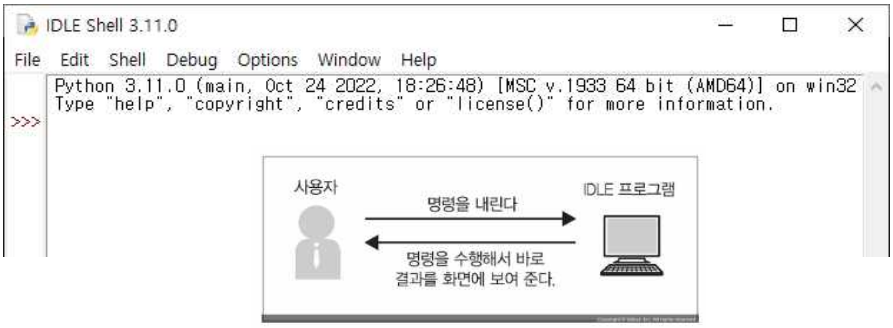
목 차

I. 파이썬 시작하기

1. 파이썬 쉘

파이썬 IDLE(Integrated Development and Learning Environment)은 파이썬 프로그램 작성을 도와주는 통합 개발 환경이다. [시작 → 모든 프로그램 → Python 3.11 → IDLE]을 선택해 파이썬 IDLE을 실행해 보자.



파이썬 **쉘**에서는 >>> 뒤에 명령어를 입력하고 엔터키를 누르면 명령어가 실행되고 실행 결과가 화면에 출력된다.

1) 파이썬과 대화하기

- 파이썬 셀에서 아래의 명령을 입력하여 실행 결과를 적어봅시다.

```
>>> 3+2

>>> 13-2+3

>>> 5/2

>>> (1+300)/0.54321
```

```
>>> 'Hi'

>>> '안녕하세요'

>>> print('Hi')

>>> print('안녕하세요')
```

* 이 외 다양한 숫자나 글자, 연산자를 입력해 출력값을 확인해 봅시다.

- 파이썬 셀에서 아래의 명령을 입력하여 실행 결과의 마지막줄 메시지를 적어봅시다.

```
>>> PRINT('HI')

>>> pront('안녕')

>>> print( 1 'hi' )

>>> print( 1 + 'Hi')

>>> 5 / 0

>>> int( 's' )
```

** 오류 메시지를 자세히 보면 오류 위치나 문장, 발생 원인 등 오류를 수정하는데 유용한 정보를 알수 있다.*

2) print() 함수

```
print( '문장' )
print( 숫자 또는 변수 )
print( 출력1, 출력2, 출력3 )
```

- 괄호 안의 값을 모니터에 출력할 때 사용.
- **문자열 출력시에는 (쌍)따옴표** 사용.
- 숫자나 변수 값 출력시에는 (쌍)따옴표를 사용하지 않음.
- 출력 대상이 여러 개일 때는 **쉼표**로 연결
  (출력 대상 사이 공백이 생김).

```
>>> print( "Hello" )

>>> print( "3+5=", 3+5 )
```

◆ 도전하기

print() 함수를 이용하여 아래와 같이 출력되도록 프로그래밍 해 봅시다.

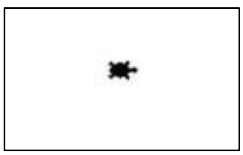
```
>>> print(                          )
    안녕하세요.

>>> print(                                                  )
    programming에 입문하신 것을 축하드립니다.

>>> print(                                      )
    3+5=? 무엇일까요?

>>> print(                                )        #단 숫자 28을 사용하지 않습니다.
    7*4 = 28 입니다.
```

3) 터틀 그래픽

파이썬은 초보자들이 쉽고 재미있게 프로그래밍을 배울 수 있도록 '거북이 그래픽'이라는 모듈을 제공합니다.
모듈이란 파이썬에서 사용하는 프로그램의 단위를 말합니다.

- IDLE 프로그램을 열고 다음과 같이 입력한 후 Enter를 누르세요.

```
>>> import turtle as t
>>> t.shape("turtle")
```



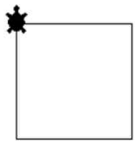
- 아래의 명령을 하나씩 입력하며 거북이의 움직임 살펴봅시다.

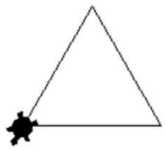
```
>>> t.forward(100)
>>> t.right(90)
>>> t.forward(100)
>>> t.left(90)
>>> t.forward(50)
```

t.forward(100)은 거북이가 100픽셀만큼 앞으로 이동합니다.
t.right(90)은 거북이가 오른쪽으로 90°(직각) 회전합니다.
t.left(90)은 거북이가 왼쪽으로 90°(직각) 회전합니다.

◆ 도전하기

'터틀그래픽'을 사용하여 사각형과 삼각형을 그려 봅시다.





2. 파이썬 스크립트 모드

- 긴 코드를 입력하거나 코드 저장이 필요할때 스크립트 모드 사용.
- 파이썬 쉘 메뉴에서 [File] -> [New File] 선택 에디터 창에서 작성 후 저장한다.
  저장 후 [Run] -> [Run Module] 또는 'F5'를 통하여 소스코드를 실행시키고 결과는 쉘 창에 출력된다.

1) 스크립트 모드 작성하기

- 제시된 소스코드를 작성해서 실행 시켜 봅시다.

```
print('스크립트 모드입니다.')
print('여러줄을 한 번에 실행시킬 수 있습니다.')
print('문자와 숫자를 출력할 때는 콤마를 사용합니다.')
print("일주일은", 7*24 , "시간 입니다." )
print('콤마를 사용하면 공백이 한 칸 생겨요.')
print("안녕","만나서","반가워!")
```

- 4번째 줄에 콤마(,)를 더하기(+)로 변경하여 실행 시켜봅시다. 몇번째 줄까지 출력되나요?
  ( )

* python은 대표적인 인터프리터 언어입니다. 인터프리터 언어는 작성한 코드를 한줄씩 읽어가며 기계어로 번역하면서 실행합니다. 따라서 오류 발생 전까지는 한 문장씩 실행하고 오류가 발견되면 오류메시지를 출력합니다.

2) 주석

- 주석은 프로그램의 진행에 전혀 영향을 주지 않는 코드로 컴퓨터에게 전달되지 않습니다.
- 본인이 작성한 코드를 나중에 다시보거나 여럿이 함께하는 프로젝트에서 다른 사람의 이해를 돕기 위해 중요한 역할을 합니다
- 주석은 # 또는 따옴표 3개 ‘’‘ 를 사용하여 표현합니다.
  - 한 줄 주석 : # (# 뒤의 문장 전체가 주석처리 되어 출력되지 않습니다.)
  - 여러 줄 주석 : ’‘’ ~ ‘’‘ / “”“ ~ ”“” (3개의 따옴표 사이에 들어간 모든 문장을 주석처리 합니다.)

① #을 추가하고 실행 시켜 봅시다.

```
#print('스크립트 모드입니다.')
print('여러줄을한번에 실행시킬 수 있습니다.')
#print('문자와 숫자를 출력할 때는 콤마를 사용합니다.')
print("일주일은" 7*24 "시간 입니다." )
print('콤마를 사용하면 공백이 한 칸 생겨요.')
print("안녕","만나서","반가워!")
```

② 소스 코드에 아래 문장을 추가하고 실행 시켜 봅시다.

```
''' 여러줄을 입력하는 스크립트 모드 연습입니다.
인터프리터가 소스파일을 한 줄씩읽어 실행시킵니다.
'''
```

③ 전체프로그램을 주석처리하려면 어떻게 하여야 할까요?

3) print() 함수

- print() 문이 없을 경우 컴퓨터가 실행은 하지만 출력 명령이 없으므로 결과 창에는 출력되지 않습니다.

- 제시된 소스코드를 스크립트 모드에서 작성해서 실행시켜 봅시다.

```
'Hello'
3+5
print('print함수 사용 출력')
print( 7 * 4 )
```

- print() 괄호 안에 아무것도 입력하지 않을 경우 빈 줄이 출력됩니다.

- 제시된 소스코드를 작성해서 실행시켜 봅시다.

```
print("# 하나만 출력합니다.")
print("HelloPythonProgramming...!")

print("# 여러 개를 출력합니다.")
print(10, 20, 30, 40, 50 )
print("안녕하세요","만나서","반갑습니다.")

print("#아무것도 출력하지 않습니다.")
print()
print()
print("--------확인용--------")
```

- 실행결과가 다음과 같이 빈 줄이 추가되도록 소스코드를 수정해 봅시다.

<실행결과>

```
# 하나만 출력합니다.
HelloPythonProgramming...!

# 여러 개를 출력합니다.
10 20 30 40 50
안녕하세요 만나서 반갑습니다.

#아무것도 출력하지 않습니다.


--------확인용--------
```

빈줄

빈줄

◆ 도전하기

① print()를 이용하여 다음과 같이 출력하시오. 단, 두 문장 사이에 빈 줄을 출력합니다.

<실행결과>

첫번째 줄입니다.

세번째 줄입니다.

② 자신의 번호가 단수인 구구단을 출력해 봅시다. 단, 5단 다음에 한줄 뛰기를 합니다. 가장 첫 줄에는 자신의 학번과 이름을 주석으로 넣습니다.

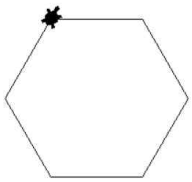
<실행결과>

```
25*1 = 25
25*2 = 50
25*3 = 75
25*4 = 100
25*5 = 125

25*6 = 150
25*7 = 175
25*8 = 200
25*9 = 225
```

4) 터틀 그래픽(2)

- 아래 프로그램에 코드를 추가하여 6각형을 완성해 봅시다.

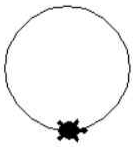
```
import turtle as t
t.shape('turtle')
t.forward(100)
t.right( 60 )
```



- **원 그리기**

- circle(반지름)

```
import turtle as t
t.shape('turtle')

t.circle(100)
```



- circle(반지름, 각도)

```
t.circle(100, 180)
```

- 오른쪽 그림과 같은 모양이 되도록 코딩해 봅시다.

```
import turtle as t
t.shape('turtle')
```

- **펜 설정하기**

- pensize(굵기)

```
t.pensize(5)
t.circle(100)
```

- pencolor('색상')

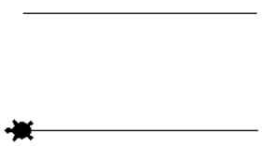
  < black, red, blue, orange, gold, green, pink, purple, brown, skyblue, yellow, bisque ...>

```
t.pencolor('green')
t.circle(100)
```

- **펜 올리고 내리기**

- up() / penup() : 펜을 올려 그림이 그려지지 않음
- down() / pendown() : 펜을 내려 그림을 그림.

```
t.forward(200)
t.up()
t.right(90)
t.forward(100)
t.right(90)
t.down()
t.forward(200)
```



◆ **도전하기**

① 거북이를 이동시켜 다음과 같은 그림을 그려봅시다.

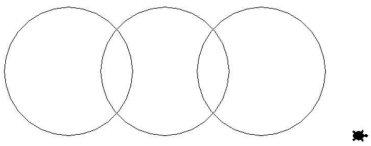
- 한변의 길이는 100, 펜의 굵기는 10, 펜 색은 ' orange'로 한다.

```
t.forward(길이)
t.right(각도)
t.left(각도)
t.pensize(굵기)
t.pencolor('색')
```

② 거북이를 이동시켜 다음과 같은 그림을 그려봅시다.

- 한 원의 반지름은 100, 각 원들의 중심간 거리는 150으로 한다.

```
t.circle(반지름)
t.up()/t.down()
t.forward(거리)
```



- **turtle 기초 함수**

동작	forward(*거리*)	거북이가 앞으로 이동합니다.
	backwoard(*거리*)	거북이가 뒤로 이동합니다.
	left(*각도*)	거북이가 왼쪽으로 회전합니다.
	right(*각도*)	거북이가 오른쪽으로 회전합니다.
	circle(*반지름, 각도*)	현재 위치에서 각도만큼 원을 그립니다.
	goto(*x, y*)	거북이를 특정 위치(좌표)로 보냅니다.
	home()	거북이의 위치와 방향을 처음 상태로 돌립니다.
	speed(*속도*)	거북이 속도를 바꿉니다. 0 : 최고 속도 1 : 가장 느린 속도 10 : 빠른 속도
펜	pendown() / down()	펜(잉크 묻힌 꼬리)를 내립니다. (그릴 수 있는 상태)
	penup() / up()	펜 (잉크 묻힌 꼬리)를 올립니다.
	pensize(*굵기*)	펜 굵기를 바꿉니다.
	fillcolor("색상명")	도형 내부를 칠하는 색을 바꿉니다.
	begin_fill()	도형 내부를 색칠할 준비를 합니다.
	end_fill()	도형 내부를 색칠합니다.
	reset()	화면을 지우고 거북이도 원래 자리와 상태로 되돌립니다.
	clear()	거북이를 그대로 둔 채 화면을 지웁니다.
	write()	현재 거북이 위치에 문자를 출력합니다. ex) t.write("Hello") #현재 위치에 Hello를 출력
상태	shape("모양명")	거북이 모양을 바꿉니다. 모양명 : classic(기본값), circle, square, triangle, arrow, turtle

II. 변수와 자료형

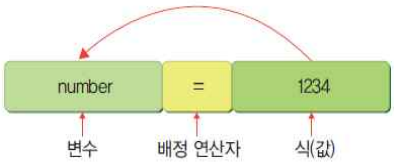
1. 변수(Variable)

1) 변수의 정의

- 변수(variable)는 값을 저장하는 상자로 컴퓨터 메모리 공간에 만들어 진다.
- 프로그램을 작성할 때 필요한 데이터를 저장하거나 가져와 사용할 수 있다.
- 변수에는 문자, 숫자 모두 저장 및 변경이 가능하다.

2) 변수 선언 및 할당, 변경

- 변수를 사용하기 위해서는 변수를 선언해 주어야 하는데 배정연산자(=) 의 왼쪽에 변수이름을 적어주고 오른쪽에 값을 입력하면 값을 변수에 할당하게 된다.
- 변수의 값은 언제든지 변경될수 있으며 **마지막에 입력된 값만을 저장**합니다.




```
x = 100     #100을 변수 x에 저장
x = 0.2     #0.2를 변수 x에 저장
name = '홍길동'  #홍길동을 변수 name에 저장
```

3) 변수를 이용한 연산

- 변수는 <변수와 변수>, <변수와 값> 연산이 가능하며 그 연산한 결과를 다시 변수에 저장할 수 있다.
- 배정연산자(=)의 왼쪽에는 숫자나 식이 올 수 없다.

```
>>> x = 100
>>> y = 200
>>> x + y

>>> result = x + y
>>> result

>>> score = 10
>>> score = score + 1
>>> score
```

```
>>> pi = 3.14159265
>>> pi + 2

>>> pi - 2

>>> pi * 2

>>> pi / 2

>>> pi % 2

>>> pi * pi
```

- 변수 사용의 잘못된 예

```
>>> 8 = number      # = 의 왼쪽이 숫자 => Error
SyntaxError: cannot assign to literal here.
>>> x + 1 = x       # = 의 왼쪽이 식 => Error
SyntaxError: cannot assign to expression here.
```

4) 변수 이름 규칙

- 첫 글자는 영문 혹은 _(언더바)로 시작하며 영문자, 숫자, _(언더바)만으로 구성된다.
- 대소문자를 구분해서 사용하고 공백을 들어가면 안된다.
- 파이썬에서 이미 다른 용도로 사용되고 있는 예약어(for, if 등)는 변수명으로 사용할 수 없다

```
result                  # 영문 알파벳 문자로 시작
_count                  # 밑줄 문자로 시작할 수 있다.
number_of_pictures      # 중간에 밑줄 문자를 넣을 수 있다.
King3                   # 맨 처음이 아니라면 숫자도 넣을 수있다.
```

5) 연습하기

- 제시된 소스코드를 작성해서 실행시켜 봅시다.

```
pi=3.14
r = 10

print('원주율 =', pi)
print('반지름 =', r)
print('원의 둘레 =', 2*pi*r)
print('원의 넓이 =', pi*r*r)
```

- 제시된 소스코드를 작성해서 실행시켜 봅시다.

```
day = 14
month = 3
week = '수요일'

print( '오늘은', month, '월', day, '일입니다.')
```

- 실행결과가 다음과 같도록 print문을 수정해 봅시다

<실행결과>

오늘은 3월 14일 수요일 입니다.

◆ 도전하기

① 변수를 이용하여 구구단을 출력하는 프로그램을 작성해 봅시다.

```
num = 2
```

<실행결과>

2 * 1 = 2
2 * 2 = 4
2 * 3 = 6
2 * 4 = 8
2 * 5 = 10
2 * 6 = 12
2 * 7 = 14
2 * 8 = 16
2 * 9 = 18

② 양의 정수를 저장하는 변수 두개를 이용하여 사칙연산하는 프로그램을 작성해 봅시다.

```
a = 6
b = 3
```

<실행결과>

6 + 3 = 9
6 - 3 = 3
6 * 3 = 18
6 / 3 = 2.0

③ 삼각형 밑변과 높이 변수을 선언하고 값을 할당하여 삼각형의 넓이를 구하는 프로그램을 작성해 봅시다.

```
w = 10      #밑변
h = 20      #높이
```

<실행결과>

밑변 10 높이 20 의 삼각형의 넓이는 100.0

2. 자료형

1) 자료형

- 데이터의 종류, 자료의 형식
- 파이썬은 **변수**를 만들 때 사용자가 **자료형을 결정하지 않아도 파이썬 내부에서 자료형을 판단**한다.
- 각 자료형의 특징을 잘 이해하면 효율적인 코드를 작성할 수 있다.
- 필요에 따라 자료형을 변경할 수 있다.
- 자료형 확인은 type() 함수로 알수 있다.

