

컴퓨팅사고와 파이썬 프로그래밍

Ch 2. 파이썬 프로그램의 기본 구성, 프로그램 디버깅



교수 김 영 탁

영남대학교 정보통신공학과

(Tel : +82-53-810-2497; E-mail : ytkim@yu.ac.kr)

Outline

- ◆ 파이썬 프로그램의 기본 구성
- ◆ 파이썬 프로그램의 입력과 출력
- ◆ 파이썬 식별자 (identifier)
- ◆ 변수(variable), 상수 (constant)
- ◆ 조건문과 반복문 개요
- ◆ 객체 지향형 프로그래밍 개요
- ◆ 파이썬 기본 연산자와 기본 명령어
- ◆ 파이썬 프로그램의 디버깅
- ◆ VS Code 설치 및 파이썬 프로그래밍, 디버깅



파이썬 프로그램의 기본 구성

MyFirstPythonProgram.py

```
# MyFirstPythonProgram.py (1)
"""
Project : My first Python Program
Author: Cul-Soo Kim
Date of last update: Jan. 5, 2021
Update list:
    - v1.0 : Jan. 2, 2021
        . My first Python program with simple
        prints of variables
    - v1.1 : Jan. 5, 2021
        . Include str and mathematical operations
"""

print("This is my first Python program !")
my_name = "Chul-Soo Kim" # string variable
print("My name is", my_name, ".")
print("I am really happy to learn\
Python Programming !!")

# definitions of variables
x = 5 # integer variable
y = 2 # integer variable
print("x : ", x)
print("y : ", y)
```

```
# MyFirstPythonProgram.py (2)

# operations
sum_xy = x + y
sub_xy = x - y
mul_xy = x * y
div_xy = x / y
int_div_xy = x // y
print("x + y : ", sum_xy)
print("x - y : ", sub_xy)
print("x * y : ", mul_xy)
print("x / y : ", div_xy)
print("x // y : ", int_div_xy)
```

```
This is my first Python program !
My name is Chul-Soo Kim .
I am really happy to learn Python Programming !!
x : 5
y : 2
x + y : 7
x - y : 3
x * y : 10
x / y : 2.5
x // y : 2
```



기본 주석문 (basic comments)

◆ 주석문 (Comments)

- 프로그램의 목적 및 주요 기능
- 프로그램 작성자 및 작성 일자, 보완 일자
- 프로그램의 주요 수정/보완 내용

파이썬 프로그램 주석문 형식	설 명
"""" comment_line_1 comment_line_2 """"	여러 줄에 걸친 주석문
# comment to the end of line	'#' 문자로부터 그 줄 끝까지 주석문

```
"""
Project : My first Python Program
Author: Cul-Soo Kim
Date of last update: March 5, 2022
Update list:
    - v1.0 : Jan. 2, 2022
        . My first Python program with simple
          prints of variables
    - v1.1 : Jan. 5, 2022
        . Include str and mathematical operations
"""

# First Greetings
x = 5 # variable
```



데이터 입력 기능을 가지는 파이썬 프로그램 예제

```
"""
    Basic Comments (same as before)
"""
# definitions of variables
x_str = input("input x = ") # string variable
y_str = input("input y = ") # string variable
x, y = int(x_str), int(y_str)
print("x, y = {}, {}".format(x, y))

# operations
sum_xy = x + y
sub_xy = x - y
mul_xy = x * y
div_xy = x / y
int_div_xy = x // y
print("x + y : ", sum_xy)
print("x - y : ", sub_xy)
print("x * y : ", mul_xy)
print("x / y : ", div_xy)
print("x // y : ", int_div_xy)
```

```
input x = 7
input y = 3
x, y = 7, 3
x + y : 10
x - y : 4
x * y : 21
x / y : 2.3333333333333335
x // y : 2
```



변수 (variable)의 선언 및 사용, Dynamic Typing

```
# Python program variables
```

```
x = 1 # integer
print("x = ", x)
print("type(x) = ", type(x))
```

```
x = 2.345 #float
print("x = ", x)
print("type(x) = ", type(x))
```

```
x = [1, 2, 3, 4] # list
print("x = ", x)
print("type(x) = ", type(x))
```

```
x = (7, 8, 9, 10) # tuple
print("x = ", x)
print("type(x) = ", type(x))
```

```
x = {'A':1, 'B':2, 'C':3} # dict
print("x = ", x)
print("type(x) = ", type(x))
```

```
x = {1, 2, 3, 4} # set
print("x = ", x)
print("type(x) = ", type(x))
```

```
x = 1
type(x) = <class 'int'>
x = 2.345
type(x) = <class 'float'>
x = [1, 2, 3, 4]
type(x) = <class 'list'>
x = (7, 8, 9, 10)
type(x) = <class 'tuple'>
x = {'A': 1, 'B': 2, 'C': 3}
type(x) = <class 'dict'>
x = {1, 2, 3, 4}
type(x) = <class 'set'>
```



상수 (constant)의 선언 및 사용

◆ 상수 (constant)

- 상수는 초기 설정된 값을 변경하지 않고 계속 사용
- 프로그램 소스코드의 가독성 (readability)를 향상시킴

```
# variables and constants in Python program
```

```
PI = 3.141592
```

```
radius = 5.0  
circle_area = radius * radius * PI  
print("Area of circle (radius : {}) : {}".format(radius, circle_area))
```

```
width = 5.0  
length = 4.0  
rectangle_area = width * length  
print("Area of rectangle (width: {}, length: {}) : {}".format(width, length, rectangle_area))
```

```
base = 5.0  
height = 4.0  
triangle_area = (base * height) / 2.0  
print("Area of triangle (base: {}, height: {}) : {}".format(base, height, triangle_area))
```

```
Area of circle (radius : 5.0) : 78.5398  
Area of rectangle (width: 5.0, length: 4.0) : 20.0  
Area of triangle (base: 5.0, height: 4.0) : 10.0
```



과학 기술 계산에서 많이 사용되는 상수들

◆ 기호상수 정의

기호 상수의 예	설 명
PI = 3.141592653589793238	원주율의 값을 의미하며, 원면적, 원둘레, 삼각함수 등에서 사용
INT_MAX = 2147483647	32비트로 표현되며 부호가 포함된 정수 중 가장 큰 값
INT_MIN = -2147483638	32비트로 표현되며 부호가 포함된 정수 중 가장 작은 값
RAND_MAX = 32767	16비트값으로 생성되는 난수 (random number)의 최대값



**파이썬 프로그램의 식별자 (Identifier),
기본 수학연산, 함수호출**

파이썬 프로그램 식별자

◆ 식별자 (identifier)

- 변수 (variable), 상수 (constant), 함수 (function) 이름
- 의미를 전달할 수 있는 영어 단어 사용

◆ 식별자의 예

구분		식별자 예
기하, 도형	원	radius (반지름), diameter (지름), circumference(원둘레), height (원기둥 높이)
	삼각형	base(밑변), side (대각선), vertex (꼭지점), height (삼각형 높이), prism (삼각기둥), prism_height(삼각기둥 높이)
	사각형	width(가로), length(세로), height(사각기둥 높이)
도형 그리기		draw(그리기), rotate(회전), move(이동)
산술 계산		add (덧셈), subtract(뺄셈), multiply(곱셈), divide(나눗셈), modulo(모듈로)



식별자 (Identifier) 설정 원칙

◆ 식별자 설정 규정 (Rule of identifier)

- 영문자(대문자, 소문자), 밑줄 문자, 숫자로 구성
 - 대문자 'A' ~ 'Z', 소문자 'a' ~ 'z', 밑줄 문자 ('_'), 숫자 ('0' ~ '9')
- 숫자는 식별자의 첫 문자가 될 수 없음 ('0' ~ '9')
- 파이썬의 키워드 (keyword)는 식별자가 될 수 없음
- 식별자의 길이에는 제한이 없음
- 대문자와 소문자를 구분함

```
>>> import keyword
>>> keyword.kwlist
['False', 'None', 'True', 'and', 'as', 'assert', 'break', 'class', 'continue', '
def', 'del', 'elif', 'else', 'except', 'finally', 'for', 'from', 'global', 'if',
'import', 'in', 'is', 'lambda', 'nonlocal', 'not', 'or', 'pass', 'raise', 'retu
rn', 'try', 'while', 'with', 'yield']
>>> |
```



식별자 설정 방법

◆ 식별자 설정 방법

- Camel casing: 단어의 첫 문자를 대문자로 구분
- GNU 작명 관례: 단어 사이에 밑줄 문자 사용

식별자 이름 작명 방법	설 명
camel casing	단어들을 연속적으로 나열하며, 각 단어의 첫 문자를 대문자로 표시 (단 첫 번째 단어는 소문자로 사용가능) 예) printComplex(), drawRectangle(), drawPolygon()
GNU 작명 관례	단어들을 연속적으로 나열하며, 각 단어사이에 밑줄 문자를 삽입 예) print_complex(), draw_rectangle(), draw_polygon()



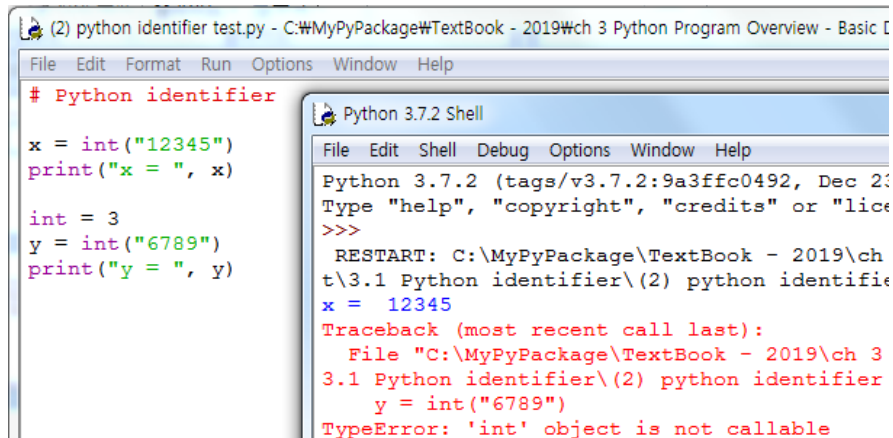
식별자의 예

◆ 파이썬 식별자의 예

파이썬 식별자 예	사용 가능 여부	설명
average, avg, avg1	사용 가능	문자와 숫자
num_data, numData,	사용 가능	문자, 밑줄
ch, ch1, ch_1	사용 가능	문자, 밑줄
i, d, f	사용 가능	
_avg, __area	사용 가능	밑줄, 문자
NUM_DATA	사용 가능	대문자, 밑줄
auto, char, int, class, while, for	사용 불가능	keyword
3avg	사용 불가능	숫자로 시작
avg!, avg%, &avg, dollar\$	사용 불가능	특수문자 포함



잘못 설정된 파이썬 식별자의 예



```
(2) python identifier test.py - C:\MyPyPackage\TextBook - 2019\ch 3 Python Program Overview - Basic I
File Edit Format Run Options Window Help

# Python identifier

x = int("12345")
print("x = ", x)

int = 3
y = int("6789")
print("y = ", y)
```

Python 3.7.2 Shell

Python 3.7.2 (tags/v3.7.2:9a3ffc0492, Dec 23 2019)

Type "help", "copyright", "credits" or "license()" for more

>>>

RESTART: C:\MyPyPackage\TextBook - 2019\ch 3 Python Program Overview - Basic I

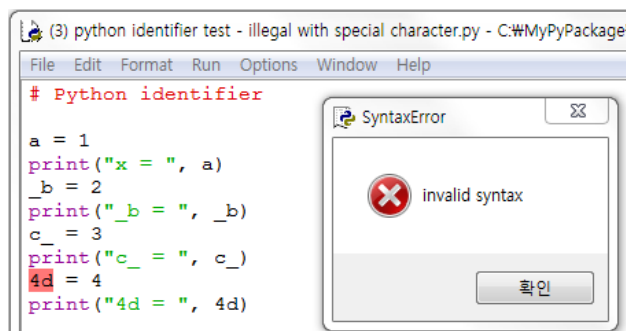
x = 12345

Traceback (most recent call last):

File "C:\MyPyPackage\TextBook - 2019\ch 3 Python Program Overview - Basic I", line 10, in <module>

y = int("6789")

TypeError: 'int' object is not callable



```
(3) python identifier test - illegal with special character.py - C:\MyPyPackage\
File Edit Format Run Options Window Help

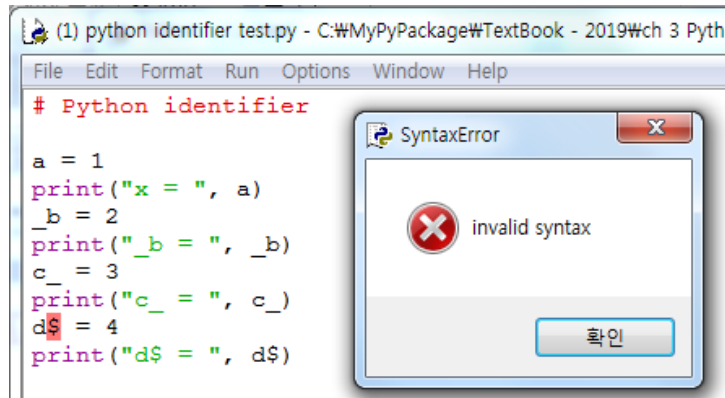
# Python identifier

a = 1
print("x = ", a)
_b = 2
print("_b = ", _b)
c_ = 3
print("c_ = ", c_)
4d = 4
print("4d = ", 4d)
```

SyntaxError

invalid syntax

확인



```
(1) python identifier test.py - C:\MyPyPackage\TextBook - 2019\ch 3 Python Program Overview - Basic I
File Edit Format Run Options Window Help

# Python identifier

a = 1
print("x = ", a)
_b = 2
print("_b = ", _b)
c_ = 3
print("c_ = ", c_)
d$ = 4
print("d$ = ", d$)
```

SyntaxError

invalid syntax

확인

기본 수학 연산

◆ 기본 수학 연산

파이썬 기본 연산자	예제 및 설명
+	주어진 데이터가 숫자 인 경우: 덧셈 주어진 데이터가 문자열일 경우: 문자열 병합
-	주어진 숫자 데이터에 대한 뺄셈
*	주어진 데이터가 숫자 인 경우: 곱셈 주어진 데이터가 문자열일 경우: 문자열 병합
/	주어진 숫자 데이터에 대한 실수형 나눗셈 (소수점 이하 값 유지)
//	주어진 숫자 데이터에 대한 정수형 나눗셈 (소수점 이하 값 생략)
%	주어진 숫자 데이터에 대한 모듈로 계산



함수의 호출 및 실행

◆ 파이썬 함수 호출의 예

파이썬 기본 함수 예시	예제 및 설명
<code>x_str = input("prompt string");</code>	표준 입력 장치로부터 문자열 입력 prompt 문자열이 지정되어 있으면 이를 먼저 출력하고, 문자열 입력
<code>x = int(x_str)</code>	주어진 숫자 문자열을 정수 (integer)로 변환
<code>y = float(y_str)</code>	주어진 숫자 문자열을 실수 (float)로 변환
<code>print("output string", data);</code>	<code>print("sum = ", sum)</code>



파이썬 프로그램의 입력과 출력

파이썬 프로그램의 데이터 입출력

◆ 파이썬 프로그램의 입력 및 출력 예

```
# Python input and output
```

```
a = input("input a data : ") # a is string variable
print("input data a = ", a)
print("type of a is ", type(a))
```

```
b = input("input an integer data : ") # b is string
variable
print("input data b = ", b)
print("type of b is ", type(b))
sum = a + b # string concatenation
print("sum is : ", sum)
```

```
input a data : 10
input data a = 10
type of a is <class 'str'>
input an integer data : 20
input data b = 20
type of b is <class 'str'>
sum is : 1020
```



입력 문자열의 숫자 데이터 변환

◆ 숫자 데이터 입력

```
# Simple Python program with number input and output
```

```
a_str = input("input a_str : ")
print("input a_str = ", a_str)
print("type of a_str is ", type(a_str))
a = int(a_str) # a = int(input("input a = ")) # a is integer variable
print("type of a is ", type(a))
```

```
b_str = input("input b_str : ") # b is integer variable
print("input b_str = ", b_str)
print("type of b_str is ", type(b_str))
b = int(b_str) # b = int(input("input b = "))
sum = a + b # integer addition
print("sum is : ", sum)
```

```
input a_str : 10
input a_str = 10
type of a_str is <class 'str'>
type of a is <class 'int'>
input b_str : 20
input b_str = 20
type of b_str is <class 'str'>
sum is : 30
```



