



Python Study

Part 1

[파이썬(Python) 이란?]

1990년 개발자 '귀도 반 로섬(Guido Van Rossum)'이 개발한 인터프리터 언어

- 컴파일러 언어 : 코드의 전체를 모두 변환하여 실행 - C, C++, JAVA
- 인터프리터 언어 : 소스 코드를 한 줄 단위로 변환하고 실행 - python, HTML, SQL, Javascript



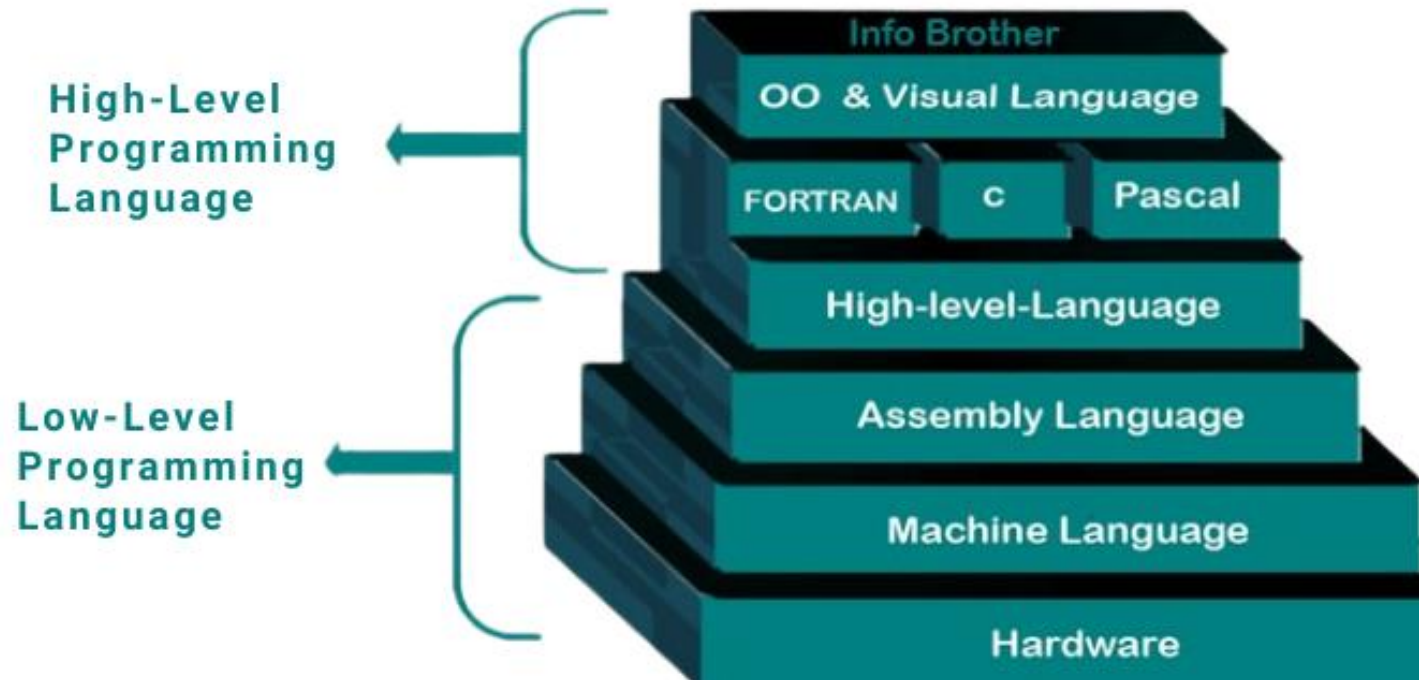
[파이썬(Python)의 장점]

Life is Short, You Need Python!