숫자형	논리형(불형)	문자열	군집형
int : 정수 float : 실수 complex:복소수	bool : 참/거짓	str : 문자열	list : 리스트 tuble :튜플 dic : 딕셔너리 set : 집합

2) 자료형의 종류

(1) 숫자형

- 정수형(int) : 소수점이 없는 수. ex> -1, 99, 1234
- 실수형(float) : 소수점이 있는 수 ex> -1.0, 9.33, 3.14e5
- 연산이 가능하다.

```
>>> type(10)

>>> type(3.14)

>>> type(3.14e5)
```

```
>>> x=10
>>> type(x)

>>> x=3.14
>>> type(x)
```

- 정수와 정수의 연산

```
>>> x=6
>>> y=3
>>> type(x+y)    #정수+정수   => 정수

>>> type(x-y)    #정수-정수   => 정수

>>> type(x*y)    #정수*정수   => 정수

>>> type(x/y)    #정수/정수   => 실수
```

- 정수와 실수의 연산

```
>>> x=6
>>> y=3.14
>>> type(x+y)    #정수+실수 => 실수

>>> type(x-y)    #정수-실수 => 실수

>>> type(x*y)    #정수*실수 => 실수

>>> type(x/y)    #정수/실수 => 실수
```

(2) 불(bool)형

- 참(True)과 거짓(False)을 나타내는 자료형
- 결과가 참(True)이나 거짓(False)으로 나타내는 식의 자료형
- **True, False** 두 가지 값만 가질 수

```
>>> type(True)

>>> type(False)

>>> a = True
>>> type(a)

>>> a = False
>>> type(b)
```

```
>>> 3 < 5

>>> type( 3 < 5 )

>>> 3 >= 5

>>> type( 3>=5)
```

(3) 문자열(str)형

- 문자열(string)은 문자들의 나열(sequence of characters)이다.
- (쌍)따옴표 안에 포함된 문자, 숫자, 특수문자들의 모임. **여는 따옴표와 닫는 따옴표의 모양이 일치**해야 함.

```
>>> type( "Hello" )

>>> msg = "Hello"
>>> type(msg)
```

- **문자열 인덱스**

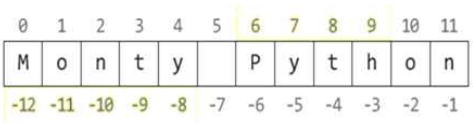
  문자열 내의 위치를 인덱스로 표현할 수 있다.

  인덱스 : 문자열 내에서의 위치

  - 양의 인덱스 : 첫번째 글자부터 **인덱스[0] 으로 시작해서 [문자열 길이-1]**까지 지정
  - 음의 인덱스 : 마지막 글자부터 인덱스[-1]로 시작해서 [-문자열 길이]까지 지정

  문자열에서 개별 문자들을 인덱스 번호를 사용하여 추출할 수 있다.

```
>>> msg = 'Monty Python'
>>> msg[0]

>>> msg[8]

>>> msg[-2]
```



- **문자열 슬라이스**

  인덱스 번호를 사용하여 문자열에서 부분 문자열을 반환

  문자열 **시작인덱스**에서 **끝인덱스-1**까지 문자열 반환

  시작인덱스가 생략되면 첫 글자부터 시작하고 **끝인덱스가 생략되면 마지막 글자까지** 반환

```
>>> msg = 'Monty Python'
>>> msg[6:10]

>>> msg[2:7]

>>> msg[:4]

>>> msg[3:]
```

◆ 도전하기

① 입력된 값의 2번째 4번째 7번째 문자를 출력해 봅시다.

```
say = 'Happy Day'
```

<실행결과>

a
p
D

② 입력된 문자를 거꾸로 출력하는 프로그램을 작성해 봅시다.

```
txt = '파이썬'
```

<실행결과>

썬 이 파

③ 주민등록번호 앞자리 6자리를 년, 월, 일 로 나누어 출력해 봅시다.

```
date = '060317'
```

<실행결과>

06 년 03 월 17 일

- **다양한 문자열 표현 방법**

① 문자열에 따옴표 포함시키기

- 문자열에 작은따옴표(‘)를 포함시키기 위해서는 문자열을 큰따옴표(“)로 둘러싸야 한다.
- 문자열에 큰따옴표(”)를 포함시키기 위해서는 문자열을 작은따옴표(’)로 둘러싸야 한다.

```
>>> msg = ' "안녕하세요"라고 말했습니다. '
>>> print(msg)

>>> msg = " Kim's my friend'
>>> print(msg)
```

② 이스케이프 문자를 사용해 문자열 표현하기

- 문자열 내부에서 특수하게 사용할 수 있도록 미리 약속으로 정해 둔 코드
- 역슬레시(\) 기호와 함께 조합해서 특수한 문자를 의미한다.
- 한국어 키보드에서 역슬레시(\)는 원화 기호(₩)를 사용한다.

\’ (역슬래시 작은따옴표) : 문자열 내부에서 작은따옴표(’)를 표현할 때 사용

\” (역슬래시 큰 따옴표) : 문자여 내부에서 큰 따옴표(")를 표현할 때 사용

\n (역슬래시 n) : 문자열 내부에서 줄바꿈할 때 사용

\t (역슬래시 t) : 문자열에서 간격을 줄 때 사용(탭 간격)

\\ (역슬래시 역슬래시) : 문자열 내부에서 역슬래시(\)를 표현할 때 사용

- 컴퓨터는 이스케이프 코드로 인식하고 출력시 미리 약속된 문장코드로 출력한다.

```
>>> msg='Python\'s favorit food is perl'
>>> msg          #컴퓨터 내부에 저장된 값

>>> print(msg)          #모니터에 출력 되는 값

>>> msg="\"Python is very easy.\" he says."
>>> msg

>>> print(msg)
```

```
>>> msg='안녕하세요.\n만나서 반갑습니다.‘
>>> msg

>>> print(msg)
```

```
>>> msg='안녕하세요.\t만나서 반갑습니다.‘
>>> msg

>>> print(msg)
```

```
>>> msg='\\ \\ \\ \\'
>>> msg

>>> print(msg)
```

◆ 도전하기

① 실행결과가 다음과 같도록 msg 변수에 알맞은 값을 입력해봅시다. <실행결과>

```
msg =



print(msg)
```

동해물과 백두산이 마르고 닳도록
하느님이 보우하사 우리나라 만세
무궁화 삼천리 화려강산 대한사람
대한으로 길이 보전하세

② 실행결과가 다음과 같도록 프로그램을 완성해 봅시다. <실행결과>

```
print("이름\t나이\t지역")
print("윤인성\t20\t남동구")
```

이름	나이	지역
윤인성	20	남동구
윤아린	24	서구
국영종	18	중구

③ 실행결과가 다음과 같도록 print문을 완성해 봅시다. <실행결과>

```
print(                              )
```

c:\\home.py

3) 자료형 변환

(1) 문자열을 숫자로 변환하기

- int(문자열) : 문자열을 정수로 변환한다.
- float(문자열) : 문자열을 실수로 변환한다.

```
>>> x = "100"
>>> type(x)

>>> x = int("100")    #문자'100' -> 정수 100
>>> x

>>> type(x)
```

```
>>> x = "100"
>>> type(x)

>>> x = float("100") #문자'100' -> 실수 100.0
>>> x

>>> type(x)
```

(2) 숫자를 문자열로 변환하기

- str(숫자) : 숫자를 문자열로 변환한다.

```
>>> x = 3.14
>>> type(x)

>>> x = str(3.14)
>>> x

>>> type(x)
```

- 변수 a의 자료형과 a의 출력결과를 적어봅시다.

문제	a의 자료형	출력결과
a=int(12.7)	정수형	12.7
a=int('123')		
a=float(456)		
a=float('65.4')		
a=str(52)		
a=str(12.34)		

(3) 불(bool)형 자료형을 숫자/문자로 변환하기

- int(True), float(True) : 숫자 1로 변환한다.
- int(False), float(False) : 숫자 0로 변환한다.

```
>>> int( True )

>>> int( False )

>>> float( True )

>>> float( False )
```

- str(True) : 문자 'True'를 반환한다.
- str(False) : 문자 'False'를 반환한다.

```
>>> str(True)


>>> str(False)
```

(4) 숫자나 문자를 불(bool)형으로 변환하기

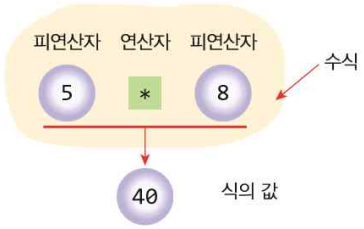
- bool(데이터) : 입력값을 True나 False로 변환한다.
- 입력값이 0이나 "" 경우 False 반환, 그 외의 값은 True로 반환한다.

```
>>> bool( 1 )

>>> bool( 2 )

>>> bool( 0 )
```

```
>>> bool('hi')

>>> bool(" ")  #공백 한칸도 True로 인식

>>> bool("")
```

III. 연산자

- 수식 : 피연산자들과 연산자의 조합
- 연산자 : 연산을 나타내는 기호
- 피연산자 : 연산의 대상이 되는 값
- 연산자의 종류 : 산술연산자, 비교연산자, 논리연산자, 비트연산자 등..

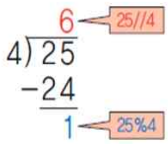


1. 산술 연산자

- 숫자를 피연산자로 하는 계산을 위한 연산자

산술 연산자	파이썬 연산자	수학의 식	파이썬의 식
덧셈	+	x + 3	x + 3
뺄셈	-	a − b	a - b
곱셈	*	bm 또는 b · m	b * m
나눗셈	/	x ÷ y	x / y
나눗셈(정수 부분만)	//		x // y
나머지(모듈로)	%		x % y
지수	**	x^y	x ** y

```
>>> 25/4

>>> 25//4

>>> 25%4
```



- 나머지 연산자를 활용한 프로그램

```
#짝수 홀수 판별
>>> number = 28
>>> number%2

>>> number = 5
>>> number%2
```

- 지수 연산자를 활용한 프로그램

```
>>> 2 ** 7     #2^7

>>> 2 ** 10
```

<원리금 계산하기>

```
>>> 원금 = 1000
>>> 이자율 = 0.05
>>> 기간 = 10
>>> 원금*(1+이자율)**기간
```

① 변수 A의 값을 변수 B의 값으로 나누었을 때 몫과 나머지를 구하는 프로그램을 작성해 봅시다.

```
A = 2023
B = 22
```

<실행결과>

2023을 22로 나누었을 때
몫은 91
나머지는 21

② 초 단위의 시간을 받아서 몇 분 몇 초 인지 계산하는 프로그램을 작성해 봅시다.

```
sec = 1000
```

<실행결과>

16 분 40 초

③ 원기둥의 반지름(r)과 높이(h)가 저장된 변수를 사용하여 부피를 계산하는 프로그램을 작성해 봅시다.

< 원기둥의 부피(V) 구하는 공식 : $V = 3.14 * r^2 * h$ >

<실행결과>

원기둥의 반지름 : 5
원기둥의 높이 : 100
원기둥의 부피 : 7850.5

• **복합연산자**

- 대입 연산자(=)와 다른 연산자를 합쳐 놓은 연산자이다.
- 연산에서 결과값을 저장되는 변수(좌변)가 연산에 사용(우변)되는 경우 간소화한 형태

복합 연산자	의미
x += y	x = x + y
x -= y	x = x - y
x *= y	x = x * y
x /= y	x = x / y
x %= y	x = x % y

```
>>> a = 20
>>> a += 10          # a = a+10
>>> a

>>> a -=15           # a=a-15
>>> a

>>> a *=2            # a = a*2
>>> a

>>> a //=4           # a = a // 4
>>>a

>>>a%=3             # a = a %3
>>>a
```

• **연산자 우선순위**

- 많은 연산들 중에서 어떤 연산을 먼저 수행할 지를 결정하는 규칙
- 우선순위 대로 연산을 하지 않기 위해서는 괄호를 사용한다.

순위	연산자	설명
1	**	지수 연산자
2	~ + -	단항 연산자
3	* / % //	곱셈, 나눗셈, 나머지 연산자
4	+ -	덧셈, 뺄셈

```
>>> 10 + 20 / 2

>>> (10 + 20) / 2
```

① 화씨온도(F)를 섭씨온도(C)로 바꾸는 프로그램을 작성해 봅시다. <공식 : $C=(F-32)*\frac{5}{9}$ >

< 실행결과>

화씨 100도
섭씨 37.77777777777778도

② 거스름돈 계산 프로그램

자동 판매기에서 동전 거스름돈을 계산하는 프로그램이다.
자판기는 동전 500원, 100원짜리만가지고 있고 거스름돈은 최소한의 동전 개수로 반환한다.
거스름 돈, 거스름돈 의 500원의 개수, 100원의 개수를 출력해 봅시다.

hint : 2200을 500으로 나누면 몫은 4, 나머지는 200입니다.

```
money = 5000
price = 2800
```

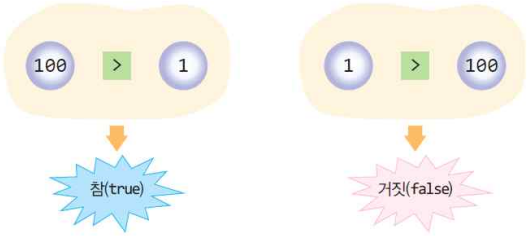
<실행결과>

거스름 돈 : 2200
500원 동전 개수 : 4
100원 동전 개수 : 2

2. 관계 연산자

- 두 개의 피연산자를 비교하는 연산자
- 주어진 두 개의 값을 비교하여 True , False 반환

관계 연산자 (파이썬)	수학 기호	의미	예	결과
>	>	큰가?	7 > 9	False
<	<	작은가?	10 < 5	False
>=	≥ (또는 ≥)	크거나 같은가?	6.0 >= 6.0	True
<=	≤ (또는 ≤)	작거나 같은가?	123 <= 126	True
==	=	같은가?	1 + 1 == 3	False
!=	≠	다른가?	3.2 != 2.5	True



- 숫자 비교하기

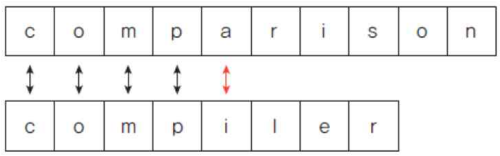
```
>>> 10 == 100
>>> 10 != 100
>>> 10 < 100
>>> 10 > 100
>>> 10 <= 100
>>> 10 >= 100
```

- 숫자 범위 비교하기

```
>>> x = 25
>>> 10 < x < 30
>>> 40 < x < 50
```

- 문자열 비교하기

```
>>>'cat' < 'dog'          # 알파벳 순서댈 크기 증가
>>> 'Dog' < 'dog'        # 대문자는 소문자보다 작다.
>>> 'Dog'  < 'cat'       # 모든 대문자는 소문자보다 작다.
>>> '알고리즘' < '파이썬'     # 한글은 가나다 순서대로 증가
>>> 'algorithm' < '알고리즘' # 알파벳은 한글보다 작다
```



[그림 4-2] 두 개의 문자열을 비교

- 문자열과 숫자는 비교할수 없다

```
>>>'cat' < 123
```

① 변수에 입력된 값이 양수인지 아닌지 확인해 봅시다.

```
x = 10
```

```
양수 인가요?  True
음수 인가요?  False
0 인가요? False
```

② 변수 a와 변수 b의 값이 같으면 True, 그렇지 않으면 False를 출력하는 프로그램을 작성해 봅시다.

```
a = 5
b = 5
```

```
입력된 두 값이 같은가? True
```

③ 변수에 입력된 값이 짝수인지 아닌지 확인해 봅시다.

```
x = 10
```

```
10 는 짝수인가요? True
```

3. 논리 연산자

- 조건식을 연결하여 새로운 조건식을 만듦
- <조건식> 논리연산자 <조건식>
- 주어진 부울값(참,거짓) 비교하여 True나 False반환
- and, or, not

X	Y	X and Y	X or Y	not X
True	True	True	True	False
True	False	False	True	False
False	True	False	True	True
False	False	False	False	True

• **'and' 연산자**

- 두 조건식이 **모두 참이면 참**, 그렇지 않으면 거짓

```
>>> 3 < 4  and  7 < 8    # True  and True

>>> 3 < 4  and  7 > 8    # True and False

>>> 3 > 4  and  7 < 8    # False and True

>>> 3 > 4  and   7 >  8   # False  and False
```

• **'or' 연산자**

- 하나만 참이여도 참, **모두 거짓이면 거짓**

```
>>> 3 < 4  or   7 < 8

>>> 3 < 4  or   7 > 8

>>> 3 > 4  or   7 < 8

>>> 3 > 4  or   7 >  8
```

• **'not' 연산자**

- 참이면 거짓, 거짓이면 참으로 변환

```
>>> not 7 < 8

>>> not 7 > 8
```

① 나이가 10살 이상이고 키가 140cm 이상일 경우 True를 출력, 그 외의 경우 False를 출력하는 프로그램.

```
age = 18
height = 165
```

True

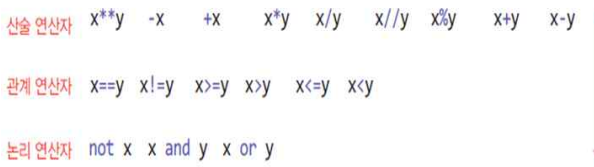
② 변수값이 짝수이거나 100보다 크면 True, 그 외의 경우 False를 출력하는 프로그램

```
a = 5
```

True

4. 연산자 우선순위

- 산술연산자 > 관계연산자 > 논리연산자
- 대입연산자(=) 오른쪽의 식을 우선순위에 따라 연산한 결과를 왼쪽의 변수(seq)에 저장.




```
>>> a = 2
>>> b = 3
>>> seq = a/b**3-2<10 or a*b>=20
>>> seq
```

5. 문자열 연산자

- **'+' 연산자**

- 두 문자열을 공백없이 연결한다.
- +연산자의 경우 두 피연산자가 같은 자료형이여야 한다.

```
>>> 'Hello' + 'World!'

>>> head = "Python"
>>> tail = " is fun!"
>>> head + tail
```

```
>>> 100 + '200'

>>> 100 + int('200')

>>> str(100) + '200'
```

- 숫자와 문자를 한줄에 출력할 결우 **숫자를 문자로 자료형 변환후 '+'연산자를 사용**하여 공백없이 출력할 수 있다

```
>>> age = 18
>>> print('나는 현재', age,'살 이다.')

>>> print('나는 현재'+str(age)+'살 이다.')
```

- **'*' 연산자**

- 문자열을 횟수만큼 반복한다.

```
>>> '안녕' * 5

>>> txt = "="
>>> txt * 20
```

① 자신이 좋아하는 과일 100번을 출력하는 프로그램을 작성해 봅시다.

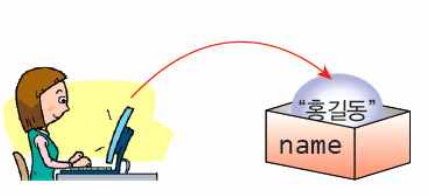
② 실행결과가 다음과 같도록 프로그램을 완성해 봅시다.

```
name = "홍길동"
age = 18
```

저의 이름은 홍길동입니다.
내년이면 19살 입니다.

