Python 04

김은진

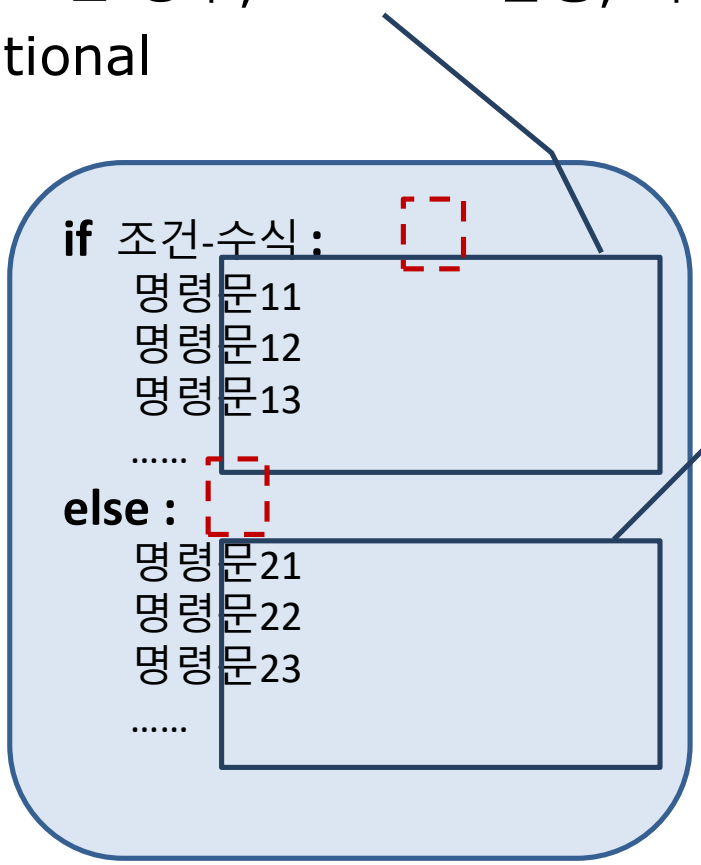
제어문(Control Statements)

- 프로그램은 명령문들의 모임
- 명령문들은 순차적으로 실행(top-down order)
- 제어문
  - 명령문 실행순서를 바꾸는 문장
  - 제어문은 하나의 문장이 위치할 수 있는 곳 어디나 작성 가능

- if / if-else / if-elif- …-else
- while
- for
- break
- continue

if 문 / if-else 문 (조건문)

- 조건-수식이 True 인 경우, if-block 실행, 아니면 else-block 실행
- else 부분은 optional



Block(일련의 명령문들)은 동일한 들여쓰기(indentation) 되어야 함

작은 수 찾기

두 개의 정수 입력 받아 큰 수 출력하기

```
x = int( input( "x: " ) )
y = int( input( "y: " ) )
if x > y :
    print("큰 수는 "+str(x)+"입니다")
if  y > x:
        print("큰 수는 ", y, "입니다")
else:
        print(" 두 수가 같다")
```

인자를 화면 출력하고 키보드에서 입력된 값을 input 함수 결과(str)로 반환

input 함수 결과값(str)을 정수형(int)로 바꿔 결과값으로 반환

실수 부분 출력

입력 숫자가 실수일때만 소수점 이하 부분 출력

```
x = eval( input("숫자 입력: " ) )
if type(x) == float :
    print( "입력 실수의 소수부: ",  x - int(x) )
```

input 함수 결과값(str)을 표현방법에 따라 적절한 Data Type의 값이나,
적절한 명령문으로 실행하여 결과값 반환

Built-in 함수 : eval( )

