

■ 제어문

● 조건문

- if
- if – else
- if – elif – else

● 반복문

- while
- for

■ 조건문 - if

● if문의 기본 구조

- if

```
if 조건식:  
    수행할 문장  
    수행할 문장  
    수행할 문장
```

- if else

```
if 조건식:  
    수행할 문장  
    수행할 문장  
    수행할 문장  
else:  
    수행할 문장  
    수행할 문장
```

- if elif else

```
if 조건식:  
    수행할 문장  
    수행할 문장  
    수행할 문장  
elif:  
    수행할 문장  
    수행할 문장  
else:  
    수행할 문장  
    수행할 문장
```

들여쓰기 주의사항

1. [:] 콜론이 등장한 다음 줄부터 들여쓰기 사용
2. 첫번째 들여쓰기의 여백만큼 다음 줄에서도 들여쓰기 사용
3. [Tab] 또는 [Spacebar] 두가지 다 사용 가능하지만 하나의 방법으로만 사용
4. 일반적으로 [Spacebar]를 사용하여 공백 4칸을 권장

■ 조건문 - if

- 점수가 60점 이상인 경우 합격 (if)

```
score = 60  
  
if score >= 60:  
    print('PASS')
```

- 점수가 60점 이상인 경우 합격, 그 외 경우 불합격 (if - else)

```
score = 59  
  
if score >= 60:  
    print('PASS')  
else:  
    print('FAIL')
```

- 점수가 70점 이상인 경우 합격, 60점 이상인 경우 재시험

그 외 경우 불합격 (if - elif - else)

```
score = 65  
  
if score >= 70:  
    print('PASS')  
elif score >= 60:  
    print('RETAKE')  
else:  
    print('FAIL')
```

■ 조건문 - if

● 조건식

- 각 자료형의 Bool 값으로 사용

```
name = 'ggoreb'

if name:
    print('PASS')
else:
    print('FAIL')
```

- 비교 연산자 사용

```
age = 20

if(age >= 20):
    print('음주 가능')
else:
    print('음주 불가')
```

Type	Value	Bool
String	'Python'	True
	''	False
Number	1	True
	0	False
List	[1, 2]	True
	[]	False
Tuple	(1, 2)	True
	()	False
Dictionary, Set	{1, 2} or {'a':1}	True
	{}	False
None	None	False

비교 연산자	설명
x == y	x 와 y 가 같다
x != y	x 와 y 가 같지 않다
x < y	x 가 y 보다 적다
x > y	x 가 y 보다 크다
x <= y	x 가 y 보다 적거나 같다
x >= y	x 가 y 보다 크거나 같다

■ 제어문에서 사용되는 연산자

● 논리 연산자

연산자	설명
x and y	x 와 y 모두 참일때 참
x or y	x 와 y 모두 거짓일때 거짓
not x	x 가 거짓일때 참

● and

x	y	x and y
true	true	true
true	false	false
false	true	false
false	false	false

● or

x	y	x or y
true	true	true
true	false	true
false	true	true
false	false	false

● not

x	not x	y	not y
true	false	true	false
true	false	true	false
false	true	false	true
false	true	false	true

■ 제어문에서 사용되는 연산자

● 논리 연산자 사용

```
native = '금수저'  
money = 8000
```

```
if native == '금수저' or money >= 10000:  
    print('슈퍼카 구매 가능')  
else:  
    print('슈퍼카 구매 불가')
```

```
age = 20  
hasIDCard = True
```

```
If age >= 20 and hasIDCard:  
    print('술 구매 가능')  
else:  
    print('술 구매 불가')
```

■ 제어문에서 사용되는 연산자

● 자료형이 가지고 있는 요소 확인

in	not in
x in String	x not in String
x in List	x not in List
x in Tuple	x not in Tuple
x in Dictionary	x not in Dictionary
x in Set	x not in Set

```
>>> s = 'python'
>>> 'th' in s
True
>>> 'P' in s
False
```

```
>>> list = [1, 2, 3, 4, 5]
>>> 1 in list
True
>>> 0 in list
False
```

```
>>> dic = { 'a' : 1, 'b' : 2, 'c' : 3 }
>>> 'd' not in dic
True
>>> 'a' not in dic
False
```

```
>>> set = { 1, 2, 3, 4, 5 }
>>> 2 in set
True
>>> '2' in set
False
```

```
items = ['sword', 'gun', 'bow']
```

```
if 'bow' in items or 'gun' in items:
    print('공격 성공')
else:
    print('공격 실패')
```

■ 조건부 표현식

- [조건식이 참인 경우] if [조건식] else [조건식이 거짓인 경우]

```
score = 40

if score >= 60:
    print('PASS')
else:
    print('FAIL')

print('PASS') if score >= 60 else print('FAIL')
```

```
score = 40

if score >= 60:
    msg = 'PASS'
else:
    msg = 'FAIL'

print(msg)

msg = 'PASS' if score >= 60 else 'FAIL'
print(msg)
```