IV. 표준입출력

1. 입력함수 : input()



1) input() 함수

- 사용자로부터 입력을 받는 함수
- 명령 프롬프트에서 키보드로 사용자가 직접 입력할 수 있다.
- input()괄호 안에 사용자에게 표시할 문자열을 넣을 수 있다.
- 반환된 값은 주로 변수에 담아 사용한다.
- 사용자가 숫자를 입력하더라도 **문자열** 형태로 입력 받는다.

```
변수 = input( )
변수 = input( '프롬프트 문자열' )
```

2) 문자열 입력 받기

```
txt = input( )
print( txt )
```

```
txt = input( '오늘의 기분은 어떤가요? ')
print( txt )
```

```
name = input("이름을 입력해 주세요> ")
print(name+'님 만나서 반갑습니다.')
hobby =
print(                                  )
```

```
이름을 입력해 주세요> 정은주
정은주님 만나서 반갑습니다.
취미는 무엇인가요? 독서
독서 저도 함께 하고 싶어요.
```

① 가장 좋아하는 노래에 대하여 질문하고 대답을 3번 출력하는 프로그램을 작성해 봅시다.

```
song =
```

```
좋아하는 노래 제목은? 신호등
신호등
신호등
신호등
```

② 나를 꾸며주는 말과 이름을 입력받아 한줄로 출력하는 프로그램을 작성해 봅시다.

```
deco =
name =
```

```
나를 꾸며주는 말은? 반짝반짝 빛나는
나의 이름은? 정은주
반짝반짝 빛나는 정은주
```

③ 자신이 꼭 가보고 싶은 곳을 입력받아 다음과 같이 출력하는 프로그램을 작성해 봅시다.

```
가보고 싶은 장소는? 지중해
지중해는 참 좋은 곳이에요.
```

3) 숫자 입력 받기

- 사용자가 숫자를 입력하더라도 문자열 형태로 입력 받습니다.
- 입력 받은 문자열을 숫자로 사용하기 위해서는 형 변환이 필요합니다. int( ) float( ) 등

(1) 정수 입력 받기

```
변수 = input( )
변수 = int( 변수 )
```

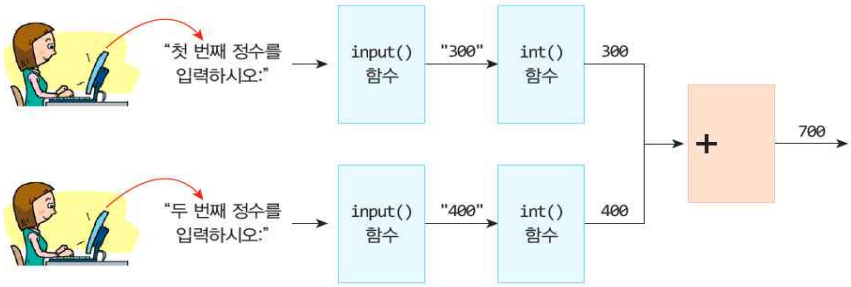
==

```
변수 = int( input( ) )
```

```
x = input("첫 번째 정수 입력 : ")
y = input("두 번째 정수 입력 : ")
x = int( x )
y = int( y )
print( x+y )
```

==

```
x = int( input("첫 번째 정수 입력 : ") )
y = int( input("두 번째 정수 입력 : ") )
print( x+y )
```



(2) 실수 입력 받기

```
변수 = input( )
변수 = float( 변수 )
```

==

```
변수 = float( input( ) )
```

```
x = float( input("첫 번째 실수 입력 : ") )
y = float( input("두 번째 실수 입력 : ") )
print( x+y )
```

① 사용자로부터 3개의 정수를 받아서 평균을 계산하고 결과를 출력하는 프로그램을 작성해 봅시다.

```
x =
y =
z =
```

```
첫 번째 숫자를 입력하시오 : 10
두 번째 숫자를 입력하시오 : 20
세 번째 숫자를 입력하시오 : 30
10 20 30 의 평균은  20.0 입니다
```

② 사용자로부터 신장과 체중을 입력받아서 BMI 값을 출력하는 프로그램을 작성해 봅시다.

$$BMI = \frac{\text{몸무게(kg)}}{(\text{키})^2}$$

```
몸무게를 kg 단위로 입력하시오: 45
키를 미터 단위로 입력하시오: 1.61
당신의 BMI= 17.360441340997646
```

◆ 연습하기1

① 20초 세어 맞히는 프로그램

```
import time

input('엔터를 누르고 20초를 셉니다.')
start=time.time()

input('20초 후에 다시 엔터를 누릅니다.')
end=time.time()

et=end-start
print('실제시간 :', et,"초")
```

엔터를 누르고 20초를 셉니다.
20초 후에 다시 엔터를 누릅니다.
실제시간 : 19.80247187614441 초

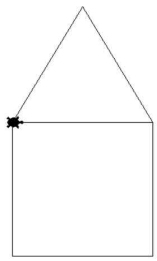
② 다음은 한 변의 길이가 100인 집을 그리는 프로그램이다.
프로그램을 수정하여 사용자로부터 집의 크기(한변의 길이)를 입력받아 집을 그리는 프로그램을 작성해봅시다.

```
import turtle as t
t.shape('turtle')

t.forward(100)
t.right(90)
t.forward(100)
t.right(90)
t.forward(100)
t.right(90)
t.forward(100)
t.right(90)

t.forward(100)
t.left(120)
t.forward(100)
t.left(120)
t.forward(100)
t.left(120)
```

➪

```
import turtle as t
t.shape('turtle')
```

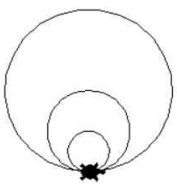
집의 크기를 얼마로 할까요? 200



③ 반지름을 입력 받아 반지름의 크기가 두배씩 증가하는 원을 3개 그리는 거북이를 만들어 봅시다.

```

```

반지름의 크기를 얼마로 할까요? 20



◆ 연습하기2

① 사용자가 입력한 문자열 중에서 처음 2글자와 마지막 2글자를 추출한 후에 이들을 합쳐서 출력하는 프로그을 만들어 봅시다.

```
txt = input('문자열을 입력하시오: ')
```

문자열을 입력하시오: Good Morning
Gong

② 사용자가 입력한 기호 2개 안에 입력한 문자열을 삽입하는 프로그램을 작성해 봅시다.

```
sign =
msg =
```

기호를 입력하시오: []
중간에 삽입할 문자열을 입력하시오: Happy Day
[Happy Day]

③ 밑(base) 과 지수(power)를 입력받아 거듭제곱 수를 표시하시오.

```
base =
power =
```

거듭제곱할 수 : 2
곱할 횟수 : 3
2 의 3 제곱은 8

④ 숫자 3개를 입력받아 세 숫자가 오름차순으로 입력되었으면 True, 그렇지 않으면 False를 출력하는 프로그램을 작성해봅시다. 단, 동일한 수가 입력되면 오름차순으로 입력되었다고 인정합니다.

```
x =
y =
z =
```

첫번째 숫자> 3.14
두번째 숫자> 17.3
세번째 숫자> 29.87
True

⑤ 동아리 환영 파티를 준비하려 한다. 파티에 참석하는 회원들을 위해 음식을 준비한다고 가정하다.
문제는 파티에 참석할 수 있는 획원들의 숫자가 변경될수 있다는 점이다.
참석자에 맞춰 필요한 치킨(1인 한 마리), 콜라(1인 2캔), 감자칩(1인 1개)의 수를 출력하는
프로그램을 작성해 봅시다.

```
count =
```

파티 참석 인원> 21
치킨 21 마리
콜라 42 캔
감자칩 21 개

추가문제> 치킨(9000), 콜라(400), 감자칩(700) 가격이 정해져 있을 때 총 주문금액을 계산해 봅시다.

파티 참석 인원> 21
치킨 21 마리
콜라 42 캔
감자칩 21 개
총 금액은 220500 원

2. 출력함수 : print()

1) print() 함수

- 괄호 안의 값을 모니터에 출력할 때 사용.
- **문자열 출력시에는 (쌍)따옴표**나 사용.
- 숫자나 변수 값 출력시에는 (쌍)따옴표를 사용하지 않음.
- print() 명령어를 그대로 사용하면 줄 바꿈을 함.
- 출력 대상이 여러 개일 때는 **쉼표**로 연결(출력 대상 사이에 **공백**이 생김).
- 출력 대상이 문자열 여러 개일 때는 **+로** 연결 (출력 대상 사이 공백 없이 출력함).
- *(반복) 연산 사용하여 문자열을 반복해서 출력할수 있음.
- 문자 내부에 큰따옴표를 넣기 위해서는 문자열를 만들때 작은따옴표를 사용한다.(반대도 동일)
- 줄 바꿈을 없애고 싶을 때는 괄호 마지막에 end=' '를 추가한다. (end 초깃값은 줄바꿈(\n) 문자)

```
print( 'No pain',  end  = ' ' )
print( 'No gain' )
```

2) 문자열 포매팅(f-string)

문자열에서 특정 부분만을 변경하려 할때, 문자열 포매팅을 이용해서 예쁘게 출력할수 있다.

- **f-string 포매팅**

```
f'문자열 {변수} 문자열'
```

- Python3.6 버전부터 사용가능
- f와 중괄호 { } 만을 이용하여 포맷팅.
- 문자열 맨 앞에 f를 붙여주고, 중괄호 안에 직접 변수 이름이나 출력하고 싶은 것을 넣는다.

```
>>> s = 'coffee'
>>> n = 5
>>> f'저는 {s}를 좋아합니다. 하루 {n}잔 마셔요.'
```

- **정렬하기**

중괄호 안에서 콜론(:) 을 통해 정렬하고자 하는 방향을 입력한다. < (왼쪽), ^ (가운데), >(오른쪽) 방향 입력후 문자열 전체 길이를 입력한다.

```
f'{변수:'방향' '문자열 전체 길이' }
```

```
>>> txt = '정렬하기'
>>> f'{ txt:<10 }'          # '정렬하기        '
>>> f'{ txt:^10 }'          # '    정렬하기    '
>>> f'{ txt:>10 }'          # '        정렬하기'
```

- **소수점 표시**

콜론(:) 뒤에 **.(점)표현할 자리수f** 를 입력한다.

```
f'{ 실수: .표기할 자리수f }
```

```
>>> a = 3.141592
>>> f"기존 값 : {a}"

>>> f"소수 첫번째 자리까지 표기: {a:.1f}"

>>> f"소수 두번째 자리까지 표기: {a:.2f}"

>>> f"소수 세번째 자리까지 표기: {a:.3f}"
```

① 두 정수를 입력받아 사칙연산하는 프로그램을 작성해 봅시다.
단, 출력은 f-string 포맷팅을 이용하여 출력하고 나누기 연산은 소숫점 둘째자리까지 표시하시오.

```
a =
b =
```

```
첫번째 수 입력> 10
두번째 수 입력> 8
10+8=18
10-8=2
10x8=80
10/8=1.25
```

② 다음 프로그램을 수정하여 특수 문자를 입력 받아 다음과 같이 마름모 모양이 나오도록 출력해 봅시다.

```
print(f'{"$":^10}')
print(f'{"$"*3:^10}')
```

```
    $
   $$$
 $$$$$
   $$$
    $
```

③ 해외 여행을 위해 한국 돈을 미국돈으로 환전하고자 한다. 1달러가 1320원이라고 할 때
원화를 입력받아 몇 달러로 받을 수 있는지 계산해 보자. 환전 후 남은 금액이 얼마인지도 출력해 봅시다.

```
rate = 1320
money =
```

```
환전할 금액을 원단위로 입력해 주세요> 1000000
1000000원은 757달러로 환전
잔액은 760원
```

④ 로봇 기자 만들기
사용자에게 경기장, 이긴 팀, 진 팀, 우수 선수, 점수를 질문하고 변수에 저장한다.
f-string 포맷팅을 활용하여 다음과 같이 출력해 봅시다.

```
place =
win =
lose =
mvp =
win_s =
lose_s =
```

```
경기장은 어디? 문학
이긴팀은? SSG
진팀은? 삼성
우수선수는? 최정
이긴팀 점수는? 7
진팀 점수는? 4

오늘 문학에서 야구 경기가 열렸습니다.
SSG과 삼성은 치열한 공방전을 펼쳤습니다.
최정이 맹활약을 하였습니다.
결국 SSG가 삼성을 3점 차로 이겼습니다.
```

⑤ 자신이 2050년에는 몇 살이 될 것인지 계산하는 프로그램을 작성해 봅시다.
f-string 포맷팅을 활용하여 다음과 같이 출력해 봅시다.

```
import time
now = time.time()
thisYear= int(1970 + now//(365*24*3600))
```

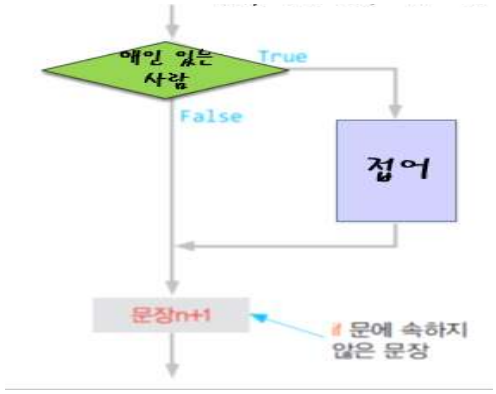
```
올해는 2024년입니다.
올해 나이를 입력해 주세요>> 18
2050년에는 44살 이시군요
```

V. 선택구조

- 참(True) 또는 거짓(False) 여부를 판단하여 선택적으로 일을 처리함

1. if 문

- if 명령 뒤에 오는 조건식이 참(True)일때만 내부 코드 실행




```
if 조건식 :
    조건이 참일 때 수행할 문장1
    조건이 참일 때 수행할 문장2
```

*들여쓰기
if문을 만들 때는 조건식 바로 아래 문장부터 if문에 속하는 모든 문장에 들여쓰기(공백4개)를 해주어야 함. 들여쓰기 밖으로 나오면 if문을 빠져나옴

- 입력받은 숫자가 양수일 경우에만 '양수 입니다.'를 출력하는 프로그램

```
num=int(input('숫자 입력: ') )
if num > 0 :
    print("양수 입니다.")
print("프로그램 끝")
```

정수 입력: 5
양수 입니다.
프로그램 끝

정수 입력: -2
프로그램 끝

추가미션 > 양수를 입력했을 때 출력메시지가 추가되도록 프로그램을 수정해 봅시다
.

정수 입력: 5
양수 입니다.
5 은 0보다 큰 수 입니다.
프로그램 끝

- 뷔페식당에서 1인당 요금을 계산하여 출력하는 if문이다.
1인당 요금은 32000원이고 나이가 3세 이하이면 요금은 0원으로 변경된다.

```
age = int( input('나이를 입력해 주세요> ') )
price = 32000
if age <= 3 :
    print("유아는 무료")
    price = 0
print("가격: ", price )
```

① 입력된 값이 짝수이면 '짝수입니다', 홀수이면 '홀수입니다'가 출력되는 프로그램을 작성해 봅시다.

```
num =
```

정수 입력> 2
짝수입니다.

정수 입력> 5
홀수입니다.

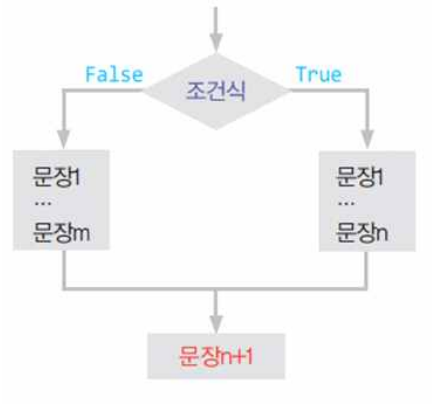
② 과제수행시간(분)과 과제점수를 입력받아
과제수행시간이 40분 미만이고 과제 점수가 80점 이상이면 '합격' 를 출력하도록 코딩해 봅시다.

```
time = int( input('과제수행시간(분) 입력> ')  )
score = int(input('과제점수 입력> ') )
```

과제수행시간(분) 입력> 35
과제점수 입력> 78

과제수행시간(분) 입력> 39
과제점수 입력> 90
합격

2. if-else 문



- if 명령 뒤에 오는 조건식이 참(True)일때는 if문 내부 코드 실행
  if 명령 뒤에 오는 조건식이 거짓(False)이면 else문 내부 코드 실행

```
if 조건식 :
    조건이 참(True)일 때 수행할 문장
else :
    조건이 거짓(False)일 때 수행할 문장
```

- 다음 if문을 <u>if-else문으로 변경하여</u> 입력된 값이 짝수이면 '짝수입니다', 홀수이면 '홀수입니다'가 출력되는 프로그램을 작성해 봅시다.

<< if 문 >>

```
num = int(input('정수 입력> ') )

if  num%2 == 0  :
      print('짝수입니다.')

if  num%2==1  :
      print('홀수입니다.')
```

⇒

<< if - else 문 >>

```

```

① 돈을 입력받아 4000원 이상의 돈을 가지고 있으면 '택시를 타고 가라'
그렇지 않으면 '걸어가라'는 메시지를 출력하도록 프로그램을 작성해 봅시다.

```
money =
```

```
현재 금액을 입력해 주세요> 5000
택시를 타고 가라

현재 금액을 입력해 주세요> 2500
걸어가라
```

② 돈와 카드를 입력받아 돈이 4000원 이상이거나 카드가 있다면 '택시를 타고 가라'
그렇지 않으면 '걸어가라'는 메시지를 출력하도록 프로그램을 작성해 봅시다.

```
money =
card =
```

```
현재 금액을 입력해 주세요> 2300
카드가 있나요?(Y/N) Y
택시를 타고 가라

현재 금액을 입력해 주세요> 3400
카드가 있나요?(Y/N) N
걸어가라
```

③ 나이가 10살 이상이고 키가 140cm 이상일 경우 '탑승 가능합니다.'
그 외의 경우 '탑승 불가능합니다.'를 출력하는 프로그램을 작성해 봅시다.

```
age =
height=
```

```
나이를 입력해 주세요> 18
키를 cm단위로 입력해주세요> 165
탑승 가능합니다.

나이를 입력해 주세요> 9
키를 cm단위로 입력해주세요> 180.2
탑승 불가능합니다.
```

④ 속도를 입력받아 속도가 50 이상이면 '과속입니다.'
50 미만이면 '정상속도입니다.'를 출력하는 프로그램을 작성해 봅시다.

```
CarSpeed =
```

현재 속도 입력> 70
과속입니다.
속도를 줄여주세요.

현재 속도 입력> 49
정상속도 입니다.
즐거운 하루 되세요.

■ random 모듈

- 랜덤으로 숫자를 생성하거나 선택하는 함수를 갖고 있는 모듈
- random() : 0.0이상 1.0이만의 실수 반환
- randint(숫자1, 숫자2) : 숫자1부터 숫자2까지 사이의 정수 반환

```
>>> import random
>>> random.random()

>>> random.random()

>>> random.randint(-5, 3)

>>> random.randint(-5, 3)
```

구구단 게임 > 컴퓨터가 1~9 사이의 두 개의 무작위 정수를 생성하고 두 수의 곱을 물어봅니다.
사용자가 답을 입력하여 맞으면 '정답입니다.' 틀리면 '구구단 공부합시다.'를 출력하는 프로그램 작성해 봅시다.

```
import random

a=random.randint(1,9)
b=random.randint(1,9)
answer=int(input(f'{a}*{b}는 무엇일까요?'))
```

4*1는 무엇일까요?4
정답입니다.