- 인자값을 표현방법에 따라 적절한 Data Type의 값이나, 적절한 명령문으로 실행하여 결과값 반환

```
>>> x = eval( input( "입력: " ) )
입력: 145
>>> x; type(x)
145
<class 'int'>
>>> x = eval( input( "입력: " ) )
입력: 5.345
>>> x; type(x)
5.345
<class 'float'>
>>> x = eval( input( "입력: " ) )
입력: kim
Traceback (most recent call last):
  File "<pyshell#21>", line 1, in <module>
    x = eval( input( "입력: " ) )
  File "<string>", line 1, in <module>
NameError: name 'kim' is not defined
```

```
>>> x = eval( input( "입력: " ) )
입력: 'kim'
>>> x; type(x)
'kim'
<class 'str'>
>>> x = eval( input( "입력: " ) )
입력: [ 1,2,3,4,5 ]
>>> x; type(x)
[1, 2, 3, 4, 5]
<class 'list'>
>>> x = eval( input( "입력: " ) )
입력: ( 10, 100, 30 )
>>> x; type(x)
(10, 100, 30)
<class 'tuple'>
>>> x = eval( input( "입력: " ) )
입력: dict( )
>>> x; type(x)
{}
<class 'dict'>
```

합격 조건

수학, 영어 점수 모두 90점 이상인 경우 'PASS' 출력,
아니면 'FAIL' 출력

```
math, eng = eval( input("수학, 영어: ") )
if math >= 90 and eng >= 90 :
    w = 'PASS'
else:
    w = 'FAIL'
print( w )
```

두 조건이 모두 만족해야 함

숫자, 숫자 형식으로 입력하면 tuple이 됨

따라서 다음과 같은 tuple 할당이 수행됨
math, eng = 숫자, 숫자

학점 결정

점수 입력받아 표와 같이 학점 출력

점수	학점
90 이상	A
80 이상 90 미만	B
70 이상 80 미만	C
60 이상 70 미만	D
60 미만	F

```
score = int(input('Enter Score: '))
if score >= 90:  90 이상
    grade = 'A'
else:            90 미만
    if score >= 80:  90 미만 and  80이상
        grade = 'B'
    else:            80 미만
        if score >= 70:  80 미만 and  70이상
            grade = 'C'
        else:            70 미만
            if score >= 60:  70 미만 and  60이상
                grade = 'D'
            else:            60 미만
                grade = 'F'
print('Grade => ', grade)
```

if - elif 문 (조건문)

- if 문의 변형
- elif 부분과 else 부분은 optional

```
if 조건-수식1 :
        명령    문11
        ……
elif 조건-수식2 :
        명령문21
        ……
elif 조건-수식3 :
        명령문31
        ……
……
else :
        령문n1
        ……
```

조건-수식1 True 이면
명령문11 …… 실행 -> 끝
아니면 조건-수식2 True 이면
명령문21 …… 실행 -> 끝
아니면 조건-수식3 True 이면
명령문31 …… 실행 -> 끝
아니면
……
마지막 조건-수식 True 아니면
명령문n1 …… 실행 -> 끝

** 조건-수식 True 성립하면 명령문들 실행하고 문장 끝으로

학점 결정을 if-elif 문으로

```
score = int(input('Enter Score: '))

if score >= 90:     90 이상
    grade = 'A'
elif score >= 80:   90 미만 and  80이상
    grade = 'B'
elif score >= 70:   80 미만 and  70이상
    grade = 'C'
elif score >= 60:   70 미만 and  60이상
    grade = 'D'
else:               60 미만
    grade = 'F'

print('Grade => ', grade)
```

Data Type에 따른 True, False

if 조건-수식 **:**

자료형	True	False
bool	True	False
int	0이 아닌 숫자	0
float	0.0 이 아닌 숫자	0.0
str	한 개 이상 문자열	""
list	[1, 2, 3]	[ ], list( )
tuple	(1, 2, 3)	( ), tuple( )
dict	{"key": 1}	{ }, dict( )
set	{1, 2, 3}	set( )