포맷 지정 데이터 출력

◆ 파이썬 출력 포맷 지정 방법

문자열 서식 지정	서식 지정 예
%d, %s, %f의 형식 지정자를 사용한 출력 형식 지정	<pre>print("%d %5d %10s, %7.3f" % (123, 456, "Python", 123.456))</pre> <p>%b : 2진수 출력 %d : 10진수를 출력 %5d : 5칸을 확보하여 오른쪽으로 정렬된 10진수로 출력 %05d : 5칸을 확보하여 오른쪽으로 정렬된 10진수로 출력하며, 왼쪽에 남은 공간에 0을 출력 %c : 문자 출력 %s : 문자열 (string) %10s : 10칸을 확보하여 오른쪽으로 정렬된 문자열을 출력 %f : 실수 출력 %7.3f : 7칸을 확보하여 오른쪽으로 정렬된 실수를 출력하며, 소수점 아래 3자리까지 출력하며, 오른쪽 빈칸은 0으로 채움</p>
{ n : x }와 .format 메소드를 사용한 서식 지정	<pre>print("{0:d} {1:5d} {2:05d} {3:10s} {4:7.3f} {5:010b}".format(123, 456, 678, "Python", 123.456, 255))</pre> <p>{i:>} i번째 항을 오른쪽 정렬 {i:^} i번째 항을 중앙 정렬 {i:<} i번째 항을 왼쪽 정렬</p>
format() 함수를 사용한 서식 지정	<pre>format(123456789, ',d') : 1000단위로 comma (,)를 삽입한 형식으로 출력</pre>



출력 포맷 지정 예제

◆ 출력 포맷 지정 예제

```
# Python print() with formatting
```

```
print("%5d %7d %07d %10s %7.3f" % (123, 456, 678, "Python", 123.456789))  
print("%5d %7d %07d %-10s %10.6f" % (12, 34, 5, "Python", 123.456789))
```

```
print("{0:5d} {1:7d} {2:07d} {3:10s} {4:7.3f}"\  
      .format(123, 456, 678, "Python", 123.456789))
```

```
print("{0:5d} {1:7d} {2:07d} {3:<10s} {4:10.6f}"\  
      .format(12, 34, 5, "Python", 123.456789))
```

```
print("{0:5d} {1:7d} {2:07d} {3:^10s} {4:10.6f}"\  
      .format(12, 34, 5, "Python", 123.456789))
```

```
print("{0:5d} {1:7d} {2:07d} {3:>10s} {4:10.6f}"\  
      .format(12, 34, 5, "Python", 123.456789))
```

```
print("Long integer with comma at each 1000 unit : "\  
      , format(123456789, ',d'))
```

```
123      456 0000678      Python 123.457  
12       34 0000005 Python    123.456789  
123      456 0000678 Python    123.457  
12       34 0000005 Python    123.456789  
12       34 0000005 Python    123.456789  
12       34 0000005 Python    123.456789  
Long integer with comma at each 1000 unit : 123,456,789
```



출력 포맷 지정 예제

◆ 정수, 이진수, 8진수, 16진수 출력

```
# print integer in decimal, binary, octal and hexadecimal

print("=====")
print("Decimal    Bit    Octal    Hexadecimal")
print("-----")
for i in range(0, 257, 8):
    print("  {:4d} {:09b} {:#05o} {:#05X}" \
          .format(i, i, i, i))
print("=====")
```

```
=====
Decimal    Bit    Octal    Hexadecimal
-----
0 000000000 0o000 0X000
8 000001000 0o010 0X008
16 000010000 0o020 0X010
24 000011000 0o030 0X018
32 000100000 0o040 0X020
40 000101000 0o050 0X028
48 000110000 0o060 0X030
56 000111000 0o070 0X038
64 001000000 0o100 0X040
72 001001000 0o110 0X048
80 001010000 0o120 0X050
88 001011000 0o130 0X058
96 001100000 0o140 0X060
104 001101000 0o150 0X068
112 001110000 0o160 0X070
120 001111000 0o170 0X078
128 010000000 0o200 0X080
136 010001000 0o210 0X088
144 010010000 0o220 0X090
152 010011000 0o230 0X098
160 010100000 0o240 0X0A0
168 010101000 0o250 0X0A8
176 010110000 0o260 0X0B0
184 010111000 0o270 0X0B8
192 011000000 0o300 0X0C0
200 011001000 0o310 0X0C8
208 011010000 0o320 0X0D0
216 011011000 0o330 0X0D8
224 011100000 0o340 0X0E0
232 011101000 0o350 0X0E8
240 011110000 0o360 0X0F0
248 011111000 0o370 0X0F8
256 100000000 0o400 0X100
=====
```



조건문 개요

조건문 개요 – if, if-else

◆ 조건문 if

```
# simple program to input two integers and compare
x = int(input("input first integer (x) : "))
y = int(input("input second integer (y) : "))
if x == y:
    print("x(%d) is equal to y(%d)"%(x, y))
if x < y:
    print("x(%d) is less than y(%d)"%(x, y))
if x > y:
    print("x(%d) is greater than y(%d)"%(x, y))
```

```
input first integer (x) : 7
input second integer (y) : 5
x(7) is greater than y(5)
```

```
input first integer (x) : 3
input second integer (y) : 3
x(3) is equal to y(3)
```

```
input first integer (x) : 4
input second integer (y) : 8
x(4) is less than y(8)
```



조건문 – if-elif-else

◆ 조건문 if-elif-else

```
# conditional branch with if - elif - else
score = int(input('course score [0..99] = '))
if 90 <= score <= 100:
    grade = 'A'
elif 80 <= score:
    grade = 'B'
elif 70 <= score:
    grade = 'C'
elif 60 <= score:
    grade = 'D'
else:
    grade = 'F'
print("score = %d, grade = %s" %(score, grade))
```

```
course score [0..99] = 95
score = 95, grade = A
```

```
course score [0..99] = 74
score = 74, grade = C
```



반복문 개요

while 반복문 개요

◆ while-loop 기본 구조

```
# while-loop for input positive integers

L = list() # creation of an empty list L
print("Input positive integers (-1 to end)")
x = int(input("data : "))
n = 0
sum = 0
while x >= 0:
    L.append(x) # append x to list L
    n = n + 1
    sum = sum + x
    x = int(input("data : "))

print("Total", n, " integers were input: ", L)
print("Sum is : ", sum)
```

```
Input positive integers (-1 to end)
data : 1
data : 3
data : 5
data : 7
data : 9
data : 2
data : 4
data : 6
data : 8
data : 10
data : -1
Total 10 integers were input: [1, 3, 5, 7, 9, 2, 4, 6, 8, 10]
Sum is : 55
```



while-loop과 continue, break

◆ while-loop with break and continue

```
# while-loop, break, continue
```

```
MAX_NUM_DATA = 100
```

```
num_data = 0
```

```
data_sum = 0
```

```
print("Input (up to {} ) integer data.".format(MAX_NUM_DATA))
```

```
while num_data < MAX_NUM_DATA:
```

```
    data = int(input("Data (-1 to finish) = "))
```

```
    num_data = num_data + 1
```

```
    if data == -1:
```

```
        break
```

```
    elif data <= 0:
```

```
        continue
```

```
    else:
```

```
        data_sum = data_sum + data
```

```
print("Total {} data input, sum of positive data = {}".\
      format(num_data, data_sum))
```

```
Input (up to 100 ) integer data.
```

```
Data (-1 to finish) = 10
```

```
Data (-1 to finish) = 2
```

```
Data (-1 to finish) = 7
```

```
Data (-1 to finish) = 1
```

```
Data (-1 to finish) = -5
```

```
Data (-1 to finish) = -9
```

```
Data (-1 to finish) = 2
```

```
Data (-1 to finish) = 3
```

```
Data (-1 to finish) = 13
```

```
Data (-1 to finish) = -1
```

```
Total 10 data input, sum of positive data = 38
```



입력 데이터 중의 최댓값, 최솟값 찾기

```
# while-loop, find min and max of input data list
```

```
TARGET_NUM_DATA = 10
num_data = 0
L_data = [] # empty list
L_sum = 0
print("Input up to {} integer data.".format(TARGET_NUM_DATA))
while num_data < TARGET_NUM_DATA:
    data = int(input("data = "))
    num_data = num_data + 1
    L_data.append(data)
    L_sum = L_sum + data
    if num_data == 1:
        data_min = data_max = data
        continue
    if data < data_min:
        data_min = data
    if data > data_max:
        data_max = data

print("Input data list = ", L_data)
print("Min = {}, Max = {}, Avg = {}".format(data_min, data_max, L_sum/num_data))
```

```
Input up to 10 integer data.
```

```
data = 32
```

```
data = 45
```

```
data = 1
```

```
data = 5
```

```
data = 10
```

```
data = 4
```

```
data = 3
```

```
data = 99
```

```
data = -5
```

```
data = -30
```

```
Input data list = [32, 45, 1, 5, 10, 4, 3, 99, -5, -30]
```

```
Min = -30, Max = 99, Avg = 16.4
```



range() Generator 함수

◆ range(start, stop, step)

- start로 부터 stop 직전까지 (stop은 포함하지 않음) step씩 증가하며 정수 값을 생성하여 제공
- 만약 인수가 2개만 제공되면, start와 stop의 값으로 사용되며, step은 1로 설정됨
- 만약 인수가 1개만 제공되면 stop의 값으로 사용되며, start는 0으로, step은 1로 설정됨
- 만약 step 값이 음수이면, start로 부터 stop 직전까지, step에 따라 감소시키며 정수값을 생성하여 제공

range() 사용 예	설 명
range(10)	0부터 9까지 1씩 증가하며 정수를 생성
range(1, 10)	1부터 9까지 1씩 증가하며 정수를 생성
range(1, 10, 2)	1부터 9까지 2씩 증가하며 정수를 생성
range(10, 100, 5)	10부터 99까지 5씩 증가하며 정수를 생성
range(100, 0, -1)	100부터 1까지 1씩 감소하며 정수를 생성



range()와 in을 사용하는 for 반복문

◆ for-loop

- for-loop에서는 in과 range() 를 사용하여 구성할 수 있음
- for-loop에서 사용하는 변수 (아래 예에서는 i)의 값이 range()에서 생성된 값에 차례로 대입되어 사용되며, 전체 반복 회수는 range()에서 생성된 숫자의 개수에 의해 제한됨

```
# for-loop with range()
n = int(input('Input n to calculate sum of [0..n] : '))
nSum = 0
for i in range(0, n+1): #nSum = sum(range(0, n+1))
    nSum = nSum + i # nSum += i
print("Sum of [0..%d] = %d" %(n, nSum))
```

```
Input n to calculate sum of [0..n] : 10
Sum of [0..10] = 55
```

```
Input n to calculate sum of [0..n] : 100
Sum of [0..100] = 5050
```



for 반복문과 continue, break

◆ Example of for-loop with break and continue

```
#for_loop with break and continue

n = int(input("Input number of data to process: "))
L = list()
sum = 0
print("Input %d non-negative integers"%(n))
for i in range(n):
    d = int(input())
    if d == 0:
        continue
    elif d < 0:
        break
    L.append(d)
    sum = sum + d # sum += d
print("Input data : ", L)
print("Sum = ", sum)
```

```
===== RESTART: C:/YTK-PythonProg/3_
Input number of data to process: 5
Input 5 non-negative integers
2
3
0
4
5
Input data : [2, 3, 4, 5]
Sum = 14
>>>
===== RESTART: C:/YTK-PythonProg/3_
Input number of data to process: 5
Input 5 non-negative integers
2
-1
Input data : [2]
Sum = 2
>>> |
```



입력데이터의 통계 분석 – 평균, 분산, 표준편차

```
# while-loop, for-loop, find min and max of input data list
import math
```

```
TARGET_NUM_DATA = 10
num_data = 0
L_data = [] # empty list
L_sum = 0
print("Input {} integer data."\
      .format(TARGET_NUM_DATA))
while num_data < TARGET_NUM_DATA:
    data = int(input("data = "))
    num_data = num_data + 1
    L_data.append(data)
    L_sum = L_sum + data

avg = L_sum / num_data
dev_sq_sum = 0
for i in range(num_data):
    dev = avg - L_data[i] # deviation from avg
    dev_sq_sum = dev_sq_sum + dev * dev
var = dev_sq_sum / num_data
std = math.sqrt(var)

print("Input data list = ", L_data)
print("avg = {}, var = {}, std = {}".format(avg, var, std))
```

```
Input 10 integer data.
data = 1
data = 2
data = 3
data = 4
data = 5
data = 6
data = 7
data = 8
data = 9
data = 10
Input data list = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
avg = 5.5, var = 8.25, std = 2.8722813232690143

Input 10 integer data.
data = -5
data = -4
data = -3
data = -2
data = -1
data = 0
data = 1
data = 2
data = 3
data = 4
Input data list = [-5, -4, -3, -2, -1, 0, 1, 2, 3, 4]
avg = -0.5, var = 8.25, std = 2.8722813232690143
```



ASCII Characters

◆ ASCII (American Standard Code for Information Interchange)

	0x00	0x01	0x02	0x03	0x04	0x05	0x06	0x07	0x08	0x09	0x0A	0x0B	0x0C	0x0D	0x0E	0x0F
0x00	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
0x10	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
0x20	SP	!	"	#	\$	%	&	'	()	*	+	,	-	.	/
0x30	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
0x40	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
0x50	P	Q	R	S	T	U	V	W	X	Y	Z	[\]	^	_
0x60	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
0x70	p	q	r	s	t	u	v	w	x	y	z	{		}	~	DEL

Printing Upper-case/Lower-case Characters and Numbers in ASCII table

Printing Upper_case / Lower_case Characters, and Digits of ASCII Code Table

```
L_upper = [] # list
for x in range(0x41, 0x5B):
    L_upper.append(chr(x))
print("Upper case alphabets : \n", L_upper)
print()
```

```
L_lower = []
for x in range(0x61, 0x7B):
    L_lower.append(chr(x))
print("Lower case alphabets : \n", L_lower)
print()
```

```
L_digits = []
for x in range(0x30, 0x3A):
    L_digits.append(chr(x))
print("Digits : \n", L_digits)
print()
```

	0x00	0x01	0x02	0x03	0x04	0x05	0x06	0x07	0x08	0x09	0x0A	0x0B	0x0C	0x0D	0x0E	0x0F
0x00	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
0x10	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
0x20	SP	!	"	#	\$	%	&	'	()	*	+	,	-	.	/
0x30	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
0x40	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
0x50	P	Q	R	S	T	U	V	W	X	Y	Z	[\]	^	_
0x60	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
0x70	p	q	r	s	t	u	v	w	x	y	z	{		}	~	DEL

```
Upper case alphabets :
['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K', 'L', 'M', 'N', 'O', 'P',
'Q', 'R', 'S', 'T', 'U', 'V', 'W', 'X', 'Y', 'Z']

Lower case alphabets :
['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm', 'n', 'o', 'p',
'q', 'r', 's', 't', 'u', 'v', 'w', 'x', 'y', 'z']

Digits :
['0', '1', '2', '3', '4', '5', '6', '7', '8', '9']
```



객체 지향형 프로그래밍 개요

객체 지향형 프로그래밍 (Object-Oriented Programming)

◆ 객체 (Object)

- 우리 주변에서 살펴볼 수 있는 다양한 사물과 생물들을 의미하며 크기, 색깔, 위치 등의 상태 (state) 정보와 스스로 또는 다른 객체에 의하여 이루어지는 행동 (behavior)을 가짐.
- 예를 들어 사람에게는 이름, 생년월일, 키, 몸무게, 현재 위치, 소속 기관 등의 상태 정보와 그 사람이 다른 곳으로 이동하거나, 키와 몸무게가 변하거나, 노래를 부르거나, 소속기관이 변경되는 등의 행동이 이루어질 수 있음.