7*8는 무엇일까요?55
구구단 공부합시다.

동전 던지기 >

random모듈을 활용하여 0이나 1을 랜덤하게 생성하여 생성 숫자가 0 이면 '앞면입니다.' 1이면 '뒷면입니다.'를 출력하는 프로그램	**사용자에게 0 또는 1을 입력받아 컴퓨터가** random모듈을 이용하여 생성한 숫자와 같으면 '맞았습니다.' 같지 않으면 '틀렸습니다.'가 출력되는 프로그램

import random
print("동전 던지기 게임을 시작합니다.")
coin = random.randint(0,1)
if coin == 0 :
  print("앞면입니다.")
else :
  print("뒷면입니다.")
print('게임이 종료되었습니다.')
	import random
	print("동전 던지기 게임을 시작합니다.")
	coin = random.randint(0,1)
	
	if coin == 0 :
	  print("앞면입니다.")
	else :
	  print("뒷면입니다.")
	print('게임이 종료되었습니다.')

동전 던지기 게임을 시작합니다. 뒷면입니다. 게임이 종료되었습니다.	동전 던지기 게임을 시작합니다. 앞면(0)/뒷면(1)을 입력해 주세요> 1 맞았습니다. 뒷면입니다. 게임이 종료되었습니다.

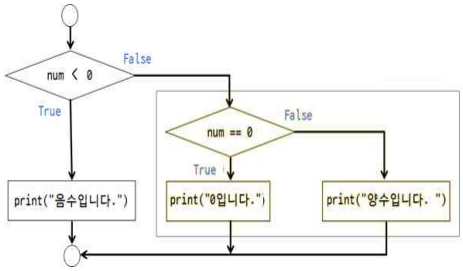
3. 중첩 if-else 문

- if, else 내부 코드에 다시 if문을 추가하여 중첩된 if문을 실행할수 있다.

```
if 조건식1 :
    if 조건식2 :
        조건식1과 2가 모두 참일 때 수행할 문장
    else :
        조건식1은 참, 조건식2가 거짓일 때 수행할 문장
else :
    if 조건식2 :
        조건식1은 거짓, 조건식2가 참일 때 수행할 문장
    else :
        조건식1과 조건식2가 모두 거짓일 때 수행할 문장
```

① 숫자를 입력받아 입력받은 값이 0보다 작으면 “음수입니다.” , 0이면 “0 입니다.” , 0보다 크면 “양수입니다” 를 출력하는 프로그램을 작성해 봅시다.

```
num = int( input('정수 입력> ') )
if  [          ] :
    print('음수입니다.')
else :
    if [          ] :
        print('0입니다.')
    else :
        print('양수입니다.')
```



정수 입력> 5
양수입니다.

정수 입력> 0
0입니다.

정수 입력> -2
음수입니다.

② 두 수를 입력받아 두 수가 같으면 '입력된 두 수는 같습니다.' 첫번째 입력된 수가 더 크면 '첫번째 수가 더 큽니다.' 두번째 입력된 수가 더 크면 '두번째 수가 더 큽니다.' 를 출력하는 프로그램을 작성해 봅시다.

```
num1 =
num2 =
```

첫번째 수 입력> 5
두번째 수 입력> 5
입력된 두 수는 같습니다.

첫번째 수 입력> 10
두번째 수 입력> 2
첫번째 수가 더 큽니다.

첫번째 수 입력> 3
두번째 수 입력> 7
두번째 수가 더 큽니다.

③ 만약 돈이 5000원 이상이 있으면 택시를 타고, 5000원은 없지만 2000원 이상 있으면 버스를 타고 2000원도 없으면 걸어 가라 는 메시지를 출력하는 프로그램을 작성해 봅시다.

```
money =
```

돈 입력> 7000
택시 타고 가라

돈 입력> 3000
버스 타고 가라

돈 입력> 600
걸어 가라

④ 은행에서 신용대출을 받기 위해서 연소득이40,000,000원 이상이고, 재직연수가2년 이상이어야 한다. 두 조건이 모두 만족하면 '대출자격있음' , 연소득이 40,000,000원 미만이면 '연봉 4000만원 이상 필요', 재직기간이 2년 미만이면 '재직기간 2년 이상 필요'를 출력하는 프로그램을 작성해 봅시다.

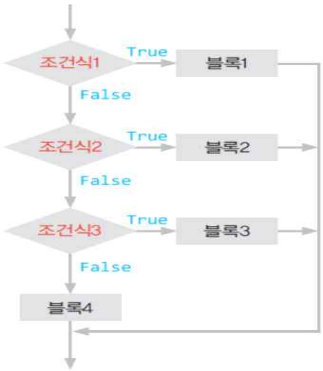
연소득 입력> 50000000
재직기간 입력> 4
대출자격 있음

연소득 입력> 20000
재직기간 입력> 5
연봉 4000만원 이상 필요

연소득 입력> 45000000
재직기간 입력> 1
재직기간 2년 이상 필요

4. if - elif - else문

- 세 개 이상의 조건을 연결해서 사용할때 활용
- if 조건문과 else구문 사이에 elif입력




```
if 조건식1 :
␣␣␣␣조건식1이참일 때 수행할 문장
elif 조건식2 :
␣␣␣␣조건식1은 거짓, 조건식 2가  참 일 때 수행할 문장
elif 조건식3 :
␣␣␣␣조건식1,2는거짓, 조건식3이  참일 때 수행할 문장
else :
␣␣␣␣조건식1,2,3이 모두 거짓일 때 수행할 문장
```

① 숫자를 입력받아 입력받은 값이 0보다 작으면 '음수입니다.' , 0이면'0 입니다.', 0보다 크면 '양수입니다'를 출력하는 프로그램을 작성해 봅시다. (중첩 if-else문을 if-elif-else문으로 변경해 봅시다.)

```
num = int( input('정수 입력> ') )
if num<0  :
    print('음수입니다.')
else :
    if num==0 :
        print('0입니다.')
    else :
        print('양수입니다.')
```

⟹

② 만약 돈이 5000원 이상이 있으면 택시를 타고, 5000원은 없지만 2000원 이상이 있으면 버스를 타고 2000원도 없으면 '걸어 가라'라는 메시지를 출력하는 프로그램을 작성해 봅시다.

```
money =
```

돈 입력>7000
택시 타고 가라

돈 입력>3000
버스 타고 가라

돈 입력>600
걸어 가라

③ 점수를 입력받아 90점 이상이면 A, 80점 이상이면 B, 70점 이상이면 C, 그 외에는 F를 출력하는 프로그램을 작성해 봅시다.

```
score =
```

당신의 점수는 몇점인가요?97
A

당신의 점수는 몇점인가요?85
B

당신의 점수는 몇점인가요?77
C

당신의 점수는 몇점인가요?34
F

④ 나이를 입력받아 나이에 따라 학령을 출력하는 프로그램을 작성해 봅시다.
미취학(~7), 초등학생(8~13), 중학생(14~16), 고등학생(17~19), 성인(20~)

```
age =
```

나이를 입력해 주세요>5
미취학

나이를 입력해 주세요>9
초등학생

나이를 입력해 주세요>18
고등학생

나이를 입력해 주세요>22
성인

① 로그인 프로그램

1단계 : 사용자로부터 아이디를 받아서 프로그램에 저장된 아이디와 일치하는지 여부를 출력하는 프로그램
(id : ilovepython)

```
id = "ilovepython"
s = input("아이디를 입력하시오: ")
```

```
아이디를 입력하시오: ilovepython
환영합니다.

아이디를 입력하시오: ilove
아이디를 찾을 수 없습니다.
```

2단계 : 사용자로부터 아이디와 비밀번호를 받아서 프로그램에 저장된 아이디와 패스워드가 일치하는지 여부를 출력하는 프로그램을(id : ilovepython , pw : 123456)

```
id = "ilovepython"
pawd = 123456
```

```
아이디를 입력하시오: ilovepython
비밀번호를 입력하시오: 123456
환영합니다.

아이디를 입력하시오: ilovepython
비밀번호를 입력하시오: 123
아이디와 비밀번호가 일치하지 않습니다.

아이디를 입력하시오: ilooo
비밀번호를 입력하시오: 123456
아이디를 찾을 수 없습니다.
```

3단계 : 사용자로부터 아이디와 비밀번호를 받아서 프로그램에 저장된 아이디와 패스워드가 일치하는지 여부를 출력하는 프로그램. 단 id가 admin일 경우 비밀번호와 상관없이 로그인 되도록 프로그램을 작성해 봅시다. (id : ilovepython , pw : 123456)

```
id = "ilovepython"
pawd = 123456
```

```
아이디를 입력하시오: ilovepython
비밀번호를 입력하시오: 123456
환영합니다.

아이디를 입력하시오: ilovepython
비밀번호를 입력하시오: 123
아이디와 비밀번호가 일치하지 않습니다.

아이디를 입력하시오: ilooo
비밀번호를 입력하시오: 123456
아이디를 찾을 수 없습니다.

아이디를 입력하시오: admin
비밀번호를 입력하시오: 11111
관리자가 로그인하였습니다.
```

② 윤년 판단 프로그램

입력된 연도가 윤년인지 아닌지를 판단하는 프로그램을 작성해 봅니다.
윤년은 아래의 조건 중 하나만이라도 만족하면 윤년이다.

(1조건) 400으로 나누어 떨어지는 연도는 윤년이다.
(2조건) 100으로 나누어 떨어지는 연도를 제외한 연도가 4로 나누어 떨어지면 윤년이다.

```
연도를 입력해 주세요 2000
2000은 윤년입니다.

연도를 입력해 주세요 2023
2023은 윤년이 아닙니다.
```

③ 사용자에게 현재 온도를 질문하고 온도가 25도 이상이면 반바지를 추천하고 25도 미만이면 긴바지를 추천하는 프로그램을 작성해 봅시다.

```
현재 온도 입력> 28.9
반바지를 입으세요.

현재 온도 입력> 10.2
긴바지를 입으세요.
```

④ 세 수를 입력 받아 가장 큰 수를 출력해 봅시다.

```
첫번째 숫자 > 10
두번째 숫자 > 30
세번째 숫자 > 50
가장 큰 수는 50

첫번째 숫자 > 89
두번째 숫자 > 105
세번째 숫자 > 20
가장 큰 수는 105
```

⑤ 사용자로부터 정수를 받아서 이 정수가 2의 배수 인지 3의 배수인지 출력하는 프로그램을 작성해 봅시다.

```
정수 입력 > 18
18은 2와 3의 배수입니다.

정수 입력 > 14
14은 2의 배수이고 3의 배수는 아닙니다.

정수 입력 > 9
9은 3의 배수이고 2의 배수는 아닙니다.

정수 입력 > 17
17은 2와 3의 배수가 아닙니다.
```

⑥ 주민등록번호가 입력되면 주민등록에 포함된 정보를 출력하는 프로그램을 작성해 봅시다.
주민등록번호 입력시 중간에 하이픈(-)을 포함해 입력합니다.

1	2	3	4	5	6	-	7	8	9	10	11	12	13
생년		월		일			성별						

1 : 1900년대 태어난 남자　　3 : 2000년대 태어난 남자
2 : 1900년대 태어난 여자　　4 : 2000년대 타어난 여자

```
주민등록번호 입력>981027-1032372
1998년 태어난 남자 입니다.

주민등록번호 입력>970522-2138599
1997년 태어난 여자 입니다.

주민등록번호 입력>060822-3099912
2006년 태어난 남자 입니다.

주민등록번호 입력>220910-4098765
2022년 태어난 여자 입니다.
```

[심화①] 윷의 결과값이 0과 1로 입력되었을 때 윷의 상태를 출력하는 코드를 작성해 봅시다. 0은 윷이 뒤집어지지 않은 상태이고 1은 윷이 뒤집어진 상태입니다.

```
도 : 윷 1개가 뒤집어진 상태  ex> 0010, 1000
개 : 윷 2개가 뒤집어진 상태  ex> 1010, 1100
걸 : 윷 3개가 뒤집어진 상태  ex> 1011, 1110
윷 : 윷 4개가 뒤집어진 상태  ex> 1111
모 : 윷이 하나도 뒤집어지지 않은 상태  ex> 0000
```

```
윷의 결과를 입력해 주세요>0100
도
윷의 결과를 입력해 주세요>1100
개
윷의 결과를 입력해 주세요>0111
걸
윷의 결과를 입력해 주세요>1111
윷
윷의 결과를 입력해 주세요>0000
모
```

[심화②] 두자리 숫자로 이루어진 복권이 있다. 사용자가 입력한 복권 번호 두자리가 모두 일치하면 100만원, 두자리 중 하나만 일치하면 50만원을 받는다. 하나도 일치하지 않으면 상금이 없다. 복권 당첨 번호는 난수로 생성하고 사용자의 입력에 따라 상금이 얼마인지 출력하는 프로그램을 작성해 봅시다.

```
복권번호를 입력하시오(0~99)45
당첨번호는 38입니다.
상금이 없습니다.
복권번호를 입력하시오(0~99)26
당첨번호는 23입니다.
상금 50만원!
복권번호를 입력하시오(0~99)23
당첨번호는 23입니다.
상금 100만원!
```

[심화③] 세 개의 직선길이를 입력으로 받습니다. 입력된 세개의 직선으로 삼각형을 만들 수 있는지 아닌지를 판단하는 프로그램을 작성해 봅시다.

<< 삼각형의 조건 : 가장 긴변의 길이가 나머지 두 변 길이의 합보다 작어야 한다. >>

```
직선1의 길이>3
직선2의 길이>2
직선3의 길이>4
삼각형 가능
```

```
직선1의 길이>5
직선2의 길이>9
직선3의 길이>3
삼각형 불가능
```

VI. 리스트

- 여러 개의 자료들을 모아서 하나의 묶음으로 저장하는 것
- 리스트에는 다양한 자료형들을 하나의 묶음으로 저장할 수 있다.
- 리스트는 **대괄호([ ])로** 데이터를 묶어서 표현하며 각 데이터들은 **쉼표(,)**로 구분하여 저장한다.

1. 리스트 만들기

```
리스트명 = [ ]                    # 대괄호를 사용하여 빈 리스트를 만든다.
리스트명 = [ 요소, 요소, 요소 ]    # 요소 3개 들어 있는 리스트를 만든다.
```

```
>>> alist = [ ]     # 빈 리스트 생성
>>> alist

>>> type(alist)

>>> alist = [ 1, 2, 3, 4, 5 ]    # 요소가 들어 있는 리스트 생성
>>> alist

>>> type(alist)
```

2. 리스트 접근하기

1) 인덱스

```
리스트명[인데스]
```

- 리스트는 인덱스를 이용하여 리스트의 각각의 데이터를 참조(활용)할수 있다.
- 리스트가 n개의 요소를 가지고 있으면 인덱스는 0 ~ n-1 로 번호를 붙인다.
- 역방향 인덱스는 마지막 요소부터 -1. -2 -3,... 순으로 번호르 붙인다.
- 리스트의 길이가 넘는 인덱스로 데이터에 접근하려고 하면 IndexError가 발생한다.

```
>>> alist = [ 1, 2, 3, 4, 5 ]
>>> alist[0]

>>> alist[1]

>>> alist[-1]

>>> alist[5]
```

	[0]	[1]	[2]	[3]	[4]
alist	1	2	3	4	5
	[-5]	[-4]	[-3]	[-2]	[-1]

- 인덱싱한 리스트는 하나의 값을 저장한 변수와 같은 역할을 한다.

```
>>> alist = [ 1, 2, 3, 4, 5 ]
>>> alist[0] + alist[4]

>>> alist[1] * alist[3]

>>> alist[-1] - alist[1]
```

2) 슬라이싱

리스트명[시작인덱스 : 끝인덱스]

- 리스트에서 연속적으로 위치한 요소들을 잘라내는 연산
- **시작인데스**의 데이터부터 **끝인덱스-1** 데이터까지의 리스트를 반환한다.

```
>>> alist = [ 1, 2, 3, 4, 5 ]
>>> alist[0:3]

>>> alist[1:4]

>>> alist[3:]

>>> alist[:3]
```

- 자신이 좋아하는 연예인 5명 이름이 들어있는 리스트를 만들어 봅시다.
  그 중 세번째에 저장되어 있는 연예인 이름을 출력해 봅시다.

```
>>> favolist =
```

3. 리스트 값 변경하기

리스트명[인덱스] = 값

- 변수의 값을 변경하는 것과 같이 리스트의 데이터를 변경할 수 있다.
- 리스트의 인덱스와 슬라이싱을 이용하여 리스트의 일부만을 변경할 수 있다.

```
>>> alist = [ 1, 2, 3, 4, 5 ]
>>> alist[0] = 10
>>> alist

>>> alist[-1]=50
>>> alist

>>> alist[1:3] = [ 20 , 30 ]
>>> alist
```

4. 리스트에 항목 추가하기

리스트명.append(요소) #리스트의 맨 뒤에 요소 추가
리스트명.insert(인덱스번호 , 요소) # 리스트의 해당 인덱스번호 위치에 요소 추가

```
>>> alist=[]
>>> alist.append(1)
>>> alist

>>> alist.append(22)
>>> alist

>>> alist.append(333)
>>> alist
```

```
>>> alist.insert(1, 10 )
>>> alist


>>> alist.insert(0,-5)
>>> alist
```

5. 리스트 항목 삭제하기

1) 인덱스를 이용하여 제거하기

```
del 리스트명[인덱스]        # 리스트에서 해당 인덱스 삭제
리스트명.pop(인덱스)        # 리스트에서 해당 인덱스의 값을 반환 후 삭제
```

```
>>>alist = [ 1, 2, 3, 4, 5 ]
>>>del alist[1]
>>>alist
```

```
>>> value=alist.pop(0)
>>> alist

>>> value
```

2) 요소값을 이용하여 제거하기

```
리스트명.remove(요소)      # 리스트에서 해당 요소를 삭제
                           # 해당 '요소값'이 여러 개일 경우 가장 앞에 있는 것만 삭제
```

```
>>>alist = [ 10, 20, 30, 40, 30 ]
>>>alist.remove(20)
>>>alist

>>>alist.remove(30)
>>>alist
```

3) 항목 모두 제거하기

```
리스트명.clear()    # 리스트 내부의 요소를 모두 삭제
```

```
>>>alist = [ 1, 2, 3, 4, 5 ]
>>>alist.clear()
>>>alist
```

◆ 리스트 연습하기

① 리스트를 출력하고 리스트 안의 숫자들의 합계를 출력하는 프로그램을 작성해 봅시다.

```
alist = [ 11, 22, 33, 44 ]
```

