

알고리즘분석 실습 자료

1주차

Photo by [Piotr Guzik](#) on [Unsplash](#)



실습 소요 시간 100분

파이썬(Python)

- 소개
 - ✓ 파이썬 설치하기
 - ✓ 파이썬 시작하기
 - ✓ 프로그램 파일 생성, 구동, 저장
 - ✓ 파이썬 파일을 불러오기
- 변수에 값 저장
- 입력, 출력
- 조건부 수행
- 반복
- 리스트
- 함수
- 출력 형식



파이썬

- 1991년 처음 만들어 짐
- 인터프리터 언어의 일종
- high-level 언어
- 객체지향(object-oriented) 개념을 포함
- 단순한 형식과 쉬운 가독성(readability)을 목표
- 오픈소스(open-source) 소프트웨어로 비영리 단체에 의해 운영
 - 무료 사용



인터프리터(Interpreter) 언어

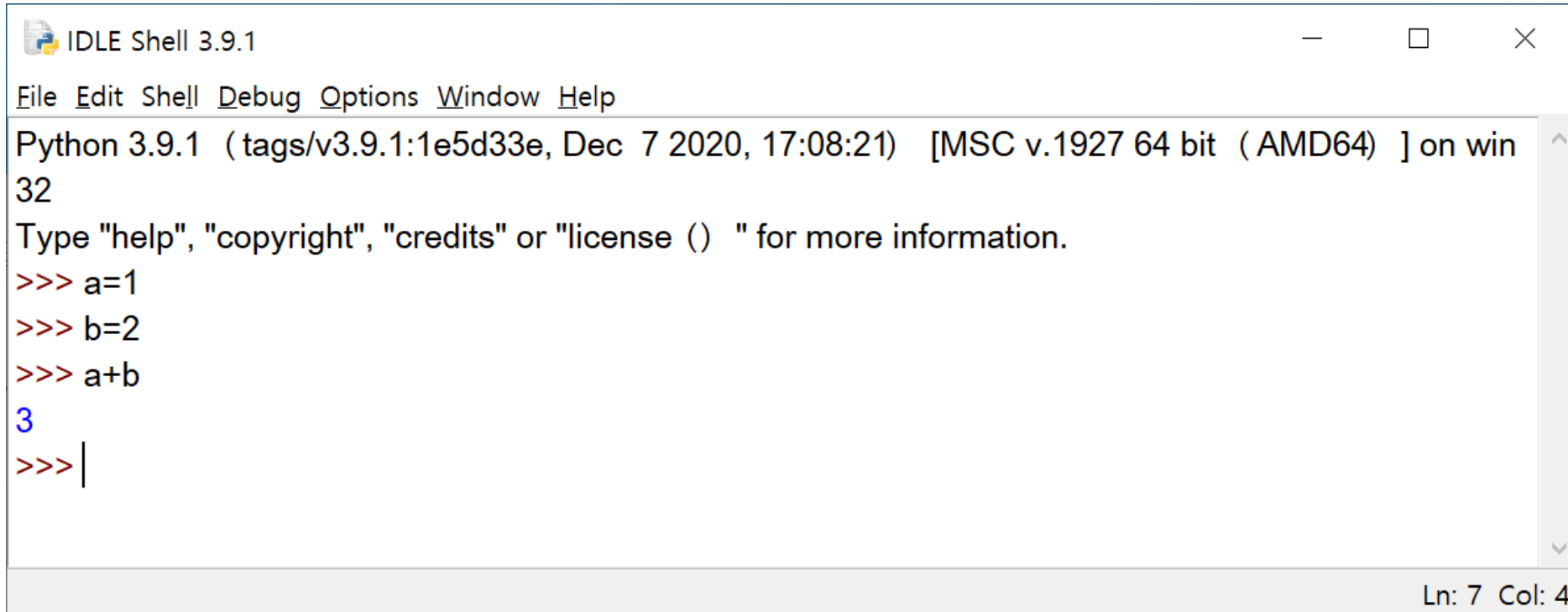
- 컴파일 과정을 거치지 않고 실행되는 컴퓨터 프로그램 언어
- Python, MATLAB, BASIC 등의 언어
- 언어가 단순하다.
- 인터프리터 : 인터프리터 언어로 작성된 프로그램을 수행하는 프로그램



| Rank | Language | Type | Score |
|------|------------|---|-------|
| 1 | Python |    | 100.0 |
| 2 | Java |    | 96.3 |
| 3 | C |    | 94.4 |
| 4 | C++ |    | 87.5 |
| 5 | R |  | 81.5 |
| 6 | JavaScript |  | 79.4 |
| 7 | C# |     | 74.5 |
| 8 | Matlab |  | 70.6 |
| 9 | Swift |   | 69.1 |
| 10 | Go |   | 68.0 |



- IDLE: Integrated Development and Learning Environment (Integrated Development Environment 의 일종)
- 파이썬을 이용하기 위해서는 IDLE를 설치해야 함



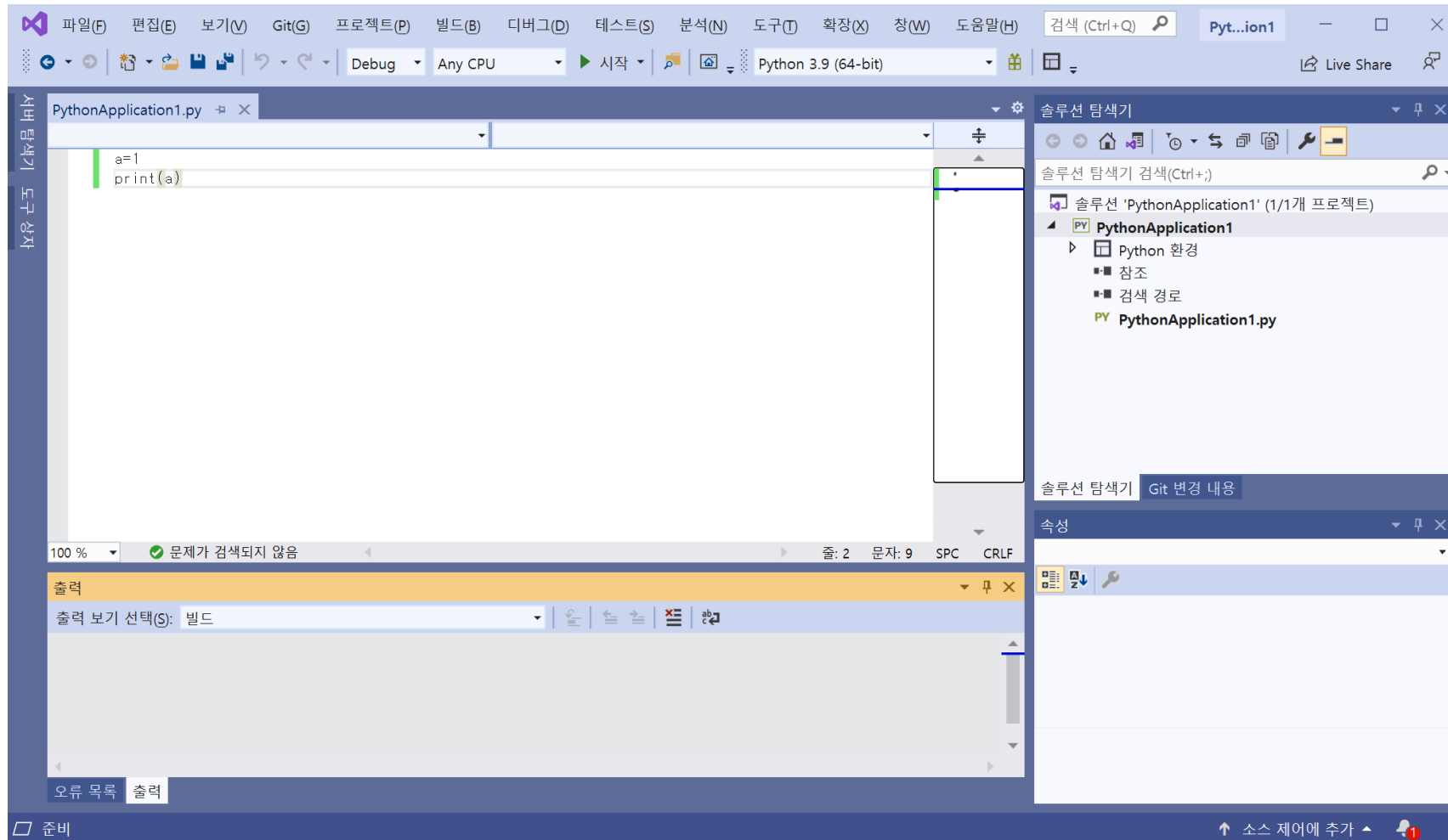
The screenshot shows the IDLE Shell 3.9.1 window. The title bar reads 'IDLE Shell 3.9.1'. The menu bar includes 'File', 'Edit', 'Shell', 'Debug', 'Options', 'Window', and 'Help'. The main text area displays the following content:

```
Python 3.9.1 (tags/v3.9.1:1e5d33e, Dec 7 2020, 17:08:21) [MSC v.1927 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license ()" for more information.
>>> a=1
>>> b=2
>>> a+b
3
>>> |
```