조건-수식의 True, False 판단

```
if False :
    print( 'True!' )
else:
    print( 'False!')

if [ ] :
    print( 'True!!' )
else:
    print( 'False!!')

if "False" :
    print( 'True!!!' )
else:
    print( 'False!!!')
```

in 연산자

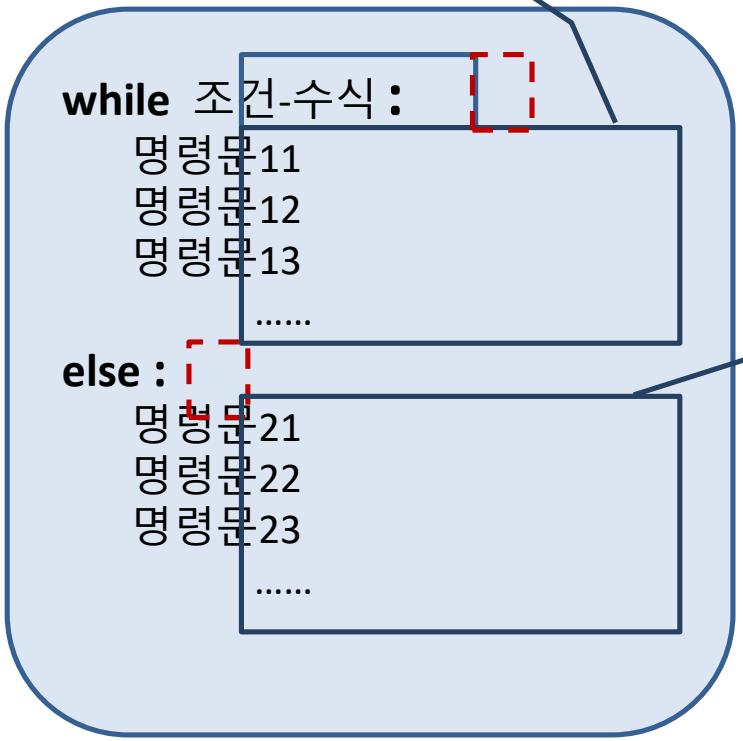
값 존재여부 확인

Data Type	True	False
x in list	1 **in** [1, 2, 3]	1 **in** [4, 5, 6]
x in tuple	1 **in** (1, 2, 3)	1 **in** [4, 5, 6]
x in str	"1" **in** "123"	"1" **in** "456"
x in dict	1 **in** { 2: 'Tue', 1: 'Mon' }	'Mon' **in** { 2: 'Tue', 1: 'Mon' }
x in set	1 **in** { 1, 2, 3 }	1 **in** { 4, 5, 6 }

```
d = input( "요일: " )
if d in ( 'Mon', 'Wed', 'Fri' ) :
    attendance = True
else:
    attendance = False
print( attendance )
```

while 문 (반복문)

- 조건-수식이 True면 block 실행을 반복
- else 부분은 optional



조건-수식이 False가 되면 else-block 실행

반복 block 내의 break 문으로 반복이 중단된 경우는 else-block 실행하지 않음

index와 반복문

- 점수 list 입력 받아 모든 item 차례로 하나씩 출력

```
scores = eval(input("점수 list: ") )
i = 0
while  i < len(scores) :
    print(scores[i], end= ' ' )
    i += 1
print( )
```

index 0 부터

print 함수 호출 방법:
print(a, b) – a값 b값 enter 출력 (줄바꿈)
print(a, b, end= '*') – a값 b값 *' 출력

index 1 증가 : 다음 item 접근 위해

enter만 출력(점수들 맨 뒤에 enter)

합격 조건 – 반복

수학, 영어 점수 모두 90점 이상인 경우 'PASS' 출력, 아니면 'FAIL' 출력
→ 이 작업을 0,0 이 입력될 때까지 반복

```
running = True
while running :
    math, eng = eval( input("수학, 영어: ") )
    if (math, eng) == (0, 0) :
        running = False
    else:
        if math >= 90 and eng >= 90 :
            w = 'PASS'
        else:
            w = 'FAIL'
        print( w )
else:
    print( 'Done' )
```