[11, 22, 33, 44]
110

② '사과', '바나나', '포도', '수박' 과일이 들어 있는 리스트를 생성하고 실행 결과가 다음과 같이 나오도록 코드를 완성해 봅시다.

```
a = ['사과', '바나나', '포도', '수박' ]
print(                    )
print(                    )
print(                    )
print(                    )
```

['사과', '바나나']
['바나나', '포도', '수박']
['바나나', '포도']
['사과', '바나나', '포도', '수박']

③ 빈 리스트 생성 후 친구 5명의 이름을 각각 입력받아 리스트에 저장한 후 출력하는 프로그램을 작성해 봅시다.

```
friend = [ ]
name = input('친구 이름 입력: ')
```

```
친구 이름 입력: 홍길동
친구 이름 입력: 유재석
친구 이름 입력: 지석진
친구 이름 입력: 김종국
친구 이름 입력: 송지효
['홍길동', '유재석', '지석진', '김종국', '송지효']
```

④ 메뉴만들기

[1단계] 빈 리스트에서 시작해서 다음과 같이 식당 메뉴를 추가한 후 메뉴를 출력합니다.

```
menu = [ ]
```

```
['치킨', '피자', '샐러드', '스테이크']
```

[2단계] 앞의 메뉴에서 '스테이크'를 '감자튀김'으로 변경 후 출력해 봅시다. (append 함수 사용)

```
['치킨', '피자', '샐러드', '스테이크']
변경=> ['치킨', '피자', '샐러드', '감자튀김']
```

[3단계] 앞의 메뉴에서 '샐러드' 메뉴를 삭제한 후 메뉴와 샐러드 삭제 메시지를 출력해 봅시다.

```
['치킨', '피자', '샐러드', '스테이크']
변경=> ['치킨', '피자', '샐러드', '감자튀김']
삭제=> ['치킨', '피자', '감자튀김']
샐러드 메뉴가 삭제 되었습니다.
```

[심화⑤] 수학문제 만들기

숫자 10개를 저장하고 있는 리스트를 작성하고 무작위로 숫자 두 개를 선택하여 사용자에게 더하기를 질문하는 프로그램을 작성해 봅시다. 문제에 대해 답이 맞으면 '맞았습니다.' 틀리면 '틀렸습니다.'메시지가 나오도록 합니다.

```
import random
number = [ 10, 20, 33, 40, 55, 60, 77, 80, 90, 100 ]
```

```
60 + 100 = ?160
맞았습니다.
80 + 33 = ?100
틀렸습니다.
```

6. 리스트 연산자

1) 연결(+) 연산자

```
리스트명 + 리스트명                    # 두 리스트를 연결하여 하나의 리스트를 만든다.
리스트명[인덱스] + 리스트명[인덱스]    # 두 부분 리스트를 연결하여 하나의 리스트를 만든다.
```

```
>>> a_list= [ 1, 2, 3 ]
>>> b_list= [ 4, 5, 6 ]
>>> a_list + b_list

>>> a_list

>>> b_list
```

2) 반복(*) 연산자

```
리스트명 * 숫자      # 리스트를 숫자만큼 반복하는 리스트를 만든다.
```

```
>>> a_list= [ 1, 2, 3 ]

>>> a_list *  3

>>> a_list
```

3) 멤버쉽(in / not in) 연산자

```
요소 in 리스트명(문자열)          #요소가 리스트에 있는지 판단
요소 not in 리스트명(문자열)      #요소가 리스트에 없는지 판단
```

- 주어진 값이 리스트나 문자열에 속하여 있는지 판단한다.
- in 연산자 : 주어진 값이 리스트에 있으면 True, 리스트에 없으면 False 반환한다.
- not in 연산자 : 주어진 값이 리스트에 없으면 True , 리스트에 있으면 False 반환한다.

```
>>>  alist= [ 1, 2, 3, 4, 5]
>>> 2 in alist

>>> 9 in alist

>>> 4 not in alist

>>> 0 not in alist
```

```
>>>  alist= [ 'abc', 'def', 'ghi', 'jkl' ]
>>> 'abc' in alist

>>> 'a' in alist

>>> 'def' not in alist

>>> 'k' not in alist
```

```
>>> mss = 'Have a Good Day'
>>> 'a' in mss

>>> 'Go' in mss

>>> ' ' in mss

>>> 'A' in mss
```

◆ 리스트 연산자 연습하기

① a, b 두 리스트의 연산 결과가 다음과 같도록 프로그램을 작성해 봅시다.

```
a = [ 7, 6, 5, 4 ]
b = [ 1, 2, 3 ]
```

```
[1, 2, 3, 7, 6, 5, 4]
[7, 6, 5, 4, 1, 2, 3, 1, 2, 3]
[1, 2, 7, 6, 7, 6, 7, 6]
```

② 1이 존재여부를 판단하여 결과가 다음과 같도록 프로그램을 완성해 봅시다.

```
a = [ 1, 2, 3, 4, 5 ]
if [          ] :
    print('1이 존재합니다')
else:
    print('1이 존재하지 않습니다')
```

```
1이 존재합니다.
```

③ 0이 존재여부를 판단하여 결과가 다음과 같도록 프로그램을 완성해 봅시다.

```
a = [ 1, 2, 3, 4, 5 ]
if [          ] :
    print('0이 존재하지 않습니다')
else:
    print('0이 존재합니다')
```

```
0이 존재하지 않습니다.
```

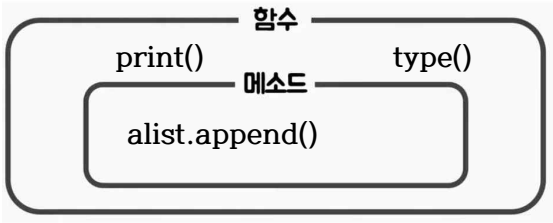
④ 빈 리스트를 만든 후 랜덤으로 1 ~ 100 사이 숫자 5개를 추가합니다. 숫자에 50이 포함되어 있는지 확인하는 프로그램을 작성해 봅시다.

```
import random
alist = [ ]
num = random.randint(1, 100)
```

```
50이 존재하지 않습니다.
리스트 숫자: [35, 69, 95, 45, 74]
```

7. 리스트의 함수와 메소드

- **함수** : 특정 작업을 수행하기 위해 설계된 기능 , 직접 호출한다.
- **메소드** : 특정 객체에서만 수행가능하게 설계된 함수, 점 표기법으로 호출한다.



1) 리스트 메소드

(1) 요소 위치찾기

리스트 . index(요소) # 리스트에서 요소의 위치(인덱스)를 반환
 # 중복된 값이 있을 경우 가장 최소의 위치 반환

```
>>>alist = [ 'No', 'pain', 'No', 'gain']
>>>alist.index('gain')

>>>alist

>>>alist.index('No')
```

(2) 리스트 특정 요소의 개수 세기

리스트.count(요소) # 리스트 안에 요소의 갯수를 반환

```
>>>alist = [ 'No', 'pain', 'No', 'gain']
>>>alist.count('gain')

>>>alist.count('No')
```

(3) 리스트 요소 정렬하기

리스트.sort() # 리스트의 요소를 오름차순으로 정렬
리스트.sort(reverse=True) # 리스트의 요소를 역순으로 정렬

```
>>>alist = [ 5,3,2,4,1]
>>>alist.sort()
>>>alist


>>>alist.sort(reverse=True)
>>>alist
```

◆ **리스트 메소드 연습하기**

① 프리미어리그의 순위대로 팀명이 저장되어 있는 리스트에서 맨유가 몇번째 순위인지 출력하는 프로그램을 작성해 봅시다.

```
premire=['맨시티', '아스널', '뉴캐슬', '맨유', '리버플' ]
```

4 위

② 주사위를 10번 던져 나온 숫자를 리스트에 추가합시다. 10번 중 6이 나온 횟수를 출력하고 리스트를 오름차순으로 정렬하여 출력하는 프로그램을 작성해 봅시다.

```
import random
alist= [ ]
dice = random.randint(1, 6)
```

2 번
[1, 2, 3, 3, 4, 4, 4, 5, 6, 6]

2) 리스트에 적용할수 있는 함수

- len(리스트) : 리스트 요소의 개수를 반환한다.
- max(리스트) : 리스트 요소 중 최대값을 반환한다.
- min(리스트) : 리스트 요소 중 최소값을 반환한다.
- sum(리스트) : 리스트 요소의 합을 반환한다.

```
>>>alist = [ 200, 300, 700, 400, 100 ]
>>>len( alist )

>>>max( alist )

>>>min( alist )

>>>sum( alist )
```

◆ **연습하기**

① 서울, 인천, 부산, 대전의 인구가 들어있는 변수를 이용하여 리스트를 생성하고
인구가 가장 많은 도시, 가장 적은 도시, 네 도시 인구의 평균을 출력하는 프로그램을 작성해 봅시다.

```
seoul = 9876
incheon = 2954
busan=3441
daejun=2391
```

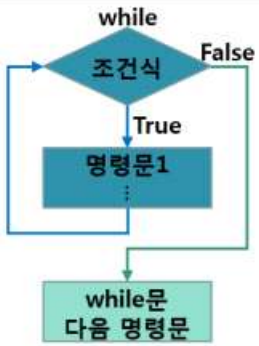
인구 최대 도시: 9876
인구 최소 도시: 2391
평균 인구수: 4665.5

VII. 반복구조

- 같은 문장을 여러번 실행할 때 사용함.
- 특정 조건이 참일동안이나 특정 횟수만큼 반복 함.
- 조건에 '항상 참인 조건'을 사용하면 무한루프 상태에 빠질 수 있음.

1. while문

```
while 조건식 :
    ␣␣␣␣조건이 참일 때 수행할 문장1
    ␣␣␣␣조건이 참일 때 수행할 문장2
```

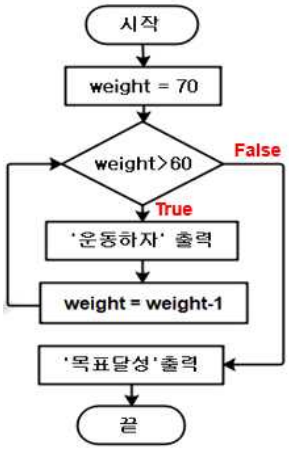


- while 명령 뒤에 오는 조건식이 참(True)일 동안 내부 코드 실행
- 조건식이 참이면 내부 문장을 수행한 후 다시 조건 검사를 함.
- 조건이 거짓이면 반복을 종료함.
- **조건식에 변수**를 사용할 경우 변수는 **초기값**을 가지고 있어야 하며, 조건식에서 사용된 변수는 **반복문 안에서 그 값을 변경** 할 수 있다.

1) 반복문 안에서 산술연산을 사용하여 변수값 변경

- 10kg이 빠질 때까지 '운동하자'를 반복하는 프로그램

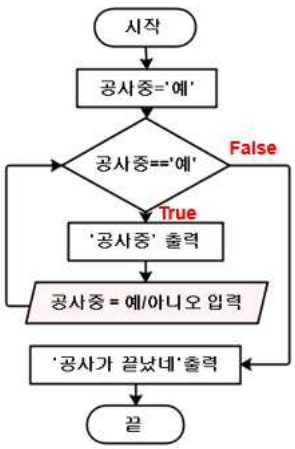
```
weight = 70
while                  :
        print('운동하자')

print('목표달성!')
```



2) 반복문 안에서 입력함수(input())를 사용하여 변수값 변경

- 공사가 끝날 때까지 계속 반복하는 프로그램.

```
공사중 = '예'
while 공사중 == '예':
    print('공사중')
    공사중= input( " 공사중입니까?(예/아니오) " )
print("공사가 끝났네")
```

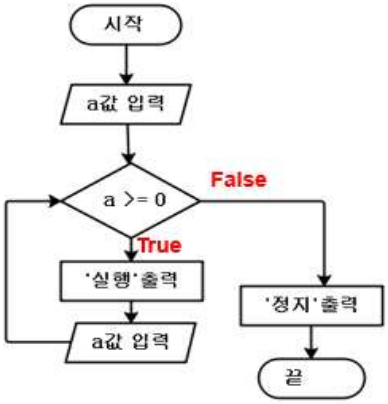


◆ while 반복문 연습하기1

① 양수가 입력되는 동안 반복하는 프로그램
a의 값을 입력받아 a가 0보다 같거나 크면 '실행'을 출력하고 다시 입력받는다.
a가 0보다 작으면 '정지'를 출력하고 프로그램을 종료한다.

```
a = int( input('a:'))
while a  >= 0  :
    print( '실행')
    a = int( input('a:'))
print('정지')
```

```
a:5
실행
a:8
실행
a:-2
정지
```



② 다음과 같이 a의 초기값을 5로 설정하고 프로그램을 실행했을 때
'실행'은 몇 번 출력 될까요? ( ) 번

```
a = 5
while a >= 0 :
        print('실행' )
        a = a-1
print('정지!')
```

a	a >= 0	while문 실행	a=a-1
5	참	출력	4

◆ **도전하기**

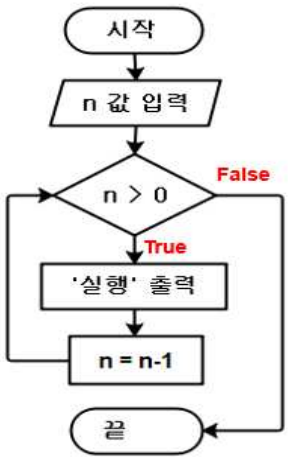
① while 반복문을 활용하여 '안녕하세요'를 5번 출력하는 프로그램을 작성해 봅시다.

```
a = 
while                :
    print('안녕하세요')
```

② 숫자를 입력받아 그 수만큼 '실행'을 출력하는 프로그램을 작성해 봅시다.

```
n = int( input('n:'))
while
```

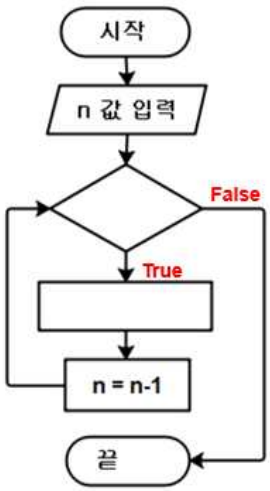
n:4
실행
실행
실행
실행



③ 숫자를 입력받아 그 수만큼 카운트다운 하는 프로그램을 작성해 봅시다.
0이 되면 카운트다운은 정지합니다.

```
n = int( input('n:'))
while
```

n:5
5
4
3
2
1
0

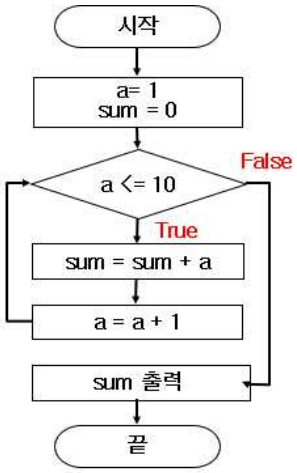


④ y가 입력될 때 까지 계속 입력받는 문자열을 출력하는 프로그램을 작성해 봅시다.

y 입력시 종료>조건이
조건이 입력시 종료>참일동안
참일동안 입력시 종료>반복해요
반복해요 입력시 종료>y

◆ while 반복문 연습하기2

① 1부터 10까지의 숫자의 합(1+2+...+10)을 계산하는 알고리즘을 보고 프로그램을 작성해 봅시다.



sum	a	a<=10
0	1	True
0+1	2	True
0+1+2	3	True
0+1+2+3	4	True
0+1+2+3+4	5	True
0+1+2+3+4+5	6	True
0+1+2+3+4+5+6	7	True
0+1+2+3+4+5+6+7	8	True
0+1+2+3+4+5+6+7+8	9	True
0+1+2+3+4+5+6+7+8+9	10	True
0+1+2+3+4+5+6+7+8+9+10	11	False

```
sum = 0
a = 1

while  a <= 10  :
    sum = sum + a
    a = a+1

print(sum)
```

추가미션> N 값을 입력받아 1부터 N까지 합계를 구하는 프로그램을 작성해 봅시다.

```
sum = 0
a = 1
[                    ]
while [                    ] :
    sum = sum + a
    a = a+1
print(sum)
```

```
N값 입력: 100
5050
```

② while문 안에 조건문 >

1부터 100까지 순서대로 숫자를 출력한다. 단, 3의 배수일 때만 '짝'을 출력하는 프로그램을 작성해 봅시다.

```
a=1
while a < = 100 :
    if [                ] :
        [                ]
    else ;
        print(a, end = ' ')
    a = a+ 1
```

```
1 2 짝 4 5 짝 7 8 짝 10 11 짝 13 14 짝 16
17 짝 19 20 짝 22 23 짝 25 26 짝 28 29 짝
31 32 짝 34 35 짝 37 38 짝 40 41 짝 43 44
짝 46 47 짝 49 50 짝 52 53 짝 55 56 짝 58
59 짝 61 62 짝 64 65 짝 67 68 짝 70 71 짝
73 74 짝 76 77 짝 79 80 짝 82 83 짝 85 86
짝 88 89 짝 91 92 짝 94 95 짝 97 98 짝 100
```

** print()함수 끝에 end=' ' 를 추가하면 줄바꿈을 하지 않고 한칸 띄움을 출력할 수 있다.*

③ while문과 리스트 >

리스트에 들어 있는 숫자 2를 모두 제거하는 프로그램을 작성해 봅시다.

```
alist = [ 1, 2, 1, 2 ]
while [                    ] :
    alist.remove(2)
print(alist)
```

```
[1, 1]
```

④ while문과 리스트 >

리스트의 요소를 모두 출력하는 프로그램을 작성해봅시다.

```
bts= [ '정국','뷔','지민','슈가','진','RM','제이홉']
i=0
while                              :
    print(   bts[i]    )
    i+=1
```

```
정국
뷔
지민
슈가
진
RM
제이홉
```

◆ **도전하기**

① 구구단 출력 >

구구단 중에서 원하는 단을 입력받아 반복문을 이용하여 출력해 봅시다.

```
dan = int(input("원하는 단은: "))
i = 1
while
```

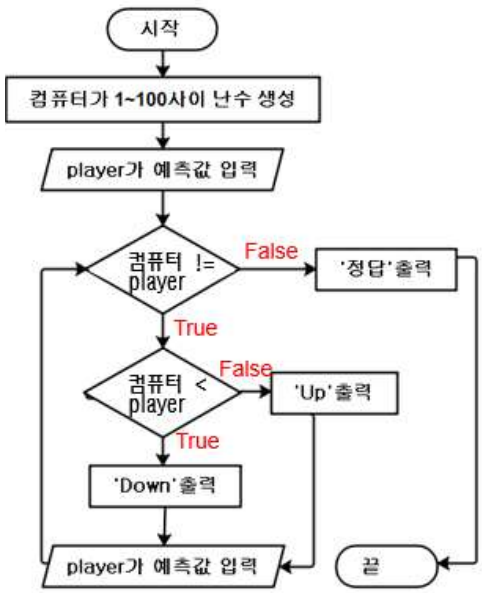
```
원하는 단은: 3
3 * 1 = 3
3 * 2 = 6
3 * 3 = 9
3 * 4 = 12
3 * 5 = 15
3 * 6 = 18
3 * 7 = 21
3 * 8 = 24
3 * 9 = 27
```

② Up & Down 게임 만들기 >

컴퓨터가 랜덤 숫자를 생성하고 사용자가 그 숫자를 맞히는 프로그램을 작성해 봅시다.