1. 비전공자 또한 읽기 쉽다. 문맥 파악이 쉽다.
2. 한 줄 단위로 실행되므로 사용자가 쉽게 결과 확인 가능
3. 문법이 쉽고 간단하다.
4. 빠른 속도
5. 매우 풍부한 라이브러리
6. 오픈 소스이다. (무료)



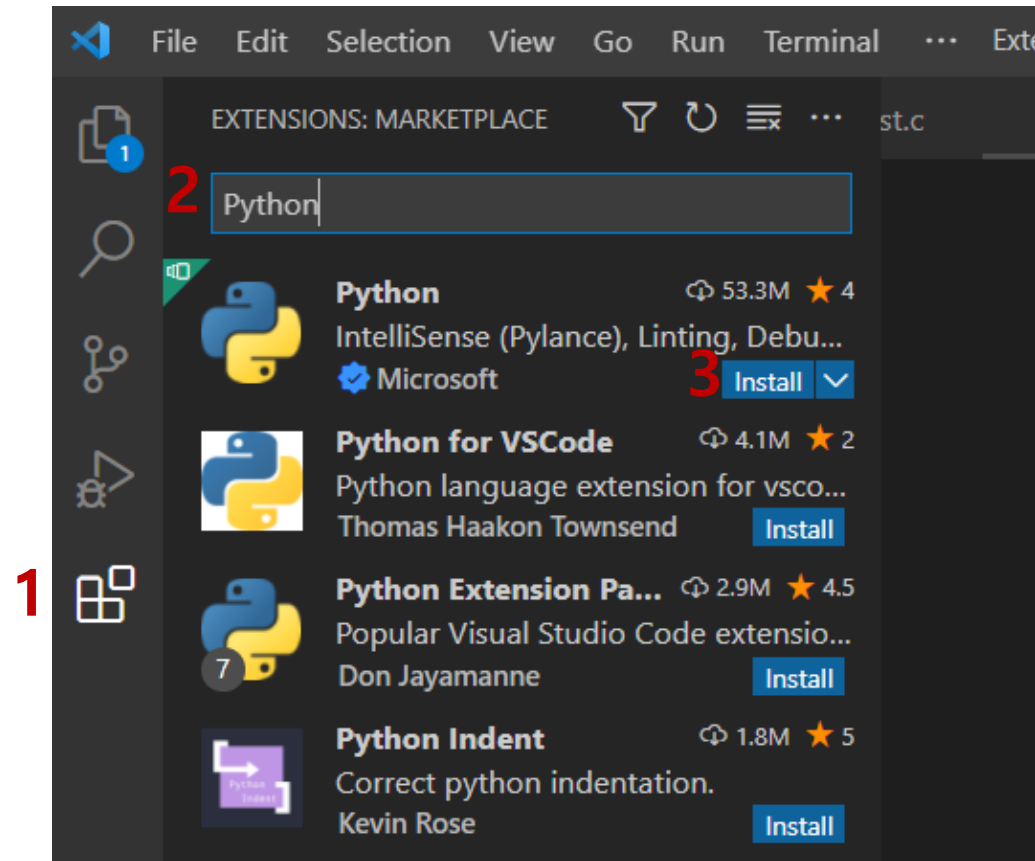
- Python : High-level programming language로, 어떤 컴퓨터 언어보다 배우기 쉽고 인간친화적



- C언어는 세미콜론(;)으로 한 줄의 끝을 표현하는데, 파이썬은 '들여쓰기'로 표현한다. 그러므로 들여쓰기가 매우 중요!
- 주석 처리는 #을 사용하거나, Ctrl+/ 을 누르면 자동 주석 처리

[Install Python]

1. 왼쪽 메뉴의 익스텐션 아이콘 누르기
2. 상단 검색창에 Python
3. Microsoft의 Python Install 버튼 누르기



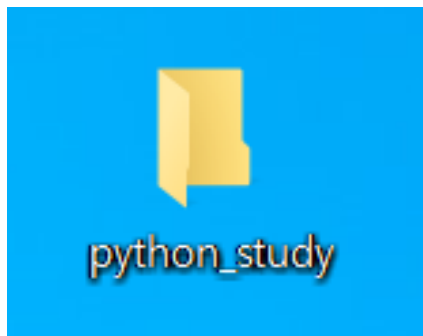
[Open Folder]

