

임베디드 보드 실습과 응용 프로그램

Chapter 2.

파이썬 패키지 사용하기

최영근

010-5898-3202

파이썬 언어의 특징

- 대화식 인터프리터 언어
 - 플랫폼에 독립적인 언어
 - 쉬운 문법과 다양한 자료형을 제공
 - 대규모의 라이브러리
-

파이썬 환경 구축

- 홈페이지(<https://www.python.org/downloads/>)에 접속하여 [Download] 버튼을 클릭하여 설치 파일 다운로드



Donate



Search

GO

Socialize

About

Downloads

Documentation

Community

Success Stories

News

Events

Download the latest version for Windows

Download Python 3.11.4

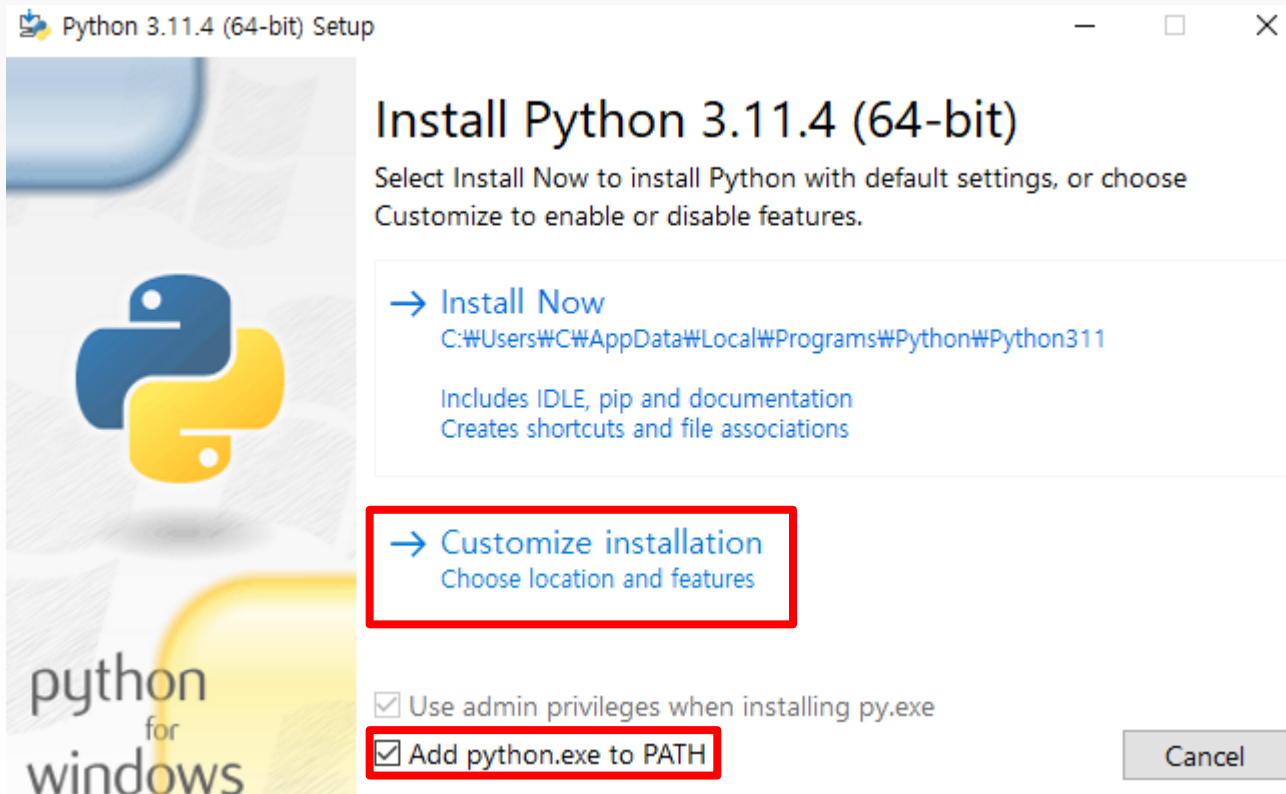
Looking for Python with a different OS? Python for [Windows](#),
[Linux/UNIX](#), [macOS](#), [Other](#)

Want to help test development versions of Python 3.12? [Prereleases](#),
[Docker images](#)



