

머신러닝/딥러닝을 위한

파이썬(Python)

– if • for • while –

조건문 - if

if condition:

```
a = 1

if a > 0:
    print("a ==", a)
    print("positive number")
elif a == 0:
    print("a ==", a)
    print("zero")
else:
    print("a ==", a)
    print("negative number")
```

```
a == 1
positive number
```

if condition in list, dict...:

```
list_data = [ 10, 20, 30, 40, 50 ]
dict_data = { 'key1': 1, 'key2' : 2 }

if 45 in list_data:
    print("45 is in list_data")
else:
    print("45 is not in list_data")

if 'key1' in dict_data:
    print("key1 is in dict_data")
else:
    print("key1 is not in dict_data")
```

```
45 is not in list_data
key1 is in dict_data
```

- 파이썬은 코딩블럭을 표시하기 위해 **들여쓰기(indentation)**를 사용함. 즉 C, Java 등과 같이 코딩블럭을 { } 로 나타내지 않음
- 동일한 블럭의 들여쓰기는 모두 동일한 수의 공백을 사용해야 함.
예를들어 4개의 공백을 사용하다가 하나의 블럭만 5개의 공백을 사용하거나, 공백과 탭을 혼용해서 사용한다면 **IndentationError: unexpected indent** 에러가 발생함

반복문 - for

for variable **in** range(...):

```
for data in range(10): # range() 함수는 시작 값~ 마지막 값 -1  
    print(data, " ", end='')
```

0 1 2 3 4 5 6 7 8 9

```
for data in range(0, 10): # range() 함수는 시작 값~ 마지막 값 -1  
    print(data, " ", end='')
```

0 1 2 3 4 5 6 7 8 9

```
for data in range(0, 10, 2): # range() 함수는 시작 값~ 마지막 값 -1  
    print(data, " ", end='')
```

0 2 4 6 8

for variable **in** list, dict...:

```
list_data = [ 10, 20, 30, 40, 50 ]
```

```
for data in list_data:  
    print(data, " ", end='') # 줄 바꿈 없음
```

10 20 30 40 50

```
dict_data = { 'key1': 1, 'key2' : 2 }
```

```
for data in dict_data:  
    print(data, " ", end='') # 줄 바꿈 없음
```

key1 key2

```
for key, value in dict_data.items():  
    print(key, " ", value)
```

key1 1
key2 2

반복문 – list comprehension

- 리스트의 [...] 괄호 안에 for 루프를 사용하여 반복적으로 표현식 (expression)을 실행해서 리스트 요소들을 정의하는 방법을 **List Comprehension** 이라 하며, 머신러닝 코드에서 자주 사용되는 기법

```
list_data = [ x**2 for x in range(5) ]  
print(list_data)
```

```
[0, 1, 4, 9, 16]
```

```
raw_data = [ [1, 10], [2, 15], [3, 30], [4, 55] ]  
  
all_data = [ x for x in raw_data ]  
x_data = [ x[0] for x in raw_data ]  
y_data = [ x[1] for x in raw_data ]  
  
print("all_data ==", all_data)  
print("x_data ==", x_data)  
print("y_data ==", y_data)
```

```
all_data == [[1, 10], [2, 15], [3, 30], [4, 55]]  
x_data == [1, 2, 3, 4]  
y_data == [10, 15, 30, 55]
```

반복문 – while, break, continue

while condition:

```
data = 5

while data >= 0:
    print("data ==", data)
    data -= 1
```

```
data == 5
data == 4
data == 3
data == 2
data == 1
data == 0
```

break, continue

```
data = 5

while data >= 0:
    print("data ==", data)
    data -= 1

    if data == 2:
        print("break here")
        break
    else:
        print("continue here")
        continue
```

```
data == 5
continue here
data == 4
continue here
data == 3
break here
```

[예제 1] 아래와 같은 패턴의 별(*)을 출력하는 프로그램을 end=' ' 을 이용하여 작성하시오

```
*****
```

[예제 2] 아래와 같은 패턴의 별(*)을 출력하는 프로그램을 end=' '을 이용하여 작성하시오

```
*****
```

```
****
```

```
***
```

```
**
```

```
*
```

[예제 3] 중첩 루프를 이용해 신문 배달을 하는 프로그램을 작성하시오. 단, 아래에서 arrears 리스트는 신문 구독료가 미납된 세대에 대한 정보를 포함하고 있는데, 해당 세대에는 신문을 배달하지 않아야 함

```
apart = [[101, 102, 103, 104],[201, 202, 203, 204],[301, 302, 303, 304], [401, 402, 403, 404]]  
arrears = [101, 203, 301, 404]
```

[예제 4] 짝수 출력하는 다음 코드를 List Comprehension 으로 구현하시오

```
even_number = []  
  
for data in range(10):  
    if data % 2 == 0:  
        even_number.append(data)  
  
print(even_number)  
  
[0, 2, 4, 6, 8]
```

[예제 5] my_list = ['A', 'B', 'C', 'D'] 에서, 파이썬 for 문과 슬라이싱을 이용하여 첫 번째 데이터를 제외하고 나머지 데이터를 출력하시오

[예제 6] my_list = [3, 1, 7, 12, 5, 16] 에서, 3의 배수이거나 4의 배수를 화면에 출력하시오

[예제 7] test_data= [[10, 20, 30], [1, 2, 3], [100, 200, 300]] 에 대하여, 다음의 xdata , tdata 를 구하시오

xdata= [x[1:-1] for x in test_data]





tdata= [x[-1] for x in test_data]

[예제 8] train 디렉토리에 다음과 같은 파일이 있을 때,
labels 값을 구하시오 (윈도우에서는 디렉토리 분리자로
'\\', 리눅스에서는 '/' 를 사용하시오)

```
import glob
```

```
files = glob.glob('train/*')
```

```
labels = [ fn.split('\\')[1].split('.')[0].strip() for fn in files ]
```

이름	유형
 cat.6	JPG 파일
 cat.9	JPG 파일
 dog.6316	JPG 파일
 dog.6331	JPG 파일