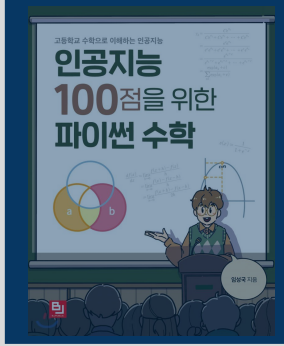
03. 파이썬 기초문법

인공지능 100점을 위한 파이썬 수학



Contents

1. 주석

2. 변수

3. 산술연산

4. 자료구조

5. 조건문

6. 반복문

Contents

7. 함수

8. 클래스와 객체

9. numpy

10. matplotlib

1. 주석

01. 주석

- 주석이란?

프로그래밍 내부에 삽입된 사람을 위한 메모

뒤의 내용은 python 인터프리터가 읽지 않는다.

01. 주석

주석이란?

프로그래밍 내부에 삽입된 메모

CODE

```
# 아래 줄에 대한 설명
print('Hello World')     # 옆 줄에 대한 설명
```

2. 변수

02. 변수

- 변수와 상수

변수 - 변할 수 있는 수(값)
특정 값을 수정하기 위해 저장할 수 있는 메모리상의 공간

상수 - 변할 수 없는 수(값)
일단 만들어진 후 수정될 수 없는 값, 읽기만 가능

02. 변수

- 변수만들기

1. 알파벳으로 시작,
2. 중간은 알파벳, 숫자, 특수문자 중 _
3. 대문자, 소문자 구분
4. 예약어 사용 불가

02. 변수

변수사용예

CODE

```
# 변수
x = 10
print(x)

x = 100
print(x)

y = 3.14
print(x*y)
print(type(x*y))
```

3. 산술연산

03. 산술연산

- 기본연산

파이썬의 산술연산으로는 기초적인 사칙연산, 정수나누기,
나머지연산, 거듭제곱등이 있다.

+, -, *, /, //, %, **

4. 자료구조

04. 자료구조

리스트

리스트는 여러 요소를 담을 수 있고, 수정, 삭제가 가능합니다.
데이터들은 [ ] 안에 담게 됩니다.

```
a = [1, 2, 3, 4, 5, 6, 7, 8]
```

리스트의 각 요소는 앞에서부터 인덱스 번호가 0부터 1씩 증가하면서 붙여집니다. 즉 a[0]은 첫 번째 요소인 1입니다. 세 번째 요소인 3은 a[2]입니다. 0부터 시작하기 때문에 n번째 요소의 인덱스번호는 n-1이 됩니다.

04. 자료구조

- 인덱스, 슬라이싱

```
a = [1, 2, 3, 4, 5, 6, 7, 8]
```

인덱스 번호를 이용해 특정 위치값을 읽거나 수정할 수 있습니다.

```
a[0]    # 1
```

인덱스 번호의 시작과 끝을 지정하면 사이의 값을 읽어옵니다.

```
a[0:3]    # [1,2,3]  - 0번부터 시작(이상) 3번 전까지(미만)
```

04. 자료구조

튜플

```
b = (1, 2, 3, 4, 5, 6, 7, 8)
```

튜플은 인덱스와 달리 수정할 수 없습니다. 한번 만들어진 튜플은 상수처럼 읽기만 가능합니다.

인덱스, 슬라이싱 모두 리스트와 같은 방식으로 사용됩니다.

04. 자료구조

딕셔너리

딕셔너리는 하나의 요소가 key와 value로 구성되어 저장, 요소는 그 요소를 가리키는 key와 설명에 해당하는 value로 구성

```
a = {'name':'Joy', 'phone':'010-0000-0123'}
```

딕셔너리는 { }로 묶이고, key와 value 사이에 :으로 구분됩니다.
키(key)를 사용해서 값(value)을 읽어옵니다.