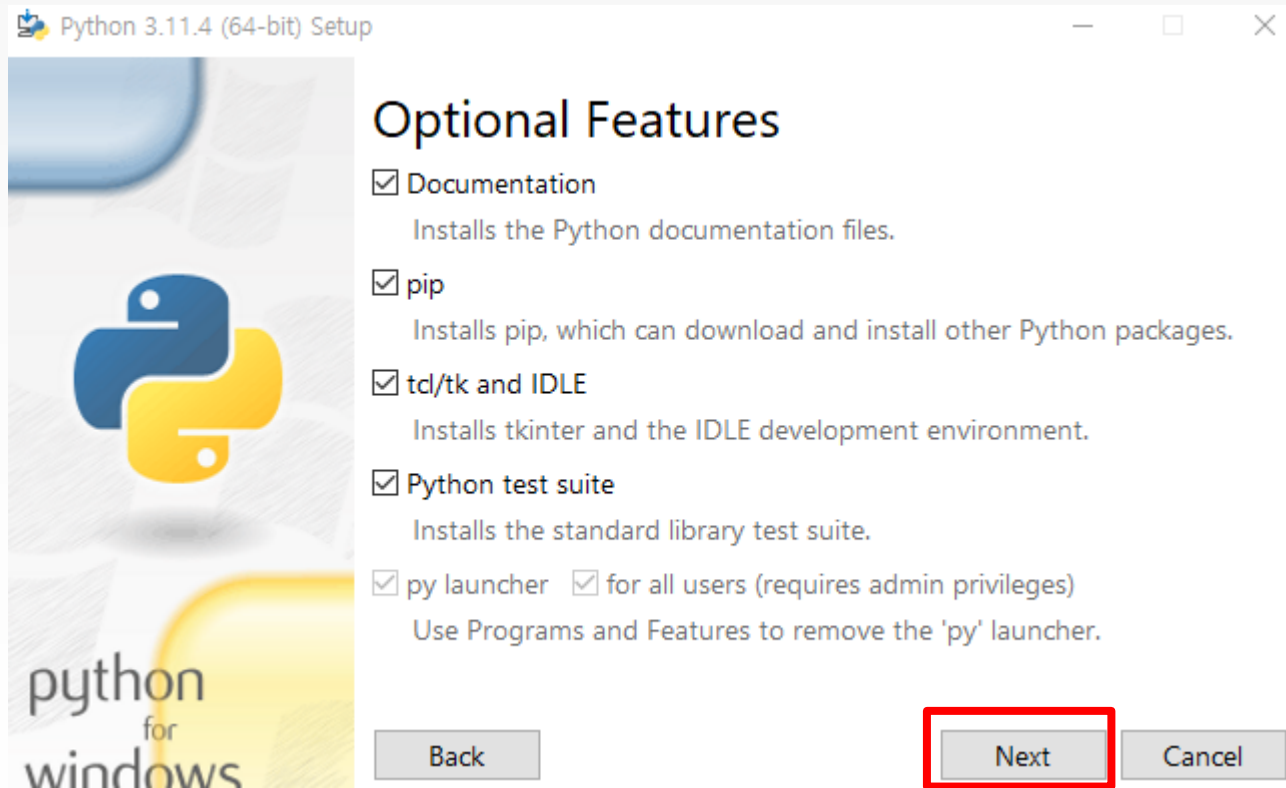
파이썬 환경 구축

- 설치 창에서 Add python.exe to PATH를 체크하고 [Customize installation]를 선택하여 설치를 진행



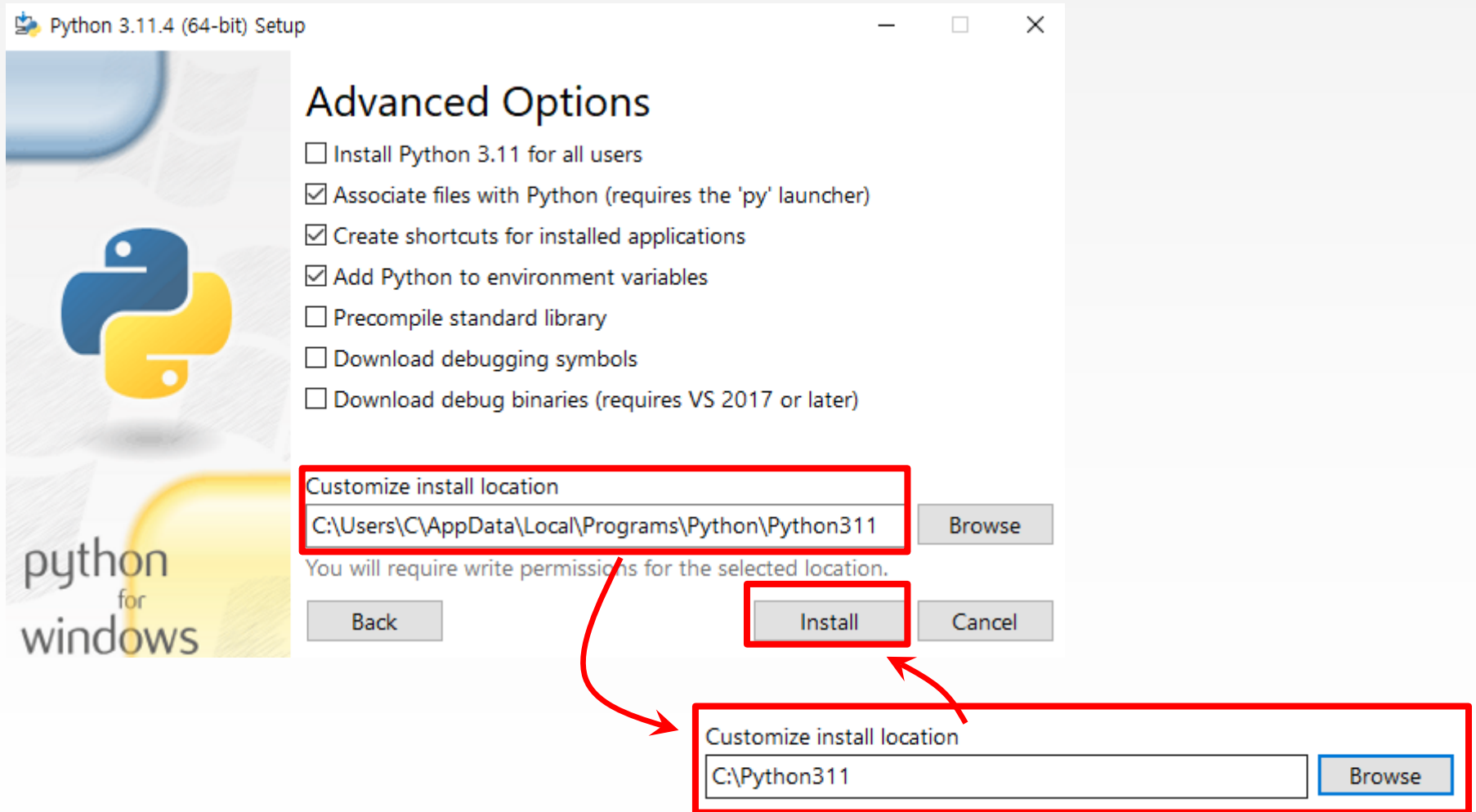
파이썬 환경 구축

- [Next]를 선택하여 설치를 진행