[게임규칙]

컴퓨터는 1~100 사이의 퀴즈 숫자를 생성
사용자는 숫자를 입력
입력한 숫자가 퀴즈 보다 크면 Down을 출력
입력한 숫자가 퀴즈 보다 작으면 UP을 출력
입력한 숫자가 퀴즈와 같으면 정답!을 출력
답을 맞출 때 까지 반복




```
import random
com= random.randint(1,100)
player= int(input("1부터 100사이의 숫자야, 맞춰봐>  "))
```

```
1부터 100사이의 숫자야, 맞춰봐> 40
UP
1부터 100사이의 숫자야, 맞춰봐> 70
UP
1부터 100사이의 숫자야, 맞춰봐> 90
DOWN
1부터 100사이의 숫자야, 맞춰봐> 80
UP
1부터 100사이의 숫자야, 맞춰봐> 85
UP
1부터 100사이의 숫자야, 맞춰봐> 88
UP
1부터 100사이의 숫자야, 맞춰봐> 89
정답!
```

③ 타자 게임 만들기>

단어 리스트에서 임의로 단어를 뽑아 단어를 똑같이 입력 했을 때 통과하는 타자 게임을 만들어 봅시다.
5번을 통과하면 소요된 시간이 출력되고 프로그램은 종료 됩니다.

*** random.choice(리스트)**
: 리스트의 요소 중 임의로 하나를 선택하는 함수

```
>>> alist= ['a', 'b', 'c', 'd']
>>> import random
>>> random.choice( alist )
```

```
import random
import time

wlist = ['cat','dog','fog','monkey','mouse','panda','frog','snake','wolf']
     #리스트의 단어는 자신이 원하는 것으로 변경해봅시다. 문장도 가능합니다.
print('[타자게임]준비되면 엔터!')
input()                                  # 사용자가 엔터를 누를 때까지 기다립니다.
start = time.time()                      # 시작 시간을 기록

com=random.choice(wlist)                 # 단어리스트에서 임의로 하나 뽑기
n=1                                      # 문제번호
while n <= 5 :                           # 문제를 다섯 번 반복
    print(com)                           # 문제를 출력
    player=input()                       # 사용자 입력 받기
    if                                   :   # 문제와 입력이 같을 때
        print('통과!')                       # "통과!"출력
        n = n + 1                            # 문제 번호 1 증가
        com=random.choice(wlist)             # 새 문제 뽑기
    else:
        print('오타! 다시 도전!')

end = time.time()                        # 끝난 시간 기록
wt = end - start                         # 실제 걸린 시간 계산
print(f'타자 시간 : {wt:.2f} 초')          # 소수점 둘째자리 까지 표기
```

```
[타자게임]준비되면 엔터!

panda
panda
통과!
wolf
wo
오타! 다시 도전!
wolf
wolf
통과!
panda
panda
통과!
cat
cat
통과!
cat
cat
통과!
타자 시간 : 21.78 초
```

추가미션> 동일한 단어가 중복되어 나오지 않도록 수정해 봅시다. *hint> 리스트명.remove(요소)*

3) 무한반복

- while 문의 조건식에 True를 넣으면 항상 True 이므로 무한반복 프로그램이 만들어 진다.

```
a = 1
while True :
    print( a,'번 반복', end=' ')
    a = a + 1
```

while의 조건식에 1 을 넣으면 어떻게 될까요?

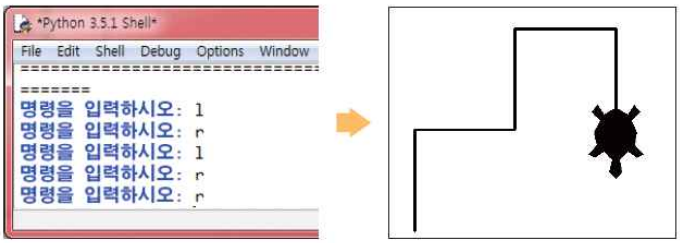
- 빨강, 주황, 초록 신호들이 2초 간격으로 바뀌는 무한반복 프로그램을 작성해 봅시다.

```
import time

while   True  :
    print('빨강')
    time.sleep(2)
    print('주황')
    time.sleep(2)
    print('초록')
    time.sleep(2)
```

time 모듈의 sleep(초) 는 초 시간동안 프로그램을 정지시키는 명령을 수행한다.

[turtle] while문 이용하여 거북이 제어하기

파이썬 쉘에서 “l” 을 입력하면 거북이가 왼쪽으로 100픽셀 이동하고
“r” 을 입력하면 거북이가 오른쪽으로 100픽셀 이동하는 프로그램을 작성해 봅시다.

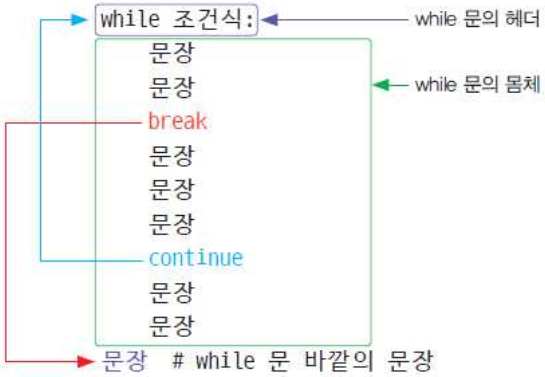



```
import turtle

t = turtle.Turtle()
t.shape("turtle")
t.shapesize(3, 3)       # 거북이를 3배 확대한다.

while True:
    cmd  =  input("명령을 입력하시오: ")
    if [          ]  : # 사용자가 "l"을 입력하였으면
        t.left(90)
        t.forward(100)
    if [          ]  : # 사용자가 "r"을 입력하였으면
        [          ]
```

추가미션 > 현재 위치에서 직진하는 명령(s)을 추가해 봅시다.

4) 반복문 제어

- **break 문** : 반복문을 벗어날 때 사용
- **continue 문** : 현재 반복을 생략하고 다음 반복으로 넘어갈 때 사용



break 연습 > 1부터 숫자를 출력하는 프로그램에서 a의 값이 100보다 크면 반복문을 종료

```
a = 1
while True :
    if a > 100 :
        break
    print(a, end=' ')
    a = a+1
```

continue 연습 > 1부터 100까지 출력하는 프로그램에서 짝수는 출력에서 제외하기

```
a = 0
while True :
    a = a+1
    if a > 100 :  break
    if a % 2 == 0 : continue
    print(a, end=' ')
```

① 숫자를 입력받아 입력받은 수들의 합을 출력하는 프로그램을 작성합니다.
단, 그 합이 1000이상이 되면 '1000이 넘었어'를 출력하고 종료하는 프로그램을 작성해 봅시다.

```
s = 0
while True:
    n = int(input( '숫자입력:' ))
    s = s + n
    print(s)
```

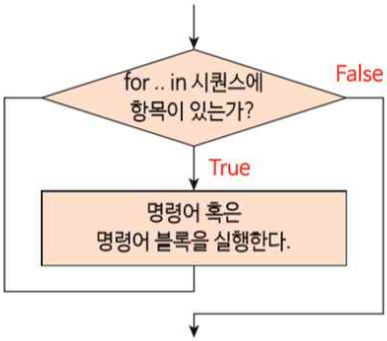
```
숫자입력:40
40
숫자입력:100
140
숫자입력:700
840
숫자입력:200
1040
1000이 넘었어!
```

② 위의 프로그램에서 입력된 숫자가 홀수일 경우에는 '홀수는 pass!'메시지를 출력하고 그 수를 더하지 않는 프로그램을 작성해 봅시다.

```
숫자입력:100
100
숫자입력:11
홀수는 pass!
숫자입력:19
홀수는 pass!
숫자입력:900
1000
숫자입력:1
홀수는 pass!
숫자입력:2
1002
1000이 넘었어!
```

2. for문

```
for 변수 in 시퀀스 :
    ␣␣␣␣반복하여 실행할 문장1
    ␣␣␣␣반복하여 실행할 문장2
```



- 특정 범위만큼 문장을 반복하여 실행.
- 시퀀스에 포함된 원소들의 개수만큼 for문 내부의 문장을 반복해서 수행함.
- 시퀀스에는 문자열, 리스트, 숫자 범위 등이 올 수 있다.

1) 시퀀스가 문자열인 경우

- 문자열의 문자를 하나씩 변수에 대입하며 반복 문장 실행
- 더이상 변수에 대입할 문자가 없으면 반복문 종료

```
for 변수 in 문자열 :
    ␣␣␣␣반복하여 실행할 문장1
    ␣␣␣␣반복하여 실행할 문장2
```

```
my_string= "프로그램"
for a in my_string:
    print(a)
```

for-문자열> 문자열을 입력받아 각각의 문자를 순서대로 출력하는 프로그램을 작성해 봅시다.

```
s = input('문자열 입력>')
for                          :
```

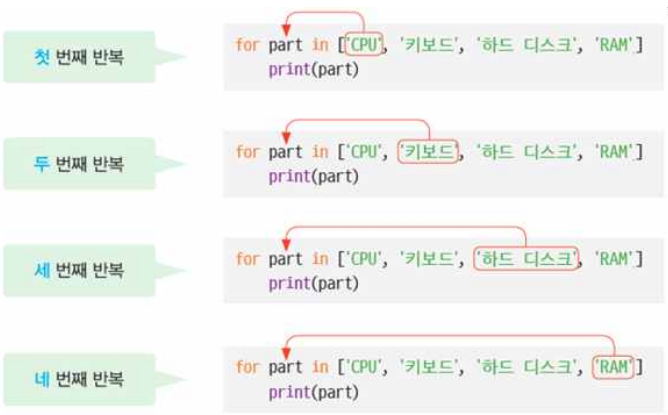
```
문자열 입력>파이썬
파
이
썬
```

2) 시퀀스가 리스트인 경우

- 리스트의 요소를 하나씩 변수에 대입하며 반복 문장 실행
- 더이상 변수에 대입할 요소가 없으면 반복문 종료

```
for 변수 in 리스트 :
    ␣␣␣␣반복하여 실행할 문장1
    ␣␣␣␣반복하여 실행할 문장2
```

```
hardware = ['cpu', '키보드','하드 디스크', 'RAM']
for part in hardware :
    print(part)
```



for-리스트> 리스트의 각 점수를 출력해 봅시다.

```
score = [ 90, 60, 50, 40, 10, 70 ]
for
```

```
90
60
50
40
10
70
```

추가미션> 60점 이상인 점수만 출력되도록 프로그램을 작성해 봅시다.

```
90
60
70
```

3) 시퀀스가 숫자범위(range)인 경우

- range()함수가 생성한 수의 집합을 차례대로 변수에 대입하며 반복문을 실행

```
for 변수 in range(초기값, 최종값, 증가값) :
    반복하여 실행할 문장1
    반복하여 실행할 문장2
```

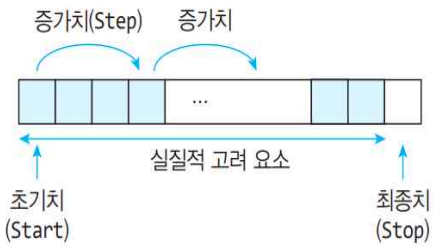

```
for i in range(5) :
    print( i )
```

*** range() 함수**

```
range( 초기값, 최종값, 증가값 )   # 초기값, 초기값+증가값, 초기값+증가값*2....., 최종값-1
```

- 숫자 범위(start ~ stop-1) 내에서 규칙적으로 증가하는 수들의 집합을 만들어 주는 함수
- 초기값부터, 초기값에 증가값을 더해서 최종값 보다 하나 작은 수까지의 수들의 집합 생성
- 초기값은 생략될 수 있으며 생략시 초기값은 0 이다.
- 증가값은 생략될 수 있으며 생략시 증가값은 1 이다.

```
>>> list( range(8) )
>>> list( range(2,8) )
>>> list( range(0, 8, 2) )
>>> list( range(0, 8, 3) )
>>> list( range(6, 0, -1) )
```



for-range> range()함수를 활용하여 1~100 수 중에서 홀수만 출력하는 프로그램을 작성해 봅시다.

```
for                          :
    print( i , end=' ')
```

```
1 3 5 7 9 11 13 15 17 19 21 23 25 27 29 31 33
35 37 39 41 43 45 47 49 51 53 55 57 59 61 63
65 67 69 71 73 75 77 79 81 83 85 87 89 91 93
95 97 99
```

for-range> 원하는 단을 입력받아 구구단을 출력하는 프로그램을 작성해 봅시다.

```
dan = int(input("원하는 단은: "))
for                              :
    print(f'{dan} * {i} = {dan*i}' )
```

```
원하는 단은: 3
3 * 1 = 3
3 * 2 = 6
3 * 3 = 9
3 * 4 = 12
3 * 5 = 15
3 * 6 = 18
3 * 7 = 21
3 * 8 = 24
3 * 9 = 27
```

for-range> 1부터 100까지의 합계를 구하는 프로그램을 for문을 이용하여 작성해 봅시다.

<while문을 이용한 합 구하기>

```
sum = 0
a = 1
while  a <= 100  :
    sum = sum + a
    a = a+1
print(sum)
```

< for문을 이용한 합 구하기>

```

```

```
5050
```

도전하기> for문 활용

① 문자열을 입력받아 입력받은 문자열에 문자 'a'가 몇 개 있는지 확인하는 프로그램을 작성해 봅시다.

```

```

```
language
2
```

② 임이의 정수를 입력받아 0부터 그 수까지 순서대로 출력하는 프로그램을 작성해 봅시다.

```
n = int( input() )
```

```
4
0 1 2 3 4
```

③ 임이의 정수를 입력받아 그 수만큼 '*'을 출력하는 프로그램을 작성해 봅시다.

```

```

```
6
******
```

④ 리스트의 인덱스 번호와 요소를 쌍으로 출력하는 프로그램을 작성해 봅시다.

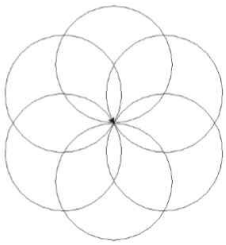
```
array = [ 273, 74, 103, 57, 52]
```

```
0 273
1 74
2 103
3 57
4 52
```

◆ for문을 활용한 도형그리기

① 60도 만큼 회전시키면서 6개의 원 그리기

```
import turtle
t = turtle.Turtle()
for count in range( □ ):
    t.circle(100)
    □
```



② 한변의 길이가 100인 사각형 그리기

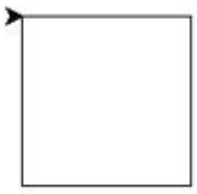
<순차구조>

```
import turtle
t = turtle.Turtle()

t.forward(100)
t.right(90)
t..forward(100)
t.right(90)
t.forward(100)
t.right(90)
t.forward(100)
t.right(90)
```

⇨

<반복구조>

```
import turtle
t = turtle.Turtle()

for i in range( □ ):
    □
```

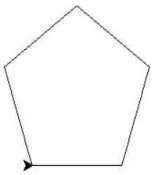


③ 사용자로부터 정수 n을 입력 받아서 n 각형을 그리는 프로그램을 작성해 봅시다.

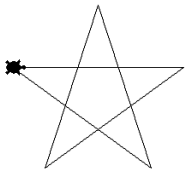
```
import turtle
t = turtle.Turtle()
n = int( input('정수입력>') ) )
```

```
*hint
for i in range( □ ):
    t.forward(100)
    t.left( □ )
```

정수입력>5



④ 한변의 길이가 100인 별 그리기 (for문 활용) *hint : 144도 회전시 별이 그려집니다.*



[심화①] 임이의 정수를 입력받아 팩토리얼을 계산하는 프로그램을 작성해 봅시다.

