방학 1주차

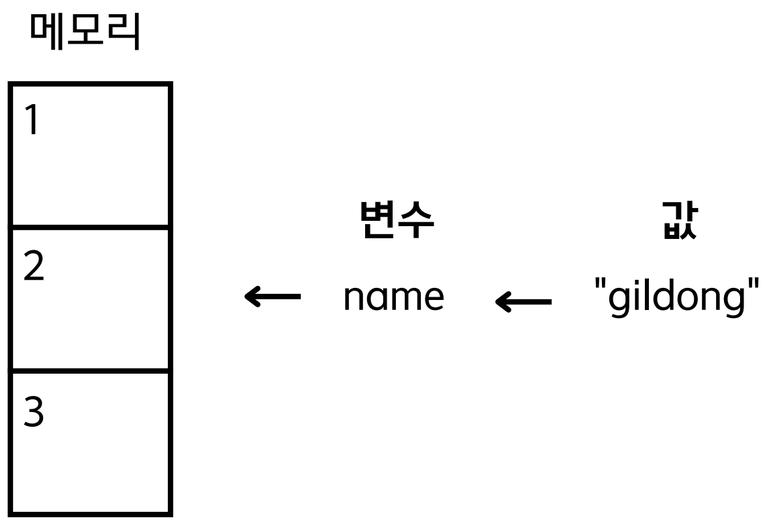
ML/DS에서 사용하는 라이브러리

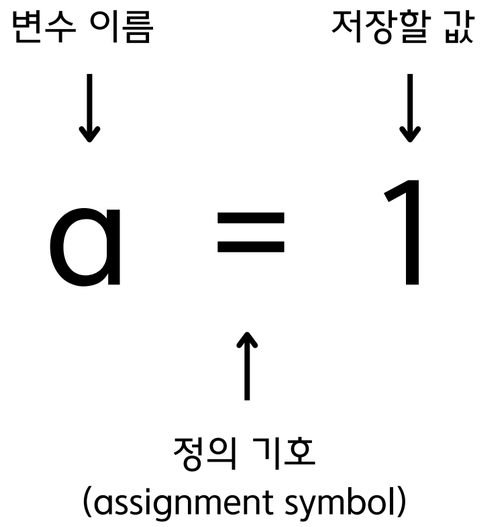
NEKA

Python 기초

(1) 변수 선언하기 (Assign)

변수 = 데이터 값을 저장하는 공간





Python 기초

(2) 변수 작명 협약 (Naming Convention)

작명 스타일 (PEP 8)

(1) ㅁ : 단일 소문자

(2) A : 단일 대문자

(3) productcode : 소문자로 표기 (lowercase) → 모듈 이름을 지을 때 권장됨 (ex - import numpy)

(4) PRODUCTCODE : 대문자로 표기 (uppercase) → 상수를 정의할 때 권장됨 (ex - LED1 = 3)

(5) product_code : 밑줄로 구분된 소문자 → 변수 이름을 지을 때 권장됨 (ex - temp_value = 0)

(6) PRODUCT_CODE : 밑줄로 구분된 대문자

(7) ProductCode : CamelCase / CapWords → 클래스 이름을 지을 때 권장됨

Python 기초

(3) 자주 쓰이는 내장형(Built-in) Types

Numeric Types

int

정수 값을 표현함

가능한 메소드 :
isinstance(), issubclass(), iter(), len()

float

실수 값을 표현함

파이썬에는 double이 X

가능한 메소드 :
hash(), hex(), id() 등

Python 기초

(3) 자주 쓰이는 내장형(Built-in) Types

Iterator Type

객체가 가지고 있는 요소들을
순차적으로 접근하게 해주는 타입

iterator가 작동 가능한(=iterable) 객체 종류

string, list, tuple, dictionary, set, range, ...

Python 기초

(3) 자주 쓰이는 내장형(Built-in) Types

Sequence Types

list

Mutable Suquence
(복사해도 메모리 위치가 같음)

일반적으로 비슷한 애들끼리
묶을 때 씀

ex : [1, 2, 3, 4, 5]

tuple

Immutable Suquence
(복사하면 메모리 위치가 달라짐)

일반적으로 다른 애들끼리
묶을 때 씀

ex : (1, 2, 3, 4, 5)

range

범위에 들어가는
정수들을 생성함

Python 기초

(3) 자주 쓰이는 내장형(Built-in) Types

Sequence Type

str

텍스트 데이터를 다룰 때 씀.