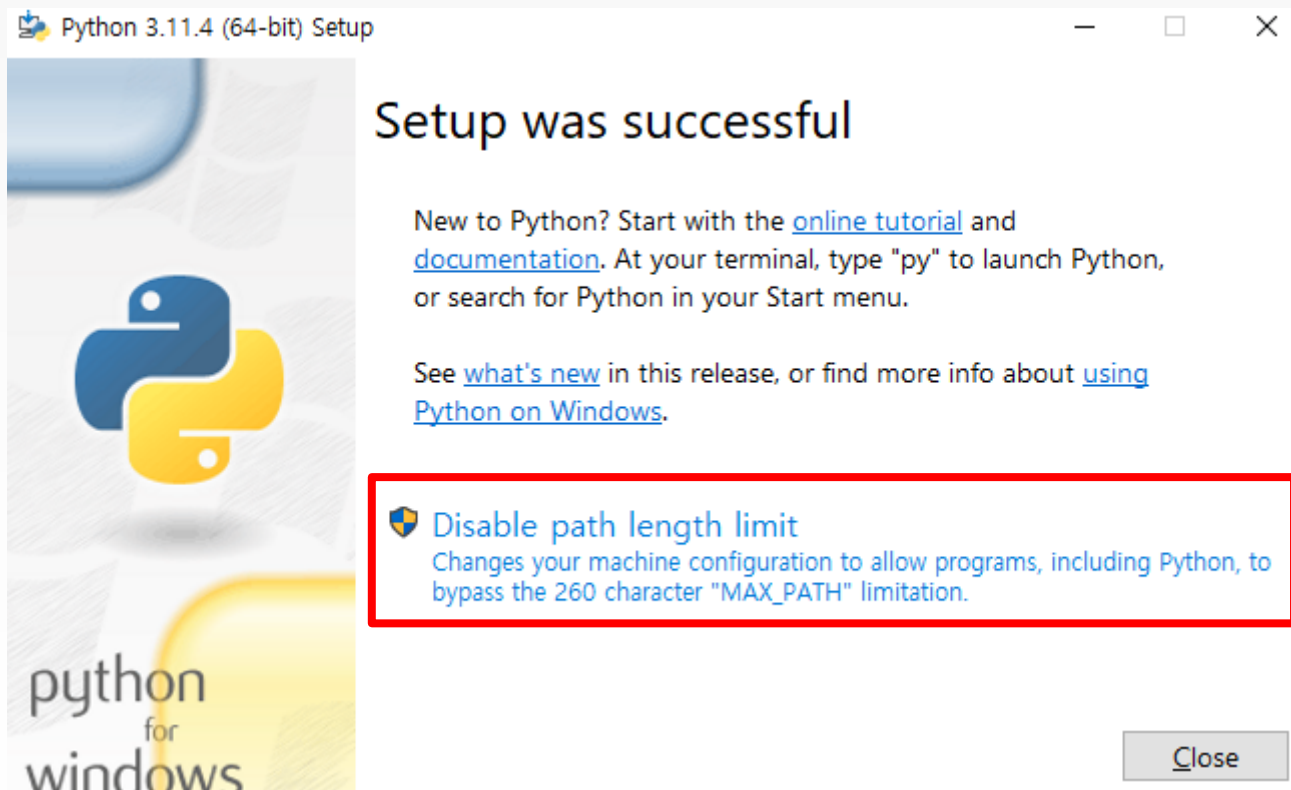
파이썬 환경 구축

- [Customize install location]를 짧게 변경한 후 Install



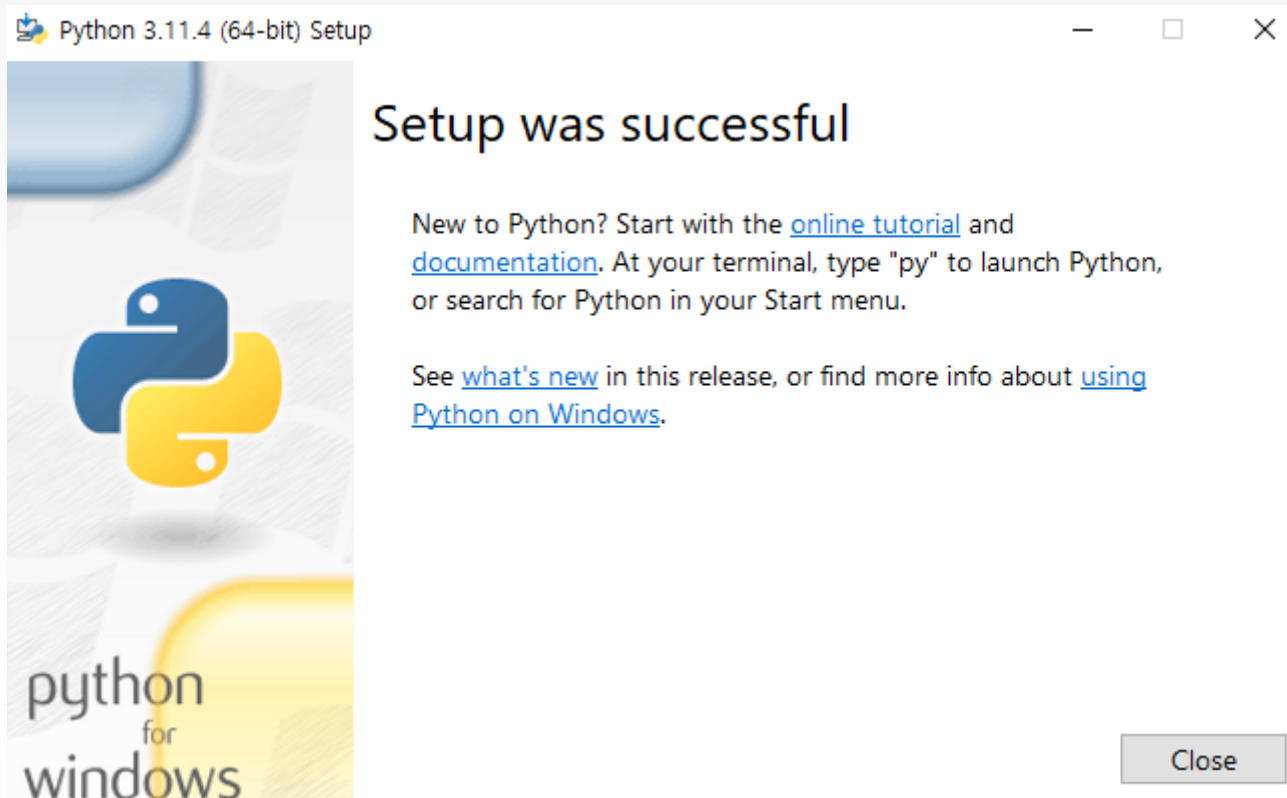
파이썬 환경 구축

- 앞 페이지의 [Customize install location]를 짧게 변경하지 않으면 설치는 되지만 아래와 같이 경로 길이를 초과했다는 메시지가 보임