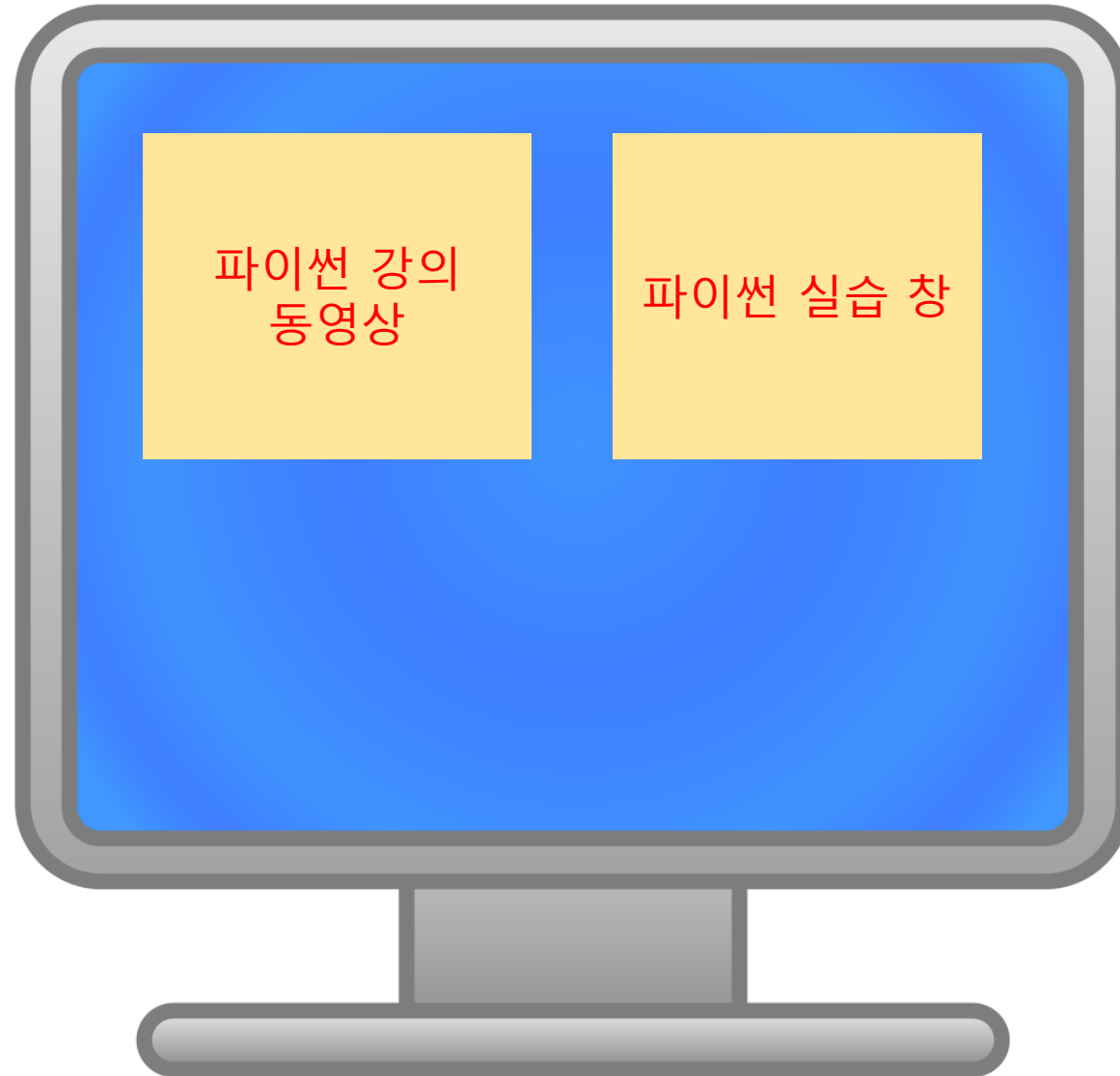
The status bar at the bottom right indicates 'Ln: 7 Col: 4'.



visual studio

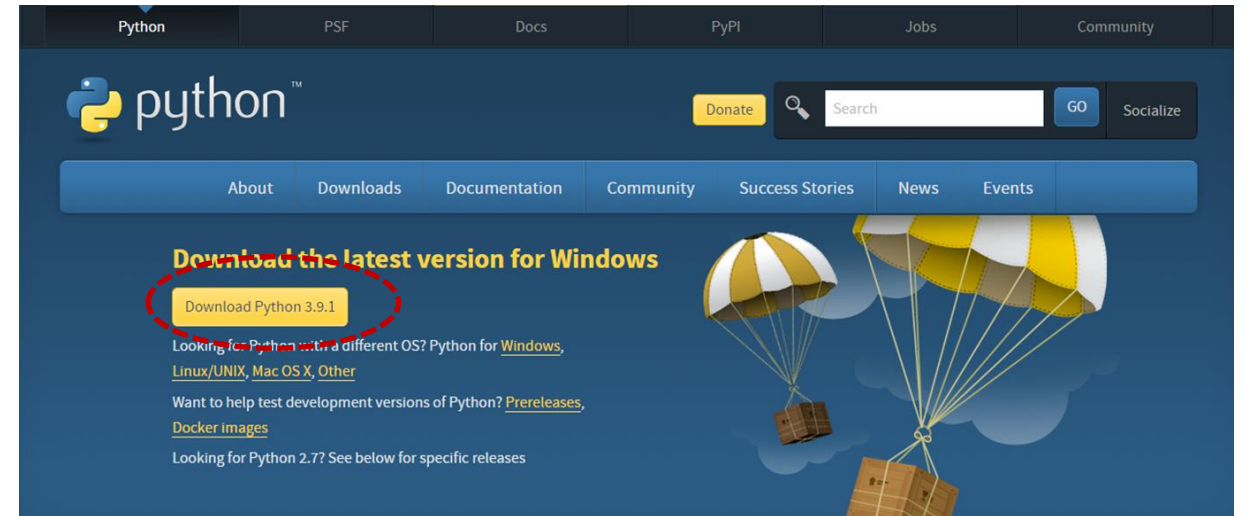
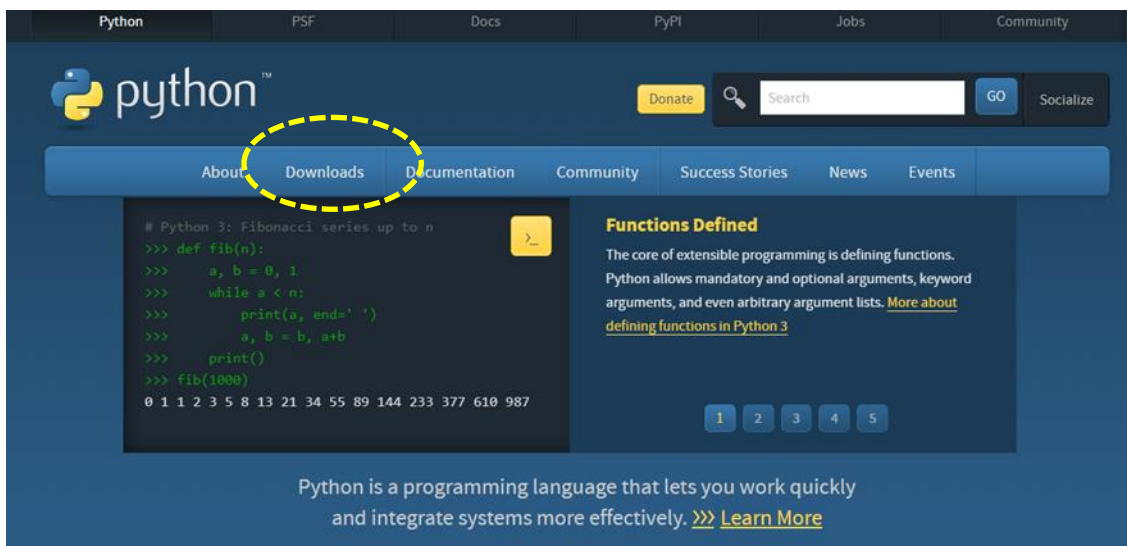