■ 반복문 - while

● while문의 기본 구조

while 조건식:

■ 수행할 문장

■ 수행할 문장

■ 수행할 문장

들여쓰기 주의사항

1. [:] 콜론이 등장한 다음 줄부터 들여쓰기 사용
2. 첫번째 들여쓰기의 여백만큼 다음 줄에서도 들여쓰기 사용
3. [Tab] 또는 [Spacebar] 두가지 다 사용 가능하지만 하나의 방법으로만 사용
4. 일반적으로 [Spacebar]를 사용하여 공백 4칸을 권장

■ 반복문 - while

● 10번 반복 실행

```
count = 0
```

```
while count < 10:
```

```
    count = count + 1
```

```
    print('%d번 반복' % count)
```

```
print('%d번 반복 후 종료' % count)
```

```
1번 반복
2번 반복
3번 반복
4번 반복
5번 반복
6번 반복
7번 반복
8번 반복
9번 반복
10번 반복
10번 반복 후 종료
```

count	조건식	수행문장 1	수행문장 2
0	0 < 10	count = count + 1	1번 반복
1	1 < 10	count = count + 1	2번 반복
2	2 < 10	count = count + 1	3번 반복
3	3 < 10	count = count + 1	4번 반복
4	4 < 10	count = count + 1	5번 반복
5	5 < 10	count = count + 1	6번 반복
6	6 < 10	count = count + 1	7번 반복
7	7 < 10	count = count + 1	8번 반복
8	8 < 10	count = count + 1	9번 반복
9	9 < 10	count = count + 1	10번 반복
10	10 < 10	X	X

■ 반복문 - while

● break : 반복문 강제 종료

- 1 ~ 10 범위의 숫자를 랜덤으로 추출, 5가 나오면 종료

```
import random

while True:
    number = random.randint(1, 10)
    print('추출된 숫자 : %d' % number)
    if number == 5:
        break
```

```
추출된 숫자 : 6
추출된 숫자 : 4
추출된 숫자 : 1
추출된 숫자 : 4
추출된 숫자 : 5
```

- 종료 메뉴가 선택 될 때까지 반복

```
while True:
    print('=' * 10)
    print('1. 입력')
    print('2. 출력')
    print('3. 종료')
    print('=' * 10)

    s = input('메뉴를 선택해주세요. > ')

    if s == '3': break

print('종료되었습니다.')
```

```
=====
1. 입력
2. 출력
3. 종료
=====
메뉴를 선택해주세요. > 1
=====
1. 입력
2. 출력
3. 종료
=====
메뉴를 선택해주세요. > 2
=====
1. 입력
2. 출력
3. 종료
=====
메뉴를 선택해주세요. > 3
종료되었습니다.
```

■ 반복문 - while

● continue : 반복문의 처음으로 돌아가기

- 1 ~ 10 범위의 숫자 중 홀수값만 출력

```
num = 0

while num < 10:
    num = num + 1

    if num % 2 == 0: # 짝수 (2로 나누어서 떨어지는 경우)
        continue # 반복문의 맨 위로 이동

    print(num) # 홀수만 출력
```

1
3
5
7
9

- 1 ~ 40 범위의 숫자 중 3의 배수만 출력

```
num = 0

while num < 40:
    num = num + 1

    if num % 3 != 0: # 3의 배수가 아닌 경우
        continue # 반복문의 맨 위로 이동

    print(num) # 3의 배수만 출력
```

3
6
9
12
15
18
21
24
27
30
33
36
39

■ 반복문 - for

● for문의 기본 구조

```
for [변수] in [자료형]: # String, List, Tuple, Dictionary, Set
```

```
    ■ 수행할 문장
```

```
    ■ 수행할 문장
```

```
    ■ 수행할 문장
```

- 첫번째 요소부터 마지막 요소까지 차례로 변수에 대입되어 반복 실행
- 반복 횟수가 이미 정해져 있음

```
# data = 'Python'
# data = [1, 2, 3]
# data = (1, 2, 3)
# data = { 'a' : 1, 'b' : 2, 'c' : 3 }
data = {1, 2, 3}
```

```
for item in data:
    print(item)
```

■ 반복문 - for

● 두 개 이상의 변수 사용

- List [Tuple, Tuple, ...]

```
data = [ (1, 2), (3, 4), (5, 6) ]  
for a, b in data:  
    print(a, b)
```

- Tuple [List, List, ...]

```
data = ( [1, 2, 'a'], [3, 4, 'b'], [5, 6, 'c'] )  
for a, b, c in data:  
    print(a, b, c)
```

- List [List, List, ...]

```
data = [ [1, 2, 'a'], [3, 4, 'b'], [5, 6, 'c'] ]  
for a, b, c in data:  
    print(a, b, c)
```