5. 조건문

05. 조건문

if 조건문

조건이 '참'인 경우에 실행되는 구조

```
# 1.3.7 if문
hungry = True
hungry = not hungry
if hungry:
    print("i'm hungry")
else :
    print("i'm not hungry")
    print("i'm sleepy")
```

6. 반복문

06. 반복문

- for

조건이 만족하는 동안 혹은 정해진 횟수가 될 때까지 실행되는 구조

```
# 1.3.8 for문
for myNum in [1,20,3]:
    print(myNum)

for myIter in range(10):
    print(myIter)
```

7. 함수

07. 함수

def 함수 작성 예

```
# 함수1
def hello():
    print("hello world")

hello()

# 함수2
def hello(name):
    print("Hello " + name +"!")

hello("cat")
```

8. 클래스와 객체

08. 클래스와 객체

- 함수와 클래스 차이

함수 = ‘기능’ 모음

클래스 = ‘정보’ + ‘기능’ 모음

08. 클래스와 객체

구조

```
class 클래스이름:
  def __init__(self, 인수, ...):
    ...
  def 메소드이름1(self, 인수, ...):
    ...
  def 메소드이름2(self, 인수, ...):
    ...
```

9. numpy

09. numpy

numpy란?

- Numpy는 C언어로 구현된 파이썬 라이브러리
- Numerical Python 줄임말
- 고성능의 수치계산을 위해 제작
- 벡터 및 행렬 연산에 효율적

09. numpy

3.9.1 배열 만들기

CODE

```
# 3.9 배열

import numpy as np

a = np.array([1, 2, 3, 4])
print(a)
print(type(a))
```

09. numpy

3.9.2 산술연산

- 리스트와 numpy 배열의 연산 비교

CODE

```
a = [1,2,3,4]
b = [1,2,3,4]
print(a+b)
```

CODE

```
a = np.array([1,2,3,4])
b = np.array([1,2,3,4])
print(a+b)
```

09. numpy

3.9.2 산술연산

CODE

```
# 3.9.2. 산술연산
import numpy as np
a = np.array([1,2,3,4])
b = np.array([.4, .4, .3, .2])
print(a+b)
print(a-b)
print(a*b)
print(a/b)
```

배열의 사칙연산
VS
배열요소의 사칙연산

.4 는 0.4 와 동일

09. numpy

3.9.3 2차원 배열과 행렬

```
A = np.array([[1,2],
              [3,4]])
```

09. numpy

3.9.3 2차원 배열과 행렬

CODE

```
# 3.9.3. 2차원 배열과 행렬
import numpy as np
A = np.array([[1,2],[3,4]])
B = np.array([[5,6],[7,8]])
print(A+B)
print(A*B)
```

배열의 사칙연산 -- X
VS
배열요소의 사칙연산 -- O

@echo 처음코딩 in YouTube

09. numpy

3.9.4 행렬의 곱

$$\begin{pmatrix} a_1 & a_2 \\ a_3 & a_4 \end{pmatrix} \cdot \begin{pmatrix} b_1 & b_2 \\ b_3 & b_4 \end{pmatrix} = \begin{pmatrix} c_1 & c_2 \\ c_3 & c_4 \end{pmatrix}$$

$c_1 = a_1b_1+a_2b_3$

$c_2 = a_1b_2+a_2b_4$

$c_3 = a_3b_1+a_4b_3$

$c_4 = a_3b_2+a_4b_4$

09. numpy

3.9.4 행렬의 곱

$$\begin{pmatrix} a_1 & a_2 \\ a_3 & a_4 \end{pmatrix} \cdot \begin{pmatrix} b_1 & b_2 \\ b_3 & b_4 \end{pmatrix} = \begin{pmatrix} c_1 & c_2 \\ c_3 & c_4 \end{pmatrix}$$

$c_1 = a_1b_1+a_2b_3$

$c_2 = a_1b_2+a_2b_4$

$c_3 = a_3b_1+a_4b_3$

$c_4 = a_3b_2+a_4b_4$

C(1,1) = 앞 행렬의 1행과 뒤행렬의 1열을 곱한다음 더한것
C(1,2) = 앞 행렬의 1행과 뒤행렬의 2열을 곱한다음 더한것
C(2,1) = 앞 행렬의 2행과 뒤행렬의 1열을 곱한다음 더한것
C(2,2) = 앞 행렬의 2행과 뒤행렬의 2열을 곱한다음 더한것

앞 행렬의 열의 크기, 뒤 행렬의 행의 크기가 같아야 행렬곱이 가능
(2,**2**) x (**2**,2) -- OK [2,2]
(2,3) x (3,2) -- OK [2,2]
(1,10)x(10x1) -- OK [1,1]