따옴표 1~3개까지 모두 가능함.
(여러줄로 쓸 때는 3개를 씀)

가능한 메소드 :

capitalize(), find(), isdigit() 등

Set Type

set

hashable collection
(해싱이 가능한 객체)

순서 X, 중복 X

ex : {1, 2, 3, 4, 5}

Mapping Type

dict

hashable collection
(해싱이 가능한 객체)

key와 value를 저장함

ex : {'a' : 1, 'b'' : 2 }

Python 기초

(3) 자주 쓰이는 내장형(Built-in) Types

Data Type	int	float	str	list	tuple	set	dict
iterable	X	X	O	O	O	O	O
mutable	X	X	X	O	X	O	O
ordered	X	X	O	O	O	X	X
hashable	O	O	O	X	O	X	X

Python 기초

(4) Python에서 라이브러리를 사용하는 법

anaconda

라이브러리를 쉽게 관리하고 설치
할 수 있도록 도와주는 도구

설치 예시 : **conda install pandas**

pip

python의 패키지 관리 도구

설치 예시 : **pip install pandas**

NumPy

(1) NumPy란?

- 수학/과학에서의 연산을 위한 라이브러리
- 호출 방식 : import numpy
- ndarray라는 객체를 기반으로 어려가지 기능을 제공함.
- 대표적인 함수 :

  (1) np.array(list) : ndarray를 생성함.

  (2) np.zeros(int) : int 개수 만큼의 0으로 이루어진 ndarray를 생성함.

  (3) np.arange() : range 와 유사하지만 ndarray를 출력함.

  (4) np.shape() : ndarray의 형태를 알려줌.

NumPy

(2) 실습하기

```
import numpy as np

arr = np.array([1, 2, 3])

print(arr)
print(type(arr))
```

RESULT

```
>>> import numpy as np
>>>
>>> arr = np.array([1, 2, 3])
...
>>> print(arr)
[1 2 3]
>>>
>>> print(type(arr))
<class 'numpy.ndarray'>
>>>
```

NumPy

(2) 실습하기

```
import numpy as np

arr = np.arange(16)
print(arr)

arr = arr.reshape(4, 4)
print(arr)

arr = arr.reshape(2, 8)
print(arr)

arr = arr.reshape(-1, 2)
print(arr)
```

RESULT

```
import numpy as np

arr = np.arange(16)

print(arr)
[ 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15]

arr = arr.reshape(4, 4)
print(arr)
[[ 0  1  2  3]
 [ 4  5  6  7]
 [ 8  9 10 11]
 [12 13 14 15]]

arr = arr.reshape(2, 8)
arr
array([[ 0,  1,  2,  3,  4,  5,  6,  7],
       [ 8,  9, 10, 11, 12, 13, 14, 15]])

arr = arr.reshape(-1, 2)
arr
array([[ 0,  1],
       [ 2,  3],
       [ 4,  5],
       [ 6,  7],
       [ 8,  9],
       [10, 11],
       [12, 13],
       [14, 15]])
```

NumPy

(2) 실습하기

```
import numpy as np

arr = np.array([1, 5, 3, 7, 9])

sort1 = np.sort(arr)
print(sort1)
print(arr)

sort2 = arr.sort()
print(sort2)
print(arr)
```

RESULT

```
import numpy as np

arr = np.array([1, 5, 3, 7, 9])

sort1 = np.sort(arr)
print(sort1)
[1 3 5 7 9]
print(arr)
[1 5 3 7 9]

sort2 = arr.sort()
print(sort2)
None
print(arr)
[1 3 5 7 9]
```

Pandas

(1) Pandas란?

- 데이터 분석을 위한 라이브러리
- 호출 방식 : import pandas
- dataframe 기반으로 수치형/시계열 데이터를 조작함.
- 대표적인 함수 :

  (1) pd.isna(), pd.isnull : 데이터값이 NaN 값인지 확인함.

  (2) pd.read_csv, pd.read_excel 등 : 파일로부터 데이터프레임을 생성함.

  (3) pd.Dataframe(dict) : 새로운 데이터프레임을 생성함.

  (4) (Dataframe).info() : 해당 데이터프레임에 대한 정보를 제공함.

Pandas

(2) 실습하기

```
import pandas as pd

data = {
     '이름' : ["홍길동", "성춘향", "이몽룡", "심청"]
    , '성적' : [85, 95, 75, 70]
}

df = pd.DataFrame(data)
print(df)
```

RESULT

```
import pandas as pd

data = {
     '이름' : ["홍길동", "성춘향", "이몽룡", "심청"]
    , '성적' : [85, 95, 75, 70]
}

df = pd.DataFrame(data)
print(df)
    이름  성적
0  홍길동  85
1  성춘향  95
2  이몽룡  75
3   심청  70
```

Pandas

(2) 실습하기

```
import pandas as pd

data = {
     '이름' : ["홍길동", "성춘향", "이몽룡", "심청"]
   , '성적' : [85, 95, 75, 70]
}

df = pd.DataFrame(data)

print(df.describe())
print(df['성적'].unique())
```