동영상 진행에 맞추어 프로그래밍 실습 진행

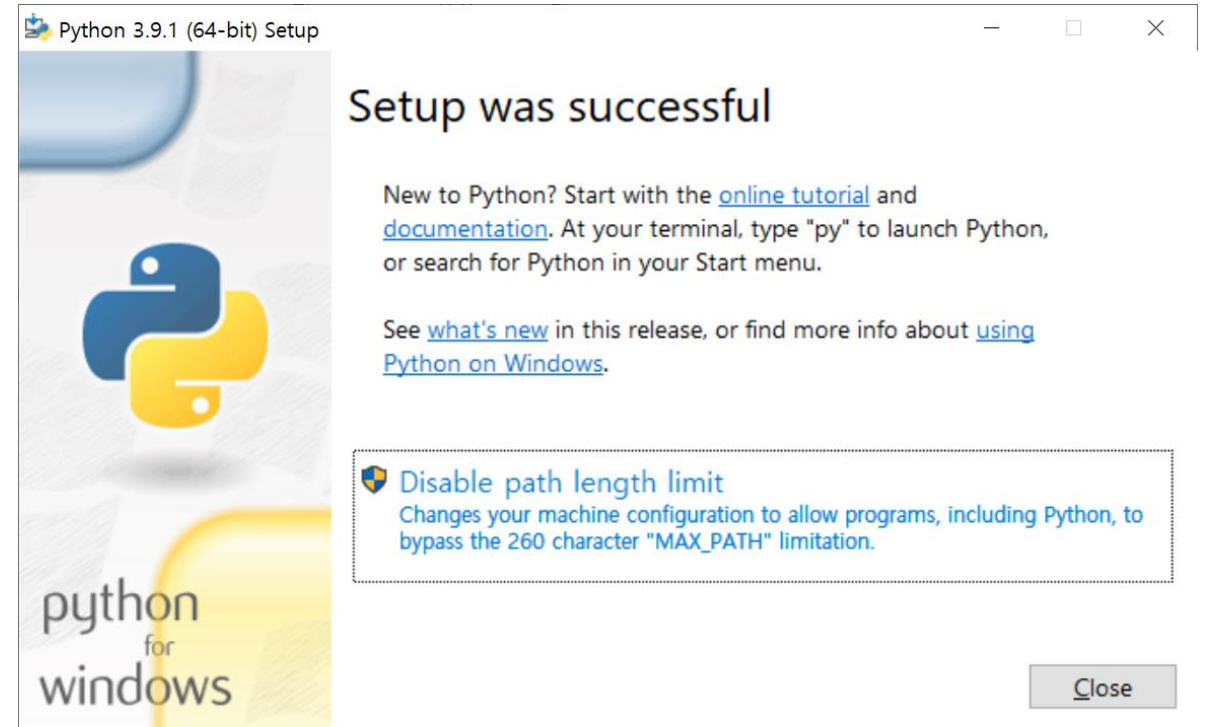
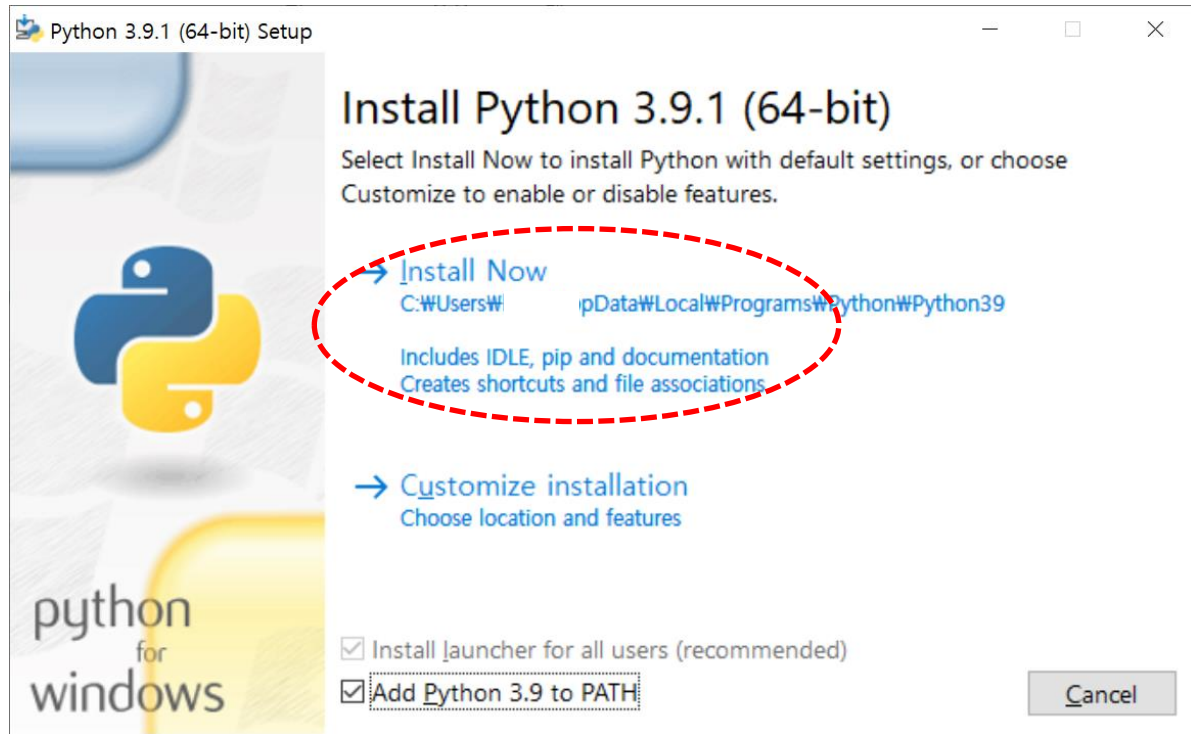


- 파이썬 다운로드

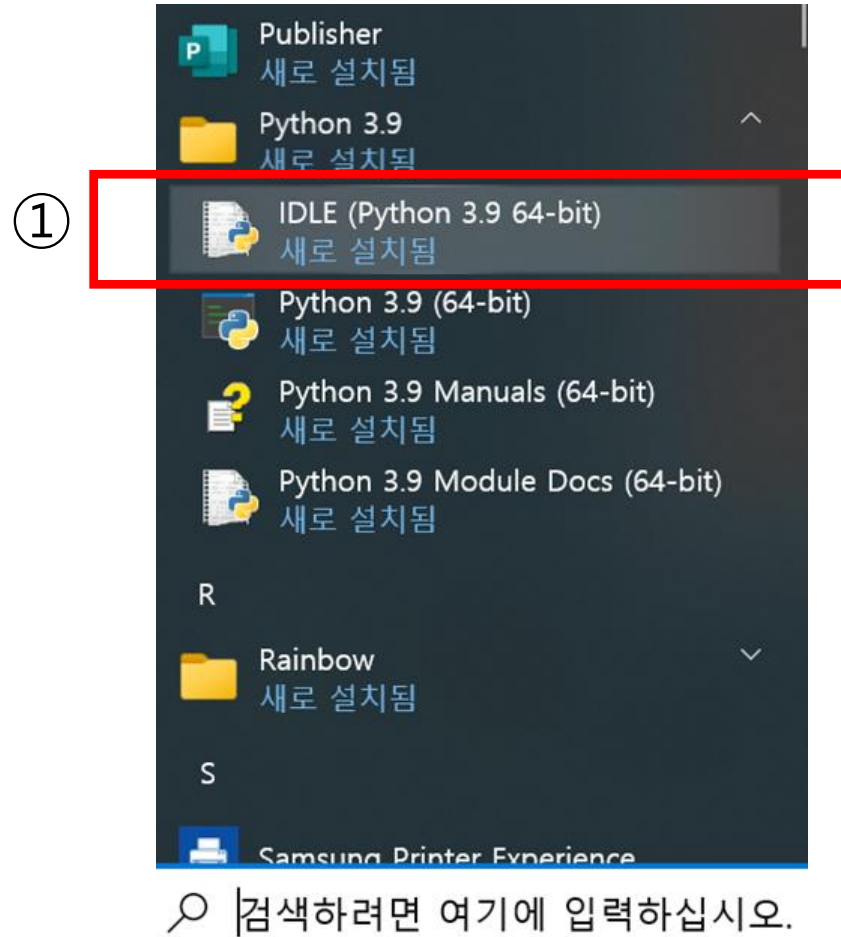
✓ www.python.org



- 파이썬 설치



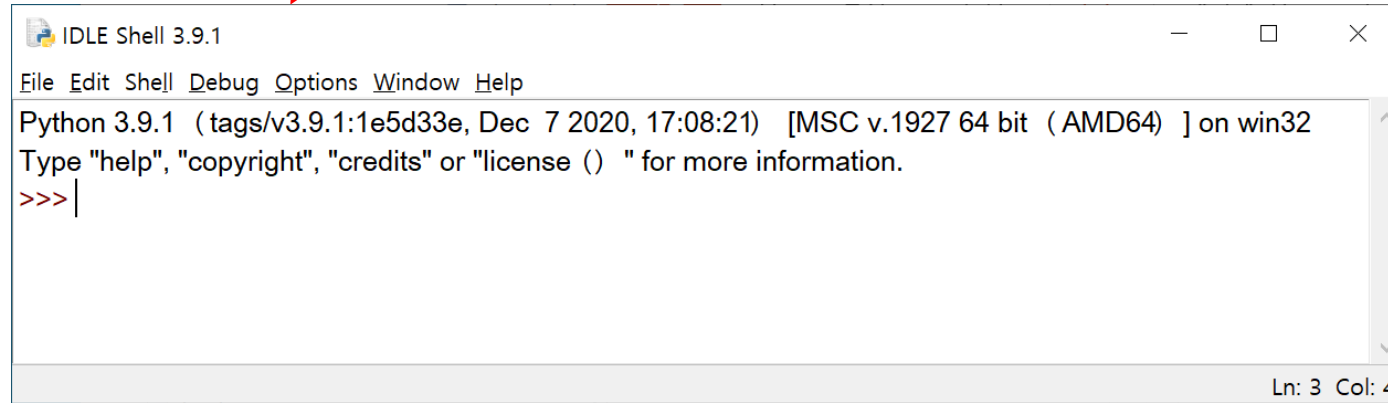
- 파이썬 시작



① IDLE 아이콘을 누르면 실행된다.



shell



IDLE shell window(창)

- shell: OS와 연결할 수 있는 user interface를 말한다.
- 명령어를 주면 값을 반환하게 된다.
- IDLE는 shell window와 editor window를 갖는다.