1. 코드를 저장할 폴더 만들기

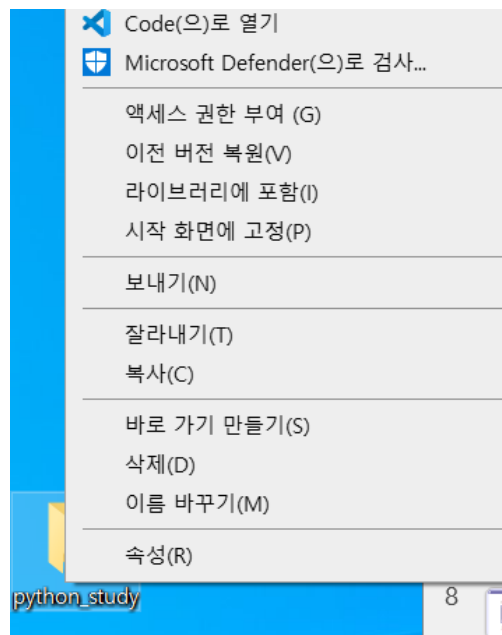
2-1. 해당 폴더 우클릭하여 vscode 열거나

2-2. vscode에서 Open Folder 하여 해당 폴더 열기

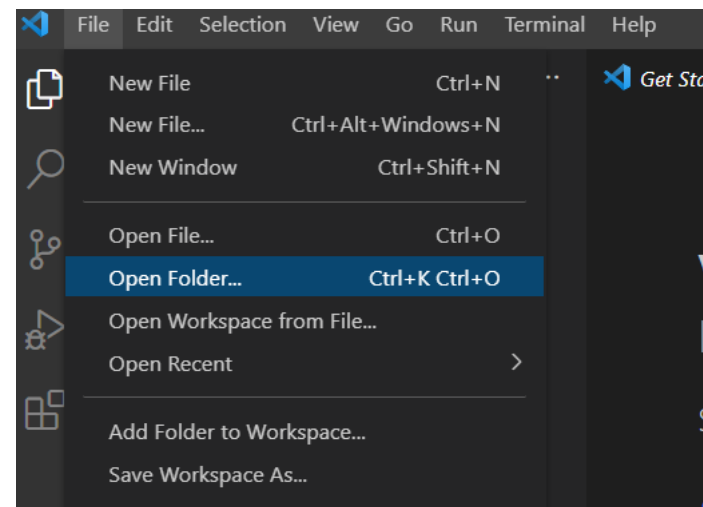
1



2-1



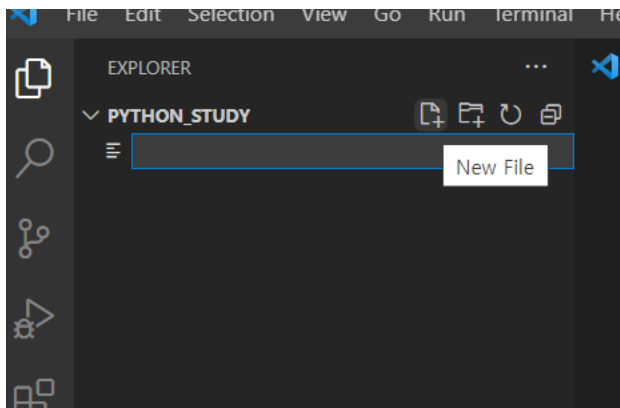
2-2



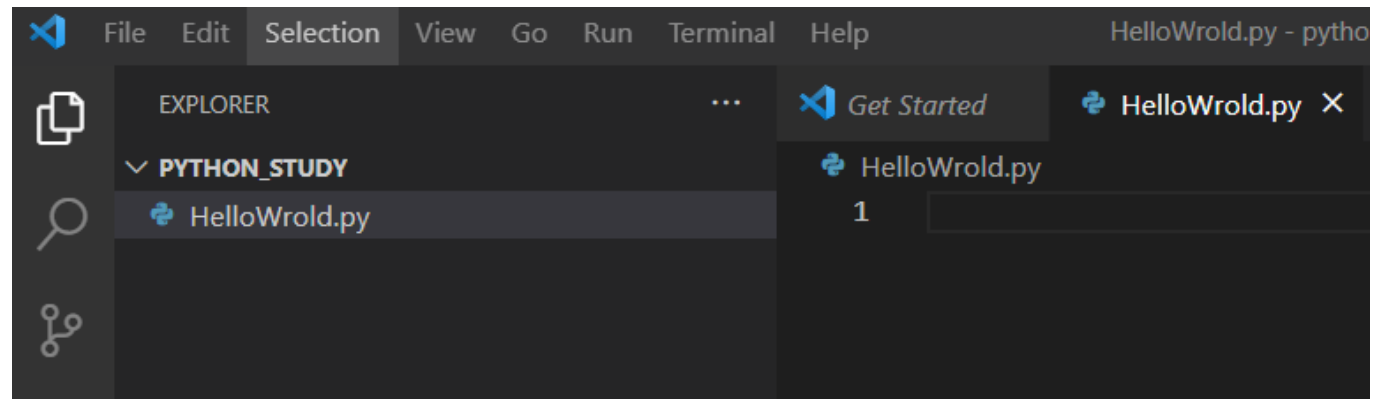
[Hello World.py]

