파이썬 입문

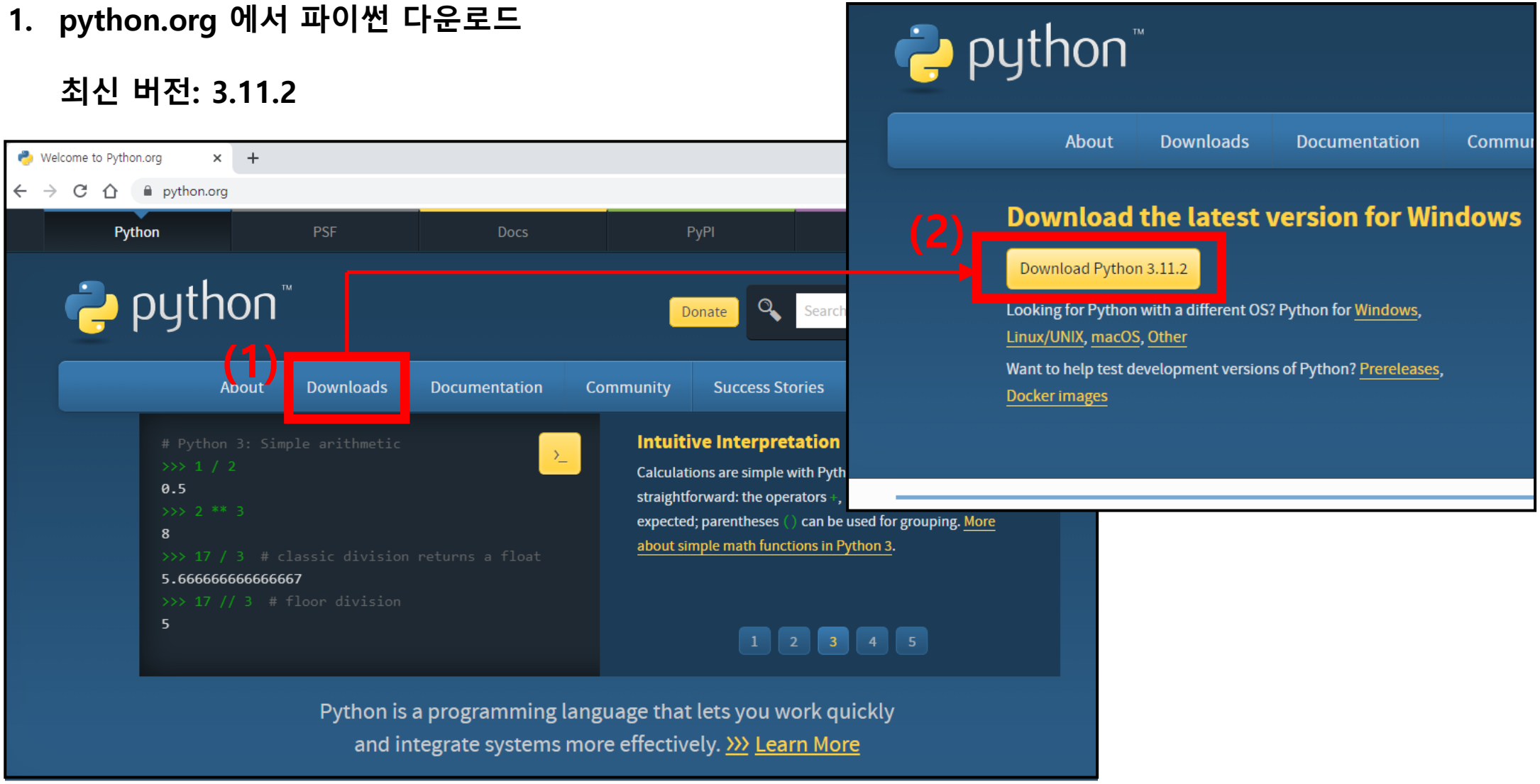
한국폴리텍대학

2023.03.10

설치하기

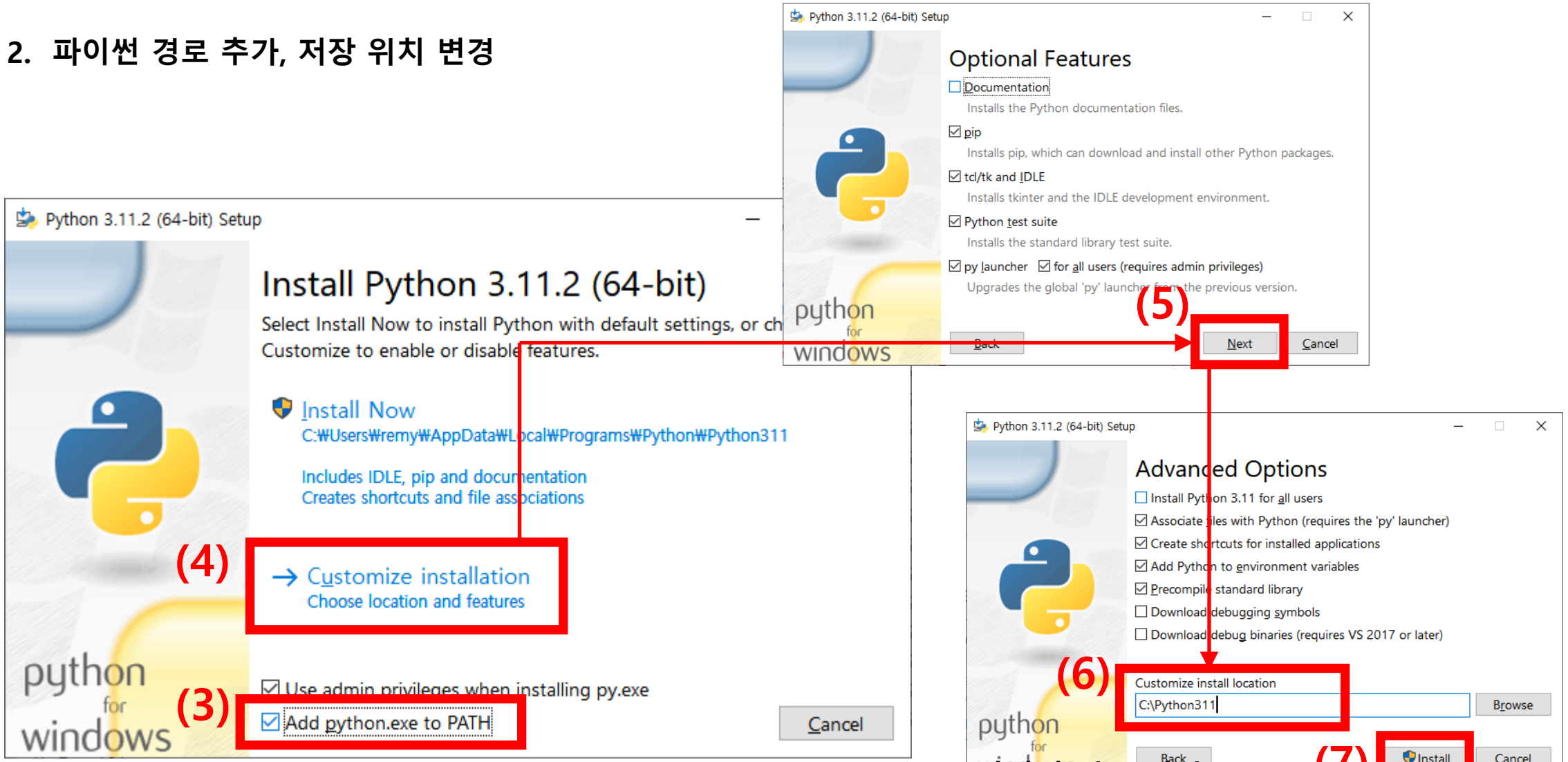
