가톨릭대학교
THE CATHOLIC UNIVERSITY OF KOREA

파이썬 프로그래밍

미디어기술콘텐츠학과
강호철

Last updated on: Feb 11, 2020

파이썬

- 프로그래밍
  - 미국인과의 대화는 영어로
  - 일본인과의 대화는 일어로
  - 컴퓨터와의 대화는 프로그래밍 언어로
    - C/C++, Java, **python,** …

파이썬

- 파이썬?
  - 1990년 암스테르담의 귀도 반 로섬이 개발한 인터프리터 언어
  - 문법이 쉽고 빠르게 배울 수 있음
  - 무료이면서 강력하고 간결
  - 수학, 공학 관련 많은 기능들이 라이브러리로 이미 구현이 되어 있음
  - TIOBE : https://www.tiobe.com/tiobe-index/
  - IEEE Spectrum : https://spectrum.ieee.org/at-work/innovation/the-2018-top-programming-languages

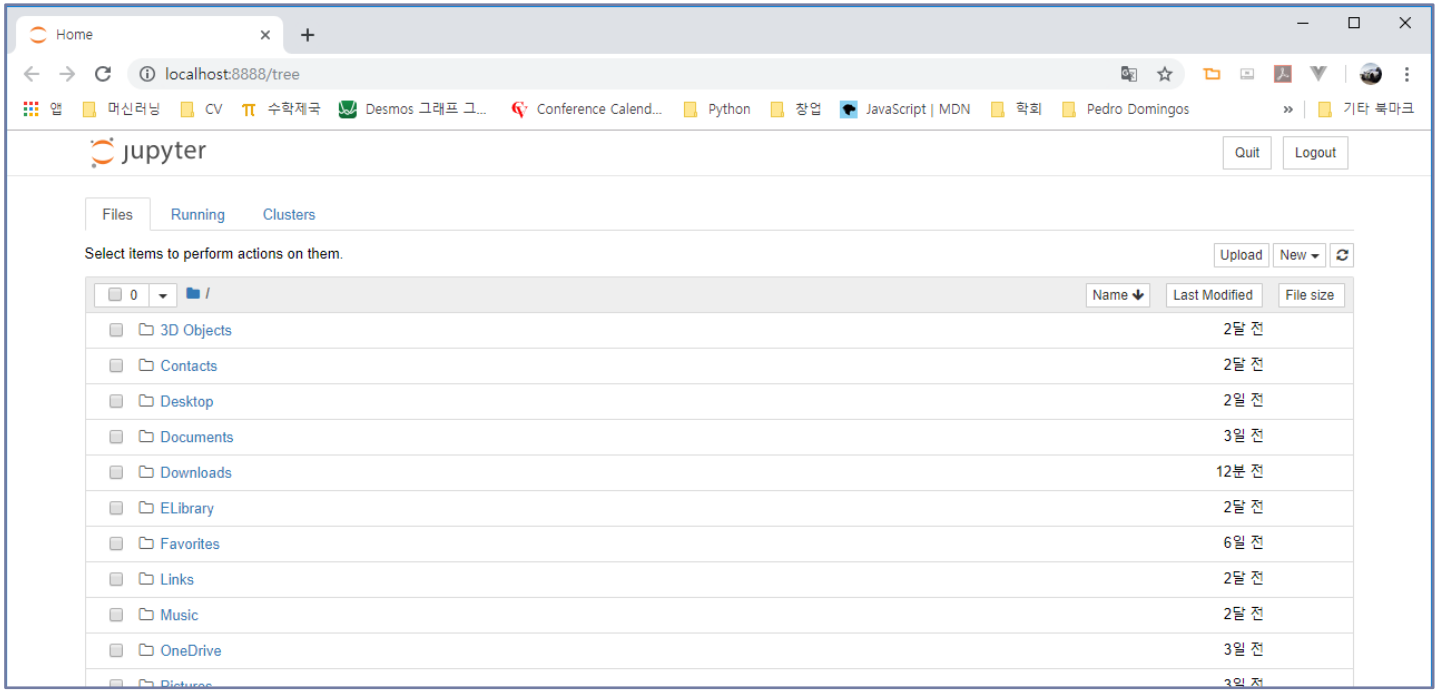
- 파이썬 설치
  - 아나콘다 (파이썬 통합 패키지) 설치
  - 파이썬 3.7, 아나콘다 2019.10
  - https://repo.anaconda.com/archive/Anaconda3-2019.10-Windows-x86_64.exe

파이썬

- OpenCV 설치 (필수)
  - anaconda prompt>pip install opencv-python
- PIL 설치 (옵션)
  - anaconda prompt>pip install pillow
- 그밖에 주요 packages
  - conda install nb_conda : JN에서 package를 관리하는 브라우저 생성
  - numpy, pandas, matplotlib, scipy, imutils, …
  - sympy, pgmpy, …

파이썬 개발 툴

- 주피터 노트북
  - 실행창에서 jupyter notebook
  - 혹은 시작 메뉴에서 anaconda3> jupyter notebook 선택
  - 실행 화면