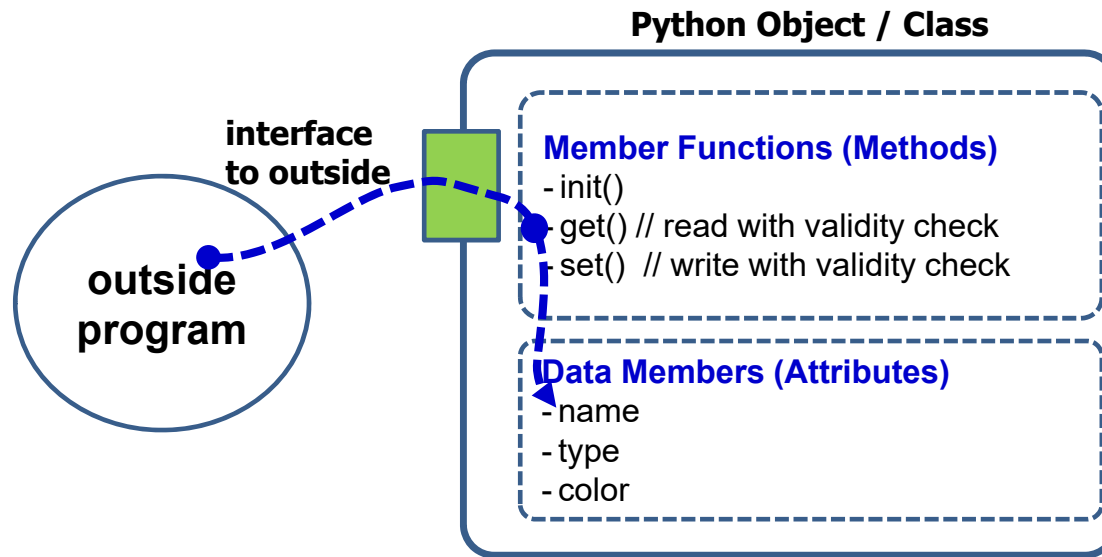
◆ 객체 지향형 프로그래밍

- 소프트웨어 시스템을 객체 (object) 단위로 나누고, 각 객체는 그 내부에 데이터와 관련 함수를 함께 포함하게 함으로써 독립성을 유지하게 하는 프로그래밍 기법
- 객체 지향형 프로그래밍에서는 문제를 해결하는 함수 호출의 절차적 실행 순서보다는 각 객체가 어떻게 상호 연관성을 가지며, 각 객체 내부에서 데이터를 어떻게 정상적인 범위 내에 유지하게 하고, 그 데이터를 어떤 자료구조를 사용하여 가장 효율적으로 표현하며, 어떤 알고리즘을 사용하여 가장 효율적으로 처리할 수 있는가에 대하여 중점을 두게 됨.



파이썬 클래스

◆ 클래스 (class)



클래스와 인스턴스

◆ 클래스 (class)

- 객체를 소프트웨어에서 표현하고 구현하기 위하여 사용하는 틀/모델
- 클래스는 객체 지향형 프로그래밍으로 알고리즘 및 자료구조를 구현하기 위한 기본 단위
- 객체의 데이터를 저장하는 데이터 멤버와 데이터를 처리하는 멤버 함수를 하나의 클래스 내에 묶음으로 관리

◆ 인스턴스 (instance)

- 클래스를 사용하여 실체를 만든 것
- 자료형 (예: int, float)을 사용하여 변수 (예: x, y)를 만드는 것과 같이 클래스 (예: list)를 사용하여 다수의 인스턴스 (예: lst_name, lst_value)를 생성할 수 있음



파이썬 클래스

◆ 클래스의 기본 멤버 함수

클래스의 기본 멤버 함수	설 명
생성자 (constructor)	클래스를 사용하여 객체가 생성될 때 실행되며, 초기화 기능 수행
데이터 멤버 접근자 (accessor)	클래스의 데이터 멤버 값을 읽기 위하여 접근하는 기능 제공. 예) getXXX()
데이터 멤버 변경자 (mutator)	클래스의 데이터 멤버 값을 변경하기 위한 기능 제공. 예) setXXX()

◆ dot (.) 연산자

- 클래스나 객체의 멤버 데이터와 멤버 함수를 사용할 때 dot(.) 연산자를 사용하여 멤버 관계를 나타냄



파이썬 패키지와 모듈

◆ 파이썬 확장 패키지의 예

파이썬 확장 패키지	설 명
NumPy	수학 계산을 위한 패키지
SciPy	과학 기술 계산을 위한 패키지이며 NumPy, SciPy, Matplotlib, IPython, Sympy, pandas 등을 포함
Matplotlib	2차원 평면에 그래프와 도형을 그리는 패키지
OpenCV	영상처리용 패키지
TensorFlow	AI (인공지능) 패키지
Keras	파이썬으로 구현된 딥러닝 라이브러리이며, TensorFlow 위에서 딥러닝 모델을 쉽게 구성할 수 있게 함
Pandas	데이터 분석 확장 패키지 https://pandas.pydata.org/pandas-docs/stable/
R	통계처리 패키지



속성 (attribute)과 메소드 (method)

◆ 속성 (attribute)

- 클래스 (class)와 클래스로 부터 생성된 인스턴스(instance)가 가지는 멤버 데이터 또는 멤버 함수
- 클래스가 구현하는 기능과 정보를 의미
- 클래스/인스턴스의 이름에 dot(.)을 붙이고 속성을 지정
- 클래스/인스턴스의 속성 설정/변경 및 접근을 위한 멤버함수 (메소드)를 클래스가 제공
- 예)
t = turtle.Turtle() # turtle 모듈의 Turtle() 클래스를 사용하여
 # 터틀 객체 생성
t.color('red') # 터틀 객체 t의 색상 (color)를 red로 설정

◆ 메소드 (method)

- 클래스와 인스턴스가 제공하는 멤버 함수
- 클래스/인스턴스의 이름에 dot(.)을 붙이고 메소드를 지정
- 예)
t.forward(100) # 터틀 객체 t의 멤버함수 forward()를 사용하여
 # 현재 터틀의 방향으로 100 픽셀만큼 앞으로 전진 이동



파이썬과 객체지향형 프로그래밍

◆ 파이썬은 객체지향형 프로그래밍 기반

- 파이썬에서 사용되는 모든 자료형, 모듈 및 패키지는 객체지향형으로 구현되어 있음
- 따라서 자료형, 모듈 및 패키지에 dot(.) 연산자를 사용하여 속성(멤버 함수와 멤버 데이터)를 지정하게 됨
- 가장 기본적인 객체: PyObject (파이썬 객체)
- 객체 지향형 프로그래밍을 사용함으로써 모든 자료형을 하나의 체계로 관리할 수 있고, 동적 자료형 (dynamic typing)이 가능함



프로그램의 디버깅 (Debugging)

문법적 오류와 오류메시지

◆ 문법적 오류 (Syntax Error)

```
# python program - syntax error & debugging
```

```
a = 10  
c = a + b
```

```
Traceback (most recent call last):  
  File "C:\MyPyPackage\TextBook - 2019\ch 2  
  in <module>  
    c = a + b  
NameError: name 'b' is not defined
```

```
# python program - syntax error & debugging
```

```
a = input("input integer data = ")  
b = 20  
c = a + b  
print("a = ", a)  
print("b = ", b)  
print("a + b = ", c)
```

```
input integer data = 10  
Traceback (most recent call last):  
  File "C:\MyPyPackage\TextBook - 2019\ch 2 Python Overv:  
  function.py", line 5, in <module>  
    c = a + b  
TypeError: can only concatenate str (not "int") to str
```



논리적 오류 (logical error)

◆ 알고리즘이 실행 순서상에 논리적인 오류가 있는 경우

/* 올바른 제빵 알고리즘 */

- 1: 재료 준비: 밀가루, 우유, 계란
- 2: 그릇 준비: 반죽용 그릇
- 3: 밀가루, 우유, 계란 반죽하여
그릇에 담기
- 4: 그릇을 오븐에 넣기
- 5: 오븐을 1시간 동안 가열
- 6: 오븐을 끄기
- 7: 그릇을 꺼내기

(a) 올바른 제빵 알고리즘

/* 논리적 오류가 포함된
제빵 알고리즘 */

- 1: 재료 준비: 밀가루, 우유, 계란
- 2: 그릇 준비: 반죽용 그릇
- 3: 그릇을 오븐에 넣기
- 4: 오븐을 1시간 동안 가열
- 5: 오븐을 끄기
- 6: 그릇을 꺼내기
- 7: 밀가루, 우유, 계란 반죽하여
그릇에 담기

(b) 논리적 오류가 있는
제빵 알고리즘



논리적 오류의 디버깅 – 중간 단계의 결과 출력

```
# python program - logical error in while-loop & debugging

count = 0
total_sum = 0
while count < 10:
    total_sum = total_sum + count
    print("At count {:3}, total_sum = {:3}".format(count, total_sum))
```

```
At count 0, total_sum = 0
At count 0, total_sum = 0
At count 0, total_sum = 0
At count 0, total_sum = 0
At count 0, total_sum = 0
At count 0, total_sum = 0
At count 0, total_sum = 0
At count 0, total_sum = 0
At count 0, total_sum = 0
At count 0, total_sum = 0
At count 0, total_sum = 0
```



파이썬 키워드를 임의로 재지정한 경우

◆ 키워드의 잘못된 재지정

```
# Python program - debugging

n_str = "100"
print("n_str= {}, type(n_str) = {}".format(n_str, type(n_str)))
n = int(n_str)
print("n = {}, type(n) = {}".format(n, type(n)))

int = 123
m_str = "200"
m = int(m_str)
print("m = {}, type(m) = {}".format(m, type(m)))
```

```
n_str= 100, type(n_str) = <class 'str'>
n = 100, type(n) = <class 'int'>
Traceback (most recent call last):
  File "C:\MyPyPackage\TextBook - 2019\ch 2
, line 10, in <module>
    m = int(m_str)
TypeError: 'int' object is not callable
```



효과적인 프로그램 개발 방법 Tip 1

◆ 프로그램 설계 (알고리즘 및 자료구조 설계)에 더 많은 시간을 투자할 것

- 주어진 문제를 해결하기 위한 효율적인 알고리즘을 먼저 구성 할 것
- 구성된 알고리즘에 적합한 자료구조를 선택할 것
- 예) 100,000개의 학생 데이터로 부터 0.01초의 데이터 처리 시간 내에 특정 조건을 만족하는 학생을 탐색 (search)하여야 하는 경우, 어떤 자료구조를 사용하며, 어떤 방식으로 탐색할 것인가 ?
 - array vs. linked list vs. binary tree
 - sequential search vs. binary search



효과적인 프로그램 개발 방법 Tip 2

◆ 프로그램 오류 수정을 위한 **debugging** 방법을 먼저 숙달 할 것

- 프로그램 소스코드 상에 존재하는 문법적 에러 및 논리적인 에러를 빨리 찾아내는 방법을 먼저 숙달
- VS Code의 debugging 기능을 먼저 확인하고, 숙달할 것
 - break point 기능
 - trace 기능: step-over (F10), step-into(F11)
 - 각 단계에서의 local variable 값 확인



파이썬의 기본 연산자와 기본 명령어

파이썬 기본 연산자

◆ 파이썬 기본 연산자

자료형	구분	사용가능 연산자
불리언	부울대수 연산	and, or, not
숫자형	산술 계산	+ (덧셈), - (뺄셈), *(곱셈), / (실수 나눗셈), //(정수 나눗셈), %(모듈로)
	비교 연산	>, >=, <, <=, ==
	비트 단위 연산	& (bit-wise and), (bit-wise or), ^(bit-wise exclusive or), ~ (bit-wise not), << (bit-wise left shift), >> (bit-wise right shift)
시퀀스 (str, bytes, bytearray, memoryview, list, tuple)	접합, 반복	+ : 두 개가 시퀀스 객체를 순차적으로 접합 (concatenation) * : 시퀀스 객체를 지정된 회수만큼 반복
	인덱싱	[i] : 시퀀스 자료형 객체의 i번째 항목
	슬라이싱	[i:j:k] : 시퀀스 자료형 객체의 i번째 항목부터 k씩 인덱스를 증가하면서 j-1번째 까지의 항목
매핑(dict)	인덱싱	[keyword] : dict 자료형 객체에서 키워드 (keyword)로 지정된 항목
집합(set)	집합 관계	<= (부분집합), < (진부분 집합), > (진부분 집합), >= (부분집합), (합집합), & (교집합), - (차집합), ^ (대칭 차집합)



List, Tuple, Set, Dict

◆ Comparison of list, tuple, set, dict

type		mutable	example, methods, remarks
sequence	list	no	<code>L = [0, 1, 2, 3]</code> <code>L = [0, 1.0, 'a', b'xyz']</code> <code>len()</code> , <code>min()</code> , <code>max()</code> , <code>L.index()</code> , <code>L.count()</code> list with different types and different length are possible
	tuple	no	<code>T = (1, 2, 1, 2, 3)</code> <code>in</code> , <code>not-in</code> duplication is possible,
set	set	yes	<code>S = {1, 2, 3}</code> <code>S1 = S2</code> , <code>S1 &= S2</code> , <code>S1 -= S2</code> , <code>S1 ^= S2</code> unordered, no duplication, collection indexing and slicing are not used
	frozenset	no	immutable collection
mapping	dict	yes	<code>D = {1:'A', 2:'B', 2:'X', 3:'C'}</code> //key:value <code>D.items()</code> , <code>D.keys()</code> , <code>D.values()</code> , <code>D.update(key=value)</code>



파이썬 연산자 우선 순위 (1)

◆ 파이썬 연산자 우선 순위 (precedence)

precedence	operator	Meanings
highest	(expressions ...), [expressions...], {key:value...}, {expressions ...}	parenthesis (binding, tuple), list, dictionary, set
	x[index], x[index:index], x(arguments...), x.attribute	indexing, slicing function call, referring attribute
	**	power (exponentiation)
	+x, -x, ~x	unary operator, negation,
	*, @, /, //, %	multiplication, division, remainder
	+, -	binary addition, subtraction
	<<, >>	shift bits left, right
	&	bit-wise AND
	^	bit-wise XOR
		bit-wise OR



파이썬 연산자 우선 순위 (2)

◆ 파이썬 연산자 우선 순위 (precedence)

precedence	operator	Meanings
	in, not in, is, is not, <, <=, >, >=, !=, ==	comparison, membership, identity
	not x	logical NOT
	and	logical AND
	or	logical OR
	if – else	conditional expression
lowest	lambda	lambda expression



파이썬 기본 명령어 (1)

◆ 파이썬 기본 명령어

기본 명령어	구분	상세설명, 예
대입 (assignment), =	creating references	a, b = 'good', 'bad' a, b = b, a # swap a, b
func(args)	invoke function with arguments	log.write("spam, ham")
print	printing objects	print("Welcome to Python")
if, elif, else	selecting actions	조건문
for	iteration	for-반복문
while	general loops	while-반복문
pass	entry placeholder	블록의 항목
break	loop exit	반복문을 탈출
continue	loop continue	반복문의 나머지 구간을 생략
def	functions and methods	함수의 정의
return	functions results	함수의 결과값 반환
yield	generator functions	제네레이터 함수의 값 전달
global	namespaces	전역 네임스페이스 지정
nonlocal	namespaces	지역 네임스페이스에 해당되지 않음을 지정



파이썬 기본 명령어 (1)

◆ 파이썬 기본 명령어

기본 명령어	구분	상세설명, 예
import	module access	모듈을 추가
with / as	context managers	모듈의 이름 간략화
from	attribute access	모듈에서의 특정 함수/속성 추가
class	building objects	클래스
try / except / else/ finally	catching exceptions	예외상황 처리
raise	triggering exceptions	예외상황 발생
assert	debugging checks	실행 조건 지정 및 검사
del	deleting references	지정된 객체를 삭제



한 줄에 여러 정수 데이터 입력 - `split()`, `map()`

```
# input multiple numbers in one line using split() and map()
```

```
input_data_str = input("input multiple data (a b c) (separated in space) = ")
decimal_strings = input_data_str.split(sep= ' ')
print("decimal_strings = ", decimal_strings)
a, b, c = map(int, decimal_strings)
print("Input a = {}, b = {}, c = {}".format(a, b, c))
```

```
input multiple data (a b c) (separated in space) = 10 20 30
decimal_strings = ['10', '20', '30']
Input a = 10, b = 20, c = 30
```



한 줄에 입력된 정수 데이터들을 사용한 list 생성

```
# input multiple numbers from one line using split() and map()
```

```
input_data_str = input("input data : ")
decimal_strings = input_data_str.split(sep=' ')
print("Input decimal_strings : ", decimal_strings)
L = list(map(int, decimal_strings))
print("Input integers : ", L)
```

```
input data : 1 2 3 4 5 6 7 8 9 10
Input decimal_strings : ['1', '2', '3', '4', '5', '6', '7', '8', '9', '10']
Input integers : [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

input data : 1 1234 6543 9876
Input decimal_strings : ['1', '1234', '6543', '9876']
Input integers : [1, 1234, 6543, 9876]
```



한줄에 여러 개의 실수 (float) 데이터 입력

```
# input multiple float data in one line using map(), split(), and float()

input_data_str = input("input width length height in floats (separated in space) = ")
float_strings = input_data_str.split(sep= ' ')
print("float_strings = ", float_strings)
width, length, height = map(float, float_strings)
print("Input width = {}, length = {}, height = {}".format(width, length, height))
```

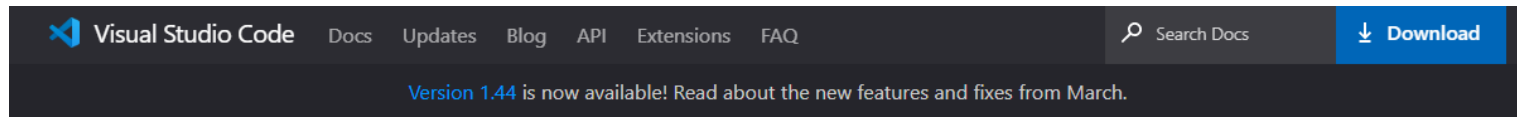
```
input width length height in floats (separated in space) = 1.23 4.56 7.89
float_strings = ['1.23', '4.56', '7.89']
Input width = 1.23, length = 4.56, height = 7.89
```



Visual Studio Code (VS Code)를 사용한 파이썬 프로그램 작성 및 디버깅


Python Program Debugging with Visual Studio Code (VS Code)

◆ <https://code.visualstudio.com/download>



Download Visual Studio Code

Free and built on open source. Integrated Git, debugging and extensions.