1. 왼쪽의 Explorer 탭에서 New File 눌러 새로운 파일 만들기
2. 파일 이름 : HelloWorld.py

1

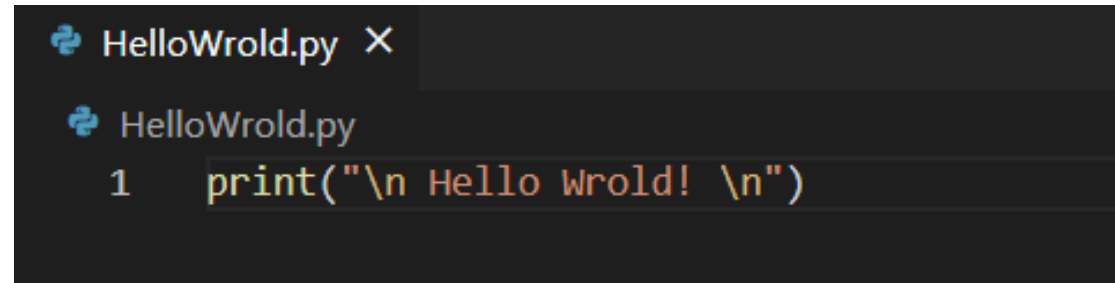


2



[Hello World.py]

1



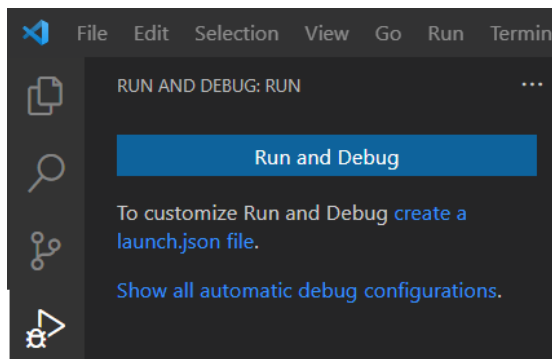
```
HelloWrold.py ×  
HelloWrold.py  
1  print("\n Hello Wrold! \n")
```

1. 코드 작성

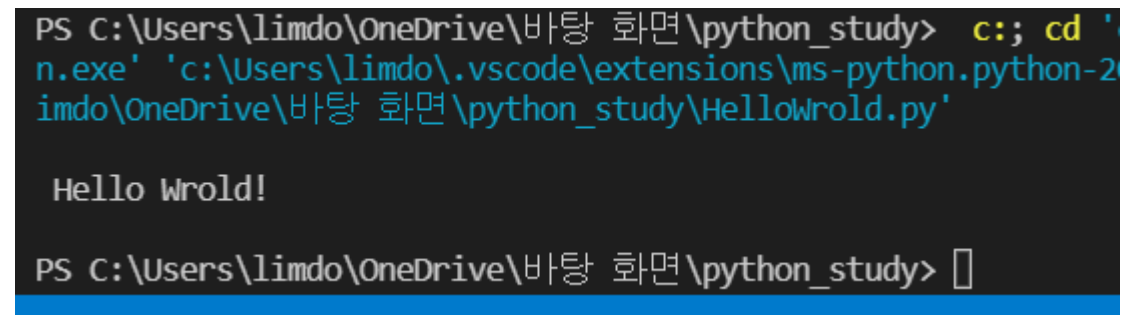
2. 왼쪽의 Run and Debug 에서 Run and Debug 누르기 (Python File)

3. Terminal에 "Hello World!" 출력 확인

2



3



[숫자형 (Number)]

항목	사용 예
정수	123, -345, 0
실수	123.45, -1234.5, 3.4e10
8진수	0o34, 0o25
16진수	0x2A, 0xFF

1. 숫자형 만들기

```
a = 1
b = -10 #음수
c = 3.14 #실수

print(a)
print(b)
print(c)
```

```
1
-10
3.14
```