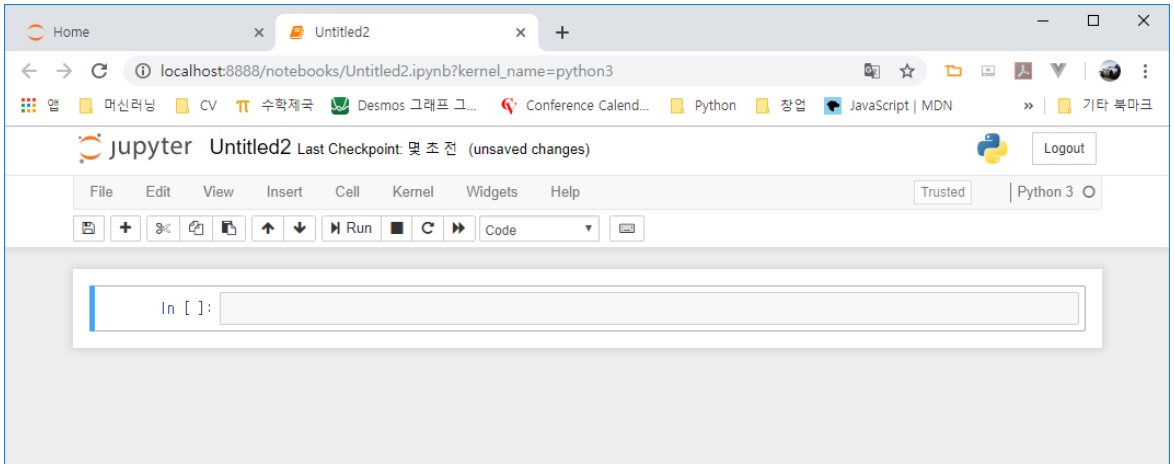


주피터 노트북 환경설정

- 시작 폴더 설정
  - 명령창 실행
  - jupyter notebook --generate-config 입력
  - 사용자 폴더에 .jupyter 폴더 열기
  - jupyter_notebook_config.py 파일 열기
  - #c.NotebookApp.notebook_dir = “ ” 찾고 # 제거
  - “ ” 안에 시작 폴더 경로 삽입
  - 저장 후 재 실행

파이썬 실행

- 주피터 노트북 실행
  - New > Python3 선택
  - 셀 안에 프로그램 작성



  - 숫자 혹은 명령 입력 후 shift-enter 혹은 run 실행
  - print("Hello world")
  - 수식 입력 후 실행
  - Markdown

기초 자료형

- 상수
  - 정수형
    - 123, -10, …
  - 실수형
    - 123.45, -10.35, …
  - 8진수, 16진수
    - 0o10, …
    - 0x10, …
  - 문자열
    - “hello world”, ‘hello world’
- 연산자
  - 사칙연산
  - **, //, %

기초 자료형

- 실습
  - 6^3 = ?, 11 나누기 4의 몫과 나머지 ?
  - 다음 수들의 평균은? 82, 70, 64, 69, 78

기초 자료형

- 변수
  - 메모리에 데이터를 저장하고 읽기 위해 할당 받는 공간
  - x = 100
  - y = 1.5
  - z = “머신러닝”
  - 대소문자 구분
    - A = 10 vs. a = 10
  - 예약어
    - 파이썬에서 미리 정의된 기능어로 변수명으로 사용 할 수 없음

False, class, return, is, finally, None, if, for, lambda, continue, True, def, from, while, nonlocal, and, del, global, not, with, as, elif, try, or, yield, assert, else, import, pass, break, except, in, raise 등

기초 자료형

- print 함수
  - 화면에 표시하는 함수
  - print("hello world")
  - print(10)
  - 문자열과 수치를 함께 표시
    - print('x = ' + str(x))
    - print('x = %d' % x)
    - print('x = {0}'.format(x))
    - print('x = {0} y = {1} z = {2}'.format(x, y, z))
    - print('x = %d y = %d z = %d' % (x, y, z))
    - print('x = {0:.1f} y = {1:.2f} z = {2:.3f}'.format(x, y, z))

기초 자료형

- 자료형
  - 데이터를 식별하는 분류
  - type 명령어로 자료형 확인
  - int 형
  - float 형
  - str 형
  - bool 형
  - list 형
  - tuple 형
  - ndarray 형

자료형	표현	예시
정수	int	450, 0, -10
실수	float	1.4, 0.01, -0.3
문자열	str	'머신러닝', '5930'
부울	bool	True, False
리스트	list	[40, '선형회귀', -1.4, -1.6, -0.4]
집합	set	{500, '분류', -1.4}
튜플	tuple	(4500, 'SVM', -1.6, -1.6, -1.4)
딕셔너리	dict	{'이름' : '홍길동', '나이' : 25}

기초 자료형

- list
  - 배열
    - 리스트명 = [요소1, 요소2, …]
    - odd = [1, 3, 5, 7, 9]
    - A = [1, 'life', 2.0]
    - B = [1, 2, ['a', 'b']]
    - 리스트의 요소로 어떠한 자료형도 포함 가능
  - 요소접근
    - odd[0] = ?, A[1] = ?, odd[-1] = ?
  - 2차원 배열 및 N 차원
    - A = [[1, 2, 3], [4, 5, 6]]
    - A[0][1] = ?