반복문 조건-수식을 위한 변수 running

while 반복 종료 조건

while 반복 종료 후 작업

합격 조건 – 반복 : 문장 구조, 실행 순서

```
running = True
while running  :
    math, eng = eval( input("수학, 영어: ") )
    if  (math, eng) == (0, 0) :
        running = False
    else:
        if math >= 90 and eng >= 90 :
            w = 'PASS'
        else:
            w = 'FAIL'
        print( w )
else:
    print( 'Done' )
```

```
if
else
    if
    else
```

부분을 if-elif-else로
바꾸면 어떻게 되나?

break

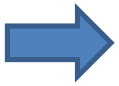
- break
  - break문을 포함하고 있는 가장 가까운 반복에서 나가 반복문 다음 문장으로
- 점수 list 입력 받아 50보다 큰 점수 역순으로 하나씩 출력

```
scores = eval(input( " 점수 list:  " ) )
scores.sort( reverse=True )
i = 0
while  i < len(scores) :
    if scores[i] <= 50:
        break
    print(scores[i], end= ' `  )
    i += 1
print()
```

list 안의 점수 역순으로 정렬

50 이하의 점수면 반복 중단
break 포함하는 while에서 나가기

while 문의 다음 문장



continue

- continue
  - continue문을 포함하고 있는 가장 가까운 반복문의 첫 줄로 가기

- 1에서 20의 정수 중 6의 배수가 아닌 정수의 합 구하기

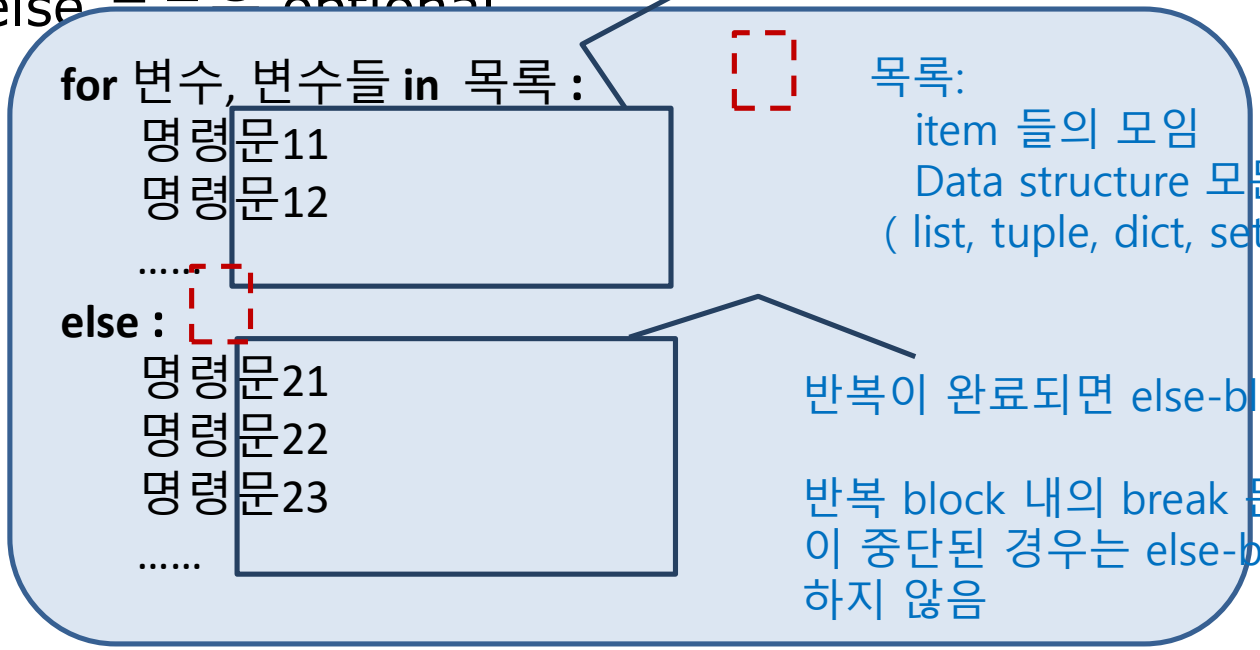
```
sum = num = 0
while num <= 20 :
    num += 1
    if num % 6 == 0 :
        continue
    sum += num
print( sum, num )
```

num이 6의 배수면,
sum += num으로 가지 않고
반복문 첫줄(while num <= 20 :) 로 감

반복문 종료 후 num 값은?