[불 자료형 (Boolean)]

1. 참(True) / 거짓(False)

```
print(1 < 3)
print(1 > 3)

print(True)
print(False)

print(not True)
print(not False)
print(not (1 < 3))
```

```
True
False

True
False

False
True
False
```

[연산자]

1. 사칙연산자

연산자	기능
+	덧셈
-	뺄셈
*	곱셈
/	나눗셈

연산자	기능
**	제곱
//	몫
%	나머지
()	우선순위

```
print(5+3)      #덧셈
print(5-3)      #뺄셈
print(3*5)      #곱셈
print(3/6)      #나눗셈
print(10//6)    #나눗셈 : 몫
print(10%6)     #나눗셈 : 나머지
print(3**2)     #제곱
print(4*(3+2))  #우선순위 연산
```

```
8
2
15
0.5
1
4
9
20
```

[연산자]

2. 비교연산자

비교 연산자	기능
>, <	큰가
>=, <=	크거나 같은가
==	같은가
!=	같지 않은가

```
print(10 > 3)
print(4 >= 7)
print(3 == 3)
print(3 == (2+1))
print(3 != 3)
```

```
a = 3*5
print(a)
a += 2 #a = a + 2와 같음
print(a)
```

```
True
False
True
True
False
```

```
15
17
```

[연산자]

3. 논리연산자

논리 연산자	기능
&, and	그리고
~, or	또는

```
print((3 < 4) and (2 > 1))  
print((3 < 4) & (2 > 1))
```

```
print((3 < 4) or (2 < 1))  
print((3 < 4) | (2 < 1))
```

```
print(5 > 4 > 3)  
print(5 > 4 > 7)
```

```
True  
True
```

```
True  
True
```

```
True  
False
```

[숫자 처리 함수]

1. 파이썬에서 제공하는 숫자처리함수

논리 연산자	기능
abs()	절댓값 반환
pow(a,b)	a^b 반환
max()	최댓값 반환
min()	최솟값 반환
round()	반올림

```
print(abs(-100))
print(pow(4,3)) #4^3 = 64 반환
print(max(5, 10, 14)) #최댓값 반환
print(min(5, 10, 14)) #최솟값 반환
print(round(3.14)) #반올림
```

```
100
64
14
5
3
```

[숫자 처리 함수]

2. math 라이브러리

논리 연산자	기능
floor()	내림
ceil()	올림
sqrt()	제곱근

- 라이브러리(library) : 미리 선언되어 있는 함수의 집합
- From math import floor : math library에서 floor 함수를 쓰겠다.
- From math import * : math library로부터 모든 함수를 이용하겠다.

```
from math import floor
from math import *

print(floor(4.99)) # 내림
print(ceil(4.99)) # 올림
print(sqrt(25)) #제곱근 #5
```

```
4
5
5.0
```

[숫자 처리 함수]

3. random

논리 연산자	기능
random()	0.0 ~ 1.0 미만의 임의의 값 생성
randrange(a,b)	a부터 b 미만의 임의의 값 생성
randint(a,b)	a부터 b 이하의 임의의 정수값 생성

```

from random import *

print(random())
print(random() * 10) #0.0 ~ 10.0 미만의 임의의 값 생성
print(int(random() * 10)) #정수형

print(randrange(1,100))
print(randint(1, 100))

```

```

0.6234423453911536
8.35562021372222
4
23
5

```


[문제 1]

다음은 두 개의 숫자를 입력 받아 더하여 돌려주는 프로그램이다.

```
input1 = input("첫번째 숫자를 입력하세요:")  
input2 = input("두번째 숫자를 입력하세요:")  
  
total = input1 + input2  
print("두 수의 합은 %s 입니다" % total)
```

이 프로그램을 수행해보면, 3과 6을 입력했을 때 9가 아닌 36이라는 결과값을 돌려준다.
이 프로그램의 오류를 수정해보자.

```
첫번째 숫자를 입력하세요:3  
두번째 숫자를 입력하세요:6  
두 수의 합은 36 입니다
```

조건 1. int 함수를 사용한다.