팩토리얼은 1부터 입력받은 수까지의 정수를 모두 곱한 것을 의미한다.(n!=1 x 2 x 3 x … x (n-1) x n)

```
n = int( input('정수입력>') ) )
```

정수입력>5
5!은 120

4) for문 제어하기

(1) break

- 반복문을 벗어나서 반복문 아래의 문장 실행
- 특정 조건일 때 반복문 종료

```
for i in range(5):
    if i==2 :
        break
    print(i)
print("for문 종료후 문장")
```

```
0
1
for문 종료후 문장
```

break 연습> 1~100까지 더하는 프로그램에서 합계가 1000이상이 되는 시작 지점이 어디인지 알려주는 프로그램

```
sum = 0
for i in range(1,101):
    sum = sum + i
    if [          ] :
        [          ]
print("1~100의 합에서 최초로 1000이 넘는 위치 : ", i )
```

1~100의 합에서 최초로 1000이 넘는 위치 : 45

(2) continue

- 반복문의 해더로 이동
- 해당 아이템을 건너뛰고 싶을 때 사용

```
for i in range(5):
    if i==2 :
        continue
    print(i)
print("for문 종료후 문장")
```

```
0
1
3
4
for문 종료후 문장
```

for 연습> 1~100사이의 합을 구하되 3의 배수를 제외하고 더하는 프로그램 (1+2+4+5++….)

```
sum = 0
for i in range(1,101):
    if [          ] :
        [          ]
    sum = sum + i
print("1~100의 합(3의배수제외) : ", sum )
```

1~100의 합(3의배수제외) : 3367

도전하기> 1~100사이의 홀수의 곱을 구하되 곱한 결괏값이 100 이상이되면 종료하는 프로그램

```
mult = 1
for a in range(1, 100) :
    [                  ]   #짝수이면 반복문 처음으로
    print( f'{mult} * {a} = {mult*a}')
    mult = mult * a
    [                  ]   #곱한 결괏값이 100이상이면
                           #반복문 종료
```

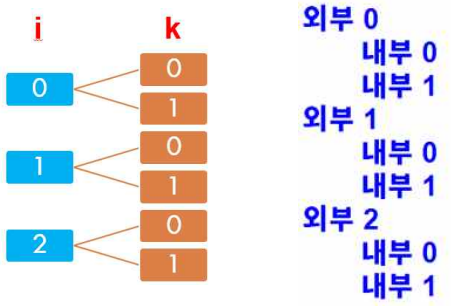
```
1 * 1 = 1
1 * 3 = 3
3 * 5 = 15
15 * 7 = 105
```

3. 중첩 반복문

- 한 반복문 안에 다른 반복문이 포함되어 있는 것

- for문 내부에 다른 for문이 들어 있는 형태

```
for i in range(0,3,1):
    print("외부", i )
    for k in range(0,2,1):
        print("        내부",k )
```
←내부반복문



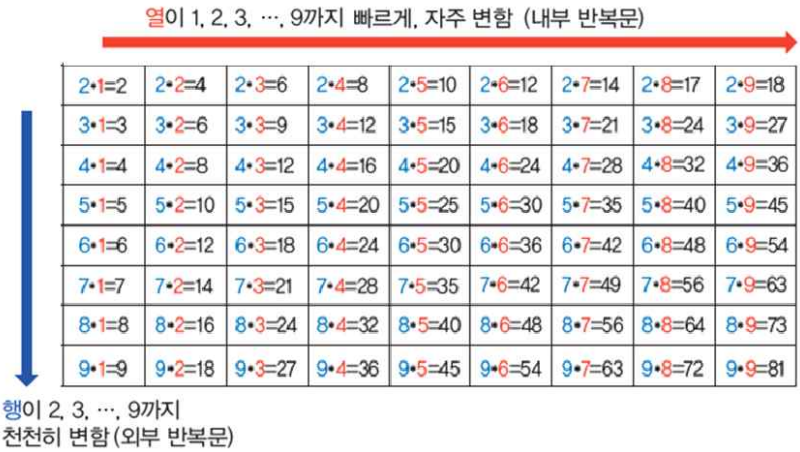
- **주사위>** 주사위 2개를 던져서 나올수 있는 모든 경우의 수 출력하기

```
for i in range(      ):
    for k in range(      ):
        print( f'({i} ,{k})' , end=' ')
    print()
```

(1, 1) (1, 2) (1, 3) (1, 4) (1, 5) (1, 6)
(2, 1) (2, 2) (2, 3) (2, 4) (2, 5) (2, 6)
(3, 1) (3, 2) (3, 3) (3, 4) (3, 5) (3, 6)
(4, 1) (4, 2) (4, 3) (4, 4) (4, 5) (4, 6)
(5, 1) (5, 2) (5, 3) (5, 4) (5, 5) (5, 6)
(6, 1) (6, 2) (6, 3) (6, 4) (6, 5) (6, 6)

- **구구단>** 2단부터 9단까지 구구단 출력하기

```
for i in range(      ):
    for k in range(      ):
        print( f'{i}*{k}={i*k}' , end='\t')
    print()
```



① 별 출력하기(5x5)

```
for i in range(5):
    for k in range(5):
        print("*", end=" ")
    print()
```

```
* * * * *
* * * * *
* * * * *
* * * * *
* * * * *
```

② 별 출력하기(5x7)

```

```

```
* * * * * *
* * * * * *
* * * * * *
* * * * * *
* * * * * *
* * * * * *
* * * * * *
```

③ 별 출력하기(계단식)

```
for i in range(5):
    for k in range(      ):
        print("*", end=" ")
    print()
```

```
*
* *
* * *
* * * *
* * * * *
```

hint :
i 가 0일 때 1개 출력
i 가 1일 때 2개 출력
i 가 2일 때 3개 출력
i 가 3일 때 4개 출력
i 가 4일 때 5개 출력

④ 1부터 10까지의 숫자를 다음의 결과와 같이 직각삼각형 모양으로 출력하는 프로그램

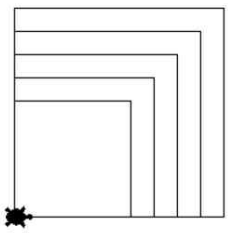
```
1
1 2
1 2 3
1 2 3 4
1 2 3 4 5
1 2 3 4 5 6
1 2 3 4 5 6 7
1 2 3 4 5 6 7 8
1 2 3 4 5 6 7 8 9
1 2 3 4 5 6 7 8 9 10
```

⑤ 별 출력하기(계단식2)

```
        *
      * *
    * * *
  * * * *
* * * * *
```

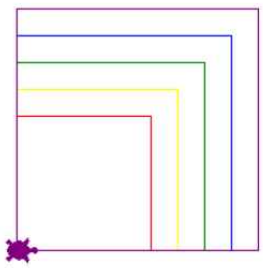
[turtle①] 다음은 사각형을 그리는 프로그램이다.
프로그램을 수정하여 한변의 길이가 20씩 증가하는 사각형 5개를 그려봅시다.

```
import turtle as t
t.shape('turtle')

side = 100
for i in range(4):
    t.forward(side)
    t.left(90)
```



추가미션> 아래의 color_lsit를 활용하여 사각형의 선색을 변경하며 커지는 사각형을 그려봅시다.
색변경함수 : t.color(색상)

```
import turtle as t
t.shape('turtle')
color_list = [ 'red', 'yellow', 'green', 'blue', 'purple' ]
```



VIII. 리스트2

1. 리스트와 for문

```
for 변수 in 리스트 :
    반복실행문장
    반복실행문장
```

- for문을 활용하여 리스트의 모든 요소를 출력하는 프로그램을 작성해 봅시다.

```
score = [ 80, 90, 75, 25, 50]
```

```
80
90
75
25
50
```

- for문과 리스트의 **인덱스 번호를 활용**하여 리스트의 모든 요소를 출력하는 프로그램을 작성해 봅시다

```
score = [ 80, 90, 75, 25, 50]
for a in range(5) :
```

```
80
90
75
25
50
```

- for문과 리스트의 **인덱스 번호를 활용**하여 리스트의 모든 요소를 출력하는 프로그램을 작성해 봅시다.
  (단 리스트의 요소의 개수가 변경되어도 올바르게 결과가 출력되도록 합니다.)

```
score = [ 80, 90, 75, 25, 50]
for a in range(              ) :
```

```
80
90
75
25
50
```

2. 문자열을 리스트로 변경

◆ 문자열.split() 메소드

: 지정한 글자를 기준으로 문자를 나눠 리스트 형식으로 저장하는 함수

```
문자열.split() # 공백을 기준으로 문장열을 나눔
문자열.split(기준)
```

```
>>>'Knowlegde is power and money'.split()

>>>'Knowlegde is power and money'.split('is')
```

```
문자열변수.split()
문자열변수.split(기준)
```

```
>>>a = '상상할 수 없는 꿈을 꾸고 있다면, 상상할 수 없는 노력을 해라.'
>>>a.split()

>>>a.split(',')
```

- 입력받은 값을 공백을 기준으로 나눠 리스트에 저장

```
리스트명 = input().split()
```

```
>>>week = input().split()

>>>week
```

① 이메일을 입력받아 아이디만 출력하는 프로그램을 작성해 봅시다.

```
email =
```

```
star5125@ice.go.kr
star5125
```

② 좋아하는 음식을 공백을 기준으로 입력받아 음식의 개수와 종류를 출력하는 프로그램을 작성해 봅시다.

```
food =
```

```
떡볶이 치킨 피자 쫄면 탕수육
좋아하는 음식의 개수: 5
떡볶이
치킨
피자
쫄면
탕수육
```

③ 정수 3개를 공백기준으로 입력받아 정수의 평균을 출력하는 프로그램을 작성해 봅시다.

```
score = input().split()
total = 0
score[0] = int(score[0])
score[1] = int(score[1])
score[2] = int(score[2])
```

```
80 90 20
63.33
```

<좀더 알아보기>

◈ map 함수

: 리스트의 요소를 지정된 함수로 처리해 주는 함수

```
map( 함수, 리스트 )
```

- 리스트의 요소를 정수로 변경

```
map( int, 리스트 )
```

```
>>> alist= ['80', '90', '10']
>>> alist= list( map( int, alist) )  #리스트 형태로 저장하기 위해서는 list함수 필요
>>> alist
```

- 공백을 기준으로 입력받은 수 중 짝수의 개수를 출력하는 프로그램을 작성해 봅시다.

```
num =
```

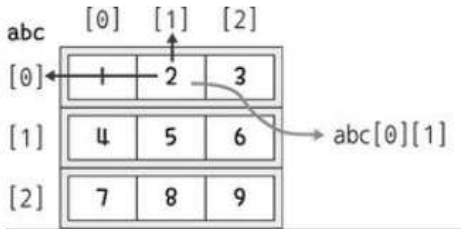
```
2 3 4 5 6 6
4
```

3. 이차원 리스트

◆ 리스트 요소로 리스트를 가진 복합 데이터 구조의 리스트

- 리스트 안에 리스트가 들어 있는 형태

```
>>>abc = [ [ 1, 2, 3 ], [ 4, 5, 6 ], [7, 8, 9] ]
>>>abc

>>>abc =[ [ 1, 2, 3 ],
          [ 4, 5, 6 ],
          [ 7, 8, 9] ]
>>>abc
```



- 이차원 리스트에서 인덱스는 2개(행과 열)

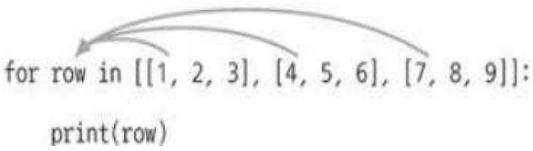
리스트명[행인덱스][열인덱스]

```
>>>abc = [ [ 1, 2, 3 ], [ 4, 5, 6 ], [7, 8, 9] ]
>>>abc[0]

>>>abc[1]

>>>abc[2]
```

```
>>>abc = [ [ 1, 2, 3 ], [ 4, 5, 6 ], [7, 8, 9] ]
>>>abc[0][1]

>>>abc[1][2]

>>>abc[2][2]
```

- for문을 활용한 이차원 리스트 탐색

프로그램	결과
abc = [[1, 2, 3],[4, 5, 6],[7, 8, 9]] for row in abc: print(row) #abc리스트의 요소 출력	



프로그램	결과
abc = [[1, 2, 3],[4, 5, 6],[7, 8, 9]] for row in abc: print(row[0]) #내부리스트의 첫번째요소 출력	

프로그램	결과

abc = [[1, 2, 3],[4, 5, 6],[7, 8, 9]]
for row in abc:
  for col in row:
  print(col, end=' ') #내부리스트의 요소 출력
  print()	

① 다음 표에서 중간고사 총점을 구하는 프로그램을 작성해 봅시다.

열(Column)

행(Row)	중간고사	기말고사	실습과제
김정호	92	80	87
박문수	94	82	86
이사부	74	65	69
장영실	87	89	81

```
Scores = [[92, 80, 87], [94, 82, 86], [74, 65, 69], [87, 89, 81]]
m_total = 0
for row in Scores :
    m_total = m_total + [      ]
print(m_total)
```

347

①-1> 다음 표에서 중간고사 평균을 구하는 프로그램을 작성해 봅시다.

```
Scores = [[92, 80, 87], [94, 82, 86], [74, 65, 69], [87, 89, 81]]
m_total = 0
for row in Scores :
    m_total = m_total + row[0]
avg = 
print( avg  )
```

86.75

열(Column)

	중간고사	기말고사	실습과제
김정호	92	80	87
박문수	94	82	86
이사부	74	65	69
장영실	87	89	81

행(Row)

①-2> 각 학생별 점수의 총점을 구하는 프로그램을 작성해 봅시다.

```
Scores = [[92, 80, 87], [94, 82, 86], [74, 65, 69], [87, 89, 81]]
for row in Scores :
    s_total = 0
    for col in row :
        s_total = s_total + col
    
```

259
262
208
257

①-3> 각 학생별 점수의 평균을 구하는 프로그램을 작성해 봅시다.

```
Scores = [[92, 80, 87], [94, 82, 86], [74, 65, 69], [87, 89, 81]]
for row in Scores :
    s_total = 0
    for col in row :
        s_total = s_total + col
    
```

86.33
87.33
69.33
85.67

①-4> 각 학생별 점수의 평균을 *학생 이름과 함께 출력*하는 프로그램을 작성해 봅시다.

```
Scores = [['김정호',92,80,87], ['박문수',94,82,86], ['이사부',74,65,69], ['장영실',87,89,81]]
for row in Scores:
```

김정호 86.33
박문수 87.33
이사부 69.33
장영실 85.67

②-1> 3국의 나라가 획득한 금,은,동 매달의 갯수를 구하는 프로그램을 작성해 봅시다.

```
medal = [ [6,4,10], [38,32,19], [26,14,17] ]
```

국가	금	은	동
대한민국	6	4	10
중국	38	32	19
일본	26	14	17

금: 70
은: 50
동: 46

②-2> 각 나라별 메달의 개수의 합을 구하는 프로그램을 작성해 봅시다.

```
medal =  [ ['대한민국', 6,4,10], ['중국', 38,32,19], ['일본', 26,14,17] ]
```

국가	금	은	동
대한민국	6	4	10
중국	38	32	19
일본	26	14	17

```
대한민국 20
중국 89
일본 57
```

③-1> 다음 표와 같이 학생들의 결석을 하였을 경우 1, 출석을 하였을 경우 0으로 표시하였다.
해당 데이터를 활용하여 모든 학생의 결석 횟수의 합을 출력하는 프로그램을 작성해 봅시다.

```
daylist = [ [0,1,0,0,1], [0,0,1,0,0], [0,1,0,1,0], [0,0,0,1,0],[0,0,0,0,0]]
count = 0
```

이름	월	화	수	목	금
유재석	0	1	0	0	1
지석진	0	0	1	0	0
송지효	0	1	0	1	0
김종국	0	0	0	1	0
하하	0	0	0	0	0

```
6
```

③-2> 다음 표와 같이 학생들의 결석을 하였을 경우 1, 출석을 하였을 경우 0으로 표시하였다.
해당 데이터를 활용하여 모든 학생의 화요일 결석 횟수의 합을 출력하는 프로그램을 작성해 봅시다.

```
daylist = [ [0,1,0,0,1], [0,0,1,0,0], [0,1,0,1,0], [0,0,0,1,0],[0,0,0,0,0]]
count = 0
```

이름	월	화	수	목	금
유재석	0	1	0	0	1
지석진	0	0	1	0	0
송지효	0	1	0	1	0
김종국	0	0	0	1	0
하하	0	0	0	0	0

```
2
```

[심화] 첫 번째 줄에는 학생 수, 두 번째 줄부터는 출석부 데이터가 입력된다. (출석:0, 결석:1)
입력받은 데이터를 활용하여 모든학생의 결석 횟수를 출력하는 프로그램을 작성해 봅시다. (*hint: append(요소)*)

```
n = int(input())              # 학생수 입력
daylist=[]
for i in range(n):                  #출석데이터 입력
    att = input().split()
    daylist.append(att)

count = 0
```

```
3
0 1 0 0 1
1 1 0 0 0
1 0 0 0 0
5
```

IX. 함수

1. 함수 정의

- 어떤 기능을 수행하는 작은 단위 프로그램
- 입력값을 가지고 어떤 일을 수행한 다음에 그 결과물을 내어놓은 것

2. 함수의 유형

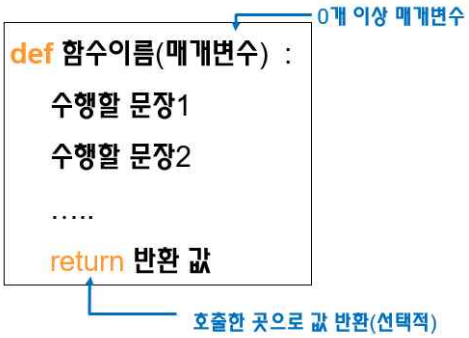
① 파이썬이 제공하는 함수 (내장 함수)
- 아무런 설정 없이 바로 사용 (print(), input(), len(), int() 등)

② 사용자가 작성하는 함수
- 사용자가 직접 코드를 작성하여 만든 함수,
- 다른 곳에 작성해 놓은 함수를 import하여 사용 가능

3. 함수 장점

① 함수 단위로 작업할 수 있으므로 큰 프로그램을 만들기 쉽다.
② 프로그램 이해하기가 쉽다.
③ 함수 별로 오류 처리가 가능하여 디버깅이 용이하다.
④ 한번 선언으로 여러 번 사용할 수 있다.
⑤ 같은 내용이 반복해서 사용되는 것을 방지할 수 있다.
⑥ 다수의 프로그래머들이 공동 개발이 쉽다.

4. 함수 정의 구조




```
# 함수 정의
def print_address():
        print("인천 중구 운남서로 " )
        print("10번길1 " )
        print("영종국제물류고등학교")

# 함수 호출
print_address()
```

1) 함수의 매개변수

- 매개변수 : 함수에 입력으로 전달된 값을 받는 변수
- 인수 : 함수를 호출할 때 전달하는 입력 값

<매개변수 없는 경우>

```
# 함수 정의
def print_address():
        print("인천 중구 운남서로 " )
        print("10번길1 " )
        print("영종국제물류고등학교")


# 함수 호출
print_address()
print_address()
print_address()
```

<매개변수 있는 경우 >

```
# 함수 정의
def print_address(name): #매개변수
        print("인천 중구 운남서로 " )
        print("10번길1 " )
        print(name)


# 함수 호출
print_address("홍길동")    #인수
print_address("정보경")    #인수
print_address("이루리")    #인수
```

① 매개변수 없는 함수 정의하기

함수를 호출하면 'Python is Fun'을 출력하는 함수를 정의하시오

```
# 함수 정의
def pyfun() :
    
# 함수 호출
pyfun()
```

Python is Fun

② 매개변수 1개를 포함한 함수 정의하기

인수의 제곱값을 출력하는 함수를 정의하시오

```
# 함수 정의
def squ(n) :
    
# 함수 호출
squ(3)
```

9

③ 매개변수 2개를 포함한 함수 정의하기

첫번째 인수(문자열) 값을 두번째 인수(정수)번 만큼 반복하여 출력하는 함수를 정의해 봅시다.

```
# 함수 정의
def print_times(value,  ) :
        for a in range(n):
            print(value)

# 함수 호출
print_times("안녕", 5)
```

안녕
안녕
안녕
안녕
안녕

④ 두 수를 전달 받아 두 수의 합을 출력하는 함수를 정의해 봅시다.

```
# 함수 정의



# 함수 호출
get_sum(3, 5)
```

8

⑤ 두 수를 전달 받아 두 수중 큰 값을 출력하는 함수를 정의해 봅시다.

```
# 함수 정의



# 함수 호출
get_max(10, 20)
```

20

2) 결과 값 반환(return)

- return 문

① return 문은 어떤 내용도 화면에 출력하지 않은 채 함수 호출문으로 값을 반환

<return문이 없는 경우>

```
# 함수 정의
def get_add(a, b) :
    result = a + b
    print(result)

# 함수 호출
get_add(3,4)
```

<return문이 있는 경우 >

```
# 함수 정의
def get_add(a, b) :
    result = a + b
    print(result)
    return result

# 함수 호출
x= get_add(3,4)
```

② 하나의 함수 안에 여러 개의 return 문이 포함될 수 있다.