1. python.org 에서 파이썬 다운로드

   최신 버전: 3.11.2



설치하기

2. 파이썬 경로 추가, 저장 위치 변경



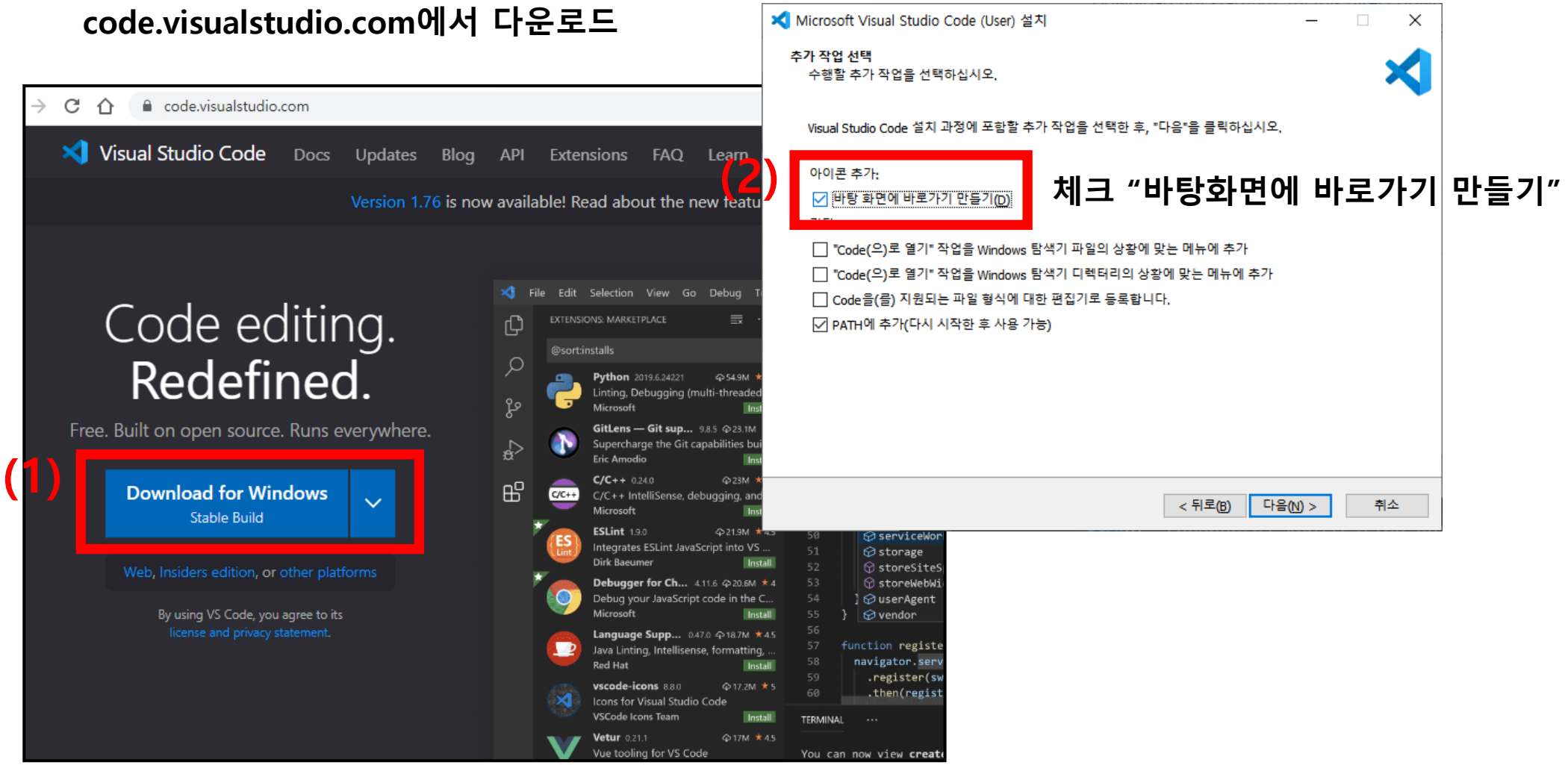
체크 "파이썬 경로 추가"

파이썬 경로 변경
C:₩Python311

설치하기

3. 비주얼 스튜디오 코드 설치 (Visual Studio Code)