RESULT

```
import pandas as pd

data = {
      '이름' : ["홍길동", "성춘향", "이몽룡", "심청"]
    , '성적' : [85, 95, 75, 70]
}

print(df.describe())
             성적
count   4.000000
mean   81.250000
std    11.086779
min    70.000000
25%    73.750000
50%    80.000000
75%    87.500000
max    95.000000

print(df['성적'].unique())
[85 95 75 70]
```

Pandas

(2) 실습하기

```
import pandas as pd

data = {
      '성' : ["Hong", "Seong", "Lee", "Shim"]
    , '이름' : ["Gildong", "Chunhyang", "Mongryong",
                "Cheong"]
}

df = pd.DataFrame(data)

df['성_소문자'] = df['성'].str.lower()
df['이름_대문자'] = df['이름'].str.upper()

print(df)
```

RESULT

```
import pandas as pd

data = {
      '성' : ["Hong", "Seong", "Lee", "Shim"]
    , '이름' : ["Gildong", "Chunhyang", "Mongryong", "Cheong"]
}

df = pd.DataFrame(data)

df['성_소문자'] = df['성'].str.lower()
df['이름_대문자'] = df['이름'].str.upper()

print(df)
      성         이름  성_소문자     이름_대문자
0   Hong    Gildong   hong    GILDONG
1  Seong  Chunhyang  seong  CHUNHYANG
2    Lee  Mongryong    lee  MONGRYONG
3   Shim     Cheong   shim     CHEONG
```

PyTorch

(1) PyTorch란?

- 딥러닝 모델을 만들기 위한 라이브러리
- 호출 방식 : import torch
- 페이스북(현재 META)에서 개발하고 있음.
- 대표적인 클래스 :

  (1) torch.nn : 신경망을 구축하기 위한 것들을 제공함.

  (2) torch.optim : optimizer을 구축하기 위한 것들을 제공함.

  (3) torch.utils : dataloader와 같은 핵심 유틸리티들을 제공함.

  (4) torch.Tensor : 행렬과 같은 역할을 하는 텐서 객체를 생성함.

PyTorch

(2) PyTorch vs Tensorflow

- 딥러닝 모델을 만들기 위한 라이브러리

	개발사	GPU 할당	**선호도**	동작 방식	**난이도**	**커뮤니티**
PyTorch	Meta	수동	Pytorch가 압도적임	동적 그래프*	PyTorch가 더 쉬움	Tensorflow 쪽이 더 큼
Tensorflow	Google	자동		정적 그래프**		

- 동적 그래프 : Define by Run. 프레임워크가 처리하는 과정에서 정의됨.
- 정적 그래프 : Define and Run. 정의는 사용자가 제공하고, 프레임워크는 이를 변환하여 처리함.

PyTorch

(2) 실습하기

```
import torch

mat1 = torch.Tensor([[1, 2, 3], [4, 5, 6]])

print(mat1)

print(mat1.shape)
```

RESULT

```
import torch

mat1 = torch.Tensor([[1, 2, 3], [4, 5, 6]])

print(mat1)
tensor([[1., 2., 3.],
        [4., 5., 6.]])

print(mat1.shape)
torch.Size([2, 3])
```

PyTorch

(2) 실습하기

```
import torch

mat1 = torch.Tensor([[1, 2, 3], [1, 2, 3], [1, 2, 3]])

mat2 = torch.Tensor([[1, 2, 3], [1, 2, 3], [1, 2, 3]])

res_matmul = torch.matmul(mat1, mat2)
print(res_matmul)

res_mul = torch.mul(mat1, mat2)
print(res_mul)
```

RESULT

```
import torch

mat1 = torch.Tensor([[1, 2, 3], [1, 2, 3], [1, 2, 3]])

mat2 = torch.Tensor([[1, 2, 3], [1, 2, 3], [1, 2, 3]])

res_matmul = torch.matmul(mat1, mat2)
print(res_matmul)
tensor([[ 6., 12., 18.],
        [ 6., 12., 18.],
        [ 6., 12., 18.]])

res_mul = torch.mul(mat1, mat2)
print(res_mul)
tensor([[1., 4., 9.],
        [1., 4., 9.],
        [1., 4., 9.]])
```

PyTorch

(2) 실습하기

```
import torch

mat = torch.Tensor([[1, 2, 3], [4, 5, 6]])

min_mat = torch.min(mat)
minloc_mat = torch.argmin(mat)
print("Minimum Value : ", min_mat, " / Location : ",
          minloc_mat)

max_mat = torch.max(mat)
maxloc_mat = torch.argmax(mat)
print("Maximum Value : ", max_mat, " / Location : ",
          maxloc_mat)

mean_mat = torch.mean(mat)
print(mean_mat)
```