@echo 처음코딩 in YouTube

09. numpy

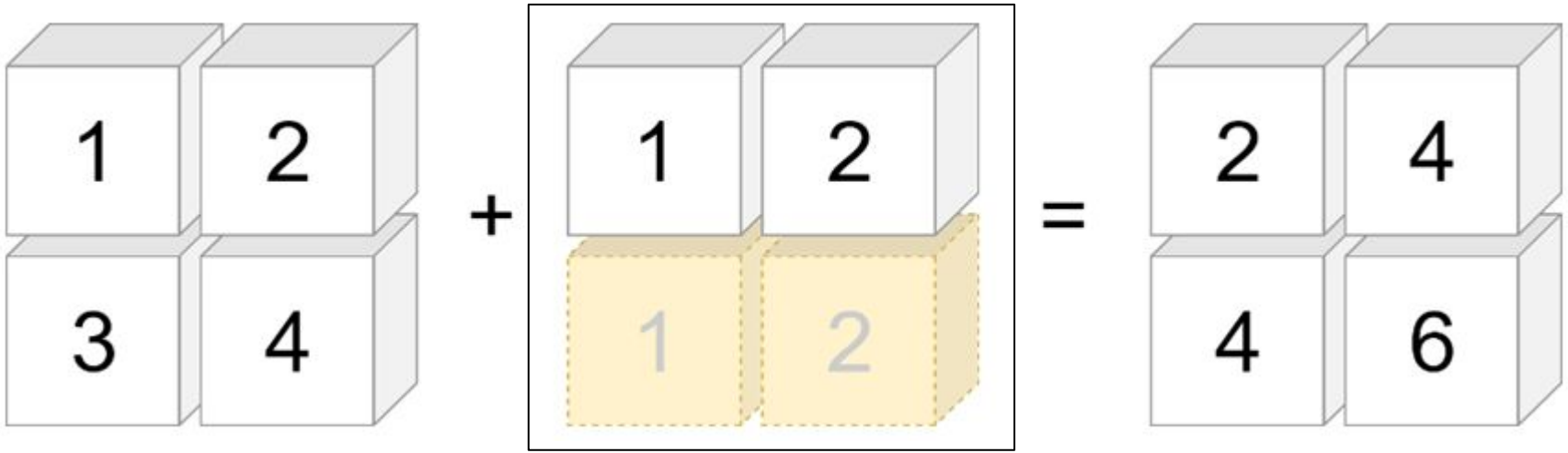
3.9.4 행렬의 곱

CODE

```
# 3.9.4. 행렬의 곱

import numpy as np

A = np.array([[1,2],[3,4]])
B = np.array([[1,2],[3,4]])
print(np.dot(A,B))
```

