)]:
    함수호출시 매개변수를 활용하는 실행내용
    return 리턴문장
```

e. 안녕하세요를 출력해보는 함수

- i. 함수를 호출하면 '안녕하세요'를 함수 내부에서 출력

```
ex01_function_basic_01.py

def hello():
    print("안녕하세요1")

hello()
```

- ii. 함수를 호출하면 '안녕하세요'를 리턴받아 리턴값을 함수 외부에서 출력

```
ex01_function_basic_02.py

def hello():
    # return "안녕하세요"
    hello = "안녕하세요"
    return hello
# print(hello())
return_var = hello()
print(return_var)
```

- iii. 함수를 호출하면서 '안녕하세요'를 전달하고 함수 내부에서 출력

```
ex01_function_basic_03.py

def hello(h):
    print(h)
# hello("안녕하세요")
param_var = "안녕하세요"
hello(param_var)
```

- iv. 함수를 호출하면서 '안녕하세요'를 전달하고 함수 내부에서 뒤에 “반갑습니다”를 붙이고 리턴하여 함수 외부에서 출력

```
ex01_function_basic_04.py

def hello(h):
    return_value = h + "반갑습니다"
    return return_value

param_var = "안녕하세요"
result = hello(param_var)
print(result)
```

2. 사용자 정의 함수, 내장함수

a. 사용자 정의 함수

- 필요한 기능을 개발자가 직접 함수로 정의. 필요할 때마다 호출하여 사용할 수 있음
- 함수이름, 매개변수, 실행내용, 리턴값 등을 정의함

b. 내장함수

- 파이썬에서 기본적으로 제공하는 함수

ii. 대표적인 내장함수(일부만 소개)

1. `print()`: 매개변수로 전달하는 값 또는 변수 값을 콘솔에 출력
2. `input()`: 사용자 입력으로 받은 값을 문자열로 반환
3. `type()`: 변수의 자료형을 반환
4. `len()`: 문자열, 리스트, 튜플, 딕셔너리 등의 길이(항목수)를 반환
5. `max()`: 매개변수로 전달받은 변수의 최댓값을 반환
6. `min()`: 매개변수로 전달받은 변수의 최솟값을 반환

6. 모듈

모듈

1. 모듈

- a. 함수, 클래스 등을 정의한 파이썬 파일
- b. 다른 파이썬 파일에서 호출하여 사용할 수 있음
- c. 문법

i. import

- 1. 다른 파일에 정의된 함수 변수 등을 모두 사용할 수 있음
- 2. 호출할 때는 파일이름.함수이름() 형태로 호출함
- 3. as를 이용하여 이름을 줄일 수 있음

```
import [해당파일이름(.py는 생략)]  
  
import [해당파일이름(.py는 생략)] as [약어]
```

1. from ~ import ~

- a. 다른 파일의 특정 함수만 지정하여 가져오는 방법

```
from [해당파일이름(.py는 생략)] import [함수이름]
```

2. 실습

- a. 모듈 사용시 실행하면 폴더 내에 __pycache__ 라는 폴더가 생성되는데 다음 번 실행시 더 빠른 실행을 위해 파이썬 자체에서 만들어내는 폴더
- b. 하나의 파일에 함수와 호출 문장이 함께 존재

```
ex01_main.py  
  
def hello():
```

```
print("안녕하세요")

hello()
```

c. 함수와 호출하는 문장이 서로 다른 파일에 존재

```
ex02_main.py

import ex02_function as ex

# hello() # 오류
# ex02_function.hello()
ex.hello()

ex02_function.py

def hello():
    print("안녕하세요")
```

d. 함수가 정의된 파일에 함수가 2개 정의된 경우

i. 파일 전체 import

```
ex03_functions.py

def hello1():
    print("안녕하세요. hello1")

def hello2():
    print("안녕하세요. hello2")

ex03_main.py

import ex03_functions

# hello1() # 오류
ex03_functions.hello1()
ex03_functions.hello2()
```

ii. hello1 함수만 import

```
ex04_functions.py

def hello1():
    print("안녕하세요. hello1")

def hello2():
    print("안녕하세요. hello2")
```

```
ex04_main.py

from ex03_functions import hello1

hello1()
# hello2() # 오류
```

iii. 함수를 모두 import 하고 함수 이름으로만 호출

```
ex05_functions.py

def hello1():
    print("안녕하세요. hello1")

def hello2():
    print("안녕하세요. hello2")

ex05_main.py

from ex05_functions import *

hello1()
hello2()
```