③ return 문이 함수의 마지막 문장일 필요는 없다.

④ return 문을 포함하지 않은 함수는 함수 호출문으로 None을 반환한다.

⑤ 특별한 상황일 때 함수를 빠져나가고 싶을 때 단독으로 사용할수 있다.

```
# 함수 정의
def return_test(a,b):
    if a == b :
        return
    else:
        return 100

# 함수 호출
x = return_test(5, 5)
print(x)
y = return_test(10, 5)
print(y)
```

```
#함수 정의
defsay_nick(nick):
    if nick == '바보' :
        return
    print("나의 별명은", nick, "입니다.")

#메인 프로그램
say_nick('야호')
say_nick('바보'))
```

① 함수를 호출하면 인수의 제곱값을 반환하는 함수를 정의하시오

```
# 함수 정의
def square(n) :
    [          ]

# 함수 호출
x = square(4)
print(x)
```

16

② 두 수를 전달 받아 두 수중 큰 값을 반환하는 함수 정의해 보기

```
# 함수 정의




# 함수 호출
x = get_max(7, 5)
print( x )
```

7

③ 원의 반지름을 입력 받아 원의 넓이 값을 반환하는 함수를 정의해 봅시다. (3.14 * 반지름 * 반지름)

```
#함수 정의



#메인 프로그램
r = int(input("원의 반지름 : "))
x = get_area(r)          #함수 호출
print( '원의 넓이 :', x )
```

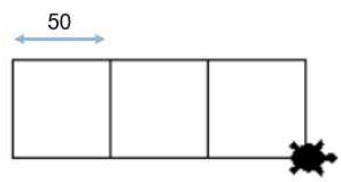
```
원의 반지름 :10
원의 넓이 : 314.0
```

④ 정수를 저장한 start와 end 변수를 받아 start부터 end까지 합을 계산하는 프로그램을 작성해 봅시다.

```
#함수 정의
def get_sum(start, end):



#메인 프로그램
result = get_sum(1, 10)
print( result  )
```

```
55
```

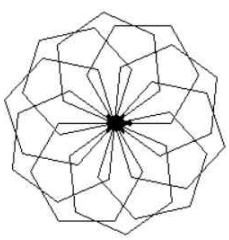
[turtle①] 한 변의 길이를 매개변수로 받아 정사각형을 그리는 함수를 이용하여 아래 그림과 같이 한변의 길이가 50인 3개의 정사각형을 그려 봅시다.




```
def square(length) :
        for i in range(4) :
                t.forward(length)
                t.left(90)
#메인 프로그램
```

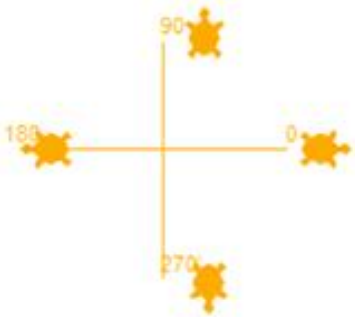
[turtle②] n-각형을 그리는 함수(n_polygon)를 작성하여 n-각형을 회전하며 10번 반복하여 그리는 프로그램을 완성해 봅시다.

```
import turtle as t
t.shape("turtle")
# 한변의 길이가 length인 n 각형 그리는 함수
def n_polygon(n, length):



#메인 프로그램
for i in range(10):
    t.left(36)
    n_polygon( 6, 50 )
```



[trutle③] 키보드로 거북이를 조종해서 그림 그리기

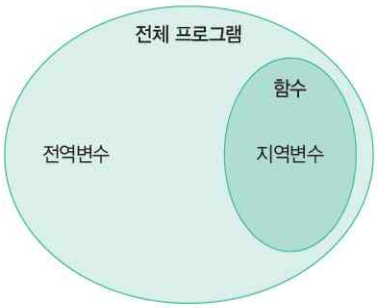
turtle 함수	설명
setheading(각도)	거북이가 바라보는 '각도'로 설정합니다.
onkeypress(함수, '키이름')	'키이름'에 해당하는 키를 누르면 함수를 호출해서 실행
listen()	거북이 그래픽 창이 키보드 입력을 받을 수 있도록 합니다.




```
import turtle as t
t.shape("turtle")
t.speed(0)

def turn_right() :              #오른쪽으로 이동하는 함수
    t.setheading(0)
    t.forward(10)

def turn_up() :                 #위로 이동하는 함수
    t.setheading(90)
    t.forward(10)

def turn_left() :               #왼쪽으로 이동하는 함수
    t.setheading(180)
    t.forward(10)

def turn_down() :               #아래로 이동하는 함수
    t.setheading(270)
    t.forward(10)

def blank():                    #화면 지우는 함수
    t.clear()

t.onkeypress(turn_right, 'Right')  # →를 누르면 turn_right 함수 실행
t.onkeypress(turn_up, 'Up')
t.onkeypress(turn_left, 'Left')
t.onkeypress(turn_down, 'Down')
t.onkeypress(blank, 'Escape')      # 'Esc'를 누르면 blank 함수 실행
t.listen()                         # 키보드 입력
```

5. 지역 변수 와 전역 변수

1) 지역 변수(local variable)



- 함수 안에서 선언되는 변수
- 함수 안에서만 사용이 가능하다.

```
def func(a, b)
    x = a + b
    y = a * b
    result = x + y
    return result
```

매개변수 a, b와 함수 안에서 정의한 x, y, result는 모두 지역 변수이고, 함수가 실행되는 동안에만 존재한다.

```
def get_area(radius):
    result = 3.14 * radius**2 #지역변수 선언
    return result

r = float(input("원의 반지름: "))
area = get_area(r)
print(result)
```

➪ NameError: name 'result' is not defined

2) 전역 변수(global variable)

- 어떤 함수에도 속하지 않는 메인프로그램(함수 외부)에서 선언되는 변수
- 메인 프로그램 및 메인 프로그램에 속한 모든 함수들이 접근할수 있는 변수

```
def get_area():
    result = 3.14 * r **2
    return result

r = float(input("원의 반지름: "))    #r: 전역변수 선언
area = get_area()
print(area)
```

```
원의 반지름: 10
314.0
```

- 함수 안에서 전역변수와 동일명으로 변수 선언시 다른 지역변수로 생성

```
result = 0        #전역변수 선언
def get_area():
    result = 3.14 * r**2     #지역변수 선언
    return result

r = float(input("원의 반지름: "))
area = get_area()
print(result)
```

```
원의 반지름: 10
0
```

- **global** 키워드 : 함수안에서 전역 변수 사용시 global 키워드를 사용하여 해당 변수가 전역변수임을 명시

```
result = 0              #전역변수 선언
def get_area():
    global result       #전역변수 명시
    result = 3.14 * r**2     #전역변수

r = float(input("원의 반지름: "))
area = get_area()
print(result)    #전역변수
```

```
원의 반지름: 10
314.0
```

- 지역변수와 전역변수 출력하기

```
def func1( ) :
    a = 10     #지역변수
    print('func1( )에서 a의 값', a)

def func2( ) :
    print('func2( )에서 a의 값', a)

a = 20       #전역변수

func1( )
func2( )
```

```
func1( )에서 a의 값 10
func2( )에서 a의 값 20
```

① 프로그램을 실행했을 때의 결과를 예측해 봅시다.

```
def func( b ) :
    b = a + 10
    return b

a = 5
c = func(3)
print(c)
print(a)
```

<실행결과>

```
def func( b ) :
    a = b + 10
    return a

a = 5
c = func(3)
print(c)
print(a)
```

<실행결과>

② 아래 프로그램을 실행했을 때의 결과를 예측해 봅시다.

```
def func( b ) :
    global a
    a = b + 10
    return a

a = 5
c = func(3)
print(c)
print(a)
```

<실행결과>

[turtle] 클릭하는 곳에 사각형 그리기

```
import turtle as t

def square(length):
    for i in range(4):
        t.forward(length)
        t.left(90)

def drawit(x,y):
    t.penup()
    t.goto(x,y)
    t.pendown()
    t.begin_fill()
    t.color('green')
    square(50)
    t.end_fill()

t.speed(0)
t.hideturtle()
t.onscreenclick(drawit)  # 마우스 버튼을 누르면 drawit함수 호출
```

추가 미션>
전역(global)변수를 선언하여 메인 프로그램에서 사각형 색을 변경시킬 수 있도록 프로그램을 수정해 봅시다.

X. 파일 입출력

1. 파일의 개념

- 파일은 컴퓨터를 이용하여 다루어지는 많은 데이터들을 일정한 묶음으로 저장하고 처리하는 자료의 단위

2. 파일을 사용하는 이유

- 사용자가 필요로 하는 데이터를 영구적으로 저장할 수 있다.
- 프로그램을 실행할 때마다 입력 데이터를 매번 타이핑할 필요가 없다.
- 방대한 데이터를 다루기 쉽다.

3.파일 입출력 : 보조기억장치에 저장된 파일을 읽어 입력하고, 파일에 출력

① 파일 열기 => open() 함수
② 파일 처리 => 파일에서 데이터를 읽거나 쓰기, read() / write()
③ 파일 닫기 => close() 함수

1) 파일 열기 open()

```
파일객체 = open( 파일이름, 파일모드 )
```

*** 파일접근 경로**
파일이름만 쓸 경우 실행파일이있는 폴더 내에서 접근
경로명이 있을 경우 해당 경로에 있는 파일 접근

파일모드	모드이름	설명
r	읽기모드	파일의 처음부터 읽는다. 해당 파일이 없으면 오류발생
w	쓰기모드	파일의 처음부터 쓴다. 파일이 없으면 생성된다. 파일이 존재하면 기존의 내용은 지워지고 새로 작성된다.
a	추가모드	파일의 새로운 내용을 추가 시킬 때 사용. 파일의 끝에 쓴다. 파일이 없으면 생성되다.

2) 파일 닫기 close()

```
파일객체.close()
```

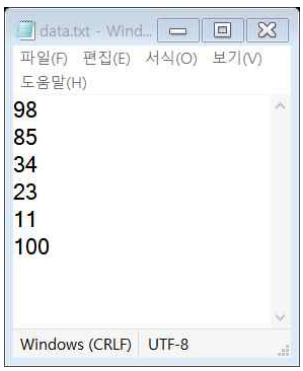
- 파일과 관련된 모든 작업이 끝나면 파일을 정상적으로 닫아야 한다.

3) 파일 읽기(입력): read()

- 파일을 읽기 위해서는 r(읽기모드)로 파일을 열어야 한다.
- **read 메소드** : 파일의 전체 내용을 하나의 큰 문자열로 반환한다.

```
변수 = 파일객체.read()
```

```
>>> infile = open( 'd:/20525/data.txt', 'r' )    #읽기모드로 파일 열기
>>> lines = infile.read()          #파일을 처음부터 끝까지 읽어 문자열로 반환
>>> print(lines)


>>> infile.close()                 #파일 닫기
```



자신의 폴더에 저장하기
d:/20525/data.txt

- **파일에서 한줄 씩 읽기**

  read()로 전체 내용을 읽은 후 **split()을 이용하여 줄바꿈문자(\n)을 기준으로 분리**하여 리스트 형태로 저장한다.

```
>>> infile = open( 'd:/20525/data.txt', 'r' )    #읽기모드로 파일 열기
>>> lines = infile.read()          #파일을 처음부터 끝까지 읽어 문자열로 반환
>>> lines

>>> lines = lines.split('\n')
>>> lines

>>> infile.close()                 #파일 닫기
```

- **for 문 활용하여 파일에서 한줄씩 읽기**

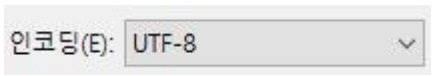
  : 파일로 부터 한 줄씩 자동으로 읽어 변수에 한 줄씩 저장한다.

```
for 루프변수 in 파일객체:
    문장1
    문장2
    ...
    문장n
```

```
infile = open( 'data.txt', 'r' )
for line in infile:
    print('한줄씩', line)
print('파일 모두 읽음')
infile.close()
```

① 아래의 글을 텍스트파일에 저장한 후 파일을 읽는 프로그램을 작성해 봅시다.

- 자신의 폴더에 poem.txt 로 저장해봅시다. (d:/20525/poem.txt)
  메모장에서 인코딩 방식을 UTF-8로 설정한 후 저장합니다.




```
괜찮다
                심동현

어느날 작은새가 나무에게 말했다
내 의자가 되어주고 내 둥지가 되어주는데
난 아무것도 해줄게 없어요
나무가 작은새에게 말했다
너의 지저귐은 좋은 노랫소리였고
너가 지은 둥지는 나의 옷이 되었다
내게 앉은 너는 나의 난로였다
그러니 괜찮다
```

①-1> 파일 전체를 읽어 모니터에 출력하는 프로그램을 작성해 봅시다.

```
infile = open( 'poem.txt', 'r' , encoding='utf8' )

print( lines )
infile.close()
```

①-2> 파일 내의 단어 리스트와 단어수를 출력하는 프로그램을 작성해 봅시다. (split() , len())

```
infile = open( 'poem.txt', 'r' , encoding='utf8' )
```

['괜찮다', '심동현', '어느날', '작은새가', '나무에게', '말했다.', '내', '의자가', '되어주고', '내', '둥지가', '되어주는데', '난', '아무것도', '해줄게', '없어요.', '나무가', '작은새에게', '말했다.', '너의', '지저귐은', '좋은', '노랫소리였고', '너가', '지은', '둥지는', '나의', '옷이', '되었다.', '내게', '앉은', '너는', '나의', '난로였다.', '그러니', '괜찮다.']

단어수 : 36

①-3> 파일 내의 라인수를 출력하는 프로그램을 작성해 봅시다.

```
infile = open( 'poem.txt', 'r' , encoding='utf8' )
count = 0
for line in infile :
```

라인수 : 11

4) 파일 쓰기(출력) : write()

- 'w'(쓰기모드), 'a'(추가모드) 로 파일을 열어야 한다.

- **write 메소드** : 문자열을 파일에 출력한다.
  숫자를 파일에 쓰기전에 문자열로 변환하여야 한다.

.

```
파일객체.write( '파일에 작성할 문자열' )
```

(1) 'w'(쓰기모드)로 파일 열기

- 해당 파일이 존재하지 않으면 새로운 파일이 생성된다.
- 해당 파일이 이미 존재할 경우 원래 있었던 내용이 모두 사라지고 새로 작성된 것만 기록된다.

```
>>> outfile = open( 'wdata.txt', 'w' )    #쓰기모드로 파일 열기, 실행파일이 있는 폴더의 파일에 접근
>>> outfile = open( 'd:/20525/wdata.txt', 'w' )    #쓰기모드로 파일 열기, 해당 경로의 파일에 접근
```

• 파일에 한줄 입력

```
outfile = open('wdata.txt', 'w' , encoding='utf8' )
outfile.write('파일을 작성합니다.')
outfile.colose()
```

• '\n'을 이용하여 여러줄을 기록
: write()메소드는 줄바꿈을 포함하지 않아 다음줄에 작성하기 위해서는 '\n'(줄바꿈) 문자를 추가한다.

```
outfile = open("wdata.txt", 'w', encoding='utf8' )
outfile.write( "파일을 작성합니다. \n 두번째줄 입력 " )    # 두 줄 입력
outfile.close()
```

```
outfile = open("wdata.txt", 'w', encoding='utf8' )
for a in range(10):                    #여러줄 입력하기
    outfile.write("여러줄 입력하기\n")
outfile.close()
```

- 정수 또는 실수는 파일에 쓰기 전에 문자열로 변환하여 저장하여야 한다.
- 따옴표나 str() 함수를 적용하여 문자열로 변환할수 있다.

```
outfile = open("wdata.txt", 'w' , encoding='utf8' )
outfile.write( 99 )      # 99 => '99'  나 str(99) 로 변환하여 보자
outfile.close()
```

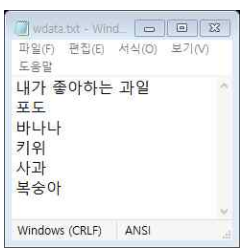
① 키보드로 문자를 입력받아 파일에 저장하는 프로그램을 작성해 봅시다.

```
outfile = open("wdata.txt", 'w' , encoding='utf8' )
msg = input('입력:')
```

② 자신이 좋아하는 과일 5개를 다섯줄로 입력하여 파일에 기록하는 프로그램을 작성해 봅시다.

```
outfile = open('wdata.txt','w', encoding='utf8' )
outfile.write('내가 좋아하는 과일\n')
```

과일:포도
과일:바나나
과일:키위
과일:사과
과일:복숭아



③ 파일 복사하기

```
infilename= input("입력 파일 이름: ")    # 복사할 원본 파일 이름 받기
outfilename= input("출력 파일 이름: ")    # 복사 후 새로 만들 파일 이름 받기

infile= open(infilename, "r" ,encoding='utf8' )       # 원본 파일을 연다.
outfile= open(outfilename, "w" , encoding='utf8' )  # 새로 만들 파일을 생성한다.

s = infile.read()    # 전체 파일을 읽는다.

outfile.write(s)       # 전체 파일을 쓴다.

infile.close()          # 파일을 닫는다.
outfile.close()
```

입력 파일 이름: wdata.txt
출력 파일 이름: copydata.txt

(2) 'a' (추가모드)로 파일 열기

- 추가모드(a)로 열어 프로그램의 출력값을 write() 함수를 이용하여 파일에 기록.
- 텍스트 파일이 원래 가지고 있던 내용 바로 다음부터 기록된다.

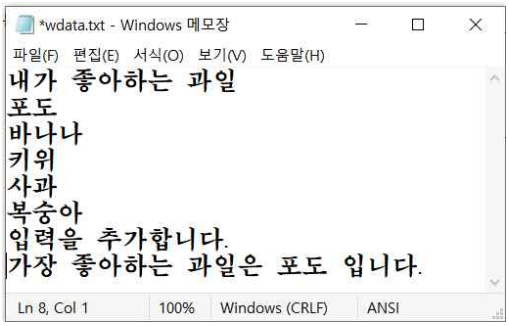
```
outfile = open("wdata.txt", 'a' , encoding='utf8' )
outfile.write( " 입력을 추가합니다. " )    # 추가 입력
outfile.close()
```

① wdata.txt. 파일에 가장 좋아하는 과일을 추가해 봅시다.

```
outfile = open('wdata.txt', 'a', encoding='utf8' )
fav = input('가장 좋아하는 과일:')
```

```
outfile.close()
```

가장 좋아하는 과일:포도



② poem.txt. 파일안에 시에 대한 감상평을 추가해 봅시다.

```
infile = open('poem.txt', 'a', encoding='utf8')
review = input('감상평을 입력합시다.\n')
```