단순 프로그램 예

```
IDLE Shell 3.9.1
File Edit Shell Debug Options Window Help
Python 3.9.1 (tags/v3.9.1:1e5d33e, Dec 7 2020, 17:08:21) [MSC v.1927 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license ()" for more information.
>>> a=1
>>> b=2
>>> a+b
3
>>> |
```

입력

출력

프롬프트

(prompt)

a=1을 친 후 enter
b=2를 친 후 enter
a+b를 친 후 enter



shell window

```
>>> a=3
>>> a
3
>>> a=a*a
>>> a
9
>>> b=2
>>> b=b*b
>>> b=b*b
>>> b
16
>>>
```

```
>>> a=2
>>> b=3
>>> a-b*a+b/a/3
-3.5
>>> c=5
>>> d=c*2
>>> d
10
>>>
```



shell window

```
>>> 3**5
243
>>> 2**4
16
>>> 2/4
0.5
>>> int(2/4)
0
>>> '5'
'5'
>>> 5
5
>>>
```

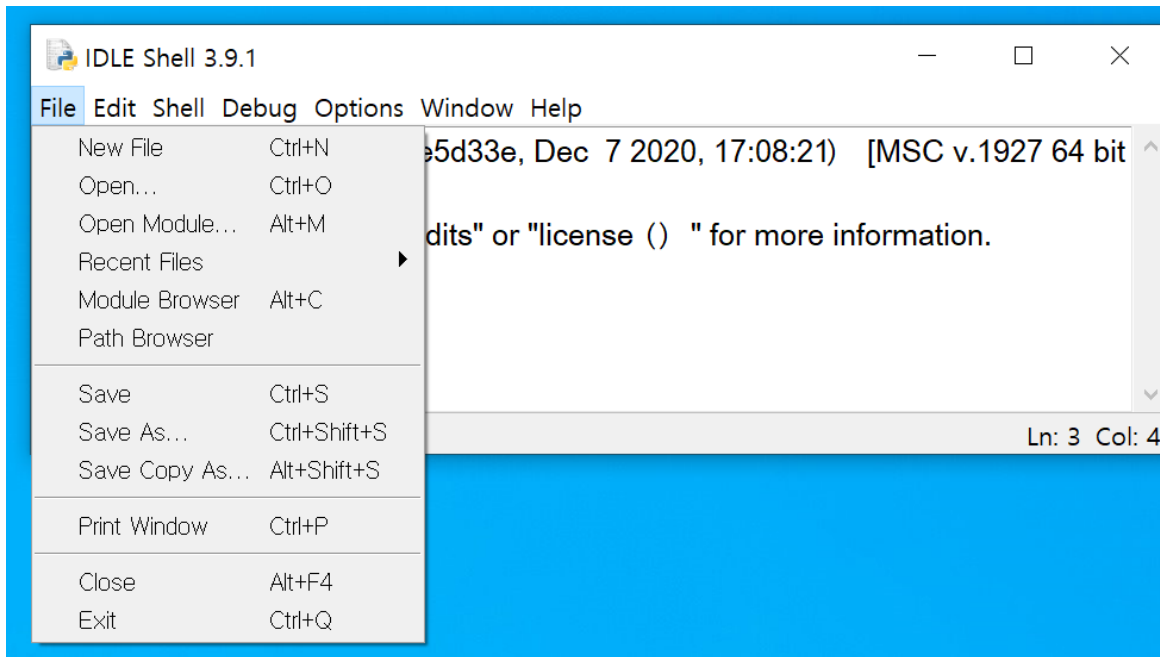
문자 5 ----->

숫자 5 ----->

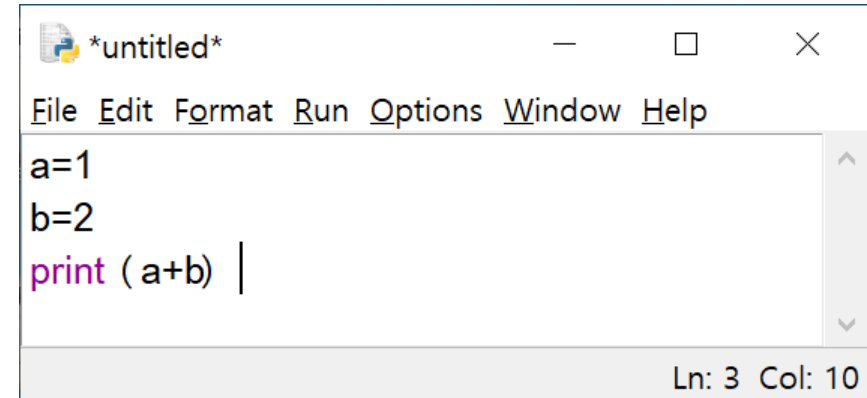
```
>>> 'abc'
'abc'
>>> "abc"
'abc'
>>> 'abc'+'def'
'abcdef'
>>> 'abc'+'d'
'abcd'
>>> abc
Traceback (most recent call last):
  File "<pyshell#53>", line 1, in <module>
    abc
NameError: name 'abc' is not defined
>>>
```



긴 프로그램을 매번 입력할 수 없으므로 파일을 생성해서 저장한다.