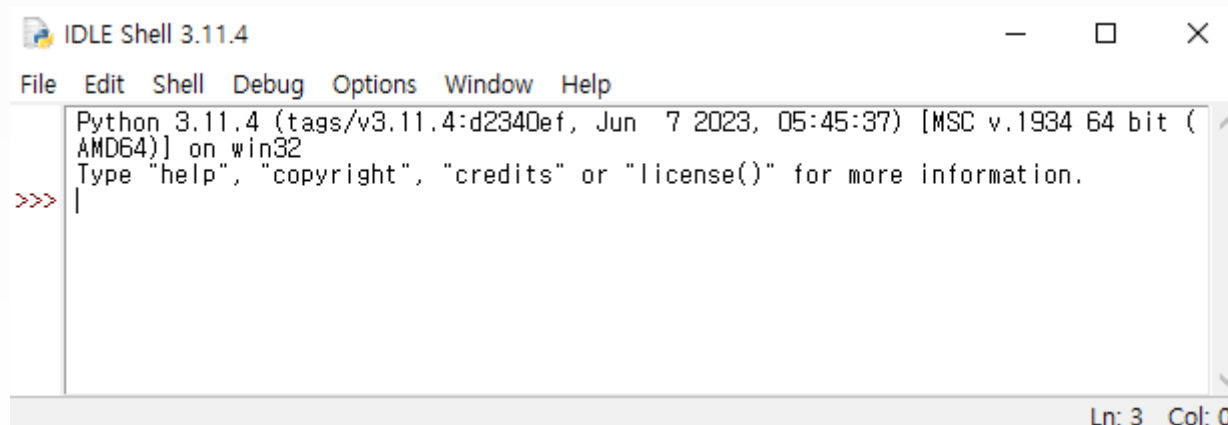
파이썬 환경 구축

- [Customize install location]를 짧게 변경했다면 성공적으로 설치가 됨



파이썬 환경 구축

- 설치된 프로그램에서 파이썬 실행 파일인 'IDLE'을 찾아 실행



파이썬 환경 구축

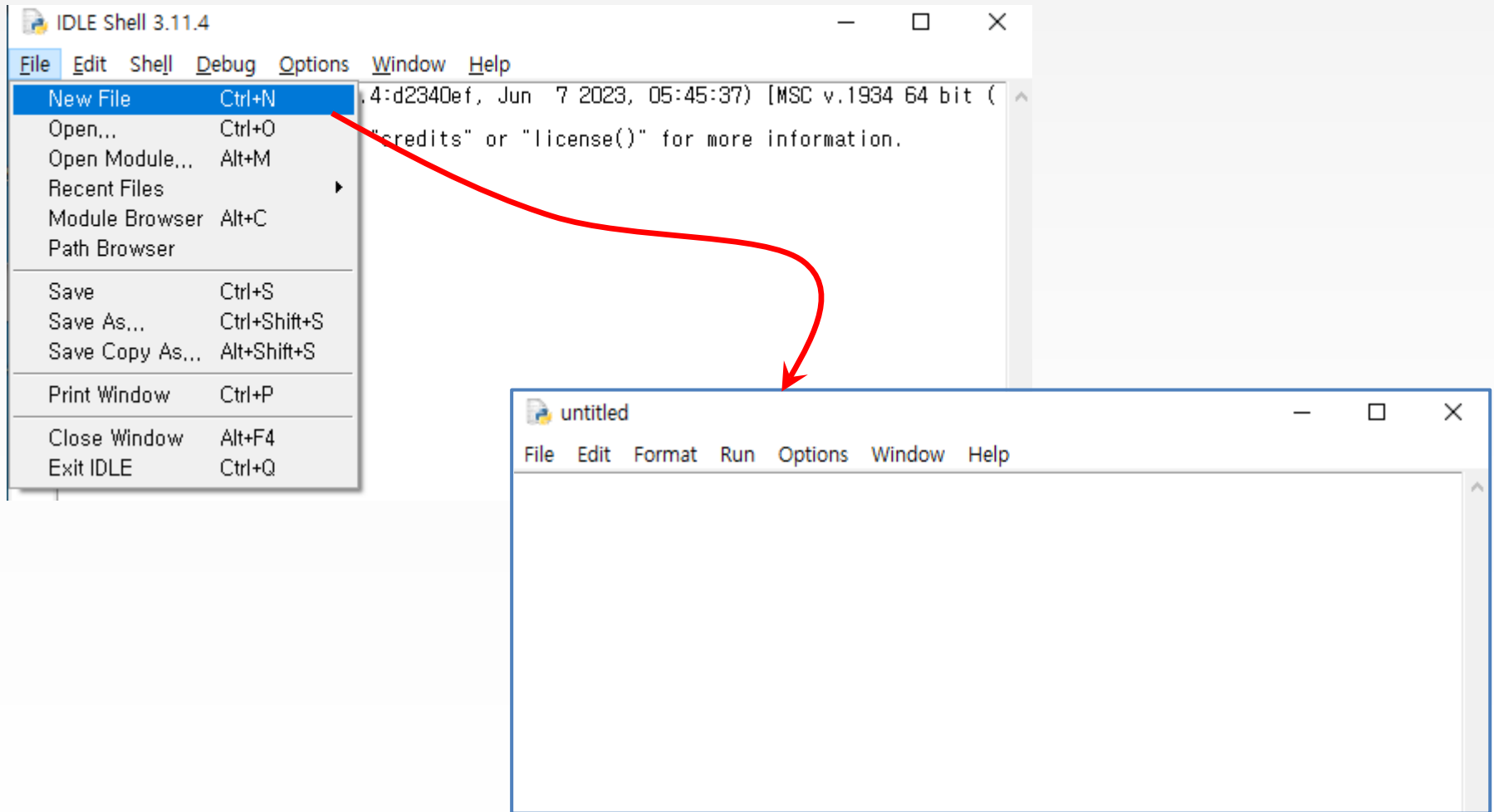
•대화형 모드



```
Python 3.11.4 (tags/v3.11.4:d2340ef, Jun 7 2023, 05:45:37) [MSC v.1934 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> a=0
>>> a
0
>>> b=10
>>> b
10
>>> sum=a+b
>>> sum
10
>>> print(a,b,sum)
0 10 10
>>> b*3
30
>>> b/3
3.3333333333333335
>>> b%3
1
>>> b//3
3
>>>
```