[문자열 (String)]

1. 문자열(String) : 문자, 단어 등으로 구성된 문자들의 집합
2. 문자열 만들고 사용하기

항목	사용 예
큰따옴표(")	"Hello World"
작은따옴표(')	'Python is fun'
큰따옴표 3개("""")	""""Life is too short, You need python""""
작은따옴표 3개(''')	'''Life is too short, You need python'''

- w" w' 는 문장 내에서 따옴표

```
print('하늘')
print("풍선")
print("나는 \'학생\'입니다.")
print('맛있는 '+"크래커")
print('hi'*3)
```

```
하늘
풍선
나는 '학생'입니다.
맛있는 크래커
hihihi
```

[문자열 (String)]

1. 문자열(String) : 문자, 단어 등으로 구성된 문자들의 집합
2. 문자열 만들고 사용하기

항목	사용 예
큰따옴표(")	"Hello World"
작은따옴표(')	'Python is fun'
큰따옴표 3개(""")	"""Life is too short, You need python"""
작은따옴표 3개(''')	'''Life is too short, You need python'''

- 따옴표가 여러 가지 있는 것은 겹치지 않기 위해

```
food = "Python's favorite food is perl"
say = '"Python is very easy." he says.'
print(food)
print(say)
```

```
Python's favorite food is perl
"Python is very easy." he says.
```

[문자열 (String)]

3. 문자열 인덱싱

- Indexing : 무엇을 '가리킨다'
- 파이썬은 숫자를 0부터 센다.
- 음수의 인덱싱은 뒤에서부터 센다.

4. 문자열 슬라이싱

- Slicing : 원하는 부분을 잘라내어 가져온다.
- [(원하는 시작 인덱싱):(원하는 마지막 인덱싱+1)]
- [:a] – 앞을 비워두면 맨 처음부터
- [b:] – 뒤를 비워두면 맨 뒤까지

```
Life is too short, You need Python
0         1         2         3
0123456789012345678901234567890123
```

```
a = "Life is too short, You need Python"
print(a[0])
print(a[-1])

print(a[0:4])
print(a[:10])
print(a[19:])
print(a[19:-7])
```

```
L
N

Life
Life is to
You need Python
You need
```

[문제 2]

홍길동 씨의 주민등록번호는 881120-1068234이다. 홍길동 씨의 주민등록번호를 연월일(YYYYMMDD) 부분과 그 뒤의 숫자 부분으로 나누어 출력해 보자.

조건 1. 문자열 슬라이싱 기법 사용

Ex)

```
a = "Life is too short, You need Python"
print(a[0])
print(a[-1])

print(a[0:4])
print(a[:10])
print(a[19:])
print(a[19:-7])
```

```
Life is too short, You need Python
0      1      2      3
0123456789012345678901234567890123
```

```
L
N

Life
Life is to
You need Python
You need
```

조건 2. 출력 화면 :

```
연월일 : 881120
나머지 : 1068234
```

[문자열 (String)]

5. 문자열 처리 함수(파이썬 기본 제공)

함수	기능
lower()	다 소문자로
upper()	다 대문자로
isupper()	대문자인가
len()	문자열 길이 반환
replace(a, b)	A를 b로 대체해 반환
index()	몇 번째 인덱스에 저장되어 있는가 (원하는 값 없으면 오류 뜨며 프로그램 종료)
find()	몇 번째 인덱스에 저장되어 있는가 (원하는 값 없으면 -1 반환)
count()	몇 번 포함되는가

```
python = "Python is amazing"
print(python.lower())
print(python.upper())
print(python[0].isupper())

print(len(python))
print(python.replace("Python", "C++"))

print(python.index("n"))
print(python.find("n"))

# print(python.index("Java"))
print(python.find("Java"))
print(python.count("n"))
```

```
python is amazing
PYTHON IS AMAZING
True
17
C++ is amazing
5
5
-1
2
```

[문제 3]

다음과 같은 문자열 a:b:c:d가 있다. 문자열의 replace 함수를 사용하여 a#b#c#d로 바꿔서 출력해 보자.

```
>>> a = "a:b:c:d"
```

조건1. replace() 함수 사용

Ex)

```
python = "Python is amazing"  
print(python)  
print(python.replace("Python", "C++"))
```

```
Python is amazing  
C++ is amazing
```

조건2. 출력 화면 :

```
a = "a#b#c#d"
```

[문자열 (String)]