code.visualstudio.com에서 다운로드

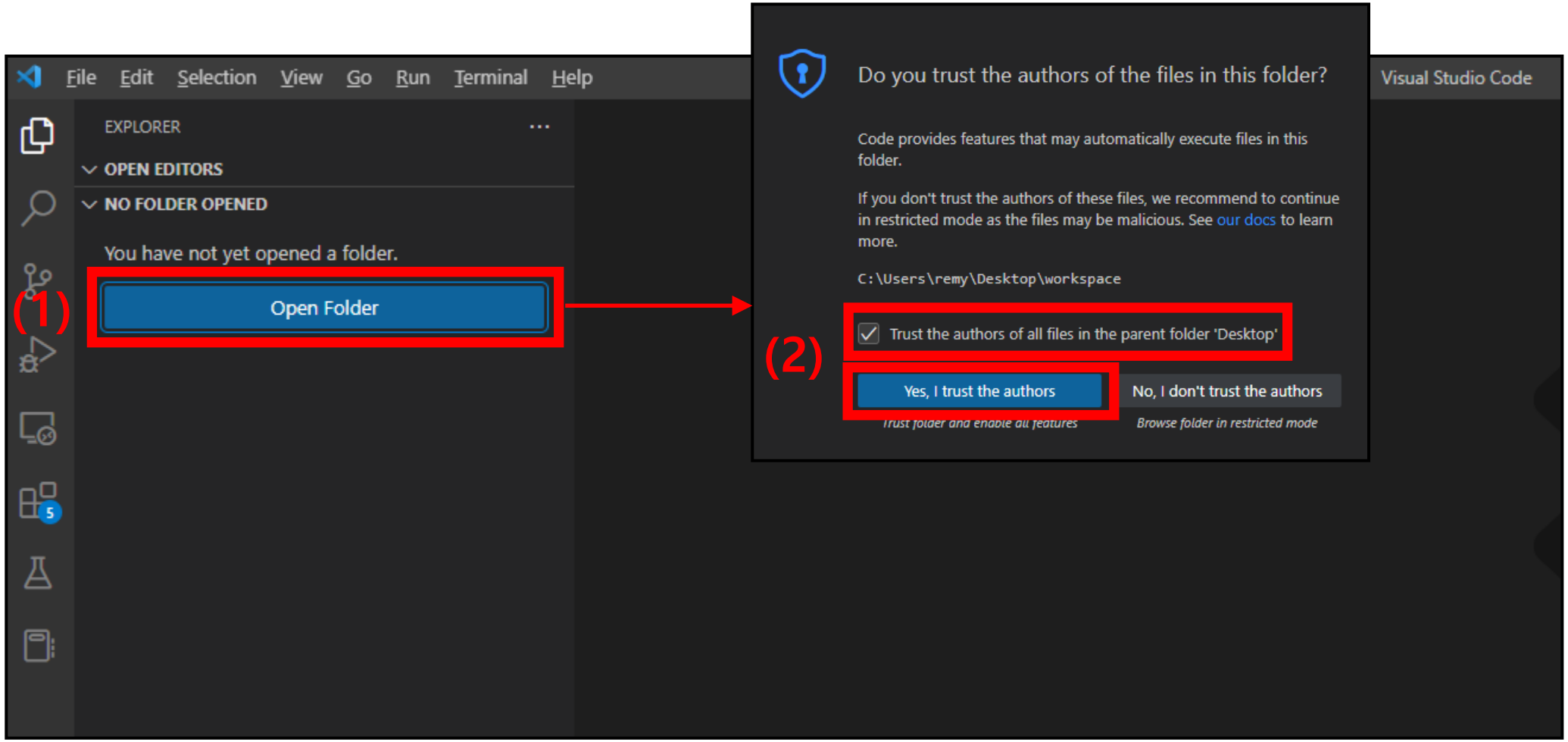


체크 "바탕화면에 바로가기 만들기"

