데이터분석 with 파이썬

Lesson 01_Numpy 기초문법

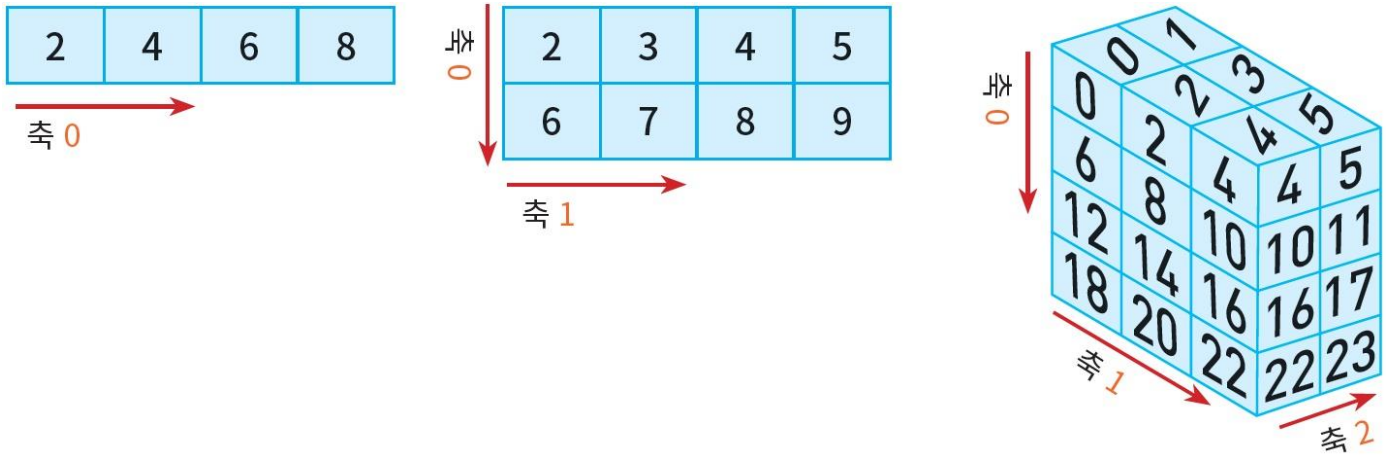
윤수연

목차

- Numpy 개요
- Numpy 설치
- Numpy의 기초

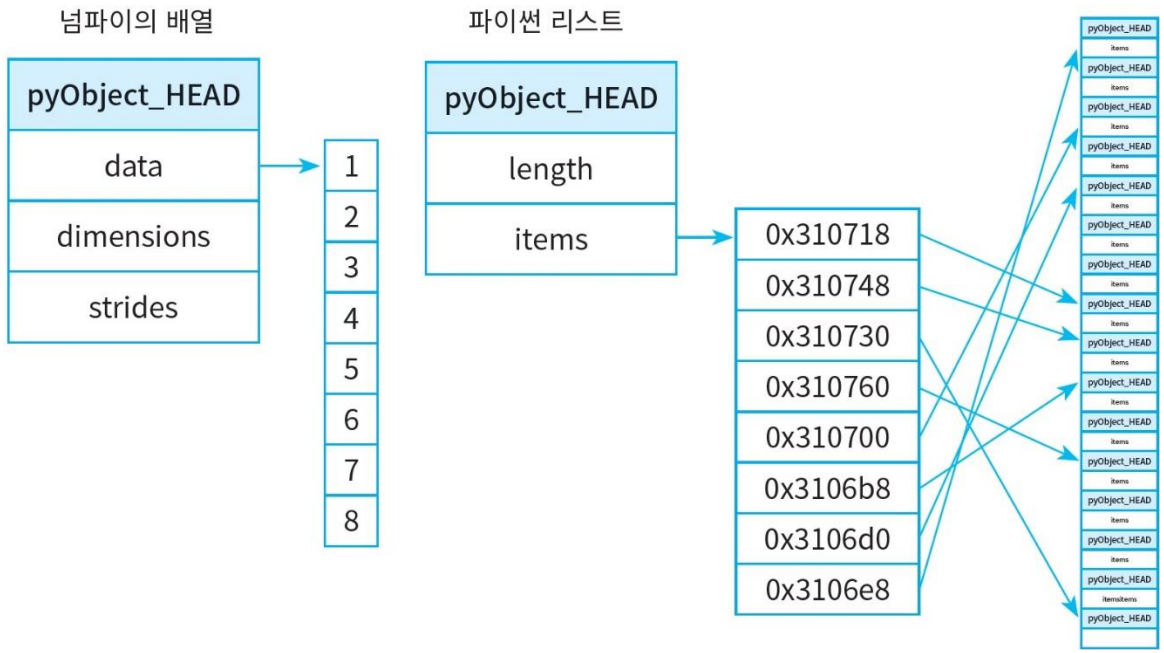
Numpy 라이브러리 개요

- Numpy (Numerical Python)
- 파이썬을 위한 행렬(matrix) 라이브러리
- 고성능의 수치계산 등 과학적 계산을 위해 C언어로 구현됨
- 이산수학과 무작위 수 생성 등 수많은 작업을 할 수 있는 기능이 제공됨
- 벡터와 행렬을 위한 특수한 배열 형식을 제공
- 생성될 때 크기가 정해짐
- 텐서플로우와 잘 어울림



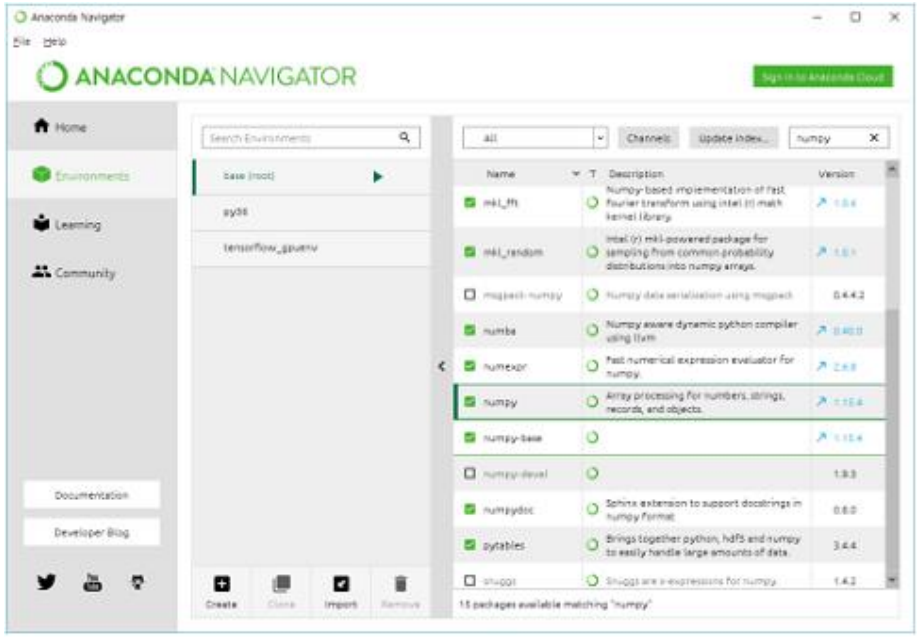
Numpy 라이브러리 개요

- 기계 학습에서는 파이썬의 기본 리스트로 충분하지 않다.
- 데이터를 처리할 때는 리스트와 리스트 간의 연산이 가능해야 하는데 파이썬의 기본 리스트는 이것을 지원하지 않기 때문이다.
- 연산 속도도 중요하기 때문에 데이터 과학자들은 기본 리스트 대신에 넘파이(Numpy)를 선호한다.



Numpy 설치 : Numpy 설정

❖ **Numpy는 일반적으로 많이 사용하는 모듈이기 때문에 기본으로 설치되어 있음**



- **아나콘다 내비게이터**
  - 박스에 체크가 되지 않았다면 체크하여 설치

- **Google Colab 환경**
  - Numpy 라이브러리가 없으면 설치하면 된다.

명령문	pip list # 설치된 라이브러리들을 보여줌
명령문	pip install numpy #라이브러리가 없다면 설치실행 import numpy as np #Numpy 라이브러리 불러오기

Numpy 문법 : 리스트 VS 넘파이 배열

- 데이터를 처리할 때는 리스트와 리스트 간의 연산이 가능해야 하는데 파이썬의 기본 리스트는 이것을 지원하지 않는다.
- 연산 속도도 중요하기 때문에 데이터 과학자들은 기본 리스트 대신에 넘파이(Numpy)를 선호한다.