6. 문자열 포맷

① 방법1

```
print("나는 %d살입니다." % 20) #정수
print("pi는 %f입니다." % 3.14) #실수
print("나는 %s를 좋아해요." % "파이썬") #문자열
print("Apple은 %c로 시작해요." % "A") #한글자만

print("나는 %s색과 숫자 %d을 좋아해요." % ("파란", 7))
```

```
나는 20살입니다.
pi는 3.140000입니다.
나는 파이썬를 좋아해요.
Apple은 A로 시작해요.
나는 파란색과 숫자 7을 좋아해요.
```

② 방법2

```
print("나는 {}살입니다.".format(20))
print("나는 {}색과 숫자 {}을 좋아해요.".format("파란", 7))
print("나는 {0}색과 숫자 {1}을 좋아해요.".format("파란", 7))
print("나는 {1}색과 숫자 {0}을 좋아해요.".format("파란", 7))
```

```
나는 20살입니다.
나는 파란색과 숫자 7을 좋아해요.
나는 파란색과 숫자 7을 좋아해요.
나는 7색과 숫자 파란을 좋아해요.
```


[문자열 (String)]

6. 문자열 포맷

③ 방법3

```
print("나는 {color}색과 숫자 {num}을 좋아해요.".format(color = "파란", num = 7))
```

나는 파란색과 숫자 7을 좋아해요.

④ 방법4

```
color = "파란"  
num = 7  
print(f"나는 {color}색과 숫자 {num}을 좋아해요.")
```

나는 파란색과 숫자 7을 좋아해요.

[변수]

1. 자료형의 값을 저장하는 공간
2. 변수만 바꿈으로써 나머지 구조는 유지 가능하므로 효율적

```
print("우리집 강아지의 이름은 사랑입니다.")  
print("사랑이는 4살이며, 산책을 좋아합니다.")  
print("사랑이는 어른일까요? True")
```



```
animal = "강아지"  
name = "사랑이"  
age = 4  
hobby = "산책"  
is_adult = age >= 3  
  
print("우리집" + animal + "의 이름은 " + name + "입니다.")  
print(name + "는 " + str(age) + "살이며, " + hobby + "을 좋아합니다.")  
print(name + "는 어른일까요? " + str(is_adult))
```

[변수]

3. 변수명 규칙

	변수명 규칙	예시
1	변수의 이름은 문자, 숫자, Underscore(_)로 구성 다른 기호를 사용하면 Syntax Error	(O) a, a2, sum, _temp, temp (X) Money\$, you're
2	변수명은 문자 또는 Underscore(_)로 시작	(X) 3up, 7hi
3	파이썬 지정 단어는 사용 불가능	아래 참조
4	대문자와 소문자를 구별	Hour과 hour은 다른 변수

4. 파이썬 지정 단어(Keyword, Reserved word)

```
import keyword
print(keyword.kwlist)
print(len(keyword.kwlist))
```

```
['False', 'None', 'True', 'and', 'as', 'assert', 'async',
'await', 'break', 'class', 'continue', 'def', 'del', 'elif',
'else', 'except', 'finally', 'for', 'from', 'global', 'if',
'import', 'in', 'is', 'lambda', 'nonlocal', 'not', 'or',
'pass', 'raise', 'return', 'try', 'while', 'with', 'yield']
35
```

[리스트 (List)]

1. 리스트를 사용하면 1, 3, 5, 7, 9 숫자 모음을 간단하게 표현할 수 있다.

2. 리스트 만들기

```
a = []  
b = [1, 2, 3]  
c = ['Life', 'is', 'too', 'short']  
d = [1, 2, 'Life', 'is']  
e = [1, 2, ['Life', 'is']]
```

3. 리스트의 인덱싱 / 슬라이싱

```
print(c[0]) #리스트 c의 첫 번째 요소값  
print(b[0]+b[1])  
print(e[2][1])  
  
print(b[1:])
```

Life

3

Is

[2, 3]

[리스트 (List)]

4. 리스트 함수

함수	기능
append()	뒤에 연장
insert()	삽입
pop()	가장 뒤에서 꺼내기
sort()	순서대로 정렬
reverse()	뒤집기
clear()	비우기

```
list = [10, 30, 20]
print(list)
```

```
list.append(50)
print(list)
```

```
list.insert(3, 40) #3번째 인덱스에 40 넣기
print(list)
```

```
print(list.pop())
print(list)
```

```
list.sort()
print(list)
```

```
list.reverse()
print(list)
```

```
# list.clear()
# print(list)
```

```
list2 = [50, 60, 70]
list3 = list + list2 #합치기
print(list3)
```

```
[10, 30, 20]
[10, 30, 20, 50]
[10, 30, 20, 40, 50]
50
[10, 30, 20, 40]
[10, 20, 30, 40]
[40, 30, 20, 10]
[]
[40, 30, 20, 10, 50,
60, 70]
```

[튜플 (tuple)]

1. 리스트와 매우 비슷하다. 차이점은 아래와 같다.