for문 (반복문)

- 목록의 item을 변수(변수들)에 할당하고 block 실행
  목록의 다음 item을 변수(변수들)에 할당하고 block 실행 …. ➔ 반복
- else 부분은 optional



while문 → for문

- for문으로: 점수 list 입력 받아 50보다 큰 점수 역순으로 하나씩 출력

```
scores = eval(input( " 점수 list:  " ) )
scores.sort( reverse=True )
for  s in scores :
   if  s  <= 50:
       break
   print( s , end= '`  )
print()
```

index가 필요 없음 !!!

for s in scores :
반복마다 scores의 item을
하나씩 s에 할당

scores [10, 20 , 100, 90, 88, 10, 30, 3, 4, 66, 77]

scores.sort(reverse=True)

scores [100, 90, 88, 77, 66, 30, 20, 10, 10, 4, 3]

s

88

.....

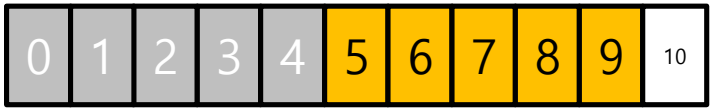
Built-in 함수 range( ) : 목록 만들기

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

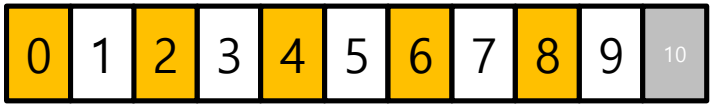
range(stop) # stop : 종료 값 (excluded)
range(start, stop)
range(start, stop [, step])

range(3)

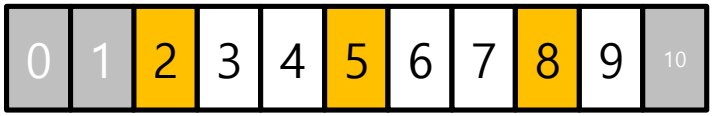
range(5, 10)



range(0, 10, 2)



range(2, 10, 3)



range() 결과값 list 만들기

- range( ) 함수 실행 결과값은 list type이 아님
- list ( ) type 변환해야 list 관련 함수(pop, append…) 사용할 수 있음

```
>>> list( range(3) )

>>> list( range(5, 10) )

>>> list( range(0, 10, 2) )

>>> list( range(2, 10, 3) )
```

50까지 짝수 출력

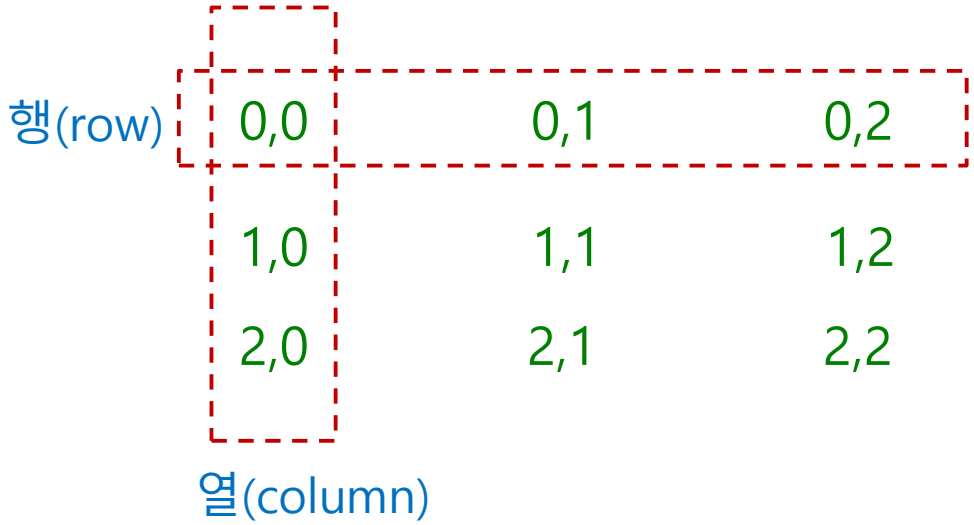
- for-range문을 사용해서 1~50 사이의 짝수만 출력
- 한 줄에 최대 10개만 출력
- Hint: print(, end=‘ ‘) 와 print( ) 사용