09. numpy

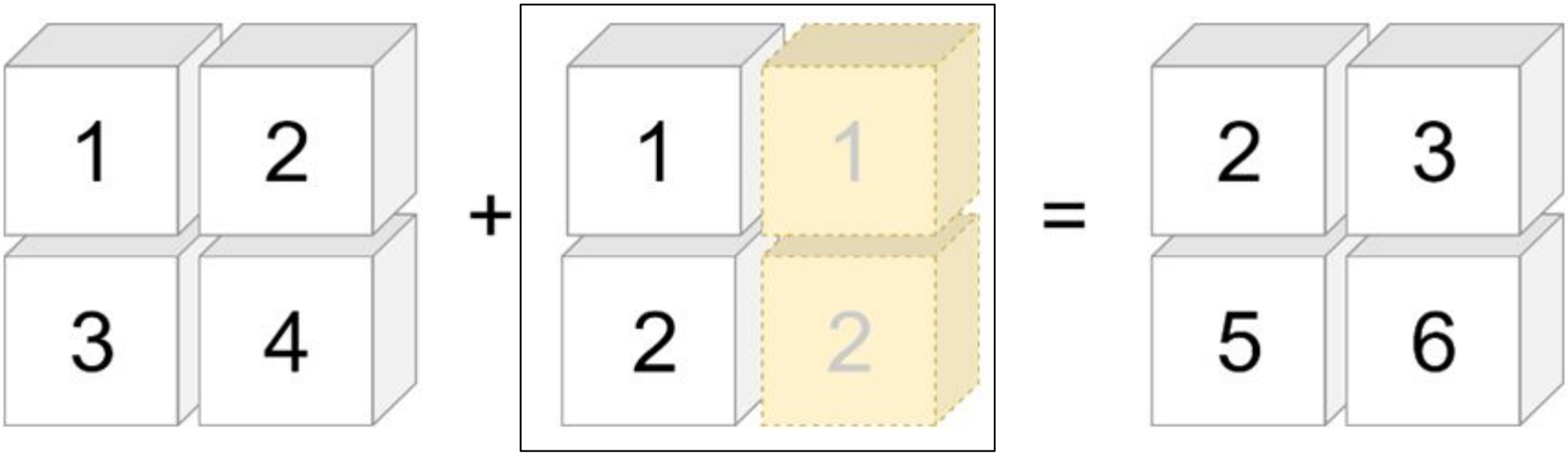
3.9.5 브로드캐스트



행 확장

09. numpy

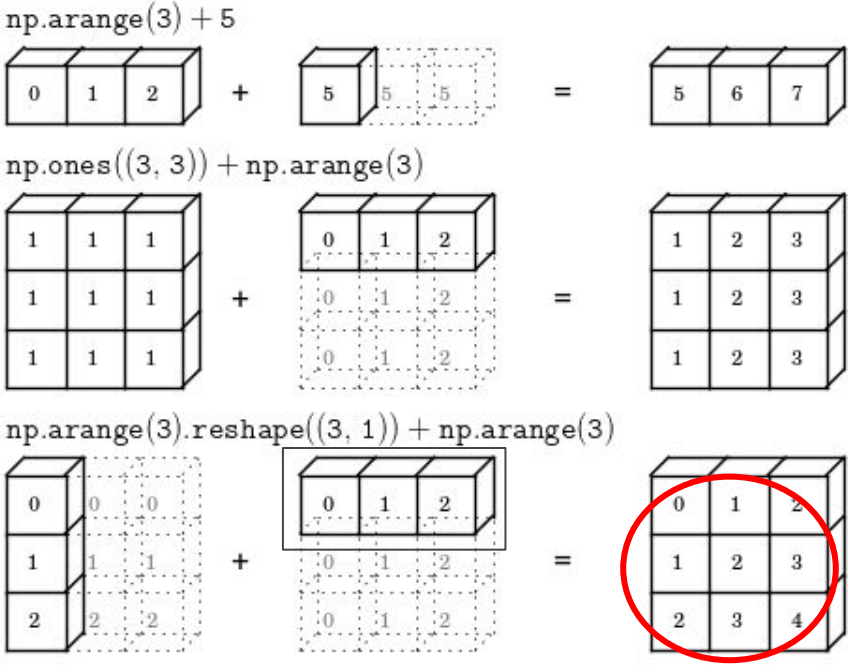
3.9.5 브로드캐스트



열 확장

09. numpy

3.9.5 브로드캐스트



CODE

```
A = np.arange(3) + 5
B = np.ones((3,3))+np.arange(3)
C = np.arange(6).reshape((2,3))+np.arange(3)

[0,0,0]          + [0,1,2] =  [0,1,2]
[1,1,1]            [0,1,2]    [1,2,3]
[2,2,2]            [0,1,2]    [2,3,4]

np.arange(3) = [0,1,2]
```

09. numpy

3.9.5 브로드 캐스트

CODE

```
# 3.9.4. 행렬의 곱

import numpy as np

A = np.array([[1,2],[3,4]])
B = np.array([[1,2],[3,4]])
print(np.dot(A,B))
```

09. numpy

리스트 중첩

CODE

```
# 리스트 중첩으로 3 보다 큰 요소를 리스트로 추출


originArray = [1,2,3,4,5]
[i for i in originArray if i > 3]
```

[결과]
[4, 5]

09. numpy

Numpy 에서 불리언 색인 추출

CODE

```
import numpy as np


originArray2 = np.array([1,2,3,4,5])
print(originArray2 > 3)
print(originArray2[originArray2 > 3])
```