↓ Windows
Windows 7, 8, 10

User Installer64 bit 32 bit

System Installer64 bit 32 bit

.zip64 bit 32 bit



↓ .deb
Debian, Ubuntu


↓ .rpm
Red Hat, Fedora, SUSE

.deb64 bit

.rpm64 bit

.tar.gz64 bit

Snap Store



↓ Mac
macOS 10.10+



Download Visual Studio Code (System Installer 32-bit)

Visual Studio Code Docs Updates Blog API Extensions FAQ

Search Docs

Download

Version 1.44 is now available! Read about the new features and fixes from March.

Overview

SETUP

GET STARTED

USER GUIDE

LANGUAGES

NODE.JS / JAVASCRIPT

TYPESCRIPT

PYTHON

JAVA

C++

CONTAINERS

AZURE

REMOTE

Thanks for downloading VS Code for Windows!

Download not starting? Try this [direct download link](#).
Please take a few seconds and help us improve ... [click to take survey](#).

Getting Started

Visual Studio Code is a lightweight but powerful source code editor which runs on your desktop and is available for Windows, macOS and Linux. It comes with built-in support for JavaScript, TypeScript and Node.js and has a rich ecosystem of extensions for other languages (such as C++, C#, Java, Python, PHP, Go) and runtimes (such as .NET and Unity). Begin your journey with VS Code with these [introductory videos](#).

Visual Studio Code in Action

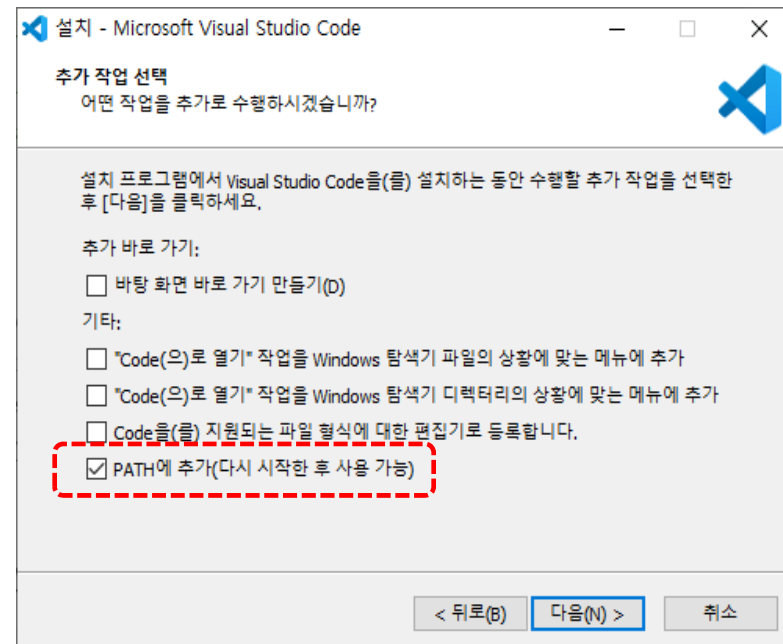
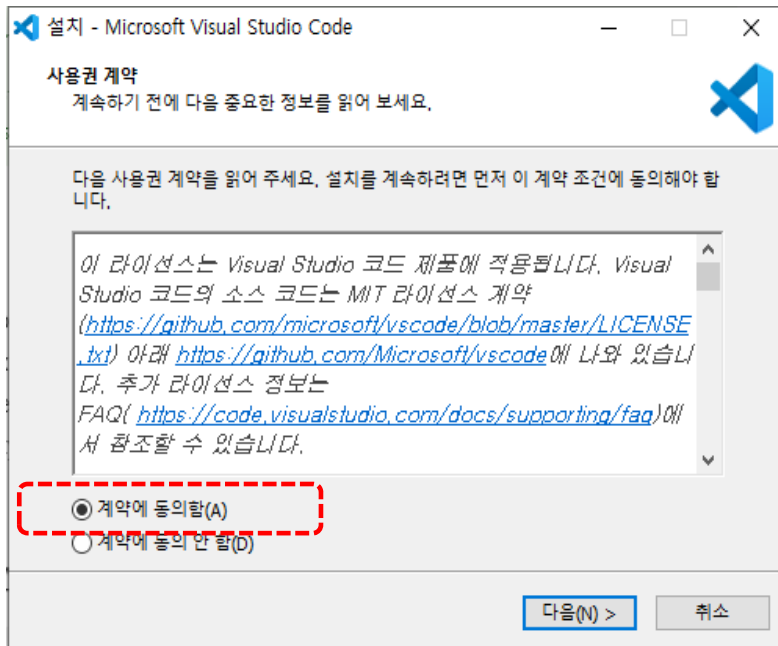
```
4 var server = express();
5 server.use(bodyParser.json);
6
7 server.get
8   get (property) Application.get: ((name: string)...)
9   getMaxListeners
10  arguments
11  engine
12  length
13  merge
14
```

VSCodeSetup-ia3...exe
5.7/54.5MB, 8분 남음

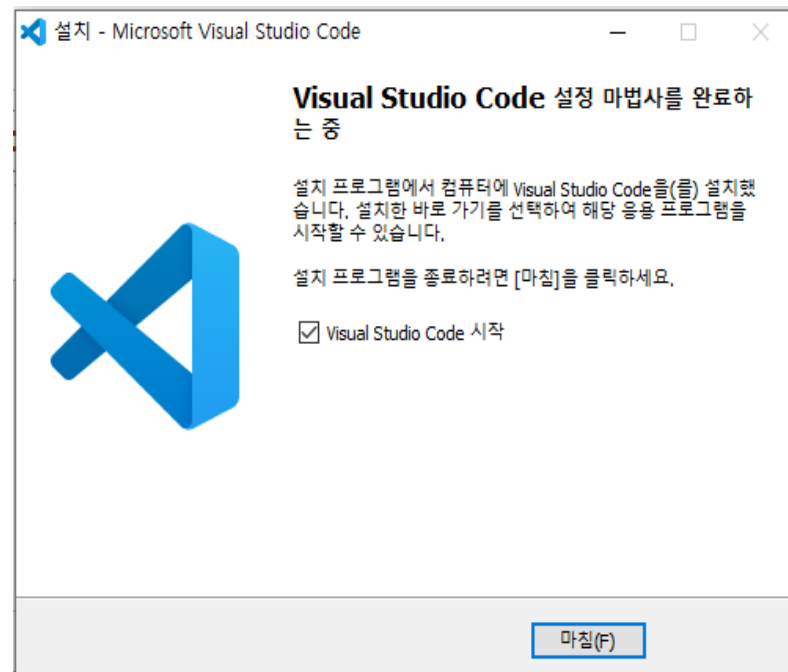
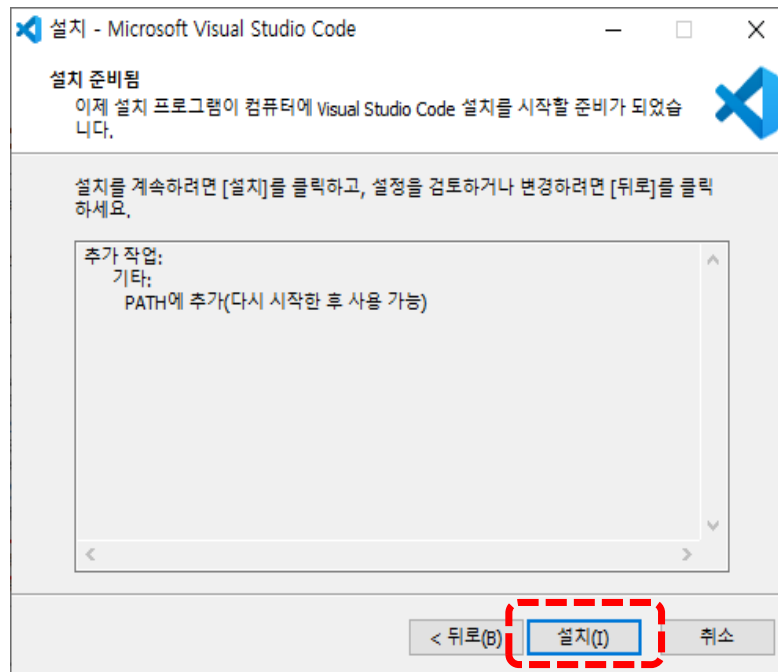
모두 표시



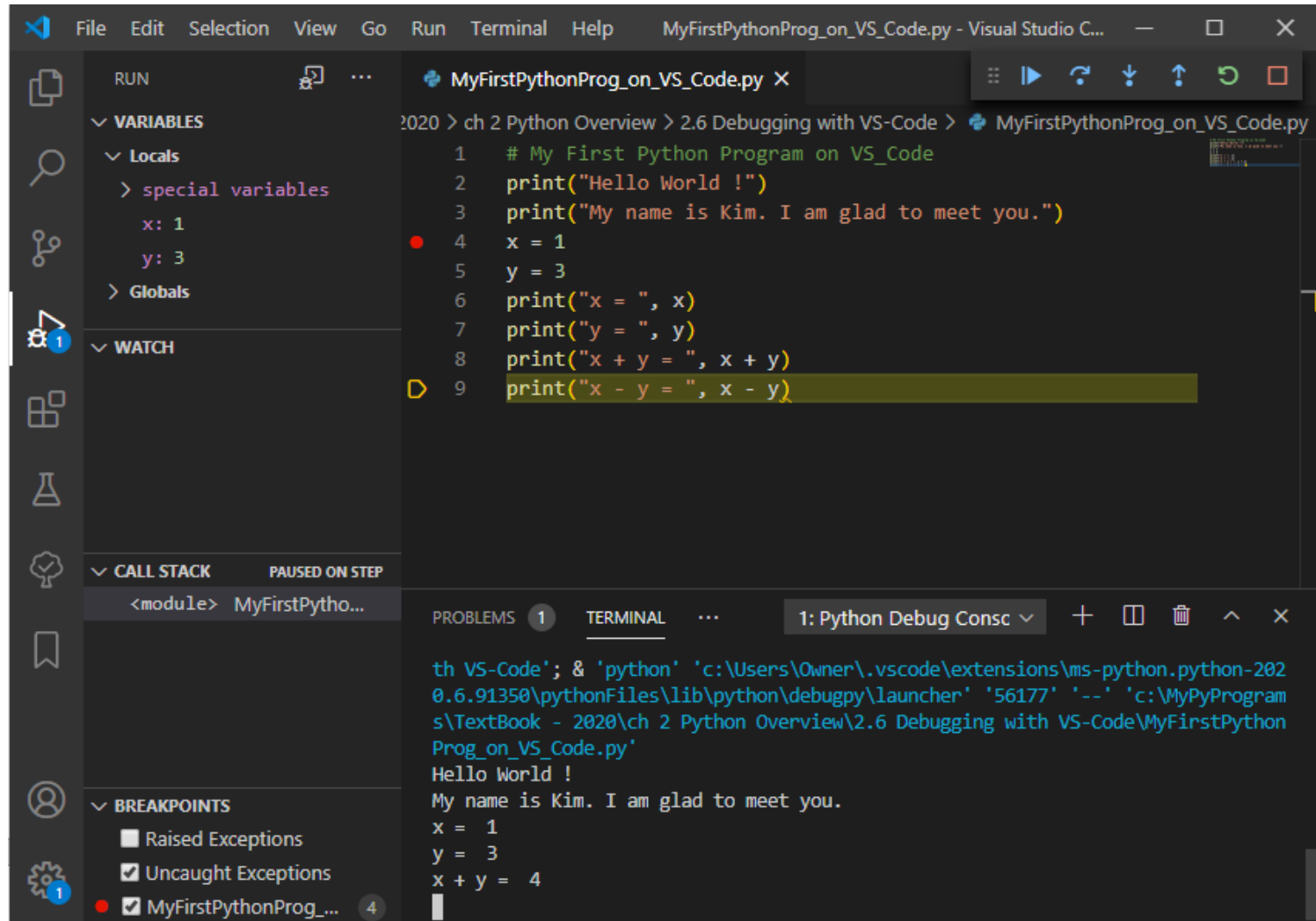
Visual Studio Code 설치



Visual Studio Code 설치



MyFirstPythonProg_on_VS_Code



The screenshot shows the Visual Studio Code interface with a Python file named `MyFirstPythonProg_on_VS_Code.py` open. The code is as follows:

```
1 # My First Python Program on VS_Code
2 print("Hello World !")
3 print("My name is Kim. I am glad to meet you.")
4 x = 1
5 y = 3
6 print("x = ", x)
7 print("y = ", y)
8 print("x + y = ", x + y)
9 print("x - y = ", x - y)
```

The left sidebar contains the following panels:

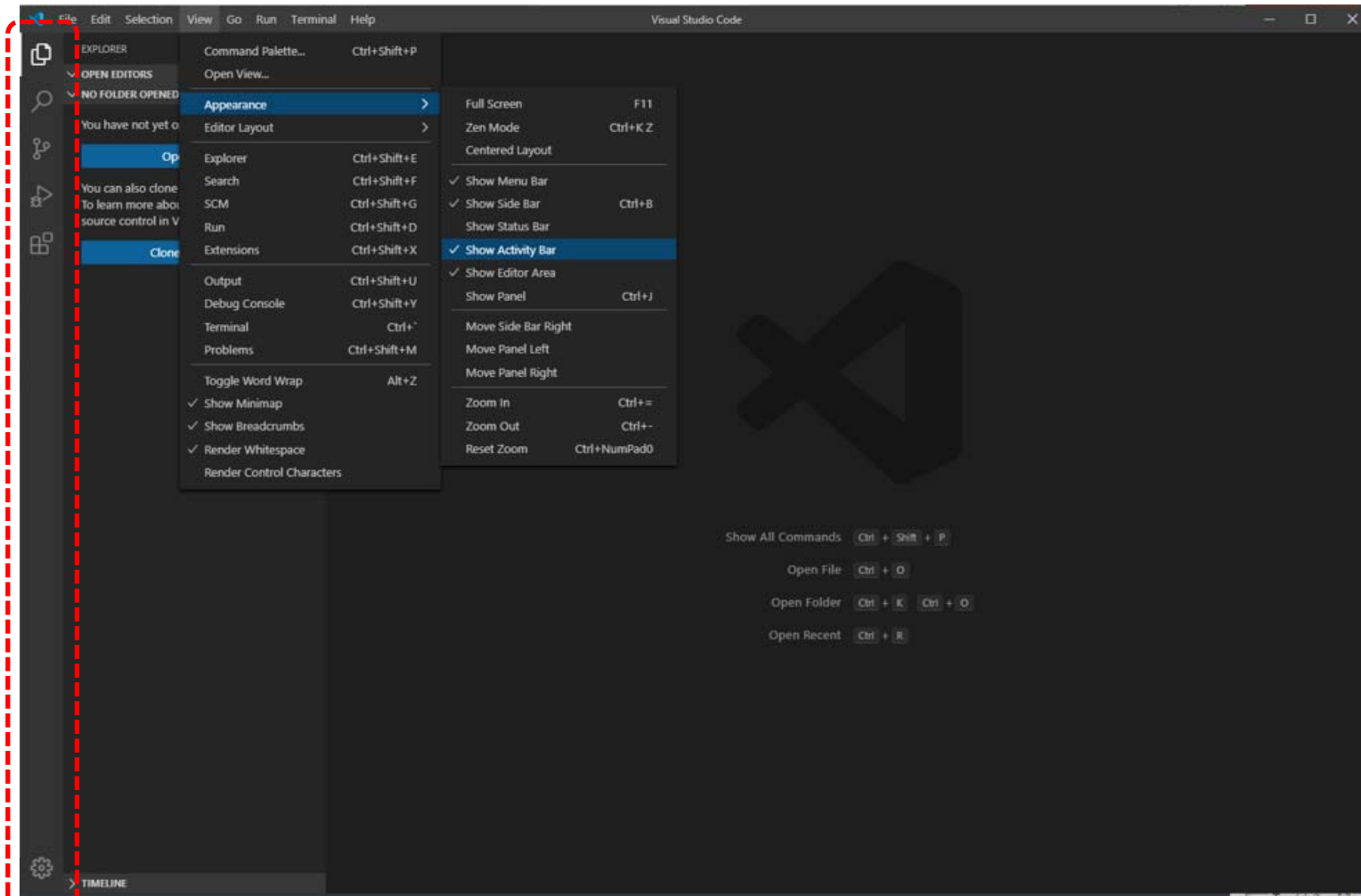
- RUN**: Shows the execution status.
- VARIABLES**: Shows the current state of variables.
 - Locals**:
 - `special variables`:
 - `x`: 1
 - `y`: 3
 - Globals**
 - WATCH**: Shows the state of watched variables.
 - CALL STACK**: Shows the call stack, currently paused on step 1.
 - BREAKPOINTS**: Shows the breakpoints, with one breakpoint set at line 4.