[출력 결과]

```
2 4 6 8 10 12 14 16 18 20
22 24 26 28 30 32 34 36 38 40
42 44 46 48 50
```

중첩 반복문(Nested Loop)

- 중첩반복문: 반복문의 block 안의 명령문으로 반복문 작성
- 다음과 같이 출력하려면?



열번호 →
행번호 ↓

	0	1	2
0	00	01	02
1	10	11	12
2	20	21	22

행번호, 열번호 를 출력

```
for i in range(3):          i 에 행번호를
    for j in range(3):      i 고정된 상태에서 j (행번호) 를 0, 1, 2 로 반복
        print ( str(i) + "," + str(j), end="  ")
    print()
```

중첩 반복문(Nested Loop)

다음과 같이 출력되도록 반복문을 완성

```
Pattern A        Pattern B
1                1 2 3 4 5 6
1 2              1 2 3 4 5
1 2 3            1 2 3 4
1 2 3 4          1 2 3
1 2 3 4 5        1 2
1 2 3 4 5 6      1
```

```
print("pattern A")
for i in range(      ):
        # display numbers
        # display newline
```

```
print("pattern B")
for i in range(          ):
        # display numbers
        # display newline
```

2차원 list에 중첩 for문 사용

- list table을 아래 예시 처럼 출력

row 값 : 0번째 item 1번째 item 2번째 item

```
table = [['00', '01', '02'], ['10', '11', '12'], ['20', '21', '22'] ]
```

cell 값 : [0] [1] [2] [0] [1] [2] [0] [1] [2]

```
00 01 02
10 11 12
20 21 22
```

```
for row in table:
    for cell in row :
        print( cell, end = `` )
    print( )
```

for 문에서의 list, tuple

```
trans = [ 'walking', 'bus', 'taxi', 'car', 'trains' ]
for t in trans :
    print ("교통수단 {0}입니다".format(t) )
```

str 함수 format( ) :
인자값을 문자열 내의 { } 대신 넣기

```
trans = ( 'walking', 'bus', 'taxi', 'car', 'trains' )
for t in trans :
    print ("교통수단 {0}입니다".format(t) )
```

list, tuple을 for ~ in 다음에 사용하면 item을 차례대로 변수에 할당

for 문에서의 dict

```
cost ={'walking': 0, 'bus': 1000, 'taxi': 3000,  'car':4000,
'trains':6000}
```

dict_items [('walking', 0) , ('bus', 1000) , ('taxi', 3000), ('car',4000), ('trains',6000)] # 순서는 임의

```
for key, value in cost.items() :
    print ("교통수단 {0}({1})".format(key, value) )
for k in cost.keys() :
    pass
for v in cost.values() :
    pass
```

dict_keys ['walking', 'bus', 'taxi', 'car', 'trains']

dict_values [0, 1000, 3000, 4000, 6000]

```
for x in cost :
    print ( x )
```

dict가 in 다음에 오면 key 값만 가져옴

control statements

- if / if-else / if-elif- …-else
- while
- for
- break
- continue

- 조건-수식 표현
  - in, not in
- for ~ range( )
- 중첩 반복문
- list, tuple, dict에 for문 사용