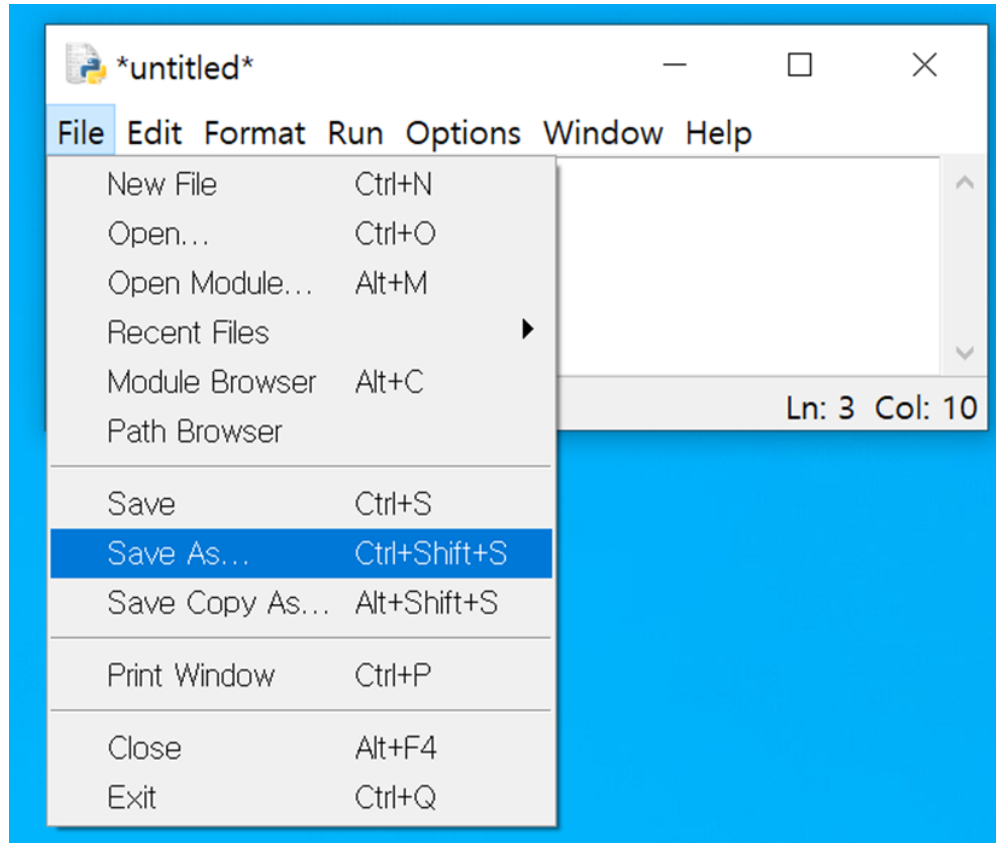
New File



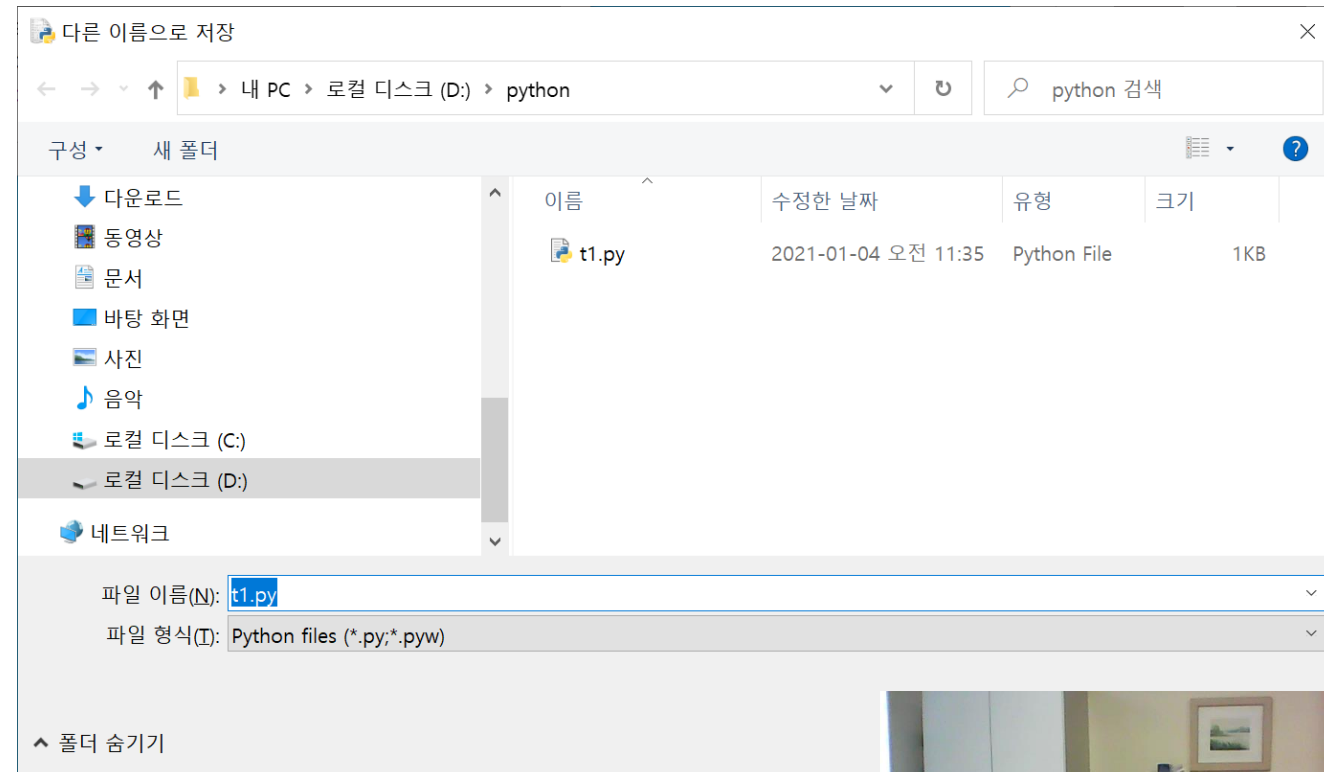
편집기 창(editor window)



t1.py로 파일 저장



Save As...