시작하기

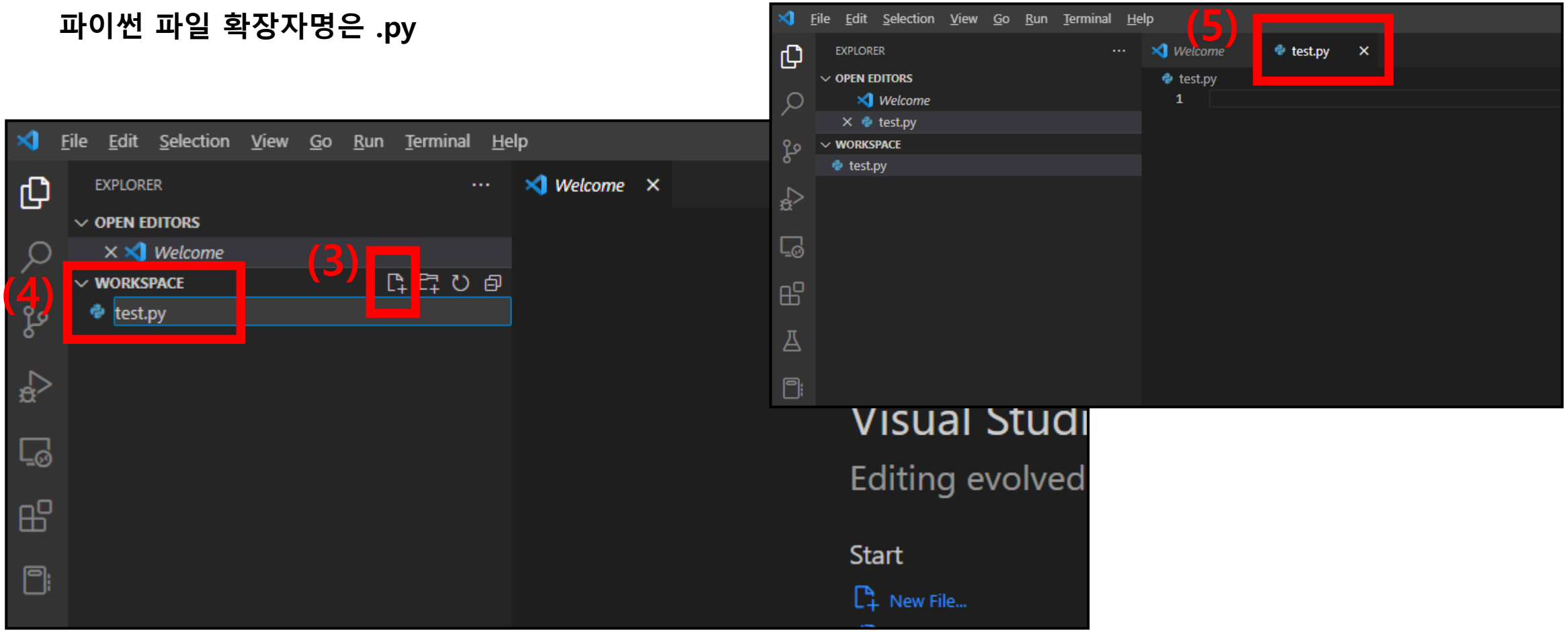
1. 비주얼 스튜디오 코드 실행

바탕화면에 workspace 폴더 생성 후, Open Folder 클릭 및 workspace 폴더 선택

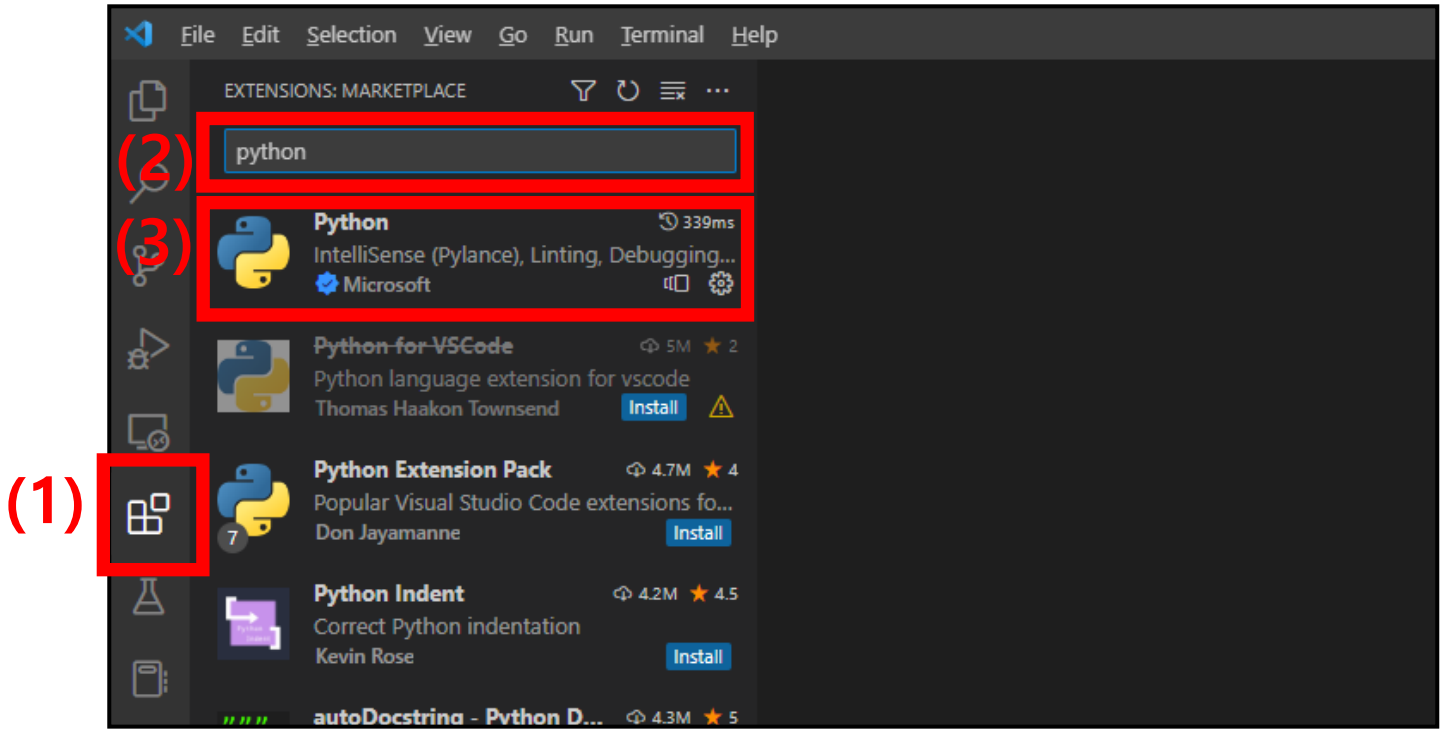


2. 새 파일 만들기

파이썬 파일 확장자명은 .py



3. Extensions 설치



주석 (comments)

1. 주석(comments) 방법

(1) 한줄 주석: #

(2) 여러줄 주석: “”” “””, ‘” ‘”

파일저장: Ctrl + S

코드실행: F5

test.py

```
# 주석 1

'''
주석 2
주석 2
주석 2
'''

"""
주석 3
주석 3
주석 3
"""

# Ctrl + / (slash)
# 주석1
# 주석2
# 주석3
```

출력 (print)

1. 출력방법

(1) 문자열 출력: print(“hello”), print(‘hello’), print(“hello “ + “world”), print(‘hello ‘ + ‘world’)

print(‘”hello world”’), print(“”“hello world”””), print(“hello “ ₩ + “world”)

print(“123” + “456”)

(2) 숫자 출력: print(65), print(3.14), print(-10),

print(5+2j), print(3+5-2), print(2*10/5),

print(3.25-2.75)

(3) 특수기호 출력: print(“C:₩₩python”),

print(“hello ₩“123₩” world”)

test.py

```python
print("hello")                # 큰따옴표 ""
print('hello')                # 작은따옴표 ''
print("hello " + "world")     # 문자열 더하기
print('hello ' + 'world')     # 문자열 더하기
print("""hello world""")      # 큰따옴표 """ """
print('''hello world''')      # 작은따옴표 ''' '''
                              # 줄바꿈
print("hello " + \
      "world")
print("123" + "456")          # 문자열 더하기
print(65)                     # 정수(양수)
print(-10)                    # 정수(음수)