- List [Dictionary, Dictionary, ...]

```
data = [ {'a' : 1, 'b' : 2}, {'c' : 3, 'd' : 4} ]  
for a, b in data:  
    print(a, b)
```

- List [Set, Set, ...]

```
data = [ {1, 2, 'a'}, {3, 4, 'b'}, {5, 6, 'c'} ]  
for a, b, c in data:  
    print(a, b, c)
```

- a, b = ('python', 'variable') # 튜플 이용 a = 'python' b = 'variable'

- a, b = ['python', 'variable'] # 리스트 이용 a = 'python' b = 'variable'

■ 반복문 - for

● 점수 합계 구하기

```
score = [ 90, 88, 54, 70, 66 ]  
  
total = 0  
  
for s in score:  
    total += s  
  
print('합계', total, '점')
```

합계 368 점

● 점수 평균 구하기

```
score = [ 90, 88, 54, 70, 66 ]  
  
total = 0  
count = 0  
  
for s in score:  
    count += 1  
    total += s  
  
average = total / count  
  
print('평균', average, '점')
```

평균 73.6 점

■ 반복문 – for

- break (5개 과목 중 60점 미만의 점수가 있으면 반복문 종료)

```
score = [ 90, 88, 54, 70, 66 ]

for s in score:
    if s < 60:
        print('60점 미만 반복문 종료')
        break
```

60점 미만 반복문 종료

- continue (5개 과목 중 60점 이상으로 합격된 점수만 출력)

```
score = [ 90, 88, 54, 70, 66 ]

for s in score:
    if s < 60:
        continue

    print('합격', s, '점')
```

합격	90	점
합격	88	점
합격	70	점
합격	66	점

■ 반복문 - for

● range() : 반복 범위 지정

```
for idx in range(0, 10): # 또는 range(10)
    print(idx + 1, '번째 반복')
```

- 숫자 목록을 만들어주는 함수
- 마지막 숫자는 포함되지 않음

```
1
2
3
4
5
6
7
8
9
10
(0)번째 반복
(1)번째 반복
(2)번째 반복
(3)번째 반복
(4)번째 반복
(5)번째 반복
(6)번째 반복
(7)번째 반복
(8)번째 반복
(9)번째 반복
```

● 1 부터 100 까지의 합 구하기

```
total = 0

for idx in range(1, 101):
    total = total + idx

print('1 ~ 100 까지의 합', total)
```

```
1 ~ 100 까지의 합 5050
```

■ 반복문 - for

● range() 역순 - reversed

```
for idx in reversed(range(0, 10)):  
    print(idx + 1, '번째 반복')
```

- 숫자 목록을 만들어주는 함수
- 마지막 숫자는 포함되지 않음

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

● range() 역순 - 2

```
for idx in range(10, 0, -1):  
    print(idx, '번째 반복')
```

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

■ 반복문 - for

● 구구단 출력 (2단)

```
for a in range(1, 10):  
    print('2 *', a, '=', (2 * a))
```

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

● 구구단 출력 (2 ~ 5단)

```
for a in range(2, 6):  
    for b in range(1, 10):  
        print(a, '*', b, '=', (a * b))
```

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
3	*	1	=	3
3	*	2	=	6
3	*	3	=	9
3	*	4	=	12
3	*	5	=	15
4	*	5	=	20
4	*	6	=	24
4	*	7	=	28
4	*	8	=	32
4	*	9	=	36
5	*	1	=	5
5	*	2	=	10
5	*	3	=	15
5	*	4	=	20
5	*	5	=	25
5	*	6	=	30
5	*	7	=	35
5	*	8	=	40
5	*	9	=	45

■ 반복문 - for

● List Comprehension

- 특정 상황에 따라 List를 생성
- 기본 표현 방법

[표현식 for 항목 in 반복가능객체 if 조건]

- List [1, 2, 3, 4]의 각 요소에 10을 곱하여 새로 List 생성

```
a = [1, 2, 3, 4]
result = []

for n in a:
    result += [n * 3] # result.append(n * 3)

print(result)
```



```
a = [1, 2, 3, 4]
result = [n * 3 for n in a]
print(result)
```

[3, 6, 9, 12]



조건문 추가 가능 (2의 배수만)

```
a = [1, 2, 3, 4]
result = [n * 3 for n in a if n % 2 == 0]
print(result)
```

[6, 12]

■ 반복문 - for

● List Comprehension

- 중첩 for문 사용 (구구단 2단)

```
result = []  
for x in range(2, 3):  
    for y in range(1, 10):  
        result.append(x * y) # result += [x * y]  
  
print(result)
```



```
result = [x * y for x in range(2, 3) for y in range(1, 10)]  
print(result)
```

```
[2, 4, 6, 8, 10, 12, 14, 16, 18]
```