기초 자료형

- list
  - 길이
    - x = [1, 1, 2, 3, 5], len(x) == 5
  - 연속된 정수 데이터
    - y = range(5, 10) → y = [5, 6, 7, 8, 9]
    - y = range(10) → y = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
    - range형으로 생성한 요소는 수정 불가
      - list형으로 변환해야 함
      - z = list(range(5, 10))
  - 슬라이싱
    - a = [1, 2, 3, 4, 5]
      - a[0:2] = [1, 2], a[:3] = [1, 2, 3], a[2:] = [3, 4, 5]

기초 자료형

- list
  - 연산
    - a = [1, 2, 3], b = [4, 5, 6]
    - a + b = [1, 2, 3, 4, 5, 6]
    - a * 3 = [1, 2, 3, 1, 2, 3, 1, 2, 3]
  - 요소 수정 및 삭제
    - a = [1, 2, 3, 4, 5]
      - del a[1] → a = [1, 3, 4, 5]
      - del a[2:] → a = [1, 2]
  - 함수
    - a = [1, 2, 3], a.append(4) → a = [1, 2, 3, 4]
    - a.append([5, 6]) → a = [1, 2, 3, 4, [5, 6]]
    - a = [3, 1, 2], a.sort() → a = [1, 2, 3]
    - a = [3, 1, 2], a.reverse() → a = [2, 1, 3]

기초 자료형

- list
  - 함수
    - a = [1, 2, 3], a.index(3) == 2
    - a = [1, 2, 3], a.insert(0, 4) → a= [4, 1, 2, 3]
    - a = [1, 2, 3, 1, 2, 3], a.remove(3) → a= [1, 2, 1, 2, 3] *# 첫 3 만 remove*
    - a = [1, 2, 3], a.pop() → a = [1, 2]
    - a = [1, 2, 3], a.pop(1) → a = [1, 3]
    - a = [1, 2, 3, 1], a.count(1) == 2 *# 1 의 개수 리턴*
    - a = [1, 2, 3], a.extend([4, 5]) → a = [1, 2, 3, 4, 5] *# 리스트를 확장*
      - a.extend([4, 5]) == a += [4, 5]
  - 실습
    - [6, 2, 8, 4, 10] 을 [10, 8, 6, 4, 2]로 만들어라.

기초 자료형

- tuple
  - list 와의 차이점
    - 리스트는 [ ], 튜플은 ( )
    - 리스트는 생성, 삭제, 수정이 가능하지만 튜플은 불가능
    - t = (1, 2, 3), 인덱싱은 리스트와 같음 – t[0], t[1], …
    - 요소가 한 개 일 경우 뒤에 쉼표를 붙임 t = (1,) why?
    - 튜플은 괄호 생략 가능 t = 1, 2, 3
    - 리스트의 인덱싱, 슬라이싱, 더하기, 곱하기 사용 가능
    - len 함수로 길이 구하기 가능
  - 실습
    - (1, 2, 3)에 4를 추가하여 (1, 2, 3, 4)를 출력 하라

기초 자료형

- Dictionary
  - list 와의 차이점
    - 단순한 데이터의 배열이 아닌 키 값을 가지고 있는 배열
    - dic = { key1:value1, key2:value2, …}
    - dic = {'name':'pey', 'phone':'01012345678', 'birth':'190310'}
    - dic = {'a':[1, 2, 3]} *#value에 리스트를 넣는 것도 가능*
  - 딕셔너리 쌍 추가
    - a = {1:'a'}, a[2] = 'b' → a = {1:'a', 2:'b'}
    - a[3] = [1, 2, 3] → a = {1:'a', 2:'b', 3:[1,2,3]}
  - 딕셔너리 요소 삭제
    - del a[1] → a = {2:'b', 3:[1,2,3]}
  - 주의사항
    - a = {1:'a', 1:'b'} → a = {1:'b'}

기초 자료형

- Dictionary
  - 함수
    - a = {'name':'pey', 'phone':'01012345678', 'birth':'190310'}
    - a.keys() == dict_keys(['name', 'phone' , 'birth'])
    - dict_keys 리스트로 변환 : list(a.keys())
    - for문에서 사용 가능

```
for k in a.keys():
    print(k)

name
phone
birth
```

    - a.values() == dict_values(['pey', '01012345678' , '190310'])
    - a.items() == dict_items([('name', 'pey'), ('phone','01012345678'), ('birth', '190310')])
    - a.clear() == { }

기초 자료형

- Dictionary
  - 함수
    - a = {'name':'pey', 'phone':'01012345678', 'birth':'190310'}
    - a.get('name') = a['name']
      - a.get('no_key') == None
      - a['no_key'] → error!!!
      - a.get('no_key', 'not existence') == 'not existence'
  - 딕셔너리 존재여부
    - 'name' in a == True, 'email' in a == False
  - 실습
    - 다음 표를 딕셔너리로 만드시오

항목	값
name	홍길동
birth	1128
age	30

제어문

- 프로그램의 흐름을 제어하기 위해 사용
  - 조건문
    - if 문
  - 반복문
    - while 문
    - for 문

조건문

- if
  - 프로그램을 조건에 따라 나누어 실행
    - if 조건문A:

      명령어 1

      명령어 2

      ……..

      elif 조건문B:

      명령어 3

      명령어 4

      ……..

      else:

      명령어 5

      명령어 6

```
In [35]: x = 11
         if x > 10:
             print('Larger than 10 ')
         else:
             print('Smaller than 10 ')

         Larger than 10

In [36]: x > 10
Out[36]: True

In [37]: x < 10
Out[37]: False
```

조건문

- if
  - 비교연산자

비교연산자	설명
x < y	x가 y보다 작다
x > y	x가 y보다 크다
x == y	x와 y가 같다
x != y	x와 y가 같지 않다
x >= y	x가 y보다 크거나 같다
x <= y	x가 y보다 작거나 같다

```
In [38]: money = 1000
         fare = 2000
         card = True
         if money < fare and card:
             print('카드로 결제')
         elif money > fare:
             print('현금으로 결제')
         else:
             print('살수없음')

         카드로 결제
```

  - 다중 비교

연산자	설명
x or y	x와 y 둘중에 하나만 참이면 참이다
x and y	x와 y 모두 참이어야 참이다
not x	x가 거짓이면 참이다

조건문

- if
  - 중첩 조건문
    - if 조건문A:

      if 조건문B:

      명령어

      ……
  - 배열에서 요소 찾기
    - if 조건문A:

      if 조건문B:

```
In [40]: ary = [1,2,3,4,5]
         if 1 in ary:
             print('1이 있음')
         else:
             print('1이 없음')

         1이 있음
```

조건문

- 실습
  - 다음 코드의 결과를 구하라

```
a = "Life is too short, you need python"
if 'wife' in a:
    print('wife')
elif 'python' in a and 'you' not in a:
    print('python')
elif 'shirt' not in a:
    print('shirt')
elif 'need' in a:
    print('need')
else:
    print('none')
```

반복문

- while 문
  - 조건문을 만족하는 동안 반복 실행
- for 문
  - 특정한 횟수만큼 반복 실행
  - 리스트, 튜플, 딕셔너리 등 자료구조의 각 항목에 대해 반복문 실행

반복문

- while 문
  - while 조건문:

    명령문 1

    명령문 2

    명령문 3

    ……

```
treeHit = 0
while treeHit < 10:
    treeHit = treeHit +1
    print("나무를 %d번 찍었습니다." % treeHit)
    if treeHit == 10:
        print("나무 넘어갑니다.")
```

```
나무를 1번 찍었습니다.
나무를 2번 찍었습니다.
나무를 3번 찍었습니다.
나무를 4번 찍었습니다.
나무를 5번 찍었습니다.
나무를 6번 찍었습니다.
나무를 7번 찍었습니다.
나무를 8번 찍었습니다.
나무를 9번 찍었습니다.
나무를 10번 찍었습니다.
나무 넘어갑니다.
```

반복문

- while 문
  - while문 강제로 빠져나가기

```
In [1]: coffee_cnt = 10
        coffee_price = 300
        while True:
            print("남은커피: %d잔" % coffee_cnt)
            money = int(input("돈:"))
            if money == coffee_price:
                print("커피한잔")
                coffee_cnt = coffee_cnt-1
            elif money > coffee_price:
                print("거스름돈 %d, 커피한잔" % (money-coffee_price))
                coffee_cnt = coffee_cnt-1
            else:
                print("돈이 모자름")
            if coffee_cnt == 0:
                print("커피가 다 떨어졌습니다.")
                break
```

반복문

- for
  - 특정 명령어를 반복하여 실행
  - for 변수 in 리스트

```
In [42]: for i in [1, 3, 5, 7, 9]:
             print(i)
         print("END")
```

```
1
3
5
7
9
END
```

```
In [43]: num = [2,4,6,8,10]
         for i in range(len(num)):
             num[i] = num[i]*2
         print(num)
```

```
[4, 8, 12, 16, 20]
```

반복문

- for
  - 중첩 반복문

```
In [2]: for i in range(3):
            for j in range(2):
                print(i, j)
        print("END")
```

```
0 0
0 1
1 0
1 1
2 0
2 1
END
```

  - 실습
    - 중첩 반복문을 이용하여 구구단을 출력하라.

반복문

- for
  - 리스트 안에 for문 포함하기 (리스트 내포, list comprehension)

```
>>> a = [1,2,3,4]
>>> result = []
>>> for num in a:
...     result.append(num*3)
...
>>> print(result)
[3, 6, 9, 12]
```

```
>>> a = [1,2,3,4]
>>> result = [num * 3 for num in a]
>>> print(result)
[3, 6, 9, 12]
```

  - 리스트 안에 for, if 문 포함하기

```
>>> a = [1,2,3,4]
>>> result = [num * 3 for num in a if num % 2 == 0]
>>> print(result)
[6, 12]
```

반복문

- 실습
  - 학생의 평균 구하기
    - A = [90, 80, 78, 86, 76, 88, 100]
  - 다음 코드를 리스트 내포로 변경하시오
    -
```
numbers = [1, 2, 3, 4, 5]

result = []
for n in numbers:
    if n % 2 == 1:
        result.append(n*2)
```