RESULT

```
import torch

mat = torch.Tensor([[1, 2, 3], [4, 5, 6]])

min_mat = torch.min(mat)
minloc_mat = torch.argmin(mat)

print("Minimum Value : ", min_mat, " / Location : ", minloc_mat)
Minimum Value :  tensor(1.)  / Location :  tensor(0)

max_mat = torch.max(mat)
maxloc_mat = torch.argmax(mat)

print("Maximum Value : ", max_mat, " / Location : ", maxloc_mat)
Maximum Value :  tensor(6.)  / Location :  tensor(5)

mean_mat = torch.mean(mat)
print(mean_mat)
tensor(3.5000)
```

HOMEWORK

1. 코드 돌려 보고 결과물 출력해보기 (1)

```
import numpy as np
import pandas as pd

score = np.array([[70, 45, 60],
                  [30, 25, 95],
                  [23, 53, 100]])

df = pd.DataFrame(score, columns=['Math', 'Chemistry', 'Physics'])

print(df)
```

HOMEWORK

1. 코드 돌려 보고 결과물 출력해보기 (2)

```
import numpy as np
import pandas as pd

data = {
    'A' : np.random.randint(1, 10, 5),
    'B' : np.random.randint(1, 10, 5)
}

df = pd.DataFrame(data)

df['C'] = df['A'] + df['B']
print(df)

df['D'] = df.mean(axis=1)
print(df)
```

HOMEWORK

2. 코드를 보고 오류 찾기 (1)

```
import pandas as pd
import numpy as np

data = {
    'Name' : ['철수', '영희', '진희', '정혁'],
    'Age'  : [17, 23, 42, 29],
    'City' : ['서울', '부산', '인천', '광주']
    }

df = pd.DataFrame(data)

df['Salary'] = [5000, 650, 2400]
```

HOMEWORK

2. 코드를 보고 오류 찾기 (2)

```
import torch

def multiply_tensors (a, b) :
    c = a * b

    return c

tensor1 = torch.tensor([2, 4, 6])
tensor2 = torch.tensor([1, 3])

result = multiply_tensors(tensor1, tensor2)
print(result)
```

HOMEWORK

3. 아래 설명을 보고 코드를 짜기 (1)

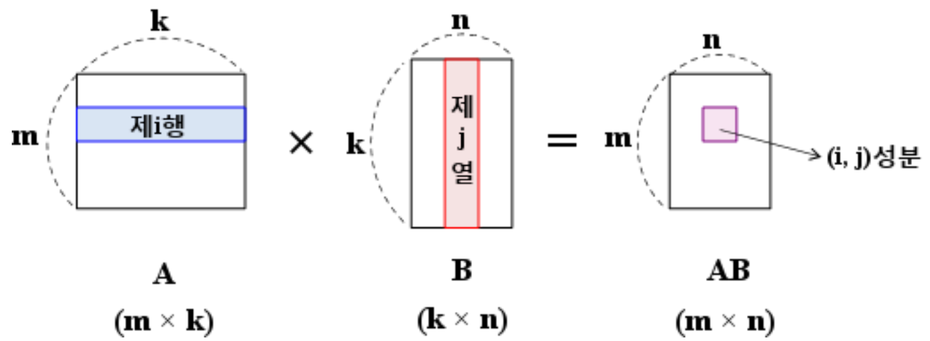
PyTorch를 이용해서 행렬 삼칙연산(덧셈, 뺄셈, 곱셈) 계산기 만들기

참고자료 :

$$A = \begin{bmatrix} 5 & -2 \\ 3 & 1 \end{bmatrix}, \quad B = \begin{bmatrix} 2 & 3 \\ 1 & -4 \end{bmatrix}$$

$$A + B = \begin{bmatrix} 7 & 1 \\ 4 & -3 \end{bmatrix}$$

$$A - B = \begin{bmatrix} 3 & -5 \\ 2 & 5 \end{bmatrix}$$



HOMEWORK

3. 아래 설명을 보고 코드를 짜기 (2)

PyTorch를 이용해서 정사각행렬을 가지고 놀아보기

(1) 정사각행렬을 만들기
(2) 90, 180, 270도 돌려보기
(3) 반대로 뒤집어보기 (flip)

참고자료 :

정사각행렬 - 행과 열의 길이가 같은 행렬

$$\begin{bmatrix} 9 & 13 & 5 & 2 \\ 1 & 11 & 7 & 6 \\ 3 & 7 & 4 & 1 \\ 6 & 0 & 7 & 10 \end{bmatrix}$$

$$A = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \quad A^T = \begin{bmatrix} a & c \\ b & d \end{bmatrix}$$

$$B = \begin{bmatrix} a & b \\ c & d \\ e & f \end{bmatrix} \quad B^T = \begin{bmatrix} a & c & e \\ b & d & f \end{bmatrix}$$

NEKA

THANK YOU