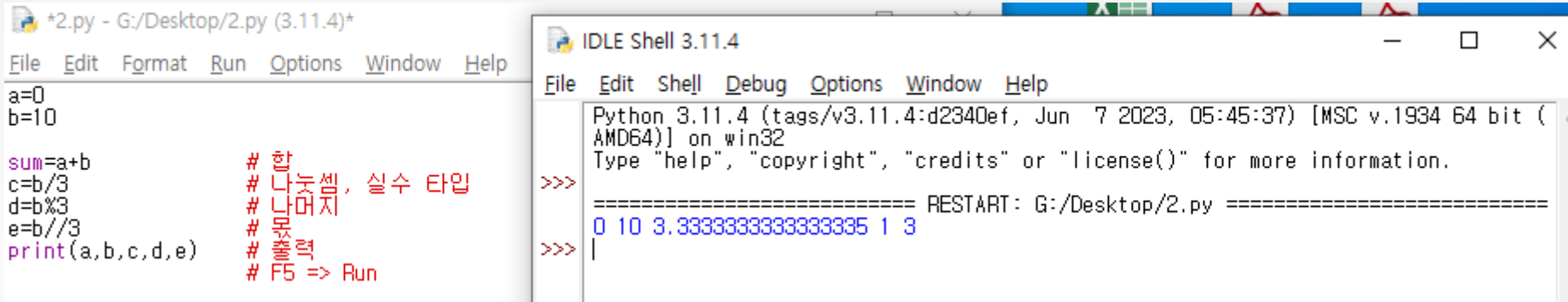
파이썬 환경 구축

•스크립트 모드



파이썬 환경 구축

•스크립트 모드



The screenshot shows two windows from the Python IDLE 3.11.4 application. The left window, titled '*2.py - G:/Desktop/2.py (3.11.4)*', contains the following Python code with red comments: `a=0`, `b=10`, `sum=a+b` (comment: # 합), `c=b/3` (comment: # 나눗셈, 실수 타입), `d=b%3` (comment: # 나머지), `e=b//3` (comment: # 몫), and `print(a,b,c,d,e)` (comment: # 출력). A red comment at the bottom says '# F5 => Run'. The right window, titled 'IDLE Shell 3.11.4', shows the execution output: 'Python 3.11.4 (tags/v3.11.4:d2340ef, Jun 7 2023, 05:45:37) [MSC v.1934 64 bit (AMD64)] on win32', 'Type "help", "copyright", "credits" or "license()" for more information.', and a prompt '>>>'. After pressing F5, the output shows '===== RESTART: G:/Desktop/2.py =====' followed by the printed values '0 10 3.3333333333333335 1 3' on a new line, with another prompt '>>>' below it.

```
*2.py - G:/Desktop/2.py (3.11.4)*
File Edit Format Run Options Window Help
a=0
b=10

sum=a+b          # 합
c=b/3            # 나눗셈, 실수 타입
d=b%3            # 나머지
e=b//3           # 몫
print(a,b,c,d,e)  # 출력
# F5 => Run

IDLE Shell 3.11.4
File Edit Shell Debug Options Window Help
Python 3.11.4 (tags/v3.11.4:d2340ef, Jun 7 2023, 05:45:37) [MSC v.1934 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: G:/Desktop/2.py =====
0 10 3.3333333333333335 1 3
>>>
```

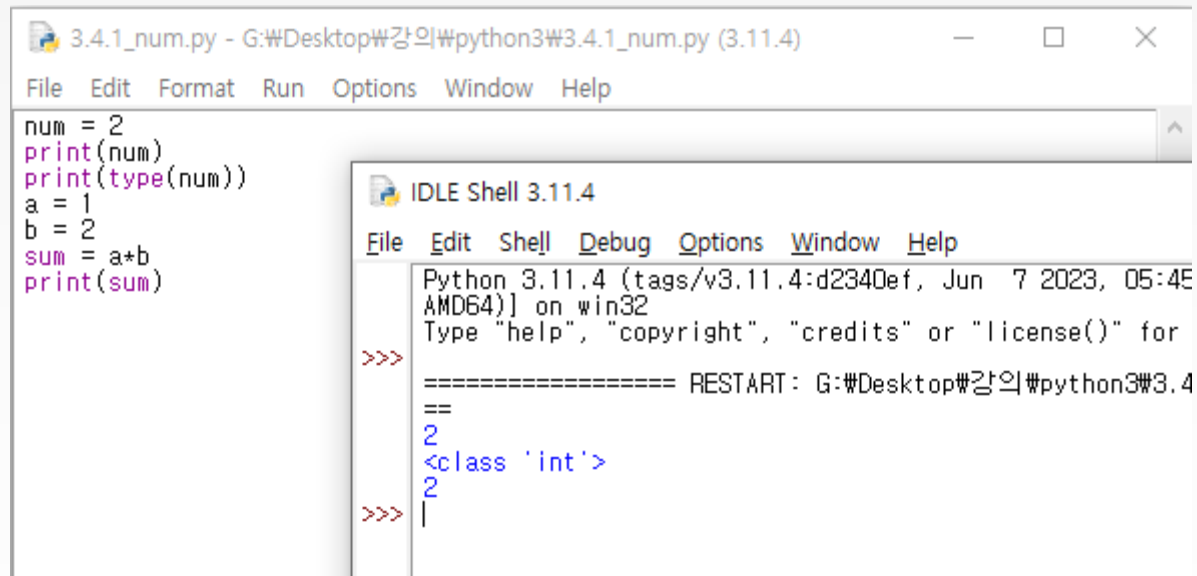
- 다음과 같이 cmd 프롬프트에서 파이썬 파일을 실행하는 것도 가능

```
C:\Users\WC>python g:\desktop\2.py
0 10 3.3333333333333335 1 3
```