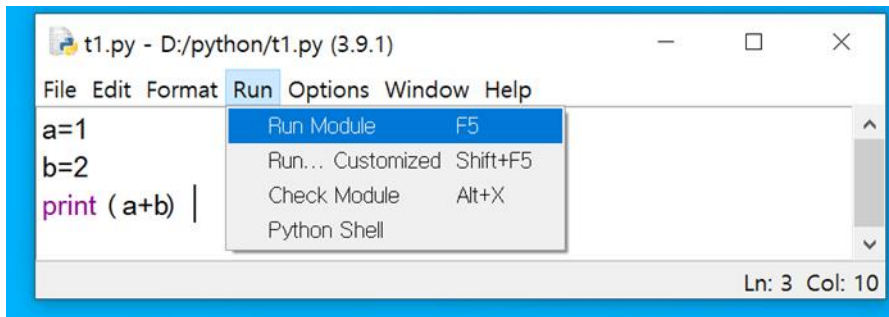


t1

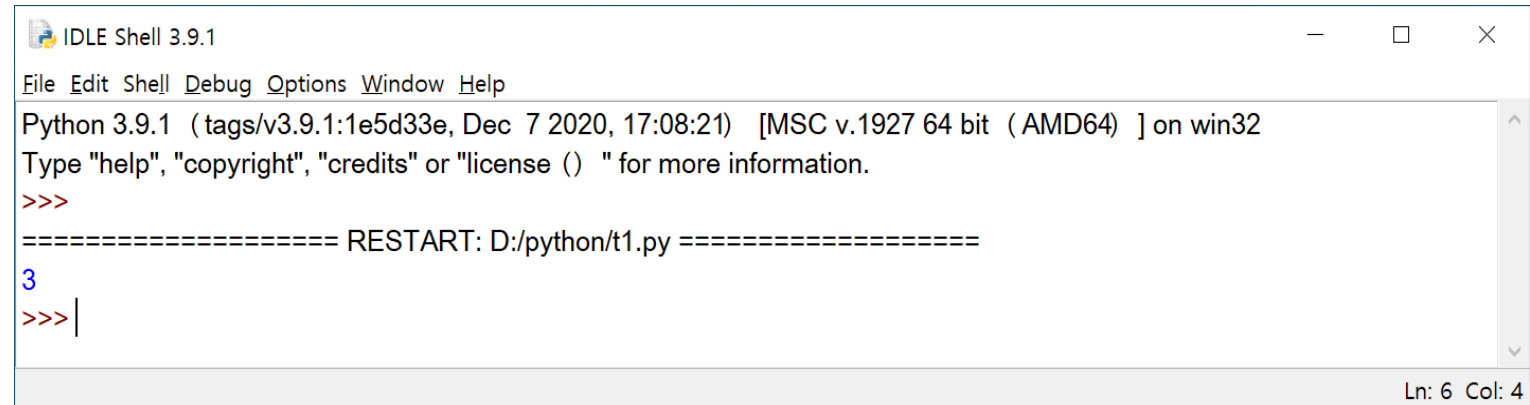


프로그램의 수행

- Run – Run Module
- F5



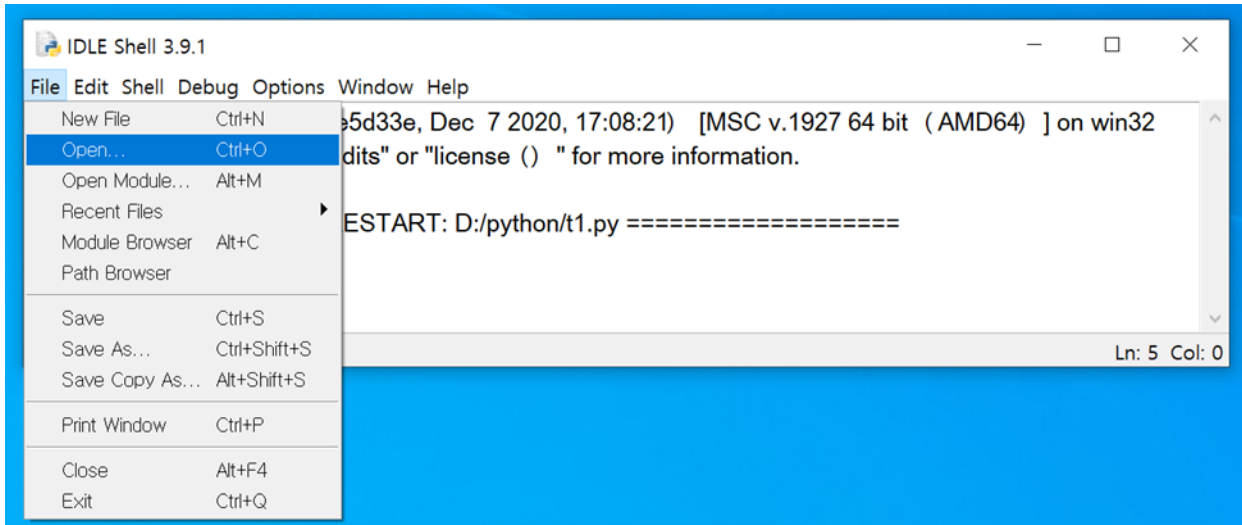
editor 창



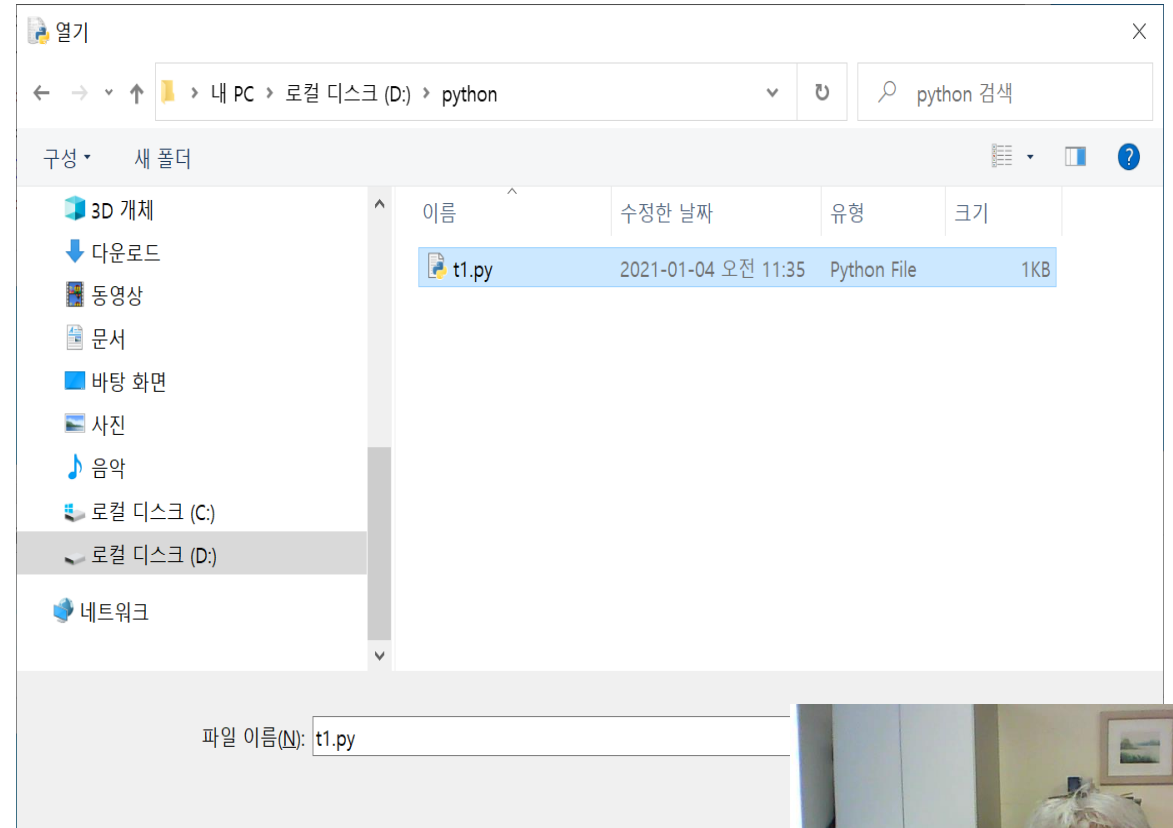
shell 창



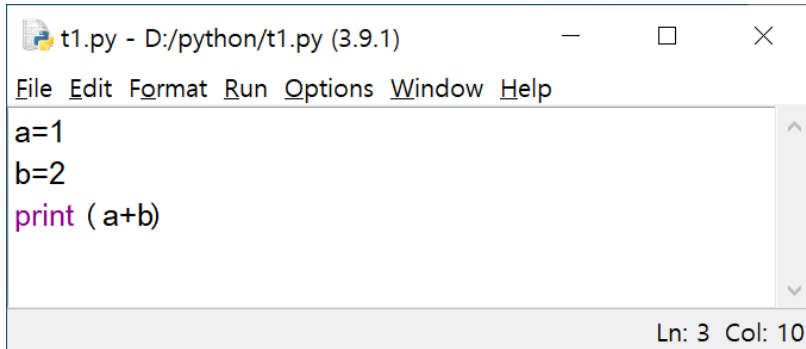
생성된 파일 t1.py를 연다



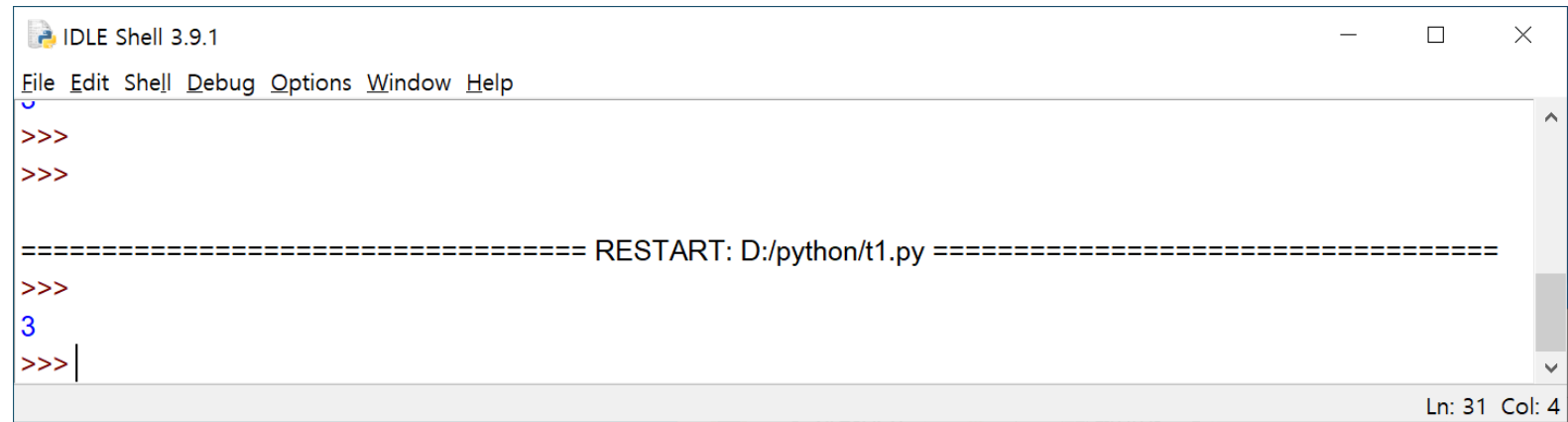
Open...



t1.py를 불러 온 후, F5를 눌러 프로그램을 수행한다.



```
t1.py - D:/python/t1.py (3.9.1)
File Edit Format Run Options Window Help
a=1
b=2
print (a+b)
Ln: 3 Col: 10
```



```
IDLE Shell 3.9.1
File Edit Shell Debug Options Window Help
>>>
>>>

===== RESTART: D:/python/t1.py =====
>>>
3
>>> |
Ln: 31 Col: 4
```

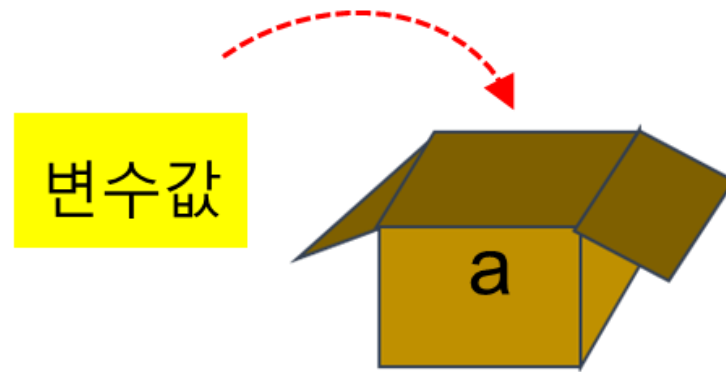


Python 공부 내용

1. 변수에 값 저장
2. 입력, 출력
3. 조건부 수행
4. 반복
5. 리스트
6. 함수



1. 변수에 값 저장



`a=53`

`word="school"`

`cost=100`

`total=sum+diff`

변수명 = 저장할 값 또는 변수명



프로그램 내부에서 변수에 값 저장

```
a=1
b=2.3
c="school"
d='day'
e=[1, "bus", 5.9]
f=a
print(a)
print(b)
print(c)
print(d)
print(e)
print(f)
```

```
1
2.3
school
day
[1, 'bus', 5.9]
1
>>>
```

✓ 변수가 갖고 있는 값은 연산에 사용 가능

```
a=1
b=2.3
print(a+1)
print(a+b)
```

```
2
3.3
>>>
```



- identifier 구성 방법

- ✓ 변수(variable)이름 이나 함수(function)이름을 identifier라고 한다.
- ✓ 알파벳 a-z, A-Z, _ (underbar, underscore)로 시작한다.
- ✓ 이후에는 복수개의 알파벳 a-z, A-Z, _, 숫자(0-9)가 올 수 있다.
- ✓ 대소문자를 구분한다. abc ≠ ABC
- ✓ 특별문자 @,\$,% 등은 사용할 수 없다.



예약어(reserved words)

- 식별자로 사용할 수 없다.

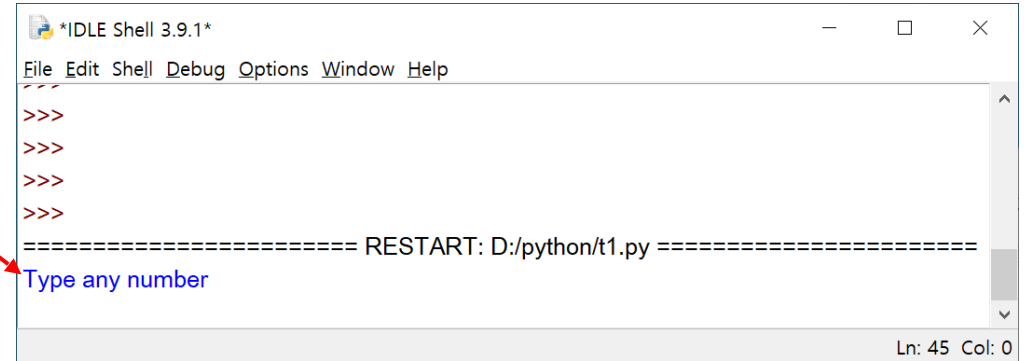
| | | | | |
|----------|---------|--------|--------|-------|
| and | del | from | not | while |
| as | elif | global | or | with |
| assert | else | if | pass | yield |
| break | except | import | print | |
| class | exec | in | raise | |
| continue | finally | is | return | |
| def | for | lambda | try | |



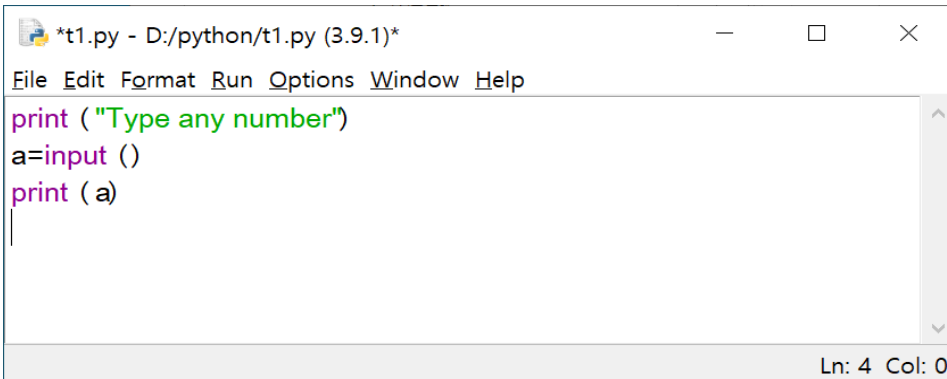
2. 입력, 출력

- ✓ input, output 사용
- ✓ input으로 받은 값은 문자로 바로 연산에 사용할 수 없다.