The bottom panel shows the **TERMINAL** output:

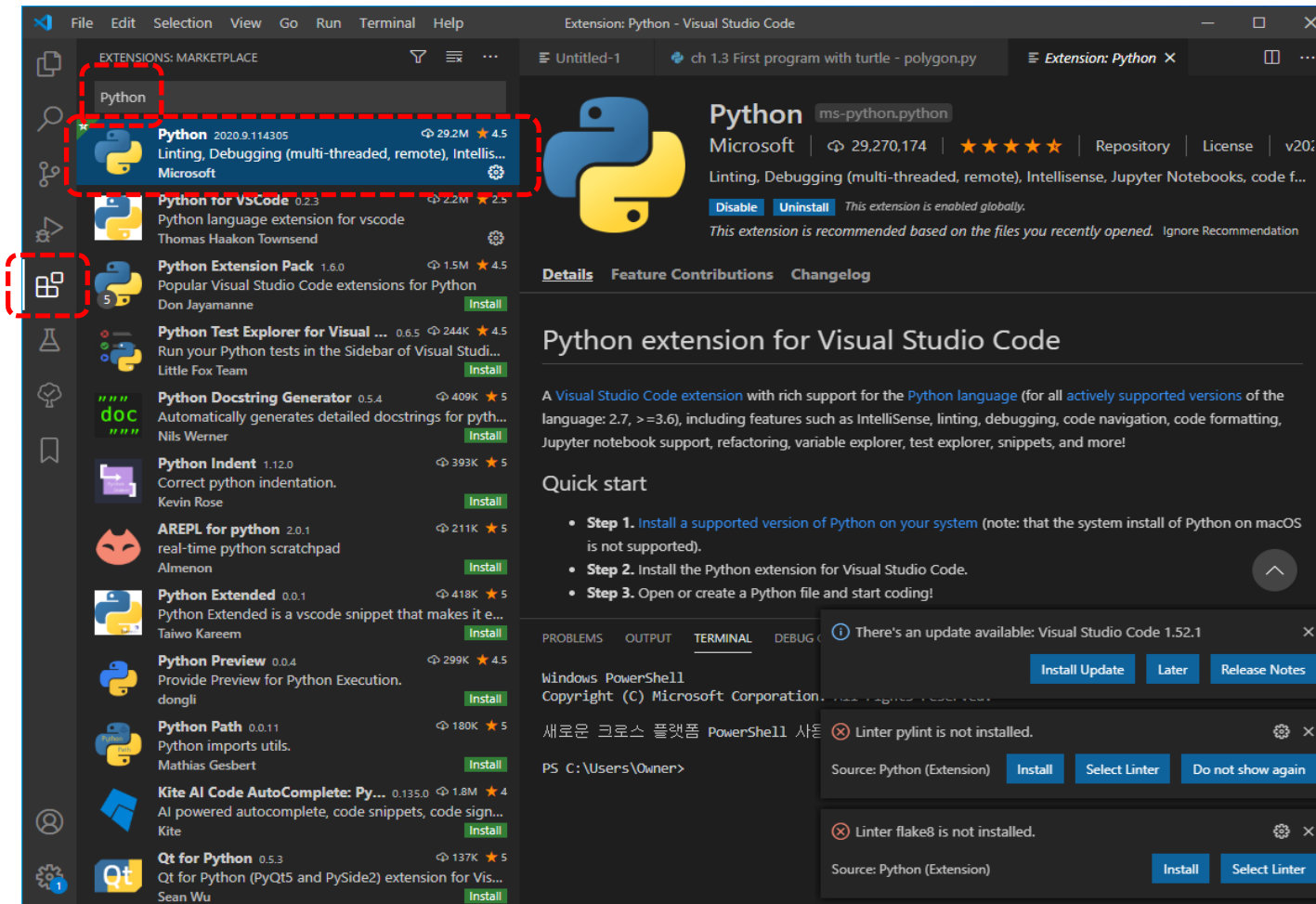
```
th VS-Code'; & 'python' 'c:\Users\Owner\.vscode\extensions\ms-python.python-2020.6.91350\pythonFiles\lib\python\debugpy\launcher' '56177' '--' 'c:\MyPyPrograms\TextBook - 2020\ch 2 Python Overview\2.6 Debugging with VS-Code\MyFirstPythonProg_on_VS_Code.py'
Hello World !
My name is Kim. I am glad to meet you.
x = 1
y = 3
x + y = 4
```



View -> appearance -> show activity bar

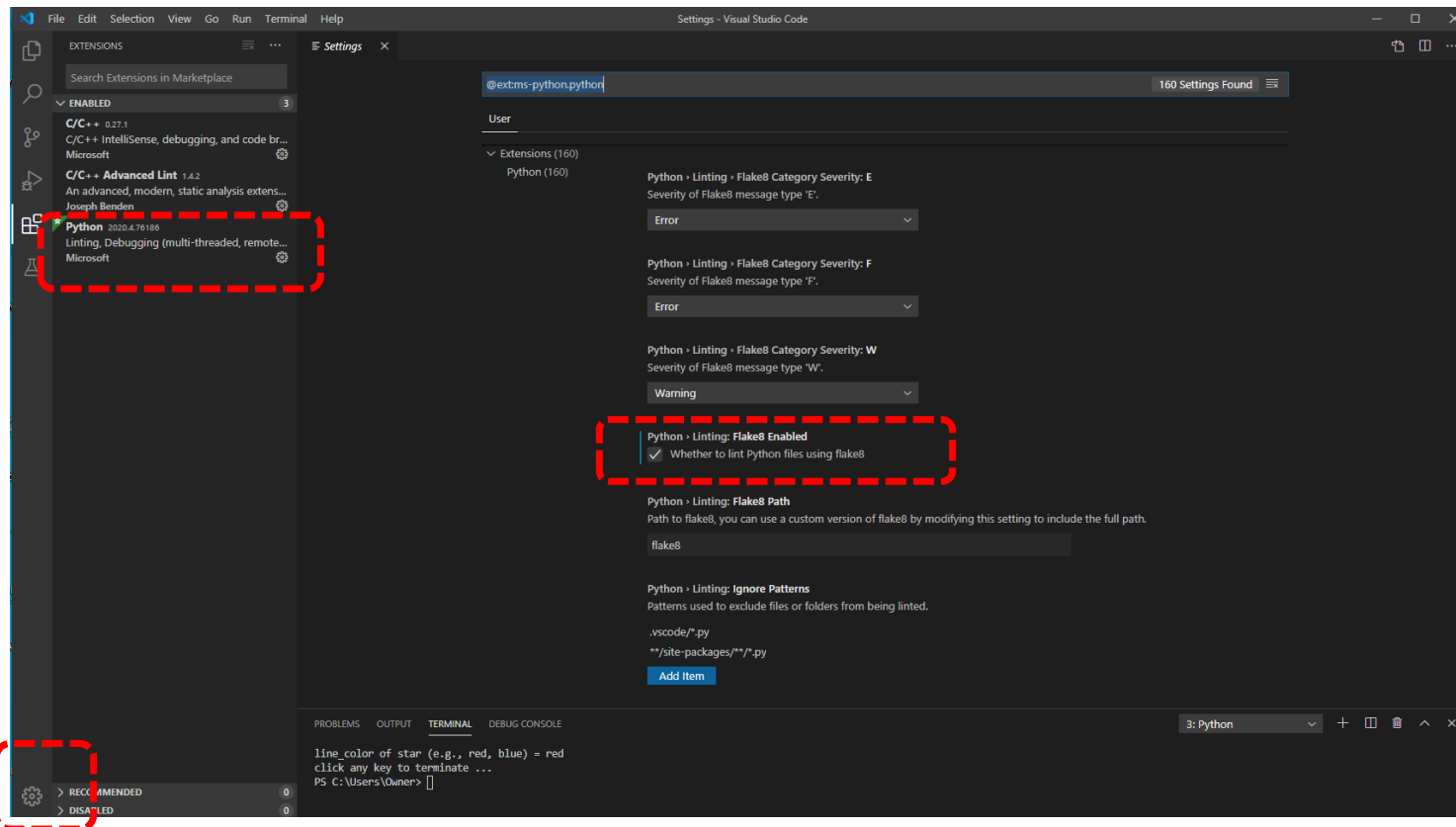


Extensions (control – shift – x) => Python 추가



Visual Studio Code에 Turtle Graphic관련 모듈 설치

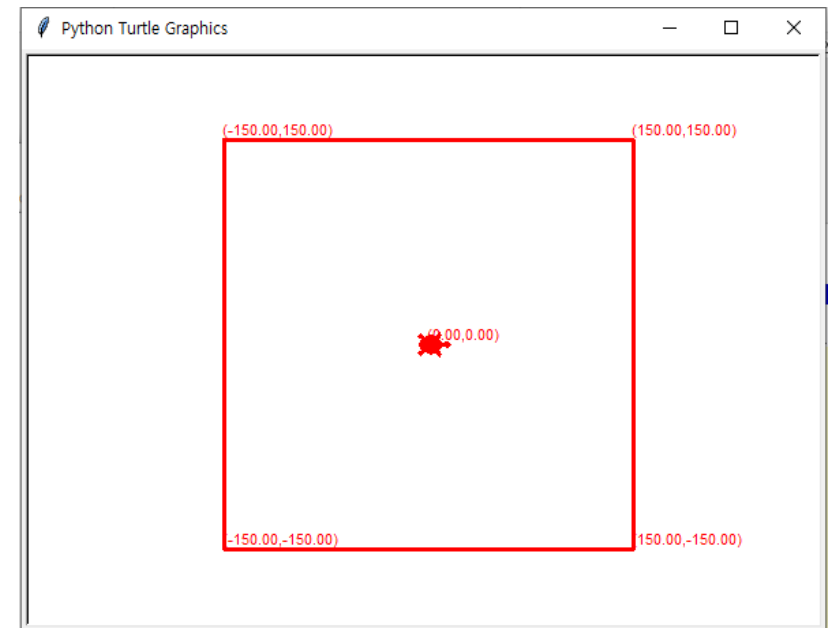
◆ Manage -> Python -> Extension Settings -> Linting : Flake8 Enabled



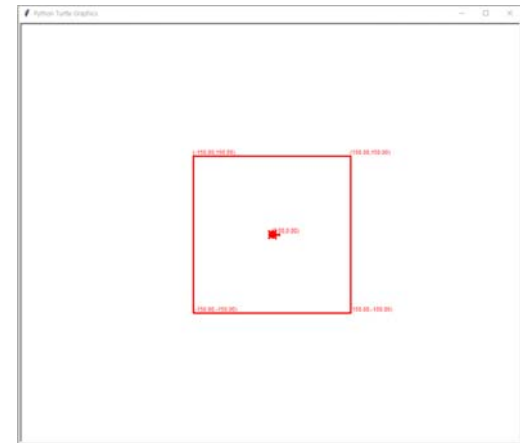
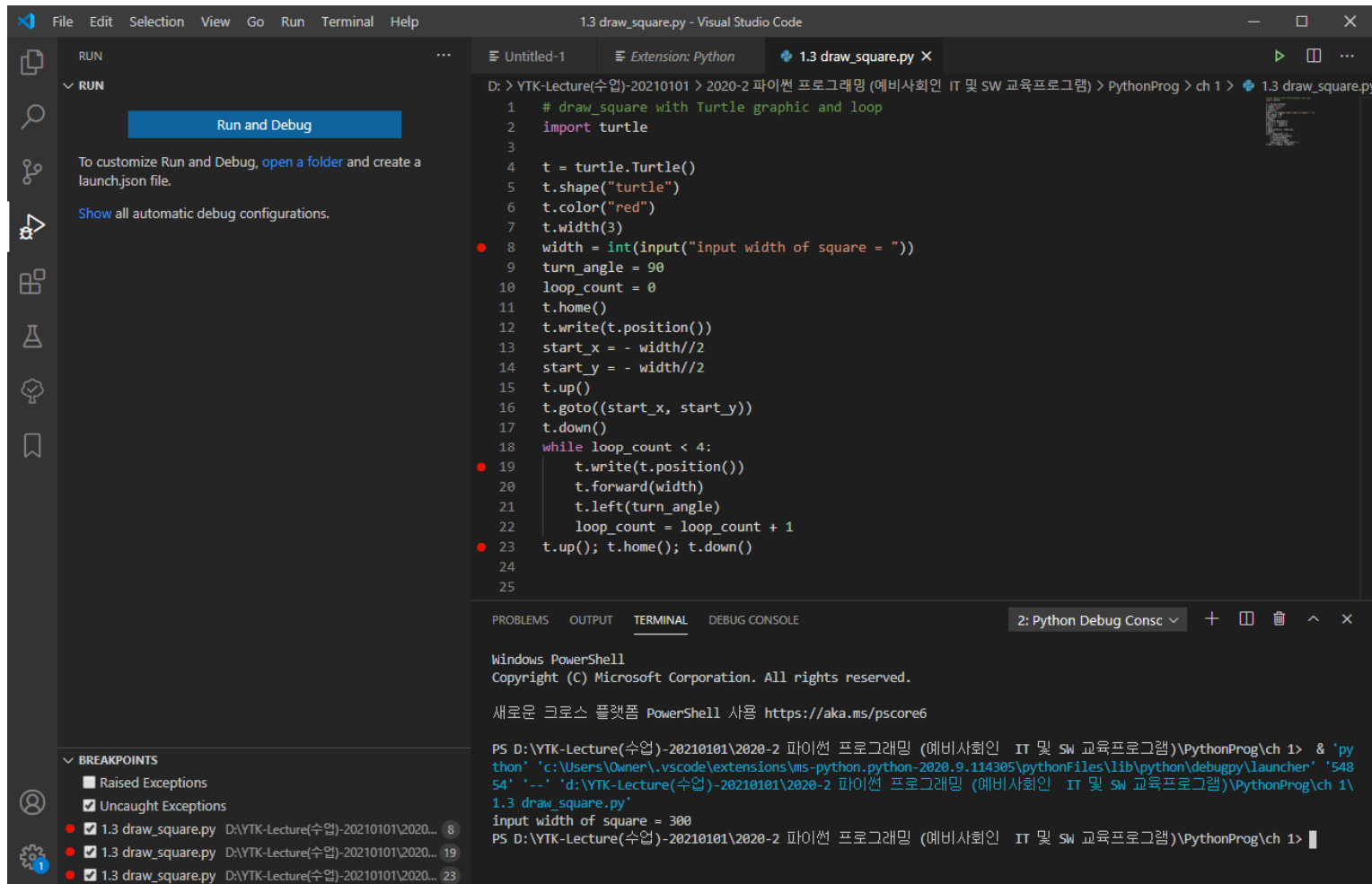
터틀 프로그래밍 예제 – draw_square.py

```
# draw_square with Turtle graphic and loop
import turtle

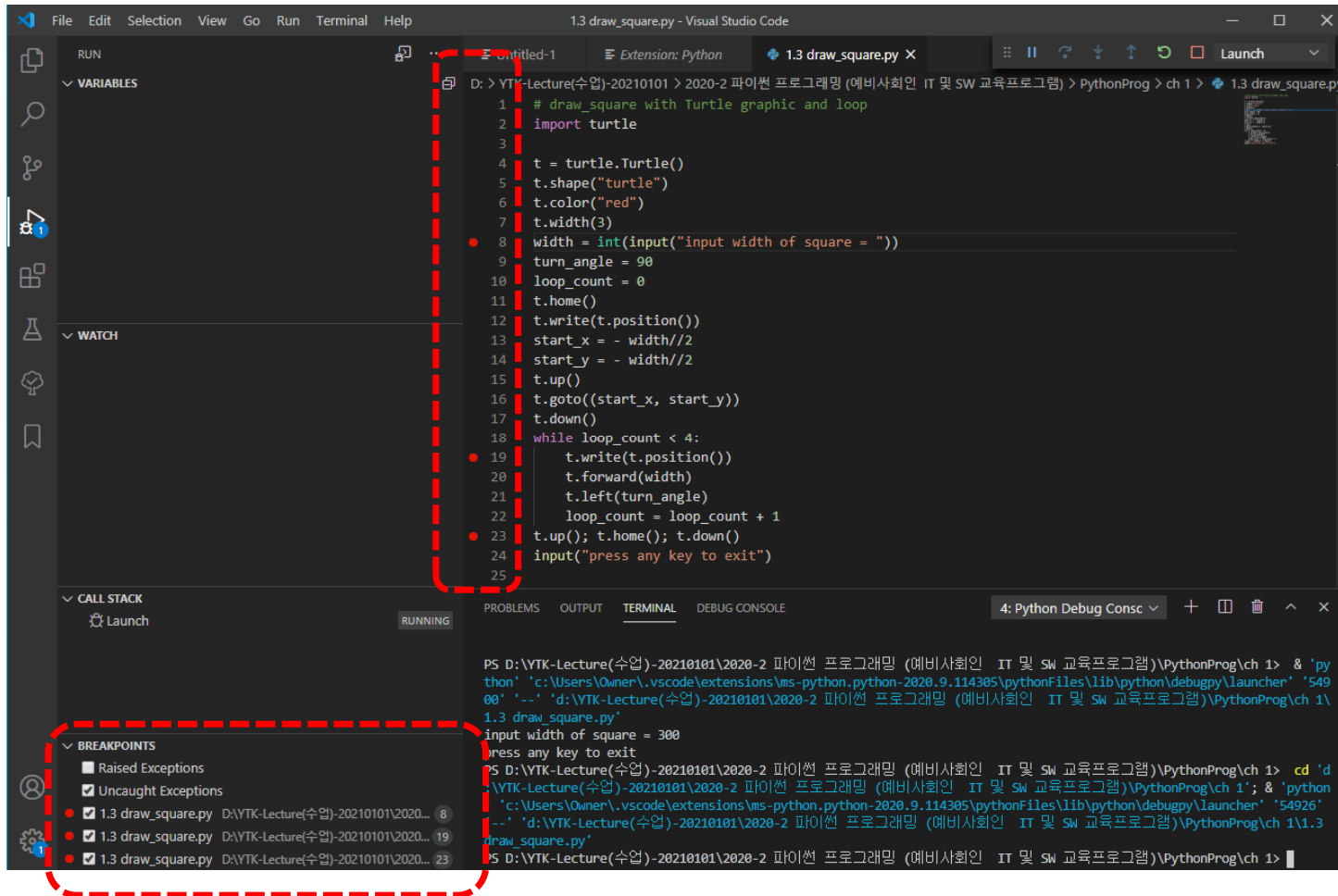
t = turtle.Turtle()
t.shape("turtle")
t.color("red")
t.width(3)
width = int(input("input width of square = "))
turn_angle = 90
loop_count = 0
t.home()
t.write(t.position())
start_x = - width//2
start_y = - width//2
t.up()
t.goto((start_x, start_y))
t.down()
while loop_count < 4:
    t.write(t.position())
    t.forward(width)
    t.left(turn_angle)
    loop_count = loop_count + 1
t.up(); t.home(); t.down()
input("press any key to exit")
```



