Python for Machine Learning - Basic

파이썬은 그 자체로 훌륭한 범용 프로그래밍 언어이면서, 몇 가지 인기있는 라이브러리(numpy, matplotlib, scipy)의 도움을 받아 과학분야 컴퓨팅을 이용한 강력한 환경을 제공한다. 그 중에서 머신러닝에 꼭 필요한 기능들만을 요약해서 설명한다.

In [ ]:
```
#Hello World!출력 예제
print("Hello World!")
```

```
Hello World!
```

1. Data type

Numbers

In [ ]:
```
x=5 #정수
print('x= ', x, type(x))#변수 값과 타입 출력
print('x+1= ', x+1)#덧셈
print('x-1', x-1)#뺄셈
print('x*1= ', x*1)#곱셈
print('x**2= ', x**2)#제곱
print('x/2= ', x/2)#나눗셈
print('x//2= ', x//2)#몫
print('x%2= ', x%2)#나머지

x+=1;print('x+=1 => ', x)#복합 연산(x=x+1)

y=2.5#실수(float)를 변수로 저장
print('y= ', y, type(y))
print(y, y+1, y-1, y*2, y**2)
```

```
x=  5 <class 'int'>
x+1=  6
x-1 4
x*1=  5
x**2=  25
x/2=  2.5
x//2=  2
x%2=  1
x+=1 =>  6
y=  2.5 <class 'float'>
2.5 3.5 1.5 5.0 6.25
```

주의: 많은 다른 프로그래밍 언어들과는 달리 파이썬에는 단항 증가(x++) 또는 단항 감소(x--) 연산자가 없다.

Booleans

파이썬은 불리언 논리에 대한 모든 일반적인 연산자를 구현하지만 기호(&&, ||등)보다는 영어단어(and, or, not)를 사용한다.

In [ ]:
```
t=True
f=False
print(type(t))#논리값 타입
print('t and f= ', t and f)#True and False
print('t or f= ', t or f)#True or False
print('not t= ', not t)#not Ture
print('t!=f', t!=f)#True !=(not equal) False
```

```
<class 'bool'>
t and f=  False
t or f=  True
not t=  False
t!=f True
```

Strings

파이썬은 문자열을 크게 지원한다.

In [ ]:
```
h="hello"
w="world"
print(h)
print(len(h))#문자열의 길이
hw=h+' '+w
print(hw)
greeting='''
Hi!
    Good morning...
            nice to meet you
```

```
'''
print(greeting)
no=2023
hw12=f'{h} {w} {no}'
print(f'Happy New {no}')
```

```
hello
5
hello world

Hi!
    Good morning...
            nice to meet you

Happy New 2023
```

String 객체는 유용한 메소드를 많이 가지고 있다.

In [ ]:
```
s="hello"
print(s.capitalize())#첫 글자를 대문자로
print(s.upper())#모두 대문자로
print(s.rjust(10))#오른쪽으로 정렬(왼쪽에 5개의 공백(전체 너비 개수=10))
print(s.center(11))#중앙정렬
print(s.replace('l', 'g'))#'l'을 'g'로 대체
print( '    world'.strip())#공백 지우기
```

```
Hello
HELLO
     hello
   hello
heggo
world
```

(링크)모든 문자열 메서드 목록

5. Containers

여러 데이터들을 묶어서 하나로 다루는데 유용한 파이썬의 기본 내장 컨테이너 유형에는 list, dictionary, set, tuple등이 있다.

5.1. List

리스트는 파이썬에서의 배열로써 크기를 변경할 수 있고(resizeable). 서로 다른 유형의 요소(숫자, 문자. 리스트 등)들을 포함할 수 있다.

In [ ]:
```
xs=[3, 1, 2]#리스트를 변수로 저장
print(xs, xs[2])#2번 인덱스(3번째)에 있는 요소
print(xs[-1])#마지막 인덱스(음수는 끝에서부터 센다.)
xs[2]='foo'#주어진 인덱스의 값을 변경
print(xs)
xs.append('bar')#마지막 값으로 추가
print(xs)
x=xs.pop()#마지막 요소를 끄집어내기
print(x, xs)
```

```
[3, 1, 2] 2
2
[3, 1, 'foo']
[3, 1, 'foo', 'bar']
bar [3, 1, 'foo']
```

In [ ]:
```
list_1=[1, 2, 3]#변수명에 올 수 있는 기호: 문자, 숫자, 언더바(_)
print(list_1)
list_1.append("a")
print(list_1)
print(list_1[:2])# 처음부터 2번 인덱스 전까지(인덱스 0에서 인덱스 1까지)
del list_1[1]
list_1.pop()
print(list_1)
```

```
[1, 2, 3]
[1, 2, 3, 'a']
[1, 2]
[1, 3]
```

Slicing: 목록 요소를 한 번에 하나씩 액세스하는 것 외에도 Python은 일부 목록에 액세스하기 위한 간결한 구문을 제공합니다. 이를 **"Slicing"**이라 부릅니다.