출력

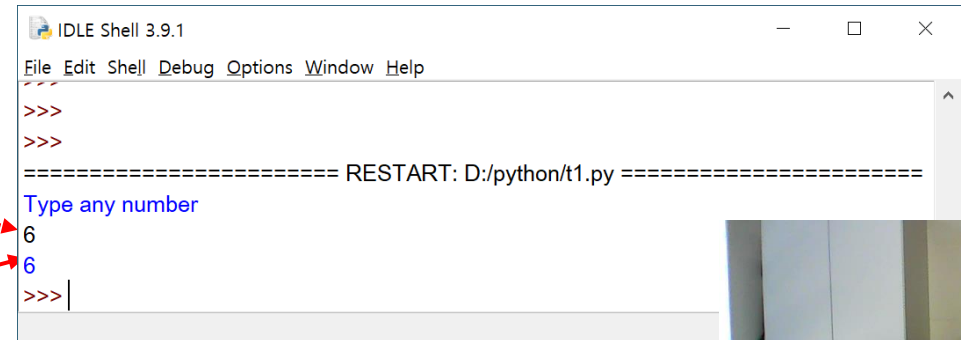


```
*IDLE Shell 3.9.1*
File Edit Shell Debug Options Window Help
>>>
>>>
>>>
>>>
===== RESTART: D:/python/t1.py =====
Type any number
Ln: 45 Col: 0
```



```
*t1.py - D:/python/t1.py (3.9.1)*
File Edit Format Run Options Window Help
print ("Type any number")
a=input ()
print (a)
Ln: 4 Col: 0
```

입력
출력



```
IDLE Shell 3.9.1
File Edit Shell Debug Options Window Help
>>>
>>>
===== RESTART: D:/python/t1.py =====
Type any number
6
6
>>> |
```



IDLE shell 창

```
>>> print("Test")
Test
>>> a=input()
xyz
>>> a
'xyz'
>>> a=input("Type any number ")
Type any number 5
>>> a
'5'
```

- Test를 출력한다.
- 사용자가 문자를 입력하기를 기다렸다가, 입력이 되면 변수 a에 저장한다.
- a값을 출력한다.
- 화면에 Type any number를 출력하고, 사용자가 문자를 입력하기를 기다렸다가, 입력이 되면 변수 a에 저장한다.

사용자가 입력한 문자

5라는 문자



즉, 숫자 입력을 하더라도 내부에서는 문자로 인식된다.

편집기 창

```
a=input("Type any number ")  
print(a)
```

출력

shell 창

입력

5라는 문자

```
Type any number 5  
5  
>>>
```

```
a=input("Type any number ")  
print(a+1)
```

```
Type any number 5  
Traceback (most recent call last):  
  File "D:\python\t1.py", line 2, in <module>  
    print(a+1)  
TypeError: Can't convert 'int' object to str implicitly  
>>>
```

```
a=input("Type any number ")  
print(int(a)+1)
```

5라는 문자를 숫자 5로 변환

```
Type any number 5  
6  
>>>
```

- 소수점있는 숫자로 변환할 때는 float() 사용



문자열의 연결(concatenation)

```
a=input("Type string1: ")
b=input("Type string2: ")
print(a+b)
print(a+"world")
```



```
Type string1: hello
Type string2: car
hellocar
helloworld }
>>>
```

입력

출력

+ 연산은 문자열에 대해서는 연결하는 작업을 수행한다.



- 소수점있는 숫자로 변환할 때는 float 사용

```
a=input("Type any number ")  
print(float(a)+1)
```

```
Type any number 3.4  
4.4  
>>>
```

- 소수점있는 두 숫자를 입력 받아 덧셈을 출력하기

```
a=input("실수 입력 = ")  
b=input("실수 입력 = ")  
a= float(a)  
b= float(b)  
print("합 = ", a+b)
```

```
실수 입력 = 1.2  
실수 입력 = 3.4  
합 = 4.6  
>>>
```

```
a, b = input("실수 입력 = "), input("실수 입력 = ")  
a, b = float(a), float(b)  
print("합 = ", a+b)
```



3. 조건부 수행

✓ if

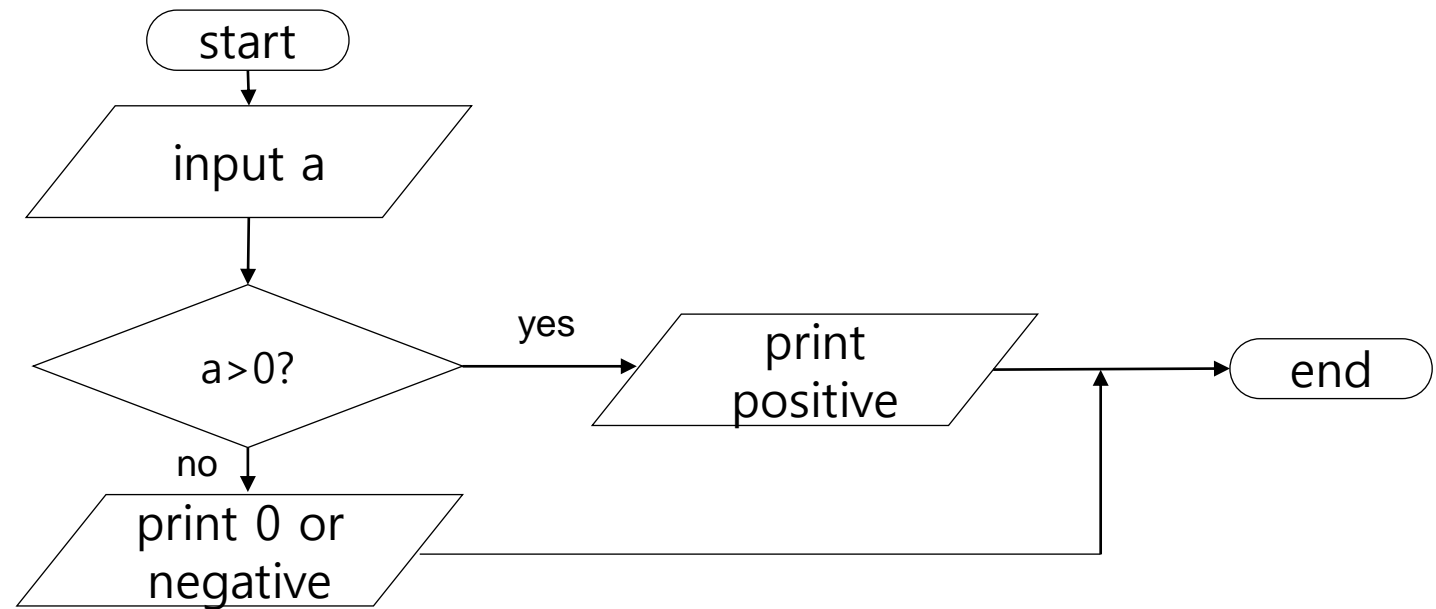
✓ else

✓ elif

```
a = 5
if(a>0):
    print("positive")
else:
    print("0 or negative")
```

if (a>0):

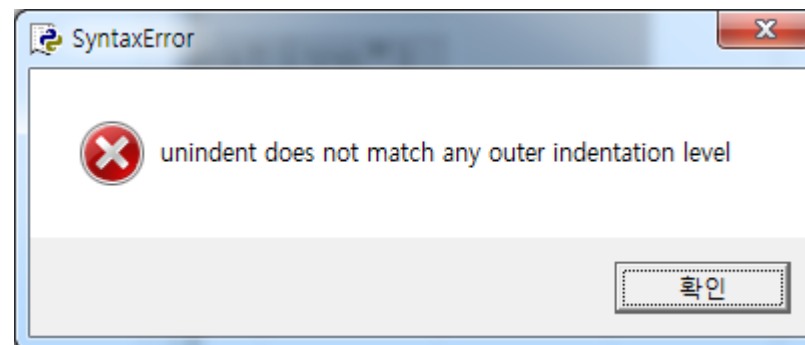
else:



들여쓰기가 달라질 때

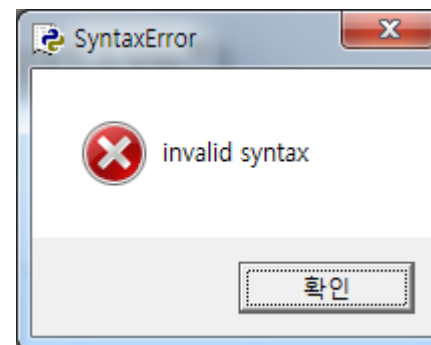
```
a = 5
if(a>0) :
    print("positive")
else:
    print("0 or negative")
```

- if 와 else가 서로 짝을 이루지 않는다.