파이썬 자료형

•숫자형 자료

- 정수형 자료



The screenshot displays the Python IDLE environment. The main editor window, titled '3.4.1_num.py', contains the following Python code:

```
num = 2
print(num)
print(type(num))
a = 1
b = 2
sum = a+b
print(sum)
```

The 'IDLE Shell 3.11.4' window shows the output of running this script. It displays the Python version and architecture, followed by a restart message. The execution results are shown as follows:

```
>>>
===== RESTART: G:\Desktop\강의\python3\3.4.1_num.py (3.11.4)
=====
2
<class 'int'>
2
>>>
```

파이썬 자료형

•숫자형 자료

- 실수형 자료
- / 은 실수 연산, //과 %는 정수 연산

<pre>print (17/3)</pre>	5.666666666666667
<pre>print (type(17/3))</pre>	<class 'float'>
<pre>print (17//3)</pre>	5
<pre>print (17%3)</pre>	2

파이썬 자료형

•숫자형 자료

- 복소수형 자료
- 실수부는 real, 허수부는 imag

```
a = 1+3j  
print (a.real, a.imag)  
print (type(a))
```

```
=====  
1.0 3.0  
<class 'complex'>
```

파이썬 자료형

•문자형 자료

- C언어 등과 달리 1개의 문자도 문자열 취급
- " 또는 ' 에 의해 문자열 출력 가능

<pre>print("Hello") print('Hi')</pre>	<pre>Hello Hi</pre>
---	-------------------------

파이썬 자료형

•형식

- C언어와 마찬가지로 %s, %d, %f 등 출력 형식을 지정 가능

```
print("%s" % "Hello. I'm Raspberry Pi~")  
print("%d" % 78)  
print("%f" % 1.23456)
```

```
Python 3.10.0 Shell  
Hello. I'm Raspberry Pi~  
78  
1.234560
```

```
print("%d" % 78)  
print("%d %x" % (78, 78))  
print("%.0f" % 1.23456)  
print("%.2f" % 1.23456)  
print("%.4f" % 1.23456)
```

```
78  
78 4e  
1  
1.23  
1.2346
```

파이썬 자료형

•형식

- 파이썬 3 버전 이후부터는 format 함수에 의해 출력 형식 지정 가능

<code>print("{}".format(78))</code>	78
<code>print("{} {}".format(78, 78))</code>	78 4e
<code>print("{:.0f}".format(1.23456))</code>	1
<code>print("{:.2f}".format(1.23456))</code>	1.23
<code>print("{:.4f}".format(1.23456))</code>	1.2346

파이썬 연산자

•논리 연산자

- and, or, not
- True는 1, False는 0 으로 대체 가능

<code>print(True and True)</code>	<code># True and True</code>	<code>= True</code>
<code>print(True and False)</code>	<code># True and False</code>	<code>= False</code>
<code>print(True or False)</code>	<code># True or False</code>	<code>= True</code>
<code>print(False or False)</code>	<code># False and False</code>	<code>= False</code>
<code>print(not False)</code>	<code># not False</code>	<code>= True</code>
<code>print(not True)</code>	<code># not True</code>	<code>= False</code>

파이썬 연산자

•비교 연산자

- ==, !=, >, >=, <, <=
- if문 등에서 사용

```
print(4 == 4)    # True
print(4 != 4)    # False
print(4 > 4)     # False
print(4 >= 4)    # True
print(4 <= 4)    # True
print(4 < 4)     # False
```

파이썬 제어문

•조건문 - if

- if 조건: 의 형식
- 블록 {} 이 따로 없으며 들여쓰기로 블록을 설정

```
x = 3
if x == 3:                # x가 3이면 if문 이하 실행
    print("x == 3")      # 반드시 들여쓰기를 해야 함
```

파이썬 제어문

•조건문 – if~else

- if 조건:
 else: 의 형식
- 블록 {} 이 따로 없으며 들여쓰기로 블록을 설정

```
x = 3
if x != 3:                                # x가 3이 아니면 if문 이하 실행
    print("x != 3")
else:
    print("x == 3")                        # x가 3이면 else문 이하 실행
```

파이썬 제어문

•조건문 – if~elif~else

- if 조건:
elif 조건:
else: 의 형식
- 블록 {} 이 따로 없으며 들여쓰기로 블록을 설정

```
x = 1
if x == 3:                # x가 3이면 if문 이하 실행
    print("x == 3")
elif x == 4:             # y가 4이면 elif문 이하 실행
    print("y == 4")
else:
    print("x != 3 and x != 4") # x가 3이 아니고,
                                # 4도 아니면 else문 이하 실행
```

파이썬 제어문

•반복문 – for

- for 변수 in range(반복 횟수):
- '변수'의 초기값은 0이며, 변수의 값이 1씩 증가하면서 'range'의 반복 횟수만큼 반복 실행
- 블록 {} 이 따로 없으며 들여쓰기로 블록을 설정

```
sum = 0
```

```
for i in range(11):    # 0 ~ 10까지 11번 반복
    sum = sum + i
    print("i = ", i)
```

```
print("sum = ", sum)    # "sum =" 문자열과 함께 출력을 위해 ", "로 구분
```


파이썬 제어문

•반복문 – for

- for 변수 in range(시작, 끝, 증가):
- '변수'의 초기값은 'range'의 '시작'이며, 변수의 값이 'range'의 '증가'만큼씩 증가하면서 'range'의 '끝' 값이 될 때까지 반복 실행
- 블록 {} 이 따로 없으며 들여쓰기로 블록을 설정

```
sum = 0
for i in range(1, 11, 2): # 1 ~ 10까지 2씩 증가하여 5회 반복
    sum = sum + i
    print("i = ", i)

print("sum = ", sum)
```

파이썬 제어문

•반복문 – for

- for 변수 in range(시작, 끝, 증가):

```
sum = 0
for i in range(10, 1, -2): # 10 ~ 2까지 2씩 감소하여 5회 반복
    sum = sum + i
    print("i = ", i)

print("sum = ", sum)
```