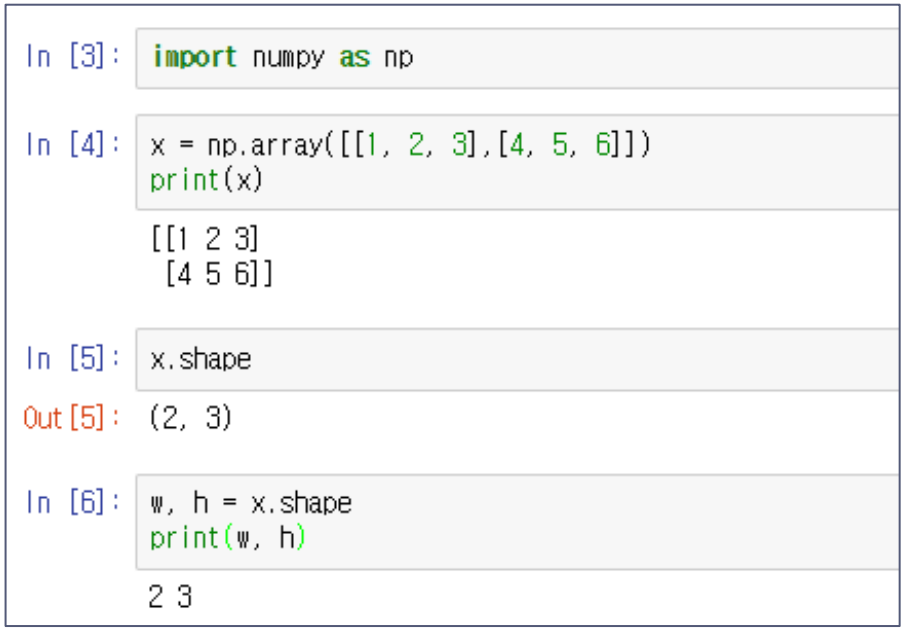
벡터와 행렬

- 벡터
  - List 를 이용한 벡터 처리
    - [1, 2] + [3, 4] = ?
  - Numpy 이용
    - 파이썬에서 벡터나 행렬을 처리하기 위한 라이브러리
    - import numpy as np
    - 벡터는 1차원 배열로 처리 가능
      - type(x) == numpy.ndarray
      - x = np.array([1, 2, 3]), y = np.array([4, 5, 6])
      - print(x+y) == [5, 7, 9]
      - 요소의 참조 및 수정은 list와 동일 x[0], x[1], …

벡터와 행렬

- 벡터
  - 연속된 정수 벡터의 생성
    - print(np.arange(10)) == [0, 1, 2, 3, 4, 5, 6, 7 ,8, 9]
    - print(np.arange(5, 10)) == [5, 6, 7, 8, 9]
  - ndarray 형의 주의점
    - ndarray는 값을 복사하지 않고 주소를 전달
      - b = a vs. b = a.copy()
      - a = np.array([1, 1])
      - b = a
      - b[0] = 2
      - a[0] = ?

벡터와 행렬

- 행렬
  - ndarray의 2차원 배열로 정의
    - 
```
In [3]: import numpy as np

In [4]: x = np.array([[1, 2, 3],[4, 5, 6]])
        print(x)

        [[1 2 3]
         [4 5 6]]

In [5]: x.shape

Out[5]: (2, 3)

In [6]: w, h = x.shape
        print(w, h)

        2 3
```
  - 요소의 참조 및 수정
    - x[1, 2] == 6
    - x[0, 2] = 10 → x = [[1 2 10], [4 5 6]]

벡터와 행렬

- 행렬
  - 요소가 0과 1인 ndarray 만들기

```
In [8]: print(np.zeros(10))
        [ 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.]

In [11]: print(np.zeros((2,3)))
         [[ 0.  0.  0.]
          [ 0.  0.  0.]]

In [12]: print(np.ones(10))
         [ 1.  1.  1.  1.  1.  1.  1.  1.  1.  1.]
```

  - 요소가 랜덤인 행렬 생성

```
In [13]: x = np.random.rand(2, 3)
         print(x)
         [[ 0.24817176  0.3601189   0.02153226]
          [ 0.8704282   0.35453953  0.95029909]]

In [15]: y = np.random.randn(2, 3)
         print(y)
         [[-0.50715046  1.63735735 -1.21522918]
          [ 0.4664811   1.66505181 -1.25612648]]

In [18]: z = np.random.randint(-3, 3, (2, 3))
         print(z)
         [[ 2 -3 -3]
          [ 2  2  1]]
```

벡터와 행렬

- 행렬
  - 크기 변경

```
In [19]: a = np.arange(10)
         print(a)

         [0 1 2 3 4 5 6 7 8 9]

In [20]: a.reshape(2, 5)

Out[20]: array([[0, 1, 2, 3, 4],
                [5, 6, 7, 8, 9]])
```

  - 사칙 연산

```
In [22]: x = np.array([[1, 1, 1],[2, 2, 2]])
         y = np.array([[3, 3, 3],[4, 4, 4]])
         z = x + y
         print(z)

         [[4 4 4]
          [6 6 6]]

In [23]: print(z*10)

         [[40 40 40]
          [60 60 60]]
```

벡터와 행렬

- 행렬
  - 산술 함수
    -

```
In [25]: x = np.array([[1, 2, 3], [4, 5, 6]])
         print(np.exp(x))

[[  2.71828183   7.3890561   20.08553692]
 [ 54.59815003 148.4131591  403.42879349]]
```

      - np.sqrt(x), np.log(x), np.round(x), np.mean(x), np.std(x), np.max(x), np.min(x)