print(3.14)                   # 실수
print(5+2j)                   # 복소수(complex)
print(3+5-2)                  # 정수 계산
print(2*10/5)                 # 정수 계산
print(3.25-2.75)              # 실수 계산
print("C:\\python")           # 특수기호 출력 \ (backslash)
print("hello \"123\' world")  # 특수기호 출력 " , '
print('ABC ' * 3)             # 문자열 곱하기
```

출력 (print)

2. 출력방법

(1) print(“hello “ + str(123) + “ world”), print(“hello “ + “123” + “ world”)

print(“hello“, 123, “world”), print(“hello“, str(123), “world”)

출력 (print)

3. 출력방법

```
(1) print("나는 %d살입니다." % 20)
    print("나는 %s살입니다." % 20)
    print("나는 %s을 좋아해요." % "파이썬")
    print("Apple은 %c로 시작해요." % "A")
    print("나는 %s색과 %s색을 좋아해요." % ("파란", "빨간"))

(2) print("나는 {}살입니다.".format(20))
    print("나는 {}색과 {}색을 좋아해요.".format("파란", "빨간"))
    print("나는 {1}색과 {0}색을 좋아해요.".format("파란", "빨간"))
    print("나는 {color1}색과 {color2}색을 좋아해요.".format(color1 = "파란", color2 = "빨간"))

(3) color1 = "파란"
    color2 = "빨간"
    print( f"나는 {color1}색과 {color2}색을 좋아해요.")

(4) print("{0:,}".format(100000000))  # 3자리마다 콤마 찍어주기
```

변수 (Variables)

```
x = 4
y = “Sally”
X = “John”      # 대소문자 구분
y = 10
2x = 5          # 변수명 오류
My-var  = 10  # 변수명 오류
My var  = 10  # 변수명 오류
```

```
x, y, z = 5, “banana”, 15
x = y = z = “apple”
```