결과

```
[False False False  True  True]
[4 5]
```

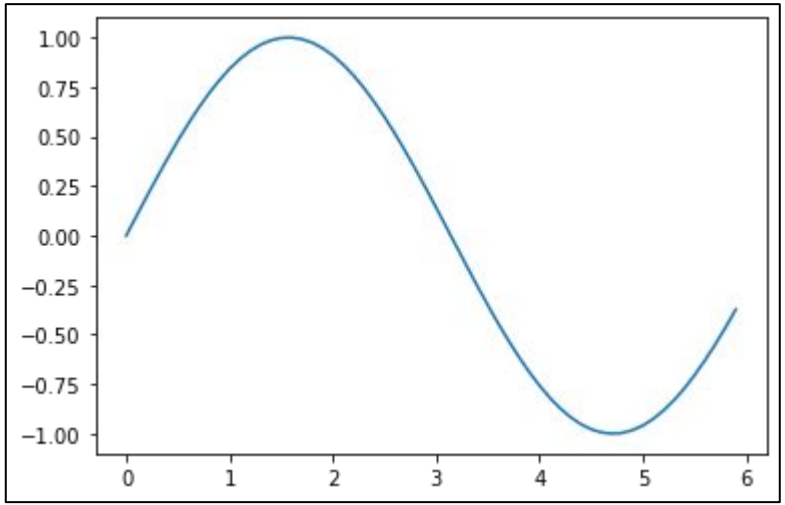
10. matplotlib 를 이용한 그래프

10. Matplotlib

3.10.1 sin 그래프

CODE

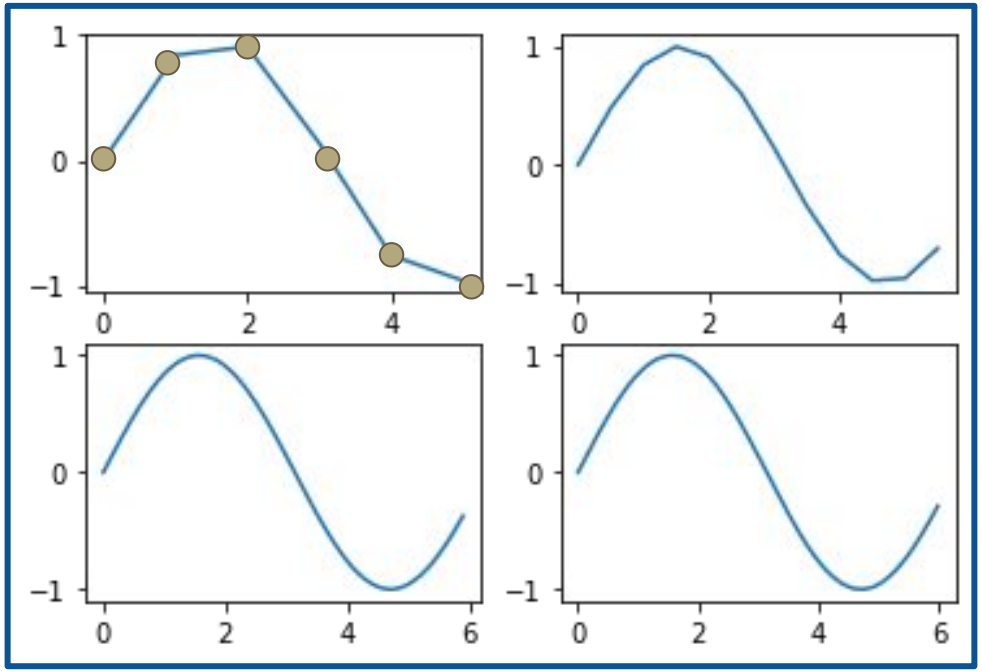
```
# 3.10.1. sin 그래프 그리기

import numpy as np
import matplotlib.pyplot as plt

x = np.arange(0, 6, 0.1)
y = np.sin(x)
plt.plot(x,y)
plt.show()
```



10. Matplotlib

3.10.2 그래프를 그리기 위한 데이터

6개, 12개, 60개, 600개




```
rate = 60
np.arange(0,6,6/rate)
```

10. Matplotlib

3.10.3 이미지파일 출력

```
# 3.10.3. CoLab 을 이용한 이미지 파일 화면 출력

import matplotlib.pyplot as plt
import matplotlib.image as pmg

from google.colab import files
upload = files.upload()

filename_key = upload.keys()
filename_key_list = list(filename_key)
filename_key_list[0]

img = pmg.imread(filename_key_list[0])
plt.imshow(img)
plt.show()
```