  - 행렬의 곱셈

```
In [26]: v = np.array([[1, 2, 3], [4, 5, 6]])
         w = np.array([[1, 1],[2, 2],[3, 3]])
         print(v.dot(w))

[[14 14]
 [32 32]]
```

벡터와 행렬

- 슬라이싱
  - 변수명[:n], 변수명[n:]
    - 0 ~ n-1, n ~ 끝
  - 변수명[n1:n2]
    - n1 ~ n2-1
  - 변수명[n1:n2:dn]
    - n1 ~ n2-1 까지 dn 간격으로 참조
  - 변수명[::-1]
    - reverse
  - 1차원 이상

```
In [27]: y = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])
         print(y)
         print(" ")
         print(y[:2, 1:2])

         [[1 2 3]
          [4 5 6]
          [7 8 9]]

         [[2]
          [5]]
```

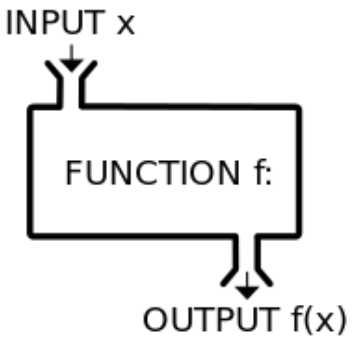
벡터와 행렬

- 조건을 이용한 데이터 수정

```
In [34]: x = np.array([1, 1, 2, 3, 5, 8, 13])
         b = x > 3
         print(b)

         [False False False False  True  True  True]

In [35]: x[x > 3]

Out[35]: array([ 5,  8, 13])

In [36]: x[x > 3] = 999
         print(x)

         [  1   1   2   3 999 999 999]

In [37]: y = x[x > 100]
         print(y)

         [999 999 999]
```

함수

- 정의
  - 재사용을 목적으로 묵은 명령 집합
  - 수학에서 함수와 같은 기능



내장 함수

```
print()
input()
type()
float()
...
```

사용자 정의 함수

```
def 함수명(x):
    명령문장1
    명령문장2
    return  y
```

함수

- 구조

```
def 함수명(매개변수):
        명령 문장1
        명령 문장2
        ...
        [return v]
```

```
def add(x1, x2):
        result = x1+x2
        print(result)

        return result
```

```
a = 5
b = 3
c = add(a, b)
```

함수

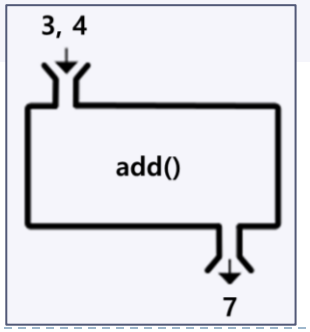
- 함수의 정의와 사용

```
def add(x1, x2):
        result = x1+x2
        print(result)
        ...
        return result
```

함수 정의

```
a = 5
b = add(a, 3)
c  = add(a, 6)
d = add(a, -3)
```

함수 사용



함수

- Lambda 함수
  - 함수를 생성할 때 사용하는 예약어로 def와 동일한 역할을 함
  - 함수를 한 줄로 간결하게 만들 때 사용
  - 사용법
    - Lambda 매개변수1, 매개변수2, … : 매개변수를 이용한 표현식
    - Ex) add = lambda a, b: a+b

      result = add(3, 4)

      print(result)

- 실습
  - 리스트의 평균값 구하는 함수 구현하고 사용하기

함수

- 인수와 반환값
  - 반환값을 두 개 이상 지정 가능
  - Ex) def myfunc(d):

```
        m = np.mean(d)
        s = np.std(d)
        return m, s
```

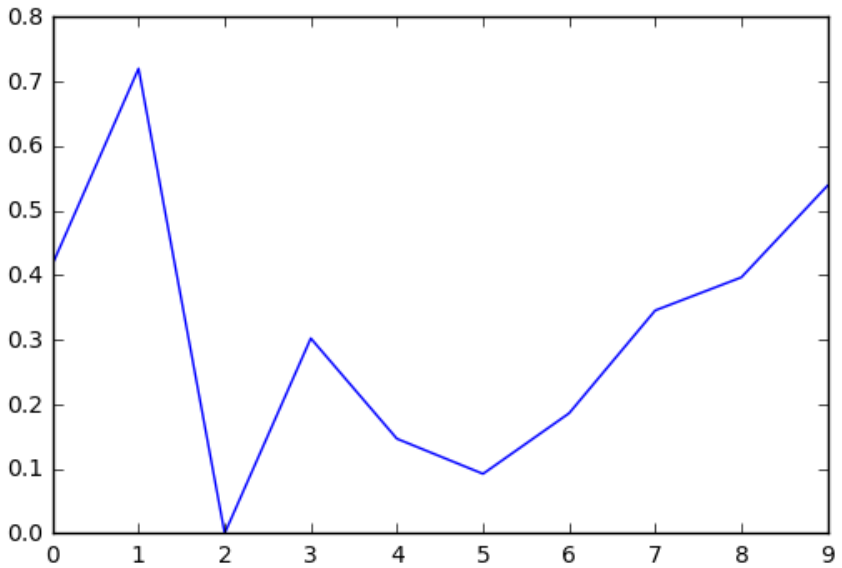
파일 저장

- 하나의 ndarray 형을 저장
  - 파일 저장
    - np.save("파일명.npy", 변수명)
  - 파일 로드
    - np.load("파일명.npy")