파이썬 제어문

•반복문 – while

- while 조건식: 의 형식

```
sum = 0
i = 1
while i < 11:           # i는 1부터 시작해서 10까지 증가
                        # 총 10번 반복
    sum = sum + i
    print("i = ", i)
    i = i + 1

print("sum = ", sum)
```

파이썬 제어문

•반복문 – break

- for나 while문에서 반복문을 빠져 나오도록 함

```
sum = 0
i = 1
while i < 11:                # 1부터 시작해서 10까지 10번 반복
    sum = sum + i
    print("i = ", i)
    if( i == 5 ) : break
    # 반복중에 i가 5가 되면 while 루프를 빠져 나감
    i = i + 1

print("sum = ", sum)
```

파이썬 제어문

•반복문 – continue

- for나 while문에서 반복문의 처음으로 실행 위치를 이동시킴

```
sum = 0
i = 0
while i < 11:                # i변수를 1부터 시작해서 10까지 10번 반복
    i = i + 1
    if i % 2 == 0:           # "%" 는 나머지 연산자, 2로 나누어서 나머지가 0이면 짝수
        print("i = ", i)
        sum = sum + i
    else:
        continue

print("sum = ", sum)
```

파이썬 제어문

•try ~ except

- 인터럽트 처리
- ex) 키보드 인터럽트 처리

```
try:
    while True:
        print("Hello")
except KeyboardInterrupt:
    pass
```

- : 이 포함된 for, while, try 등은 1개 이상의 문장을 실행해야만 하며, 아무것도 실행하고 싶지 않다면 pass 문을 사용

사용자 입력

•input() 함수

```
math = int(input("math="))  
eng = int(input("eng="))  
print(math+eng)
```

```
math=60  
eng=80  
140
```

- int 대신 float을 사용하면 실수형으로 변환

```
math = float(input("math="))  
eng = float(input("eng="))  
print(math+eng)
```

```
math=60.75  
eng=82.45  
143.2
```

파이썬 함수

•def 함수 이름():

```
def sum():  
    return (10+20)  
  
print("sum=", sum())
```

- 함수가 아래에 위치하면 오류가 발생

```
print("sum=", sum())
```

```
def sum():  
    return (10+20)
```


파이썬 함수

- 함수의 인자와 리턴 값

```
def sum(math, eng):  
    return (math+eng)  
  
print("sum=", sum(30, 40))
```

파이썬 함수

•언패킹 인자 전달

```
def sum(math, eng, kor):  
    return (math+eng+kor)  
  
score = [90, 80, 100]  
print("sum=", sum(*score))
```

- [] 는 리스트(list)로서 C언어 등의 배열(array)와 유사
서로 다른 data type의 자료를 저장 가능
- 변수명 앞에 * 를 삽입하면 리스트 내의 데이터들을 순서대로 1개씩 모
두 전달
- 결과 :
- sum == 270

파이썬 함수

•가변 인자 전달

- 함수의 인자로 전달할 데이터의 수가 가변적일 경우 사용
- 함수 선언부의 변수 앞에 * 를 삽입

```
def sum(*scores):  
    sum = 0  
    for arg in scores:                # 가변인자값은 for문을 이용  
        sum = sum + arg  
  
    return (sum)  
  
print("sum=", sum(30,40,80,90)) # 4개의 인자 전달  
print("sum=", sum(30,40,80))    # 3개의 인자 전달
```

파이썬 함수

•여러 개의 값 리턴

- return할 때 , 로 구분해서 값을 전달하면 여러 개의 값을 리턴함

```
def sum_avg(*scores):  
    sum = 0      # 합계  
    avg = 0      # 평균  
    count = 0    # 가변 인자 개수 카운트  
    for arg in scores:  
        sum = sum + arg  
        count = count + 1    # for문이 실행될 때마다 1 증가  
                                # count += 1  
    avg = sum / count        # 평균 계산  
    return (sum, avg)        # 합계, 평균. 2개의 값 리턴
```

```
ret_sum = 0  
ret_avg = 0
```

```
ret_sum, ret_avg = sum_avg(30,40,80,90)  
print("sum=", ret_sum, ", avg=", ret_avg)
```

```
ret_sum, ret_avg = sum_avg(30,40,80)  
print("sum=", ret_sum, ", avg=", ret_avg)
```

파이썬 함수

•전역 변수 / 지역 변수

```
sum = 0 # 전역 변수  
avg = 0 # 전역 변수
```

```
def sum_avg(*scores):  
    count = 0  
    global sum #global 키워드 사용  
    for arg in scores:  
        sum = sum + arg  
        count = count + 1  
  
    global avg #global 키워드 사용  
    avg = sum / count
```

```
sum = 0 # sum 변수 초기화  
sum_avg(30,40,80,90)  
print("sum=", sum, ", avg=", avg)
```

```
sum = 0 # sum 변수 초기화. 이 라인을 삭제한다면?  
sum_avg(30,40,80)  
print("sum=", sum, ", avg=", avg)
```

파이썬 모듈

•모듈

- 여러 클래스, 함수, 변수를 가지고 있는 .py 파일
- 아래 내용을 작성한 후 파일명 module_sum.py 로 저장

```
class class_sum:
    math = 0
    eng = 0

    def set_score(self, math, eng):
        self.math = math
        self.eng = eng
        return

    def do_sum(self):
        return (self.math+self.eng)
```

파이썬 모듈

•모듈

- module_sum.py 와 같은 폴더(디렉토리) 내에 아래 코드를 저장한 다음 실행하면 sum = 110 이 출력됨

```
import module_sum  
  
sum = module_sum.class_sum()  
  
sum.set_score(30, 80)  
  
print("sum=", sum.do_sum())
```

파이썬 모듈

•내장 모듈

- import 모듈명

```
import math  
  
print(math.sqrt(2.0))  
help('modules')
```

- help('modules')는 현재 사용 가능한 모든 모듈명을 표시해줌
- import 모듈명 as 내가 사용할 모듈명

```
import math as m  
  
print(m.sqrt(2.0))  
help('math')
```

- help('math')는 math 모듈에서 사용 가능한 함수들을 표시해줌

파이썬 모듈

•내장 모듈

- time

```
import time

try:
    while True:
        print("Hello")
        time.sleep(1)
except KeyboardInterrupt:
    pass
```