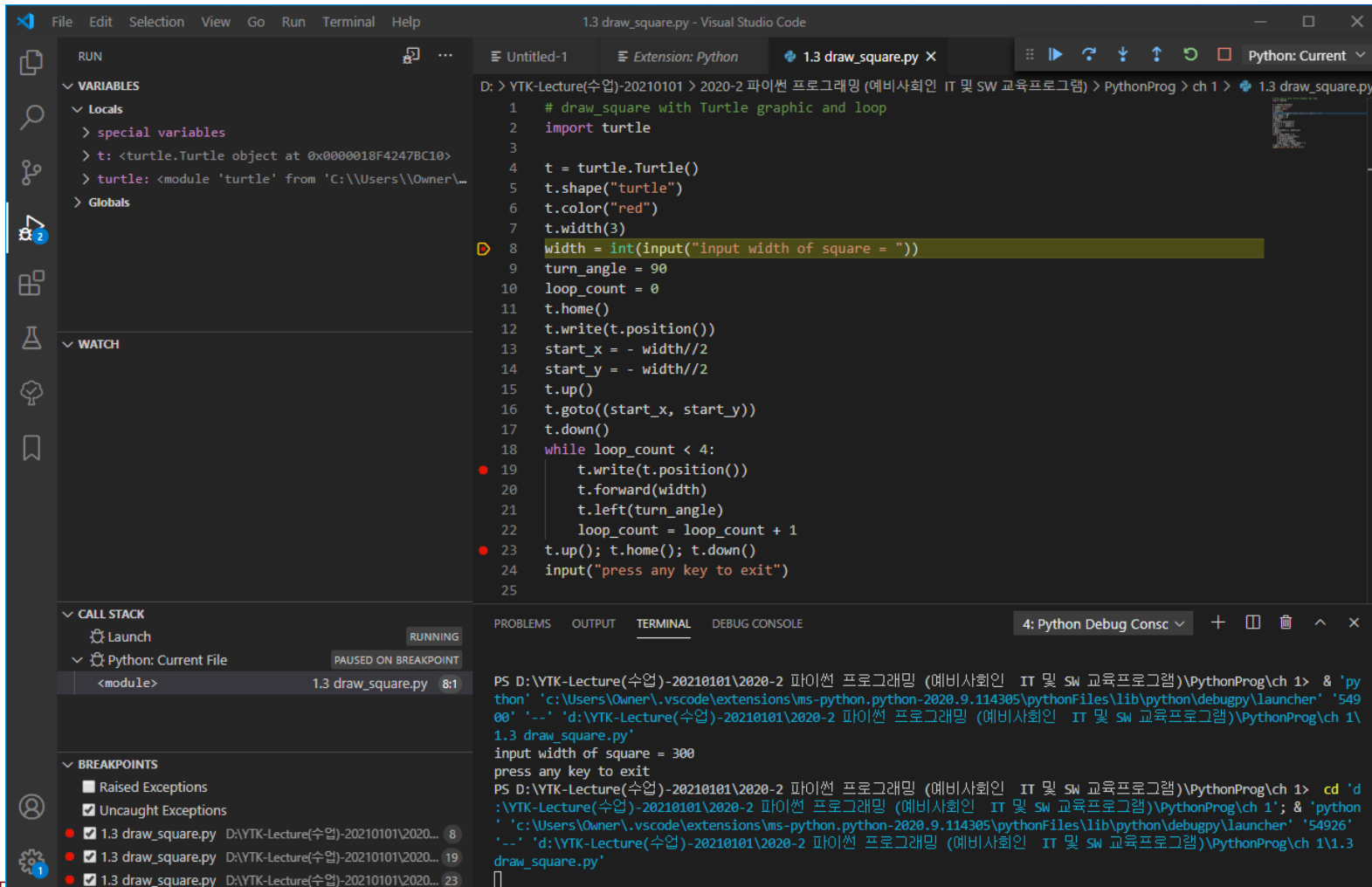
Run without Debugging (Ctrl + F5)



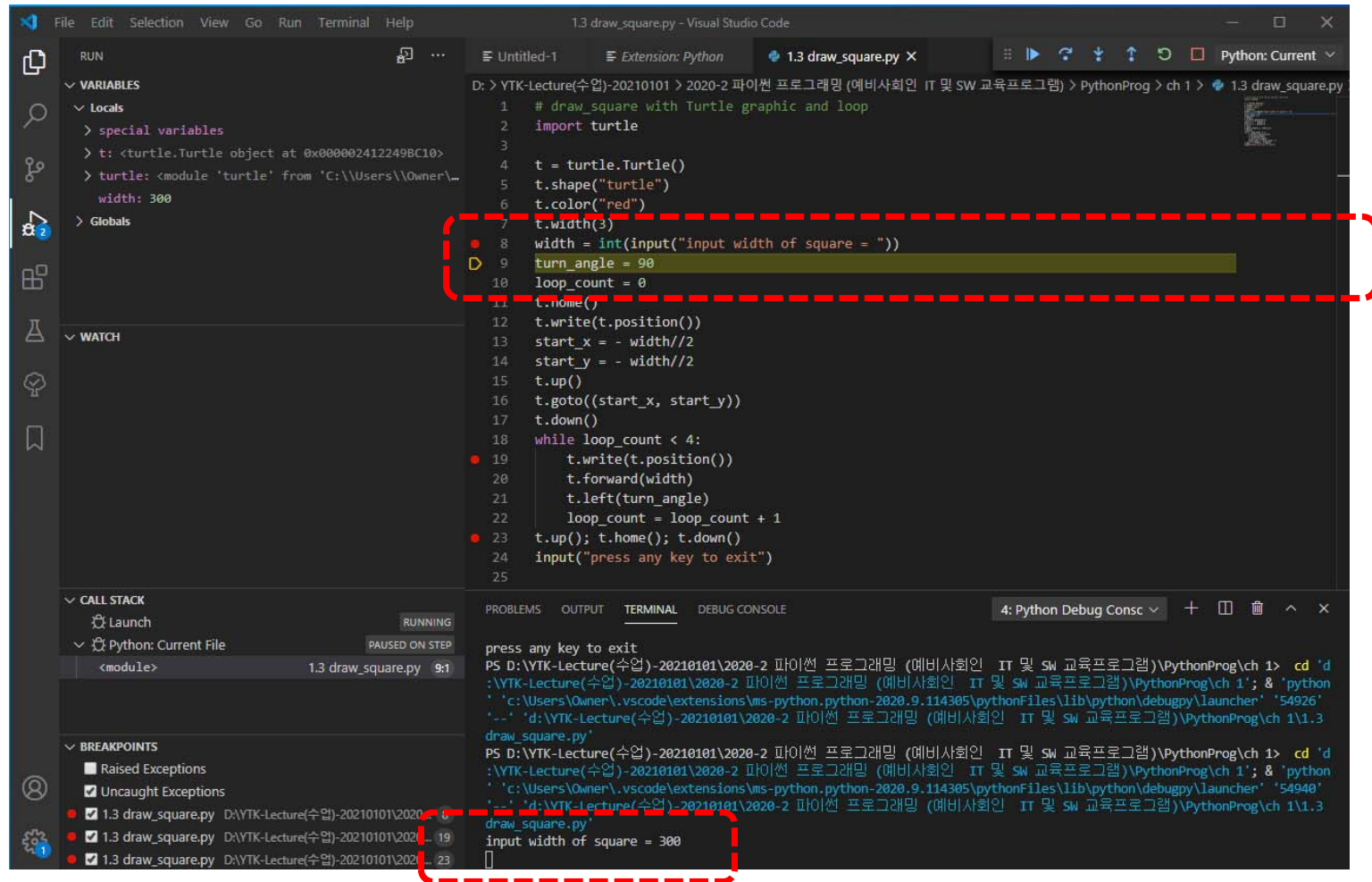
Break Points 설정



Run -> Start Debugging (F5) -> Python File



Step-over (F10)을 사용한 프로그램 실행



```
1 # draw_square with Turtle graphic and loop
2 import turtle
3
4 t = turtle.Turtle()
5 t.shape("turtle")
6 t.color("red")
7 t.width(3)
8 width = int(input("input width of square = "))
9 turn_angle = 90
10 loop_count = 0
11 t.home()
12 t.write(t.position())
13 start_x = - width//2
14 start_y = - width//2
15 t.up()
16 t.goto((start_x, start_y))
17 t.down()
18 while loop_count < 4:
19     t.write(t.position())
20     t.forward(width)
21     t.left(turn_angle)
22     loop_count = loop_count + 1
23 t.up(); t.home(); t.down()
24 input("press any key to exit")
25
```

press any key to exit

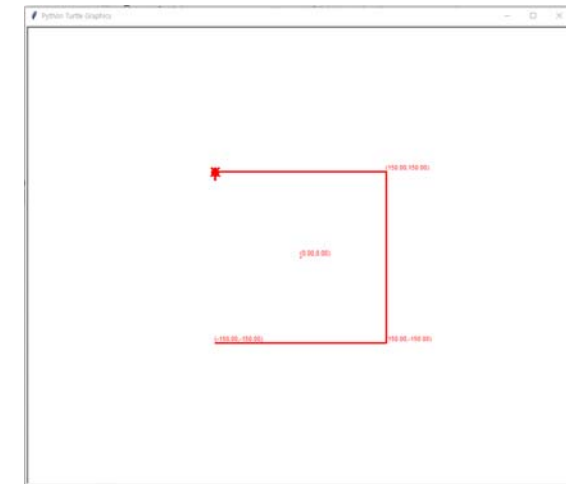
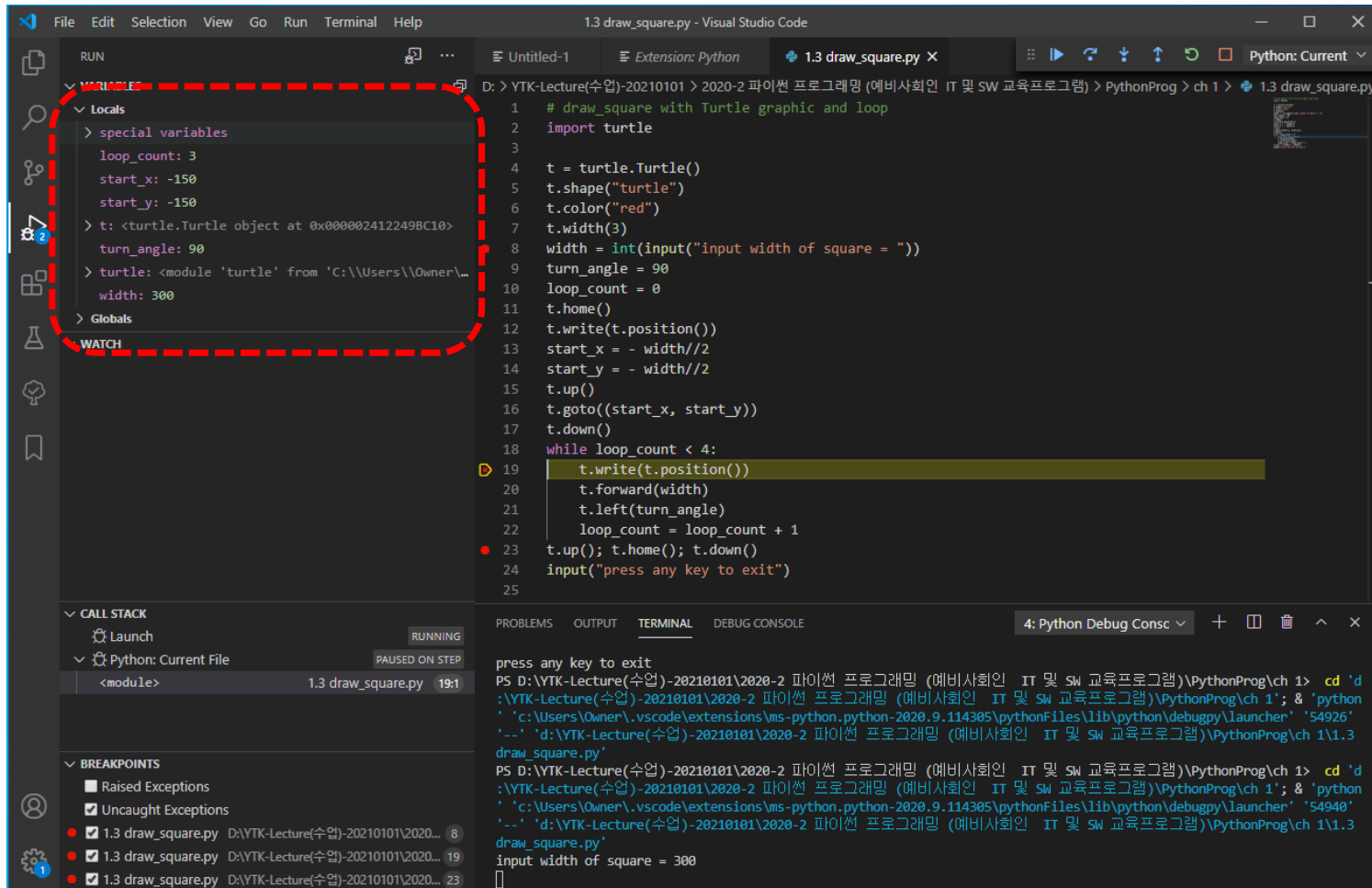
PS D:\YTK-Lecture(수업)-20210101\2020-2 파이썬 프로그래밍 (메비사하인 IT 및 SW 교육프로그램)\PythonProg\ch 1> cd 'd:\YTK-Lecture(수업)-20210101\2020-2 파이썬 프로그래밍 (메비사하인 IT 및 SW 교육프로그램)\PythonProg\ch 1'; & 'python' 'c:\Users\Owner\.vscode\extensions\ms-python.python-2020.9.114305\pythonFiles\lib\python\debugpy\launcher' '54926' '-.' 'd:\YTK-Lecture(수업)-20210101\2020-2 파이썬 프로그래밍 (메비사하인 IT 및 SW 교육프로그램)\PythonProg\ch 1\1.3 draw_square.py'