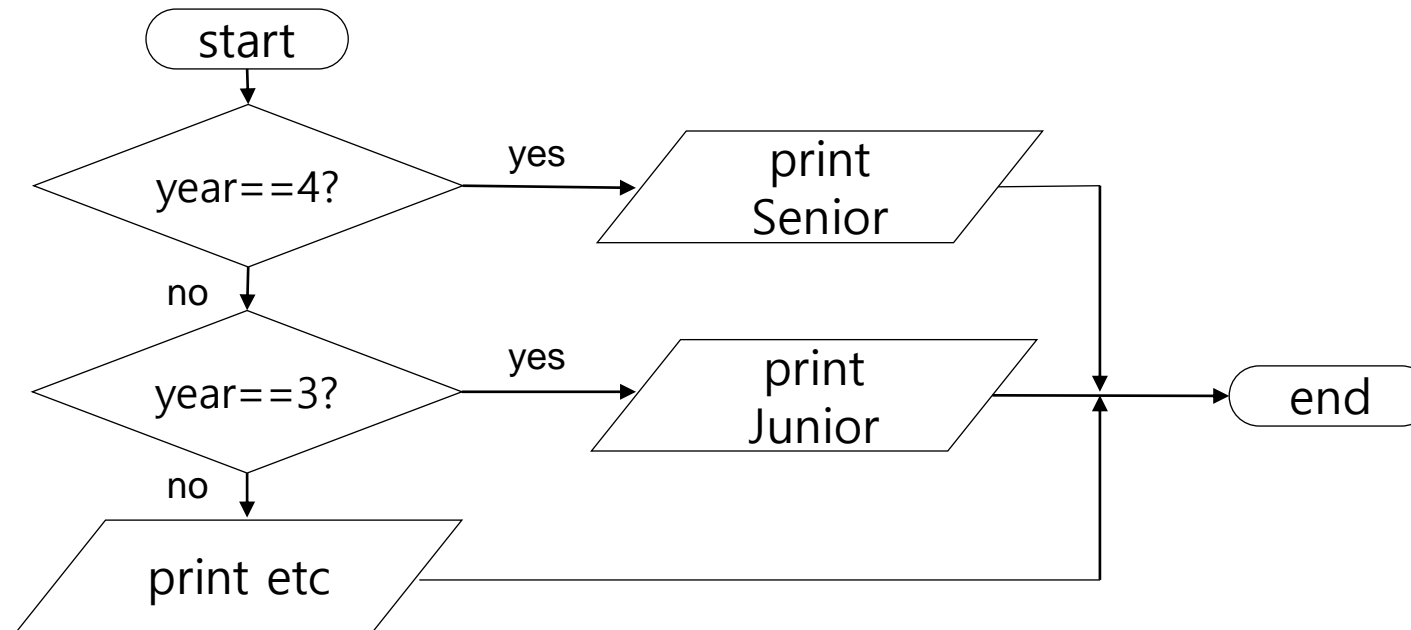
```
a = 5
if(a>0) :
    print("positive")
else:
    print("0 or negative")
```

- if 내에 else가 if 없이 사용되었다.




```
year = 3
if year == 4:
    print("Senior")
elif year == 3:
    print("Junior")
else:
    print("etc")
```

```
Junior
>>>
```



✓ 다양한 조건식을 만들 수 있다.

```
year = 3
course = "ce"
if year == 4 and course=="ce":
    print("computer Senior")
elif year == 3 or course=="ce":
    print("Junior or computer")
else:
    print("etc")
```

not A

| A | not A |
|---|-------|
| F | T |
| T | F |

T: True
F: False

Junior or computer
>>>

A or B
A and B

| A | B | A and B | A or B |
|---|---|---------|--------|
| F | F | F | F |
| F | T | F | T |
| T | F | F | T |
| T | T | T | T |



not 사용 예

```
a=8
if not (a>5):
    print("<=5")
else:
    print(">5")
```

```
>5
>>>
```



중첩된 조건문

```
kind = "bus"
size = 2000

if kind == "bus":
    if size > 3000:
        tax = 100
    else:
        tax = 50
else:
    if size > 3000:
        tax = 200
    else:
        tax = 150

print("Tax = ", tax)
```

```
Tax = 50
>>>
```

연산자 우선순위

우선순위(and) > 우선순위(or)

```
if a>0 or b<10 and c==5 or d>7:
```

A.

```
if (a>0 or b<10) and (c==5 or d>7):
```

B.

```
if a>0 or (b<10 and c==5) or d>7 :
```



4. 반복

✓ for

✓ while

```
sum=0
for d in range(10):
    sum = sum + d
print("합=", sum)
```

$d=0, 1, 2, 3, \dots, 9$

sum = sum + d의 들여쓰기는 필수

```
sum=0
d=0
while d < 10:
    sum = sum + d
    d = d+1
print("합=", sum)
```

$d=0, 1, 2, 3, \dots, 9$


sum = sum + d, d=d+1의 들여쓰기는 필수

```
합= 45
>>>
```

- `for d in range(k)`는 d가 0부터 k-1까지 변한다

✓ 들여쓰기(indentation)의 효과


```
sum=0
for d in range(10):
    sum = sum + d
    print("합=", sum)
```



들여쓰기에 의해 매번 print

```
합= 0
합= 1
합= 3
합= 6
합= 10
합= 15
합= 21
합= 28
합= 36
합= 45
>>>
```

```
sum=0
d=0
while d < 10:
    sum = sum + d
    d = d+1
    print("합=", sum)
```



들여쓰기에 의해 매번 print

for

초기값

```
for d in range(10):
```

=

```
for d in range(0, 10):
```

d=0,1,2,3,...,9

```
for d in range(2, 10):
```

d=2,3,...,9

✓ 1부터 10까지 더하기

```
sum=0
for d in range(1,11):
    sum = sum + d
print("합=", sum)
```

range(1,11)은 d=1,2,3,...,10까지 변화시킨다

```
합= 55
>>>
```

✓ 1부터 10까지의 홀수만 더하기

```
sum=0
for d in range(1,11,2):
    sum = sum + d
    print(d)
print("합=", sum)
```

```
1
3
5
7
9
합= 25
>>>
```

- range(1,11,2)은 d=1,3,...,9까지 변화시킨다
- 증가분이 2라는 뜻

```
for d in range(a,b,c):
```

- d는 a의 값에서 시작
- $c > 0$ 일 때 d는 b-1의 값까지 변화한다. $c < 0$ 일 때 d는 b+1의 값까지 변화한다.
- 증분은 c이다.
- a, b, c는 정수이어야 한다.

✓ 증분은 음수가 될 수 있다.

```
sum=0
for d in range(10,1,-2):
    sum = sum + d
    print(d)
print("합=", sum)
```

- 초기값 10, $d > 1$ 인 경우에만 반복 수행

```
10
8
6
4
2
합= 30
>>>
```

while

✓ 1부터 10까지 더하기

```
sum=0
d=1
while d < 11:
    sum = sum + d
    d = d+1
print("합=", sum)
```

- while d<=10: 과 동일 효과

```
합= 55
>>>
```

✓ 1부터 10까지의 홀수만 더하기

```
sum=0
d=1
while d <= 10:
    print(d)
    sum = sum + d
    d = d+2
print("합=", sum)
```

- d = d+2는 증가분이 2라는 뜻

```
1
3
5
7
9
합= 25
>>>
```

✓ 다양한 실수를 이용한 반복 제어 가능

- 1.5부터 3.5보다 작은 동안의 0.1씩 증가하는 실수의 합

```
sum=0
d=1.5
while d < 3.5:
    print(d)
    sum = sum + d
    d = d+0.1
print("합=", sum)
```

실수를 2진수로 저장함에 따라 발생하는 오차

```
1.5
1.6
1.700000000000000002
1.800000000000000003
1.900000000000000004
2.000000000000000004
...
...
...
3.000000000000000013
3.100000000000000014
3.200000000000000015
3.300000000000000016
3.400000000000000017
합= 49.0000000000000014
>>>
```

✓ d의 값을 단순히 증가, 감소가 아닌 것이어도 된다. – 다른 연산에 의한 변경 가능

- 100을 2로 계속 나눌 때의 값 출력
- $d > 1$ 일 때만 반복

```
d=100
while d>1:
    print(d/2)
    d=d/2
```

```
50.0
25.0
12.5
6.25
3.125
1.5625
0.78125
>>>
```

✓ 1,000보다 작은 2의 지수승 값중 가장 큰 수

```
n=2
while n <1000:
    print(n)
    n = n*2
```

- 종료 조건을 알지만, 정확한 수를 알기 어려울 때 사용

```
2
4
8
16
32
64
128
256
512
>>>
```

$\sum_{k=1}^n k \leq 1,000$ 인 최대 n ?

```
k=1
sum =0
while sum+k <=1000:
    sum = sum +k
    k = k+1
print("n의 값 = ", k-1)
```

```
n의 값 = 44
>>>
```

```
sum=0
d=1
while d<=100:
    if d%5==0:
        sum += d
        print(d)
    d +=1
print(sum)
```

sum+=d 와 sum = sum+d 은 동일

```
5
10
15
20
25
30
..
..
85
90
95
100
합 = 1050
>>>
```