- 리스트 연산 수행

```
mid_scores = [10, 20, 30]
final_scores = [70, 80, 90]
```

```
>>> total = (mid_scores + final_scores)
>>> total
[10, 20, 30, 70, 80, 90]
```

Numpy 문법 : 리스트 VS 넘파이 배열

- **Numpy array 정의 :** 기본적으로 array라는 단위로 데이터를 관리하며 이에 대해 연산을 수행
- array는 말그대로 행렬이라는 개념

- **Numpy 연산 수행**

```
import numpy as np

mid_scores  = np.array([10, 20, 30])
final_scores = np.array([70, 80, 90])

total = mid_scores + final_scores
print(total)
```

```
array([ 80, 100, 120])
```

Numpy 문법 : Numpy shape

- **shape() : array 의 형태(크기)를 확인할 수 있다.**
  - shape은 해당 클래스에서 제공해 주는 속성(attribute)
  - 1차원 배열로 3개의 원소가 있다는 결과가 나옴

명령문	print(x.shape) #(3,) 출력

  - numpy에서 지원하는 2차원 배열을 다음과 같이 만들면

명령문	a = np.array([[1, 2, 3], [2, 3, 4]])

  - a라는 배열은 2개의 원소를 가지고 있으며, 이들은 각각 3개의 원소를 가지고 있는 형태
  - "numpy 형식의 배열 a는 3개의 원소로 구성된 2개의 원소가 있다"라고 표현

Numpy 문법 : Numpy shape

- **shape()**
  - Quiz 1
    - 앞의 a 배열의 구조를 shape 속성(attribute)을 이용하여 출력하시오.

명령문	a = np.array([1, 2, 3], [2, 3, 4])
출력	(2, 3)

Numpy 문법 : numpy 자료형

- **dtype: data 의 자료형(type)을 확인할 수 있다.**
  - numpy 에서 사용되는 자료형(자료형 뒤에 붙는 숫자는 몇 비트 크기인지를 의미한다.)
  - 부호가 있는 정수 int(8, 16, 32, 64)
  - 부호가 없는 정수 uint(8 ,16, 32, 54)
  - 실수 float(16, 32, 64, 128)
  - 복소수 complex(64, 128, 256)
  - 불리언 bool
  - 문자열 string_
  - 파이썬 오프젝트 object
  - 유니코드 unicode_

출처: https://doorbw.tistory.com/171 [Tigercow.Door]

Numpy 문법 : Numpy zeros 함수

- **zeros()**
  - 괄호 안에 입력된 숫자만큼 모든 원소가 0인 array를 만든다.
  - 한 쌍의 괄호 ( ) 안에 숫자가 1개이면 벡터(1차원)를, 두 쌍의 괄호 안에 숫자가 2개이면 행렬([2차원])을 생성

명령문	np.zeros(10)
출력	array([0., 0., 0., 0., 0., 0., 0., 0., 0., 0.])

명령문	np.zeros((3, 5))
출력	array([[0., 0., 0., 0., 0.], [0., 0., 0., 0., 0.], [0., 0., 0., 0., 0.]])

Numpy 문법 : Numpy ones 함수

- **ones()**
  - 2차원 배열에 1의 값들을 채움

명령문	y = np.ones([3, 4]) print(y)
출력	[[1. 1. 1. 1.] [1. 1. 1. 1.] [1. 1. 1. 1.]]

Numpy 문법 : Numpy 평균함수

- **mean()**
  - Numpy에서 제공하는 배열을 이용하면 배열에 저장된 원소들의 평균값을 계산
  - array( )라는 메소드를 사용하여 1차원 배열 생성
  - 여기서 x라는 객체를 만들어 제공하는 mean( )이라는 메소드를 이용하여 해당값을 구할 수 있다.

명령문	① ② ③	Numpy 모듈 가져오기 파이썬의 데이터형(예 : 리스트)을 numpy 형식으로 변환 Numpy에서 제공하는 기능수행(예 : 평균)
출력	① ② ③	import numpy as np x = np.array([1, 3, 5]) print(x.mean())

Numpy 문법 : Numpy reshape 함수

- **reshape()**
  - Numpy 형식으로 배열의 원소를 입력할 때는 반드시 다음의 예시와 같이 리스트형식으로 입력

명령문	x = np.array([1, 3, 5]) (O) x = np.array(1, 3, 5) (X)

  - reshape( ) 메소드를 추가하면 다음과 같이 Numpy의 2차원 배열을 원하는 모양으로 생성할 수 있다.

명령문	x = np.array([1, 3, 5, 7, 9, 11]).reshape(3, 2) print(x)
출력	[[1 3] [5 7] [9 11]]

THANK YOU FOR YOUR ATTENTION