PS D:\YTK-Lecture(수업)-20210101\2020-2 파이썬 프로그래밍 (메비사하인 IT 및 SW 교육프로그램)\PythonProg\ch 1> cd 'd:\YTK-Lecture(수업)-20210101\2020-2 파이썬 프로그래밍 (메비사하인 IT 및 SW 교육프로그램)\PythonProg\ch 1'; & 'python' 'c:\Users\Owner\.vscode\extensions\ms-python.python-2020.9.114305\pythonFiles\lib\python\debugpy\launcher' '54926' '-.' 'd:\YTK-Lecture(수업)-20210101\2020-2 파이썬 프로그래밍 (메비사하인 IT 및 SW 교육프로그램)\PythonProg\ch 1\1.3 draw_square.py'

input width of square = 300



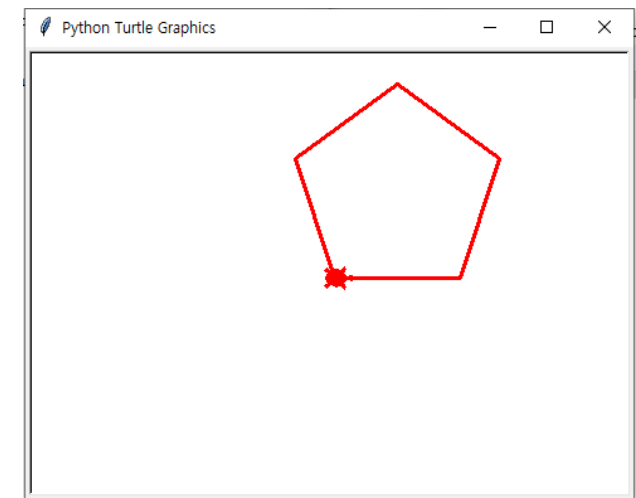
Step-over (F10)을 단계별 실행 결과 확인



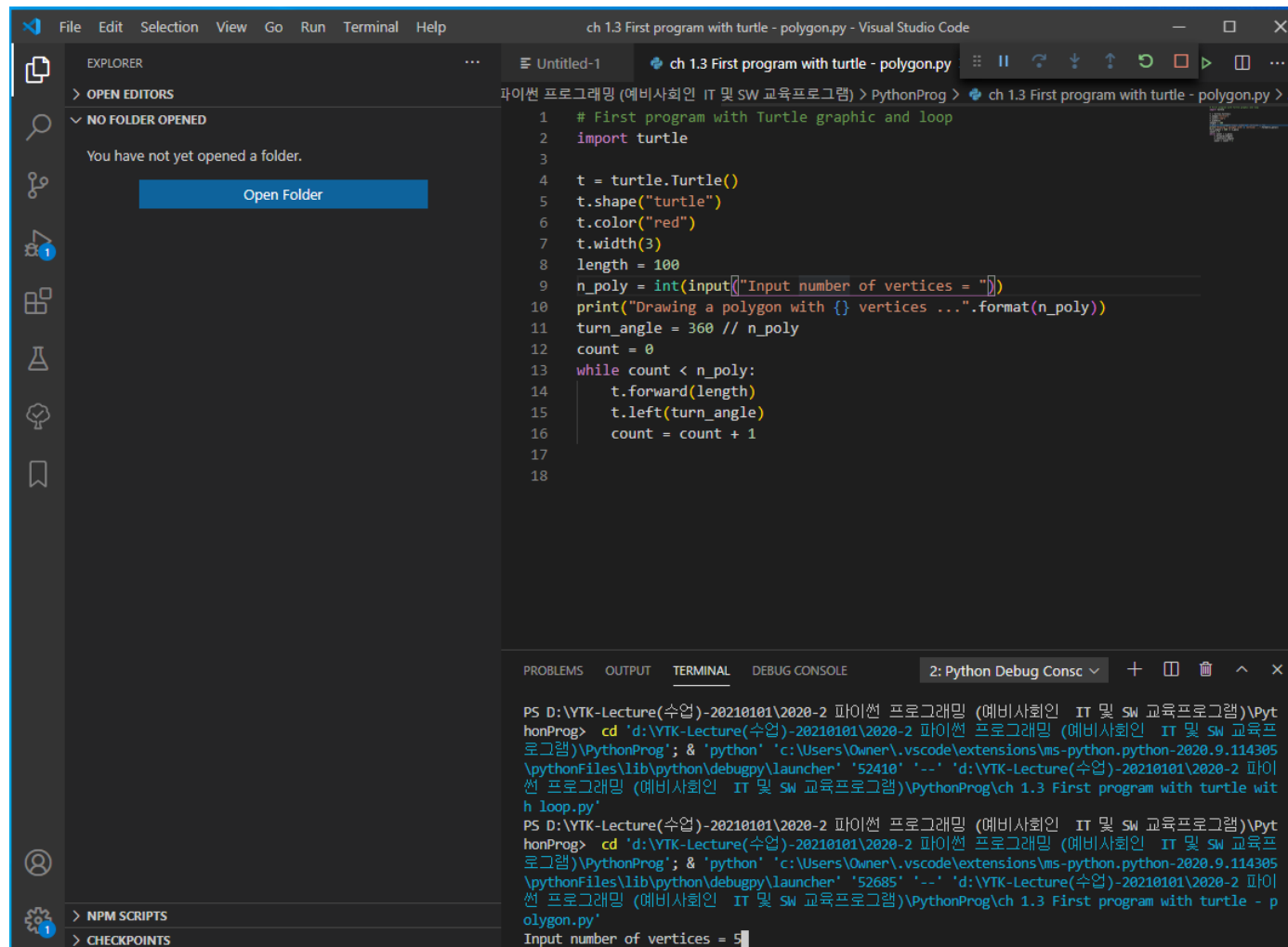
Sample Python Source – turtle_polygon.py

```
# Drawing Polygon with Turtle graphic and loop
import turtle

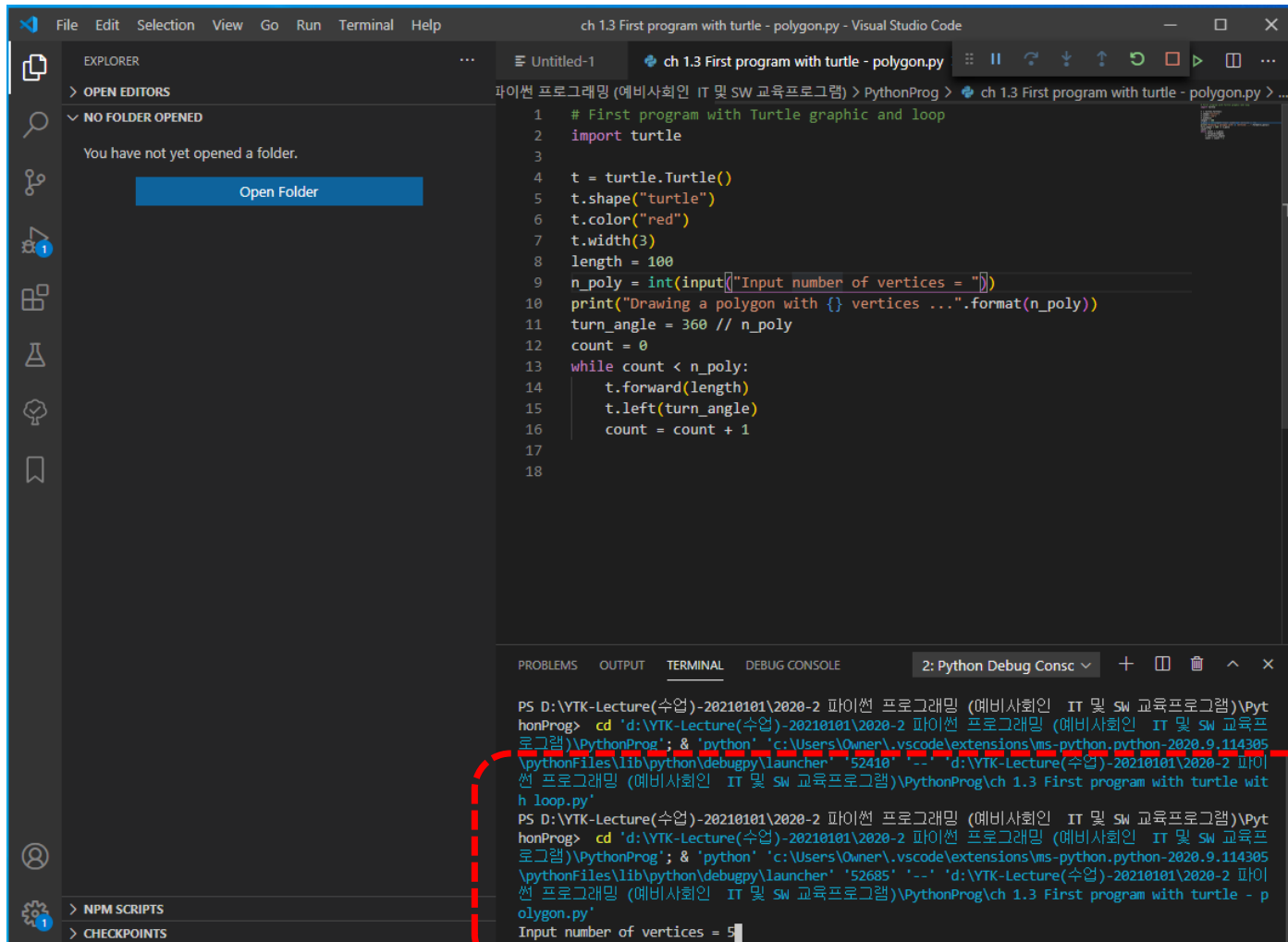
t = turtle.Turtle()
t.shape("turtle")
t.color("red")
t.width(3)
length = 100
n_poly = int(input("Input number of vertices = "))
print("Drawing a polygon with {} vertices ...".format(n_poly))
turn_angle = 360 // n_poly
loop_count = 0
while loop_count < n_poly:
    t.forward(length)
    t.left(turn_angle)
    loop_count = loop_count + 1
```



File Open on Visual Studio Code (File -> Open File (Ctrl+O))



Run without Debugging on Visual Studio Code (Ctrl + F5)



The screenshot shows the Visual Studio Code interface. The Explorer panel on the left shows 'NO FOLDER OPENED'. The main editor displays a Python file named 'ch 1.3 First program with turtle - polygon.py'. The code is as follows:

```
1 # First program with Turtle graphic and loop
2 import turtle
3
4 t = turtle.Turtle()
5 t.shape("turtle")
6 t.color("red")
7 t.width(3)
8 length = 100
9 n_poly = int(input("Input number of vertices = "))
10 print("Drawing a polygon with {} vertices ...".format(n_poly))
11 turn_angle = 360 // n_poly
12 count = 0
13 while count < n_poly:
14     t.forward(length)
15     t.left(turn_angle)
16     count = count + 1
17
18
```

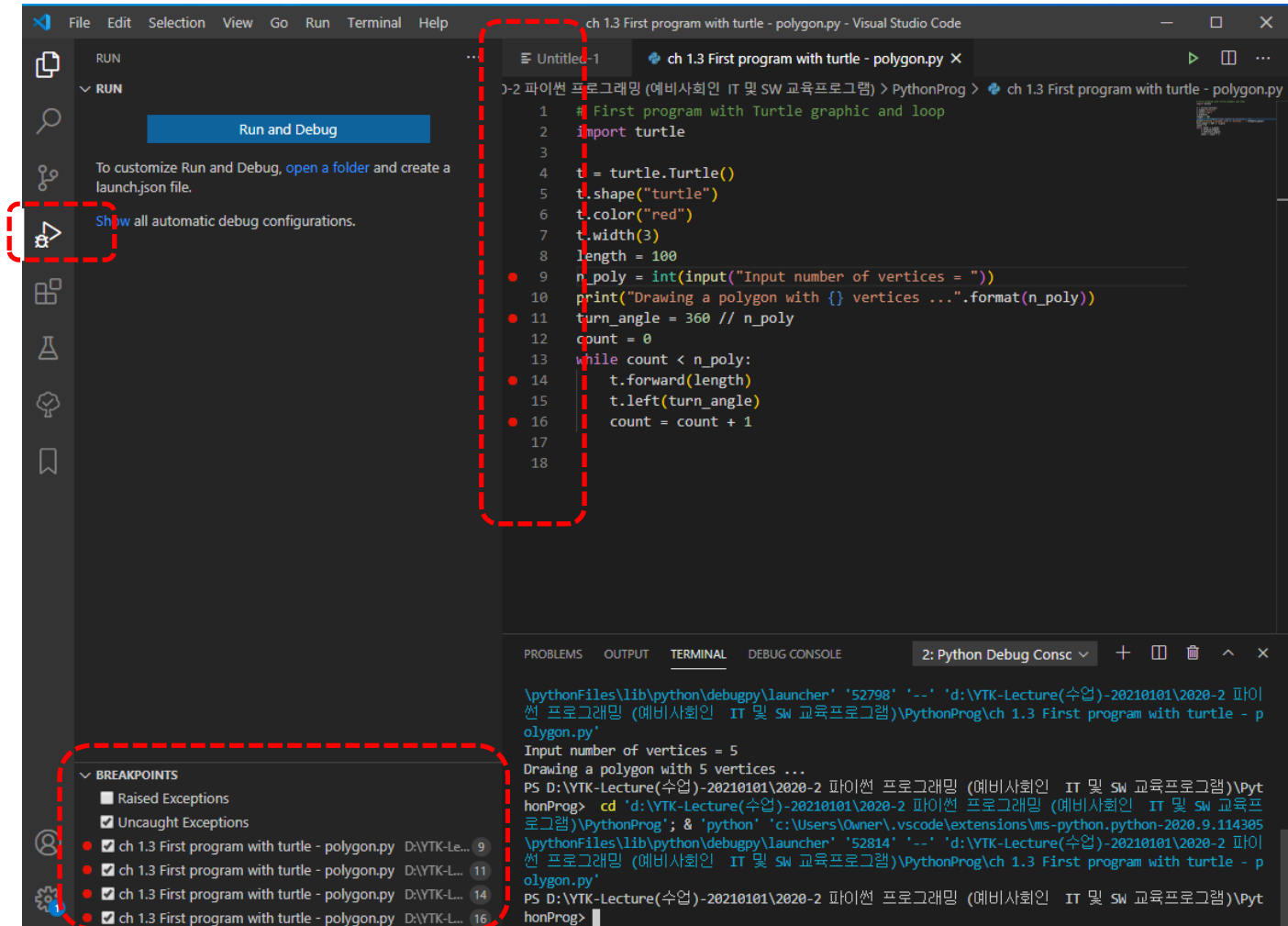
The bottom panel shows the 'TERMINAL' tab with the following output:

```
PS D:\YTK-Lecture(수업)-20210101\2020-2 파이썬 프로그래밍 (예비사회인 IT 및 SW 교육프로그램)\PythonProg> cd 'd:\YTK-Lecture(수업)-20210101\2020-2 파이썬 프로그래밍 (예비사회인 IT 및 SW 교육프로그램)\PythonProg'; & 'python' 'c:\Users\Owner\.vscode\extensions\ms-python.python-2020.9.114305\pythonFiles\lib\python\debugpy\launcher' '52410' '--' 'd:\YTK-Lecture(수업)-20210101\2020-2 파이썬 프로그래밍 (예비사회인 IT 및 SW 교육프로그램)\PythonProg\ch 1.3 First program with turtle with loop.py'
PS D:\YTK-Lecture(수업)-20210101\2020-2 파이썬 프로그래밍 (예비사회인 IT 및 SW 교육프로그램)\PythonProg> cd 'd:\YTK-Lecture(수업)-20210101\2020-2 파이썬 프로그래밍 (예비사회인 IT 및 SW 교육프로그램)\PythonProg'; & 'python' 'c:\Users\Owner\.vscode\extensions\ms-python.python-2020.9.114305\pythonFiles\lib\python\debugpy\launcher' '52685' '--' 'd:\YTK-Lecture(수업)-20210101\2020-2 파이썬 프로그래밍 (예비사회인 IT 및 SW 교육프로그램)\PythonProg\ch 1.3 First program with turtle - polygon.py'
Input number of vertices = 5
```