그래프 그리기

- Matplotlib 라이브러리
  - 그래프 그리기를 위한 라이브러리

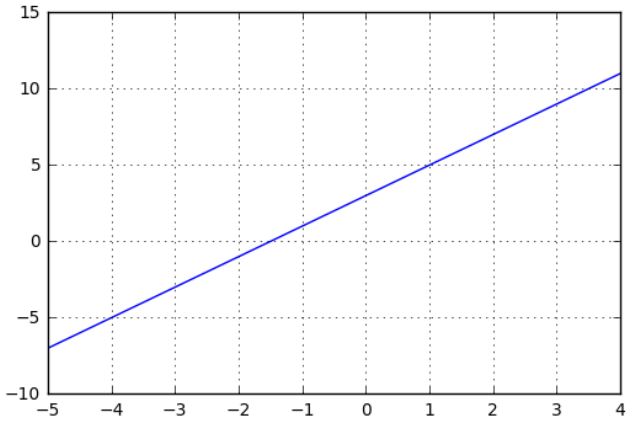
```
In [2]: import numpy as np
        import matplotlib.pyplot as plt
        %matplotlib inline

In [3]: np.random.seed(1)
        x = np.arange(10)
        y = np.random.rand(10)

In [4]: # 그래프 표시
        plt.plot(x, y)    #꺾은선 그래프를 등록
        plt.show()        #그래프 그리기
```



그래프 그리기

- Matplotlib 라이브러리
  - 함수 그리기
    - f(x) = 2x + 10

```
In [1]: import numpy as np
        import matplotlib.pyplot as plt
        %matplotlib inline

In [8]: def f(x):
            return 2*x + 3

In [9]: x = np.arange(-5, 5, 1)
        y = f(x)
        print(x, y)

        [-5 -4 -3 -2 -1  0  1  2  3  4] [-7 -5 -3 -1  1  3  5  7  9 11]

In [10]: plt.plot(x, y)
         plt.grid(True)
         plt.show()
```



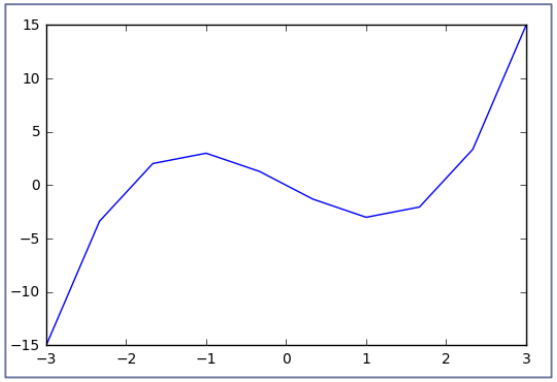
그래프 그리기

- Matplotlib 라이브러리
  - 함수 그리기
    - f(x) = (x-2)x(x+2)

그래프 그리기

- Matplotlib 라이브러리
  - 함수 그리기
    - f(x) = (x-2)x(x+2)

```
In [1]: import numpy as np
        import matplotlib.pyplot as plt
        %matplotlib inline

In [2]: def f(x):
            return (x-2)*x*(x+2)

In [3]: print(f(1))

        -3

In [6]: print(f(np.array([1, 2, 3])))

        [-3  0 15]

In [7]: x = np.arange(-3, 3.5, 0.5)
        print(x)

        [-3.  -2.5 -2.  -1.5 -1.  -0.5  0.   0.5  1.   1.5  2.   2.5  3. ]

In [8]: x = np.linspace(-3, 3, 10)
        print(np.round(x, 2))

        [-3.   -2.33 -1.67 -1.   -0.33  0.33  1.   1.67  2.33  3.  ]

In [9]: plt.plot(x, f(x))
        plt.show()
```



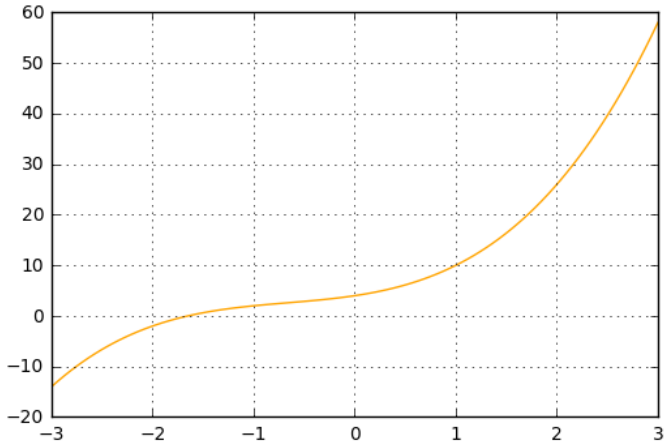
그래프 그리기

- Matplotlib 라이브러리
  - 실습
    - $f(x) = ax^3 + bx^2 + cx + d$ 를 정의하고 그래프를 그리는 프로그램을 작성하라.

그래프 그리기

- Matplotlib 라이브러리
  - 실습
    - $f(x) = ax^3 + bx^2 + cx + d$ 를 정의하고 그래프를 그리는 프로그램을 작성하라.