- ① 리스트는 [], 튜플은 ()으로 둘러싼다.
- ② 리스트는 값의 생성, 삭제, 수정이 가능하지만, 튜플은 값을 바꿀 수 없다.

2. 튜플 만들고 다루기

```
t1 = ()
t2 = (1,)
t3 = (1, 2, 3)
t4 = 1, 2, 3
t5 = ('a', 'b', ('ab', 'cd'))

print(t3[2])
print(t3[1:])

print(t2 + t5)
print(t2*3)
print(len(t5))
```

```
3
(2, 3)
(1, 'a', 'b', ('ab', 'cd'))
(1, 1, 1)
3
```

[딕셔너리 (Dictionary)]

1. Key-Value의 연관성을 가진다.

2. 기본 딕셔너리의 모습 : `{Key1:Value1, Key2:Value2, Key3:Value3, ...}`

3. 딕셔너리 예 : `dic = {'name':'pey', 'phone':'0119993323', 'birth': '1118'}`

4. 딕셔너리 다루기

#딕셔너리 쌍 추가

```
a = {1: 'a'}  
a[2] = 'b'  
a['name'] = 'pey'  
a[3] = [1,2,3]  
print(a)
```

#딕셔너리 요소 삭제

```
del a[1]  
print(a)
```

```
{1: 'a', 2: 'b', 'name': 'pey', 3: [1, 2, 3]}  
  
{2: 'b', 'name': 'pey', 3: [1, 2, 3]}
```

[딕셔너리 (Dictionary)]

5. 딕셔너리 함수

함수	기능
get()	값 가져오기
del	삭제
keys()	Key들만 출력
values()	Value들만 출력
items()	Key, value 쌍으로 출력
clear()	비우기

```
cabinet = {1:"apple", 5:"banana"}

print(cabinet[5])
print(cabinet.get(5))

# print(cabinet[3]) #에러
print(cabinet.get(3)) #None 반환
print(cabinet.get(3, "사용 가능"))
#반환값 없으면 "사용 가능" 출력

print(1 in cabinet) #True
print(3 in cabinet) #False
```

```
cabinet[10] = "grape"
print(cabinet)

del cabinet[1]
print(cabinet)

print(cabinet.keys())
print(cabinet.values())
print(cabinet.items())

cabinet.clear() #비우기
print(cabinet)
```

```
banana
banana
None
사용 가능
True
False
{1: 'apple', 5: 'banana', 10: 'grape'}
{5: 'banana', 10: 'grape'}
dict_keys([5, 10])
dict_values(['banana', 'grape'])
dict_items([(5, 'banana'), (10, 'grape')])
{}
```


[문제 4]

홍길동 시의 과목별 점수는 다음과 같다. 홍길동 씨의 평균 점수를 구해보자.

과목	점수
국어	80
영어	75
수학	55

조건1. 딕셔너리 이용

Ex)

```
cabinet = {1:"apple", 5:"banana"}  
print(cabinet.get(5))
```

banana

조건2. 출력 화면 : 홍길동 님의 평균 점수 : 70점

[자료구조 변경]

1. 자료 구조를 변경해야 하는 경우가 있다.

- List : []
- Tuple : (), 변경불가
- Set : {}, 중복 불가

```
menu = {"커피", "우유", "주스"}  
print(menu, type(menu)) #set  
  
menu = list(menu)  
print(menu, type(menu)) #list  
  
menu = tuple(menu)  
print(menu, type(menu)) #tuple  
  
menu = set(menu)  
print(menu, type(menu)) #set
```

```
{'커피', '우유', '주스'} <class 'set'>  
['커피', '우유', '주스'] <class 'list'>  
( '커피', '우유', '주스' ) <class 'tuple'>  
{ '커피', '우유', '주스' } <class 'set'>
```