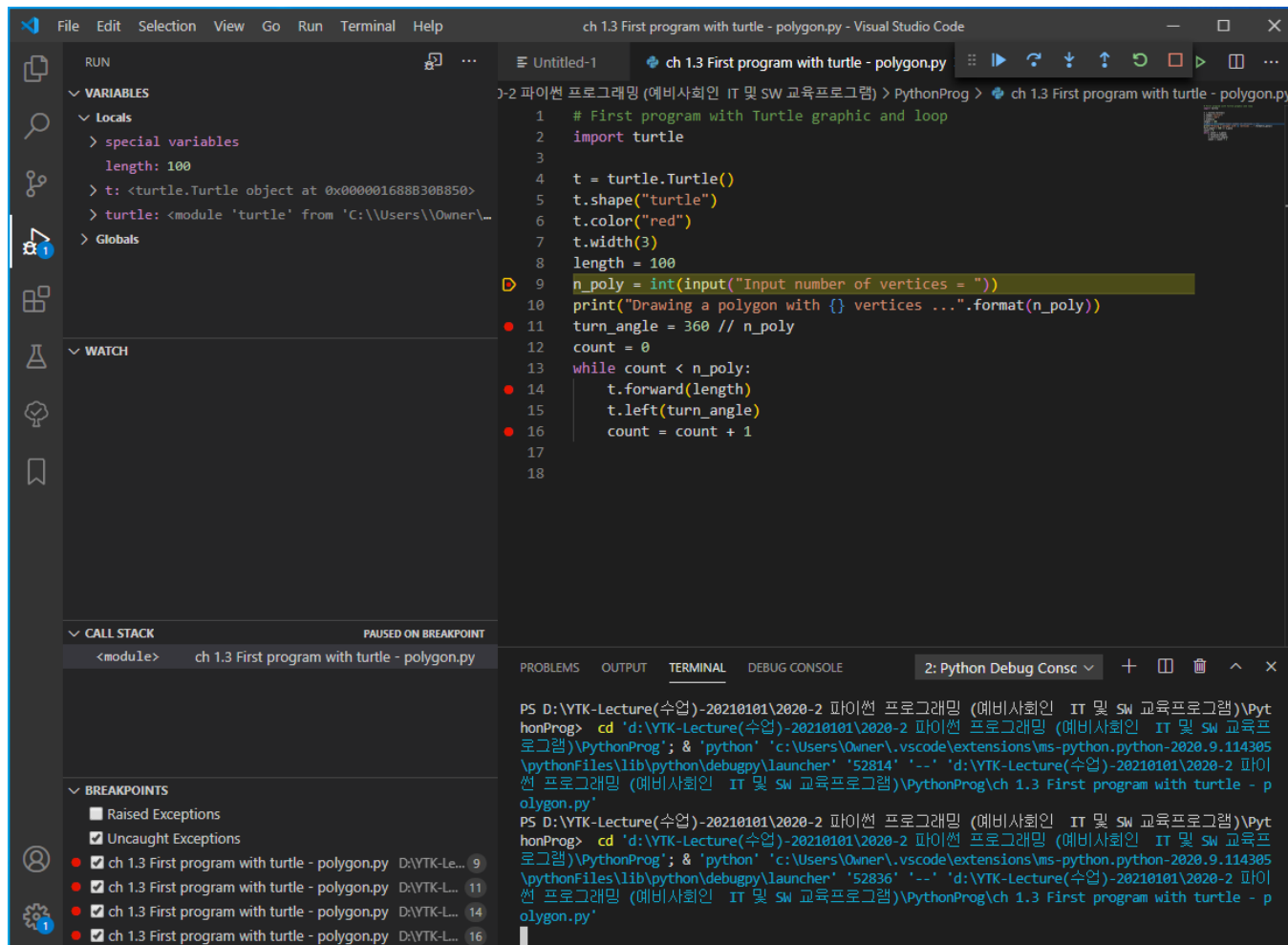
A red dashed box highlights the terminal output, and a red dashed line with the text 'ch 2-79' is drawn across the bottom of the terminal window.



Setting Breakpoint on VS-Code



(Re-)Start Debugging (F5, Cntrl+Shift+F5) with Tracing (F10- Step over, F11-Step into)



Tracing with F10 (step-over)

The screenshot shows the Visual Studio Code interface with a Python file named 'ch 1.3 First program with turtle - polygon.py'. The code is as follows:

```
1 # First program with Turtle graphic and loop
2 import turtle
3
4 t = turtle.Turtle()
5 t.shape("turtle")
6 t.color("red")
7 t.width(3)
8 length = 100
9 n_poly = int(input("Input number of vertices = "))
10 print("Drawing a polygon with {} vertices ...".format(n_poly))
11 turn_angle = 360 // n_poly
12 count = 0
13 while count < n_poly:
14     t.forward(length)
15     t.left(turn_angle)
16     count = count + 1
17
18
```

The left sidebar shows the following sections:

- VARIABLES**
 - Locals
 - special variables
 - length: 100
 - n_poly: 5
 - t: <turtle.Turtle object at 0x000001688B30B850>
 - turtle: <module 'turtle' from 'C:\\Users\\Owner\\...>
 - Globals
- WATCH**
- CALL STACK** (PAUSED ON STEP)
 - <module> ch 1.3 First program with turtle - polygon.py
- BREAKPOINTS**
 - Raised Exceptions
 - Uncaught Exceptions
 - ch 1.3 First program with turtle - polygon.py D:\\VTK-L... 9
 - ch 1.3 First program with turtle - polygon.py D:\\VTK-L... 11
 - ch 1.3 First program with turtle - polygon.py D:\\VTK-L... 14
 - ch 1.3 First program with turtle - polygon.py D:\\VTK-L... 16

The terminal shows the command to run the script with debugging:

```
honProg> cd 'd:\\VTK-Lecture(수업)-20210101\\2020-2 파이썬 프로그래밍 (예비사회인 IT 및 SW 교육프로그래밍)' & 'python' 'c:\\Users\\Owner\\.vscode\\extensions\\ms-python.python-2020.9.114305\\pythonFiles\\lib\\python\\debugpy\\launcher' '52814' '--' 'd:\\VTK-Lecture(수업)-20210101\\2020-2 파이썬 프로그래밍 (예비사회인 IT 및 SW 교육프로그래밍)\\PythonProg\\ch 1.3 First program with turtle - polygon.py'
```

The output of the script is shown in the terminal:

```
PS D:\\VTK-Lecture(수업)-20210101\\2020-2 파이썬 프로그래밍 (예비사회인 IT 및 SW 교육프로그래밍)\\PythonProg> cd 'd:\\VTK-Lecture(수업)-20210101\\2020-2 파이썬 프로그래밍 (예비사회인 IT 및 SW 교육프로그래밍)' & 'python' 'c:\\Users\\Owner\\.vscode\\extensions\\ms-python.python-2020.9.114305\\pythonFiles\\lib\\python\\debugpy\\launcher' '52836' '--' 'd:\\VTK-Lecture(수업)-20210101\\2020-2 파이썬 프로그래밍 (예비사회인 IT 및 SW 교육프로그래밍)\\PythonProg\\ch 1.3 First program with turtle - polygon.py'
```

The input value '5' is shown in the terminal, indicating the number of vertices for the polygon.



Tracing with F10 (step-over)

The screenshot shows the Visual Studio Code interface with a Python file named `ch 1.3 First program with turtle - polygon.py` open. The file contains the following code:

```
1 # First program with Turtle graphic and loop
2 import turtle
3
4 t = turtle.Turtle()
5 t.shape("turtle")
6 t.color("red")
7 t.width(3)
8 length = 100
9 n_poly = int(input("Input number of vertices = "))
10 print("Drawing a polygon with {} vertices ...".format(n_poly))
11 turn_angle = 360 // n_poly
12 count = 0
13 while count < n_poly:
14     t.forward(length)
15     t.left(turn_angle)
16     count = count + 1
17
18
```

The Run and Debug sidebar on the left shows the following variables:

- VARIABLES**
 - Locals**
 - special variables**
 - `length`: 100
 - `n_poly`: 5
 - `t`: <turtle.Turtle object at 0x000001688B308850>
 - `turn_angle`: 72
 - Globals**
- WATCH**
- CALL STACK** (PAUSED ON STEP)
 - <module> ch 1.3 First program with turtle - polygon.py
- BREAKPOINTS**
 - ☐ Raised Exceptions
 - ☒ Uncaught Exceptions
 - ☒ ch 1.3 First program with turtle - polygon.py D:\YTK-L... 9
 - ☒ ch 1.3 First program with turtle - polygon.py D:\YTK-L... 11
 - ☒ ch 1.3 First program with turtle - polygon.py D:\YTK-L... 14
 - ☒ ch 1.3 First program with turtle - polygon.py D:\YTK-L... 16

The terminal window at the bottom shows the output of the program:

```
PS D:\YTK-Lecture(수업)-20210101\2020-2 파이썬 프로그래밍 (예비사회인 IT 및 SW 교육프로그램)\PythonProg> cd 'd:\YTK-Lecture(수업)-20210101\2020-2 파이썬 프로그래밍 (예비사회인 IT 및 SW 교육프로그램)\PythonProg' & 'python' 'c:\Users\Owner\.vscode\extensions\ms-python.python-2020.9.114305\pythonFiles\lib\python\debugpy\launcher' '52814' '--' 'd:\YTK-Lecture(수업)-20210101\2020-2 파이썬 프로그래밍 (예비사회인 IT 및 SW 교육프로그램)\PythonProg\ch 1.3 First program with turtle - polygon.py'
Input number of vertices = 5
Drawing a polygon with 5 vertices ...
```



Homework 2

(주: 모든 프로그램 소스코드에 주석문 (comment)를 반드시 포함시킬 것 !!)

2.1 표준입력장치 (키보드)로부터 원의 반지름(radius)을 입력 받고, 그 원의 넓이(area)와 원둘레 (circumference)를 출력하는 파이썬 프로그램을 작성하고, 실행 결과를 제출하라.
(실행 예제)

```
radius = 10
Circle of radius (10) : area (314.1592), circumference (62.83184)
```

2.2 직사각형의 가로 (width)와 세로 (length)를 입력 받아 넓이(area)와 둘레(perimeter)를 계산하여 출력 하는 파이썬 프로그램을 작성하고, 실행 결과를 제출하라.
(실행 예제)

```
width, length = 100 50
Rectangle of width(100) and length(50) : area (5000), perimeter(300.0)
```



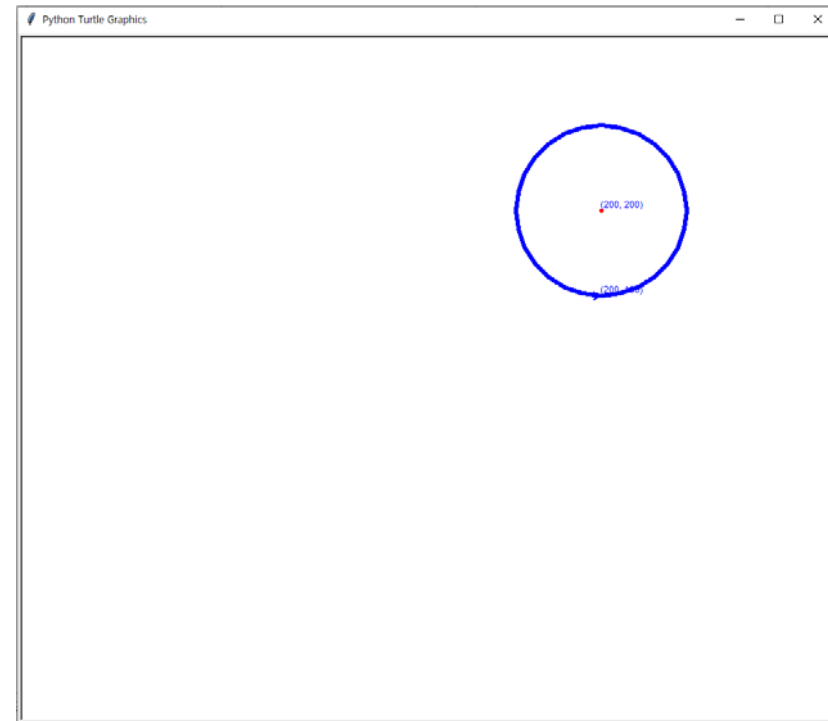
Homework 2

(주: 모든 프로그램 소스코드에 주석문 (comment)를 반드시 포함시킬 것 !!)

2.3 표준입력장치 (키보드)로부터 원의 중심 (center)의 x좌표 및 y좌표와 원의 반지름(radius)을 각각 입력 받고, 터틀 그래픽을 사용하여 지정된 중심 위치에 지정된 크기의 원을 그리는 파이썬 프로그램을 작성 하고, 실행 결과를 제출하라.

(실행 예제)

```
pos_x, pos_y : 200 200
radius = 100
Drawing a circle of radius (100) at position(200, 200)
at start_position(200, 100)
```



Homework 2

(주: 모든 프로그램 소스코드에 주석문 (comment)를 반드시 포함시킬 것 !!)

2.4 0 ~ 255의 값을 10진수, 2진수, 8진수, 16진수로 각각 출력하는 파이썬 프로그램을 작성하고, 실행결과를 제출하라. 2진수는 총 8자리, 8진수는 접두어를 포함하여 5자리, 16진수는 접두어를 포함하여 4자리로 출력하며, 앞부분에 빈자리가 있는 경우 0으로 채울 것.
(실행 예제)

```
=====
Decimal    Bit    Octal    Hexadecimal
-----
0 00000000 0o000 0X00
1 00000001 0o001 0X01
2 00000010 0o002 0X02
3 00000011 0o003 0X03
4 00000100 0o004 0X04
5 00000101 0o005 0X05
6 00000110 0o006 0X06
7 00000111 0o007 0X07
8 00001000 0o010 0X08
9 00001001 0o011 0X09
10 00001010 0o012 0X0A
11 00001011 0o013 0X0B
12 00001100 0o014 0X0C
13 00001101 0o015 0X0D
14 00001110 0o016 0X0E
15 00001111 0o017 0X0F
16 00010000 0o020 0X10
240 11110000 0o360 0XF0
241 11110001 0o361 0XF1
242 11110010 0o362 0XF2
243 11110011 0o363 0XF3
244 11110100 0o364 0XF4
245 11110101 0o365 0XF5
246 11110110 0o366 0XF6
247 11110111 0o367 0XF7
248 11111000 0o370 0XF8
249 11111001 0o371 0XF9
250 11111010 0o372 0XFA
251 11111011 0o373 0XFB
252 11111100 0o374 0XFC
253 11111101 0o375 0XFD
254 11111110 0o376 0XFE
255 11111111 0o377 0XFF
=====
```



Homework 2

(주: 모든 프로그램 소스코드에 주석문 (comment)를 반드시 포함시킬 것 !!)

2.5 2개의 정수 a와 b를 입력받아 정수연산 $a+b$, $a-b$, $a*b$, a/b , $a//b$, $a\%b$ 를 각각 계산하여 출력하는 파이썬 프로그램을 작성하고, 실행결과를 제출하라.

(실행 예제)

```
input a and b : 256 125
a(256) + b(125) = 381
a(256) - b(125) = 131
a(256) * b(125) = 32000
a(256) / b(125) = 2.048000
a(256) // b(125) = 2.000000
a(256) % b(125) = 6.000000
```

```
input a and b : 10 3
a( 10) + b( 3) = 13
a( 10) - b( 3) = 7
a( 10) * b( 3) = 30
a( 10) / b( 3) = 3.333333
a( 10) // b( 3) = 3.000000
a( 10) % b( 3) = 1.000000
```