문자열

```
jumin = “990515-1025075”
jumin[2]
jumin[1:4]
jumin[:]
jumin[:5]
jumin[-3:]
```

```
.lower()
.upper()
.isupper()
.replace()
.index()
.find()
.count()
```

데이터형 변환 (Casting)

```
a = 123
b = “456”

str(a)    # 문자열로 변환, string
int(b)    # 정수형으로 변환, integer
float(b)  # 실수형으로 변환, floating-point number
bool(a)   # 논리형으로 변환, boolean
```

연산자 (Operator)

Operator	Name	Example
+	Addition	x + y
-	Subtraction	x - y
*	Multiplication	x * y
/	Division	x / y
%	Modulus	x % y
**	Exponentiation	x ** y
//	Floor division	x // y

나머지

제곱

몫

연산자 (Operator)

Operator	Example	Same As
=	x = 5	x = 5
+=	x += 3	x = x + 3
-=	x -= 3	x = x - 3
*=	x *= 3	x = x * 3
/=	x /= 3	x = x / 3
%=	x %= 3	x = x % 3
//=	x //= 3	x = x // 3
**=	x **= 3	x = x ** 3
&=	x &= 3	x = x & 3
|=	x |= 3	x = x | 3
^=	x ^= 3	x = x ^ 3
>>=	x >>= 3	x = x >> 3
<<=	x <<= 3	x = x << 3

나머지

몫

제곱

비트 연산 (and 연산)

비트 연산 (or 연산)

비트 연산 (xor 연산) →

비트 연산 (shift 연산)

비트 연산 (shift 연산)

X	Y	XOR
0	0	0
1	0	1
0	1	1
1	1	0

연산자 (Operator)

비교연산

Operator	Name	Example
==	Equal	x == y
!=	Not equal	x != y
>	Greater than	x > y
<	Less than	x < y
>=	Greater than or equal to	x >= y
<=	Less than or equal to	x <= y

연산자 (Operator)

논리연산

Operator	Description	Example
and	Returns True if both statements are true	x < 5 and x < 10
or	Returns True if one of the statements is true	x < 5 or x < 4
not	Reverse the result, returns False if the result is true	not(x < 5 and x < 10)

Operator	Description	Example
is	Returns True if both variables are the same object	x is y
is not	Returns True if both variables are not the same object	x is not y

```
x = ["apple", "banana"]
y = ["apple", "banana"]
z = x

print(id(x), id(y), id(z))
print(x is z)     # True
Print(x is y)     # False
print(x == y)  # True
```

연산자 (Operator)

Operator	Description	Example
in	Returns True if a sequence with the specified value is present in the object	x in y
not in	Returns True if a sequence with the specified value is not present in the object	x not in y

Operator	Name	Description	Example
&	AND	Sets each bit to 1 if both bits are 1	x & y
|	OR	Sets each bit to 1 if one of two bits is 1	x | y
^	XOR	Sets each bit to 1 if only one of two bits is 1	x ^ y
~	NOT	Inverts all the bits	~x 보수
<<	Zero fill left shift	Shift left by pushing zeros in from the right and let the leftmost bits fall off	x << 2
>>	Signed right shift	Shift right by pushing copies of the leftmost bit in from the left, and let the rightmost bits fall off	x >> 2

함수 (function)

```
round()          # 반올림
floor()          # 내림
ceil()           # 올림
pow              # 제곱
sqrt()           # 루트(제곱근)
abs()            # 절대값
max()
min()

random(), randrange(), randint()
sample(), shuffle()
......
```

```
len()
print()
id()
type()
......
```

```
from math import *
from random import *

round(3.5)

import math
import random

round(3.5)  →  syntax error
math.round(3.5)
```

함수 (function)


```
x = -2.5
y = abs(x)      # absolute value
print(y)        # 2.5
```

함수 (function)

1. 함수 만들기

define parameter

```
def 함수이름 (매개변수명1, 매개변수명2, ... ):
    내용
    return y1, y2, ...
```

들여쓰기

2. 함수 사용하기1

```
def test(x1, x2, x3):
    y1 = x1 + x2 + x3
    y2 = abs(x1 - x2 - x3)
    y3 = x1 + x2 * x3
    return y1, y2, y3

print(test(5, 4, 2))        # 11, 1, 13
```

인수(argument)