In [ ]:
```
nums=list(range(5))#00부터 시작해서 5라는 숫자 앞까지 숫자로 리턴
print(nums)
print(nums[2:4])#2부터 4번 인덱스 전까지
print(nums[2: ])#2부터 마지막 인덱스 전까지
print(nums[ :2])#0부터 2번 인덱스 전까지
print(nums[:-1])#0부터 마지막 인덱스 전까지
nums[2:4]=[8, 9]#2, 3이 8, 9로 바뀌치기됨
print(nums)
```

```
[0, 1, 2, 3, 4]
[2, 3]
[2, 3, 4]
[0, 1]
[0, 1, 2, 3]
[0, 1, 8, 9, 4]
```

프로그래밍 할 때 종종 한 유형의 데이터를 다른 데이터로 변환할 경우가 있다. 간단한 예로, 제곱 값을 계산하는 다음 코드를 생각해 보자.

In [ ]:
```
nums=list(range(5))
squares=[]
for x in nums:
    squares.append(x**2)
print(squares)
```

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

List Comprehension을 사용하여, 이 코드를 더 간단하게 만들 수 있다.

In [ ]:
```
squares=[x**2 for x in nums]
print(squares)
```

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

In [ ]:
```
even_squares=[x**2 for x in nums if x%2==0]
print(even_squares)
```

```
[0, 4, 16]
```

5.2. Dictionary

Dictionary는 Java의 Map 또는 JS의 오브젝트와 유사한(키, 값)쌍을 저장한다. 다음과 같이 사용할 수 있다.

In [ ]:
```
d={'cat':'cute', 'dog':'furry'}
print(d['cat'])#'cat'이라는 key 값을 사용
print('cat' in d)#d에 'cat'값이 있는지를 확인
d['fish']='wet'#(key, value)요소 추가
print(d)
print(d['fish'])
del d['fish']
print(d)
```

```
cute
True
{'cat': 'cute', 'dog': 'furry', 'fish': 'wet'}
wet
{'cat': 'cute', 'dog': 'furry'}
```

Loops: dictionary의 키를 반복하는 것은 쉽다.

In [ ]:
```
d={'person':2, 'cat':4, 'spyder':8}
for animal in d:
    legs=d[animal]#하나의 특정한 값을 접근하기 위해 [] 사용
    print('A %s has %d legs' %(animal, legs))
```

```
A person has 2 legs
A cat has 4 legs
A spyder has 8 legs
```

키와 해당 값에 액세스하려면 items 메서드를 사용한다.

In [ ]:
```
d={'person':2, 'cat':4, 'spyder':8}
for animal, legs in d.items():
    print('A %s has %d legs' %(animal, legs))#key, value값 모두 리턴
```

```
A person has 2 legs
A cat has 4 legs
A spyder has 8 legs
```

Dictionary comprehension: 이것은 List Comprehention과 유사하지만, 쉽게 사전을 구성할 수 있게 해준다.

In [ ]:
```
nums=list(range(5))
even_num_to_squares={x:x**2 for x in nums if x%2==0}#2의 배수만 골라서 넣어주기
print(even_num_to_squares)
```

```
{0: 0, 2: 4, 4: 16}
```

5.3. Tuples

튜플을 (1), (3, 2) 또는 (4, 2, 1)... 과 같은 괄호 안의 쉼표로 구분된 숫자들로 구성되며, 값이 **불변(변경이 불가함)하는** 목록이다. 튜플은 여러 면에서 리스트와 유사하다. 가장 중요한 차이점 중 하나는 튜플을 **Dictionary**의 키로 사용할 수 있고, **list**의 요소로 사용할 수 있다는 것이다 **numpy의 배열(array)**에서는 모양을 정의할 때 사용한다.

In [ ]:
```
d={(x, x+1): x for x in range(10)}
```

```
print(d)
t=(5, 6)
print(type(t))
print(d[t])
print(d[(1, 2)])
```

```
{(0, 1): 0, (1, 2): 1, (2, 3): 2, (3, 4): 3, (4, 5): 4, (5, 6): 5, (6, 7): 6, (7, 8): 7, (8, 9): 8, (9, 10): 9}
<class 'tuple'>
5
1
```

6. Function

파이썬 함수는 **def**키워드를 사용하여 정의된다.

In [ ]:
```
def sign(x):
    if x>0:
        return 'positive'
    elif x<0:
        return 'nagative'
    else:
        return 'zero'

for x in [1, 0, -1]:
    print(sign(x))
```

```
positive
zero
nagative
```

다음과 같이 선택적 키워드 인수를 취하는 함수를 정의한다.

In [ ]:
```
def hello(name, loud=False): # loud의 default값 = False
    if loud:
        print(f'HELLO, {name.upper()}!')
    else:
        print(f'Hello, {name}!')

hello('Bob')
hello('Fred', loud=True)
```

```
Hello, Bob!
HELLO, FRED!
```