- time.sleep() 함수는 sec 단위 delay

파이썬 패키지

• 모듈들의 집합

- cmd 창에서 아래와 같이 입력해서 실행

```
C:\Users\WC>pip install opencv-python
```

- 아래와 같이 pip 버전을 업데이트하라는 에러가 발생

```
ERROR: Could not find a version that satisfies the requirement opencv (from versions: none)
ERROR: No matching distribution found for opencv
```

```
[notice] A new release of pip is available: 23.1.2 -> 23.2.1
[notice] To update, run: python.exe -m pip install --upgrade pip
```

- 녹색 명령어를 drag - 복사 - 붙여넣기 하면 업데이트됨

```
C:\Users\WC> python.exe -m pip install --upgrade pip
Requirement already satisfied: pip in c:\python311\lib\site-packages (23.1.2)
Collecting pip
  Downloading pip-23.2.1-py3-none-any.whl (2.1 MB)
  ----- 2.1/2.1 MB 9.5 MB/s eta 0:00:00
```

파이썬 패키지

• 모듈들의 집합

- 업데이트 후 다시 cmd 창에서 아래와 같이 입력해서 실행
- 다시 타이핑할 필요 없이 키보드의 방향 키로 직전에 입력했던 명령어를 선택할 수 있음

```
C:\Users\WC>pip install opencv-python
```

- 설치가 완료되면 아래와 같이 numpy, opencv가 설치되었다는 메시지가 출력됨

```
Installing collected packages: numpy, opencv-python  
Successfully installed numpy-1.25.2 opencv-python-4.8.0.74
```

파이썬 패키지

• 모듈들의 집합

- 라즈베리 파이에서는 프롬프트에서 아래와 같이 입력해서 설치
-
- `$ sudo apt update`
- `$ sudo apt install python3-opencv`

파이썬 패키지

•OpenCV 패키지 사용해보기

- photo.jpg 파일을 .py 파일과 같은 폴더(디렉토리)에 저장한 다음 아래 코드를 실행. 반드시 같은 폴더에 저장해야 함

```
import cv2

img = cv2.imread('photo.jpg')    # image 파일 읽기

cv2.imshow('photo', img)          # image 파일 출력
cv2.waitKey(0)                    # 키보드 입력 대기
cv2.destroyAllWindows()           # 키보드 입력이 되면 창 닫기
```

파이썬 패키지

•OpenCV 패키지 사용해보기

- 회색조로 변환

```
import cv2

img = cv2.imread('photo.bmp')    # image 파일 읽기
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
cv2.imshow('photo', gray)        # color을 BGR에서 회색조로 convert
cv2.waitKey(0)                   # image 파일 출력
cv2.destroyAllWindows()          # 키보드 입력 대기
cv2.destroyAllWindows()          # 키보드 입력이 되면 창 닫기
```

파이썬 패키지

•OpenCV 패키지 사용해보기

- 얼굴 인식
- haarcascade_frontalface_default.xml 파일을 .py 파일과 같은 폴더(디렉토리)에 저장한 다음 아래 코드를 실행.

```
import cv2

img = cv2.imread('photo.jpg')
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

face_cascade = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
# CascadeClassifier(다단계 분류기) 객체를 생성한 후
# 변수 face_cascade가 가리키도록 함
# haarcascade_frontalface_default.xml 파일은
# 얼굴의 앞면을 검출하기 위해
# 머신 러닝으로 미리 학습시켜 놓은 분류기 파일
faces = face_cascade.detectMultiScale(gray, 1.3, 5)
# detectMultiScale 함수는
# 지정된 그림 파일(gray)을
# 1.3배만큼씩 축소, 즉 30%씩 축소해가며 얼굴을 검출
# 그 과정에서 5회 검출이 되어야 얼굴로 최종 인식
for (x,y,w,h) in faces:
    img = cv2.rectangle(img,(x,y),(x +w,y +h),(255,0,0),2)
    # 검출된 얼굴의 좌표 : x, y
    # 검출된 얼굴의 크기 : w, h
    # rectangle 형태를 이미지 파일에 추가

cv2.imshow('photo', img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

파이썬 패키지

•OpenCV 패키지 사용해보기

- 눈 인식
- haarcascade_eye.xml 파일을 .py 파일과 같은 폴더(디렉토리)에 저장한 다음 아래 코드를 실행.

```
import cv2

img = cv2.imread('photo.jpg')
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

face_cascade = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
eye_cascade = cv2.CascadeClassifier('haarcascade_eye.xml')
# haarcascade_eye.xml는 눈을 검출하기 위한 분류기 파일

faces = face_cascade.detectMultiScale(gray, 1.3, 5)
for (x,y,w,h) in faces:
    img = cv2.rectangle(img, (x,y), (x+w,y+h), (255,0,0), 2)
    roi_gray = gray[y:y+h, x:x+w]
    # gray 변수가 가리키는 이미지 내의 얼굴 영역에 대한 객체를 roi_gray로
    # roi : region of interest
    roi_color = img[y:y+h, x:x+w]
    # img 변수가 가리키는 이미지 내의 얼굴 영역에 대한 객체를 roi_color로
    eyes = eye_cascade.detectMultiScale(roi_gray)
    # roi_gray 변수가 가리키는 회색의 얼굴 영역에서 눈을 검출
    for (ex,ey,ew,eh) in eyes:
        cv2.rectangle(roi_color, (ex,ey), (ex+ew,ey+eh), (0,255,0), 2)
        # 검출된 눈의 좌표 : ex, ey
        # 검출된 눈의 크기 : ew, eh
        # rectangle 형태를 2개 이미지 파일에 추가

cv2.imshow('photo', img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```


A minimalist black line drawing on a light gray background. On the left, a stylized figure is composed of a large circle at the top, a vertical line extending downwards from its center, and three horizontal lines branching out from the vertical line at different heights. The top horizontal line has a small circle at its end, the middle one has a larger circle, and the bottom one has a small circle. A horizontal line extends from the base of the vertical line across the bottom of the image. Two small red dots are placed on this horizontal line, one towards the right and one further to the right.

THANK YOU