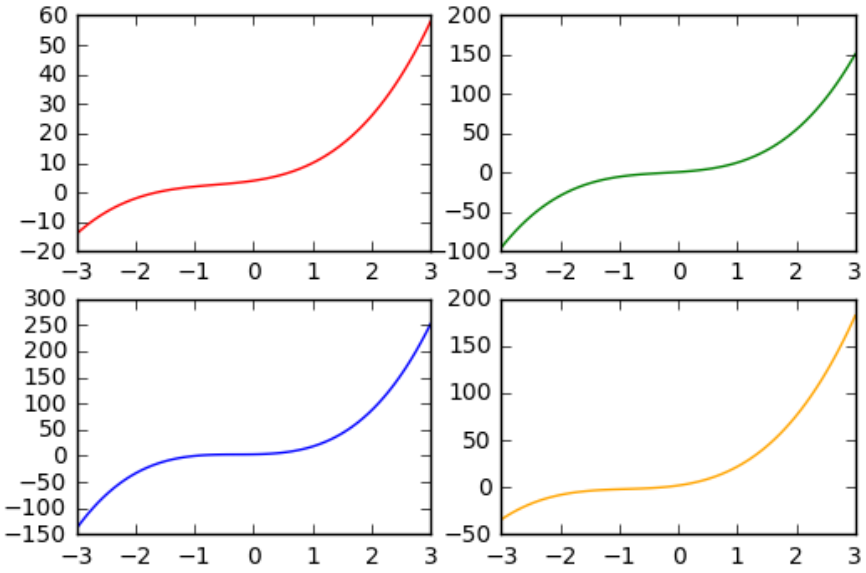
```
In [1]: import numpy as np
        import matplotlib.pyplot as plt
        %matplotlib inline

In [2]: def f(a, b, c, d, x):
            return (a*x**3 + b*x**2 + c*x + d)

In [4]: x = np.linspace(-3, 3, 100)

        a = 1
        b = 2
        c = 3
        d = 4
        plt.plot(x, f(a, b, c, d, x), color='orange')
        plt.grid(True)
        plt.show()
```



그래프 그리기

- Matplotlib 라이브러리
  - 멀티 그래프
    - 앞의 $f(x) = ax^3 + bx^2 + cx + d$ 를 여러 개 정의하고 한 화면에 그래프를 그리는 프로그램을 작성하라.

```
In [6]: l = [1, 2, 3, 4]
        m = [4, 3, 5, 1]
        n = [7, 6, 2, 3]
        k = [3, 8, 9, 2]

        plt.subplot(2, 2, 1)
        plt.plot(x, f(l[0], l[1], l[2], l[3], x), color = 'red')

        plt.subplot(2, 2, 2)
        plt.plot(x, f(m[0], m[1], m[2], m[3], x), color = 'green')

        plt.subplot(2, 2, 3)
        plt.plot(x, f(n[0], n[1], n[2], n[3], x), color = 'blue')

        plt.subplot(2, 2, 4)
        plt.plot(x, f(k[0], k[1], k[2], k[3], x), color = 'orange')
```



그래프 그리기

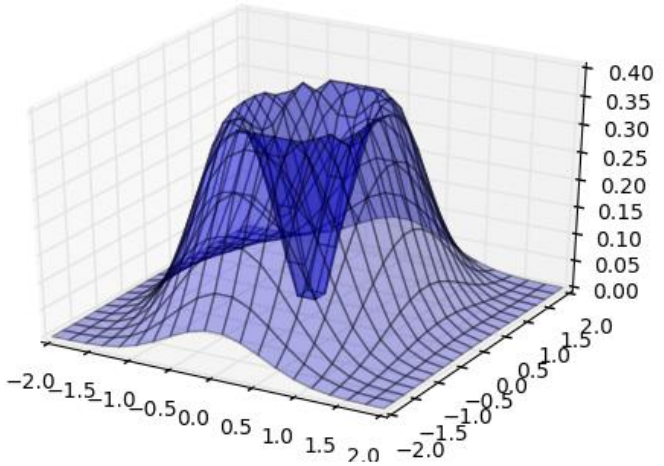
- Matplotlib 라이브러리
  - 이변수 함수 3차원 그래프
    - $f(x_0, x_1) = (2x_0^2 + x_1^2)*\exp(-(2x_0^2 + x_1^2))$

In [51]:
```
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
%matplotlib inline
```

In [52]:
```
def f(x0, x1):
    r = 2*x0**2 + x1**2
    return r*np.exp(-r)
```

In [53]:
```
xn = 20
x0 = np.linspace(-2, 2, xn)
x1 = np.linspace(-2, 2, xn)
y = np.zeros((len(x0), len(x1)))
for i0 in range(xn):
    for i1 in range(xn):
        y[i1, i0] = f(x0[i0], x1[i1])
```

In [55]:
```
xx0, xx1 = np.meshgrid(x0, x1)
ax = plt.subplot(1, 1, 1, projection='3d')
ax.plot_surface(xx0, xx1, y, color='blue', rstride=1, cstride=1,
                alpha=0.3, edgecolor='black')
plt.show()
```



참고자료

- 점프 투 파이썬
  - https://wikidocs.net/book/1

- 파이썬으로 배우는 머신러닝의 교과서
  - 이토마코토 지음, 박광수 옮김
  - 한빛미디어, 2018