3. 함수 사용하기2

```
def test(*args):
    result = ""
    for i in range(len(args)):
        result += str(args[i])
    return result

print(test("a", "b", "c", "d", "e", 123))
```

1. x = 출생년도 (예 1995), y = 월 (예 06), z = 일 (예 20)
   함수를 이용하여 출력 (예 1995-06-20)

2. data1, data2, data3의 값 중 가장 큰 값과 작은 값을
   함수를 이용하여 출력 (예 max: 25, min: 3)

범위 (Scope)

1. global(전역)
2. local(지역)
3. nonlocal(비지역)

```
num = 10

def change_num():
    # global num
    num = 100
    print(num)

change_num()    # 100
print(num)      # 10
```

범위 (Scope)

1. global(전역)
2. local(지역)
3. nonlocal(비지역)

```
def print_nums():
    num = 0
    def change_num():
        #nonlocal num
        num = 100
        print(num)
    change_num()
    print(num)
print_nums()
```

```
for i in range(start, stop, step):
    반복 코드
```

```
count = 10
for i in range(count):    # 0~9
        print("count =", i)

start = 2
stop = 10
step = 3
for i in range(start, stop, step):    # 2, 5, 8
        print("count =", i)
```

조건문

```
a = 10
b = 5
if a > b:
    print(a)
```

break, continue, pass

```
for i in range(100):
    if i == 20:
        print("stop", i)
        break

for i in range(10):
    if i == 5:
        continue
    else:
        pass
    print(i)
```

while

```
count = 0
while (count < 10):
    print(count)
    count += 1                    # +=, -=, *=, /=
```

1. 구구단 출력 (2단~9단)

1. list
   data = [1, 1, 8, 5, 3]

2. dictionary
   data = {“name”: “kim”, “age”: 30, “address”: “Seoul”}

3. tuple
   data = (100, 200, 300)

4. set
   data = {1, 2, 3, 4, 5}

1. list

data = [1, 1, 8, 5, 3]

index	0	1	2	3	4
index	-5	-4	-3	-2	-1

```
print(data[3])     # 5
print(data[-3])    # 8
print(data[:])     # 1, 1, 8, 5, 3
print(data)        # 1, 1, 8, 5, 3
print(data[1:3])   # 1, 8   (1부터 3미만)
```

→ list slicing
data[start:end:step]

1. list

data = [1, 1, 8, 5, 3]

index	0	1	2	3	4
index	-5	-4	-3	-2	-1

append(), pop()
insert(), clear()
index(), sort(), reverse()

초기화

```
data = [0 for i in range(10)]

data1 = ["kim", "Lee", "Seo"]
data1 = [len(i) for i in data1]
```

1. 컨볼루션(합성곱) 계산

```
data = [0, 1, 2, 3, 4, 5, 4, 3, 2, 1]
mask = [2, 1, 3]

result = ?        # [7, 13, 19, 25, 25, 23, 17, 11]
```

dict() : constructor

2. dictionary (key: value)

```
data = {“name”: “kim”, “age”: 30, “address”: “Seoul”}
data2 = {“pos1”: [30, 50, 20], “pos2”: [15,25,5]}
data3 = dict(name = "John", age = 36, country
             = "Norway")

print(data[“name”])    →   data.get(“name”)
print(data2[“pos2”])   →   data2.get(“pos3”, “없음”)
                          .keys(), .values(), .items(), .clear()
del data2[“pos2”]
```

3. tuple

```
data = (100, “apple”, 300)
data = (“apple”, )  # tuple
data = (“apple”)    # string

a = (1, 2, 3)
print(id(a))        # id() : 메모리 주소
a = a + (4, 5, 6)
print(id(a))
```

집합

4. set

```
data = {“apple”, “banana”, “cherry”}

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

Method

합집합 교집합

- add, clear, pop, remove, discard, union, intersection, update

error발생 (O) error 발생(x)

4. set

```
data = {“apple”, “banana”, “cherry”}

if “cherry” in data:
    print(“yes”)
```

```
data = [1, 1, 5, 3, 7, 5]
#data = {1, 1, 5, 3, 7, 5}
#data = (1, 1, 5, 3, 7, 5)

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

```
data = {1:"a", 3:"b", 5:"c", 7:"d"}

for i in data:
    print(data[i])
    #print(data.get(i))
```

```
class Person:
    def __init__(self, name, age):
        self.name = name
        self.age = age

p1 = Person("John", 36)

print(p1.name)
print(p1.age)
```

__init__ : constructor