



IoT·인공지능·빅데이터 개론 및 실습

기계학습을 위한 파이썬

서울대학교 공과대학 전기·정보공학부
윤성로 교수 / 하헌석 조교

Contents

1. 소개 및 환경 구축

- ① 기계학습과 파이썬
- ② 아이파이썬(IPython) 환경

2. 파이썬 실습

- ① 파이썬 기초
- ② 기계학습을 위한 파이썬

1 기계학습과 파이썬

(1) 기계학습

➤ 인공지능의 한 분야

➤ 데이터로부터 학습

- 학습 알고리즘

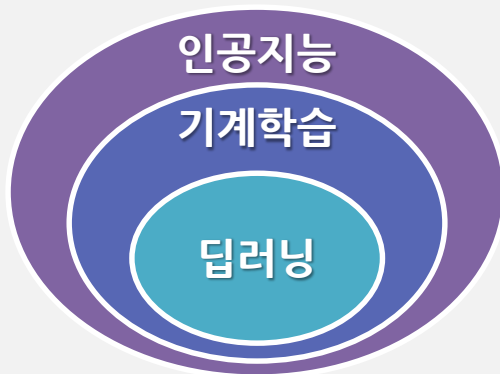
- 인공 신경망 / 딥러닝
- CNN, RNN, ...

- 빅데이터

- 대량의 다차원 행렬
- 영상, 생명정보, 시계열, ...

- 연산 가속장치

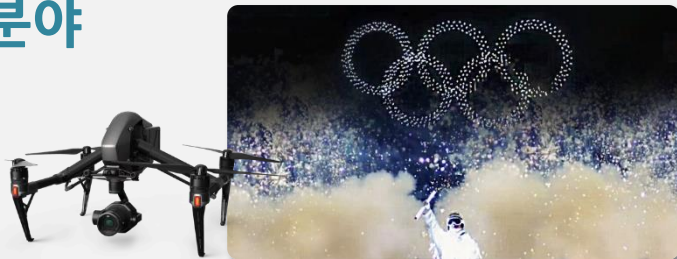
- GPGPU, FPU



1 기계학습과 파이썬

(1) 기계학습

➤ 다양한 활용 분야



[출처] <http://programs.sbs.co.kr/sports/pyeongchang2018/>
<http://eng.snu.ac.kr/>

1 기계학습과 파이썬

(2) 프로그래밍 언어

➤ 저수준(Low-level) 언어

- 고성능 수치계산에 효율적
- 컴퓨터에 대한 이해 필요
 - CUDA 프로그래밍을 위해 GPU구조 이해 필요
- C, Fortran, ...

➤ 고수준(High-level) 언어

- 유연하고 상대적으로 간단하게 작성 가능
- 실행 속도가 느림
 - 저수준 라이브러리와 결합을 통해 극복
- Python, MATLAB, ...

1 기계학습과 파이썬

(3) 파이썬

- 오픈소스 일반목적 언어
- 통역식(interpreted) 언어
 - 컴파일 과정 없이 실행
- 저수준 기반 라이브러리들과 결합
 - 실행 성능 개선
 - NumPy(배열 연산), SciPy(과학 연산), TensorFlow(딥러닝), matplotlib(그래픽),...



1 기계학습과 파이썬

(3) 파이썬

➤ 기계학습 분야에서 가장 널리 사용되는 언어

- *Github* 에서 'machine learning' 검색 결과

Languages	
Jupyter Note...	63,087
Python	52,120
HTML	19,178
MATLAB	15,427
R	5,978
Java	4,321
JavaScript	3,477
C++	2,357
C#	1,355
TeX	1,222

[출처] <https://github.com/>, 2020.03 기준.

1 딥러닝과 파이썬

(4) 설치하기

➤ 파이썬 2 vs 파이썬 3

- 서로 호환 불가
- 두 가지 버전이 공존
- 처음 학습자라면 3버전
- 과거 코드와 호환성이 중요하다면 2버전

```
print "Hello world"
```

Python 2

```
print("Hello world")
```

Python 3

1 딥러닝과 파이썬

(4) 설치하기

➤ 설치 파일 내려받아 실행

<https://www.anaconda.com/products/individual>

Anaconda Installers

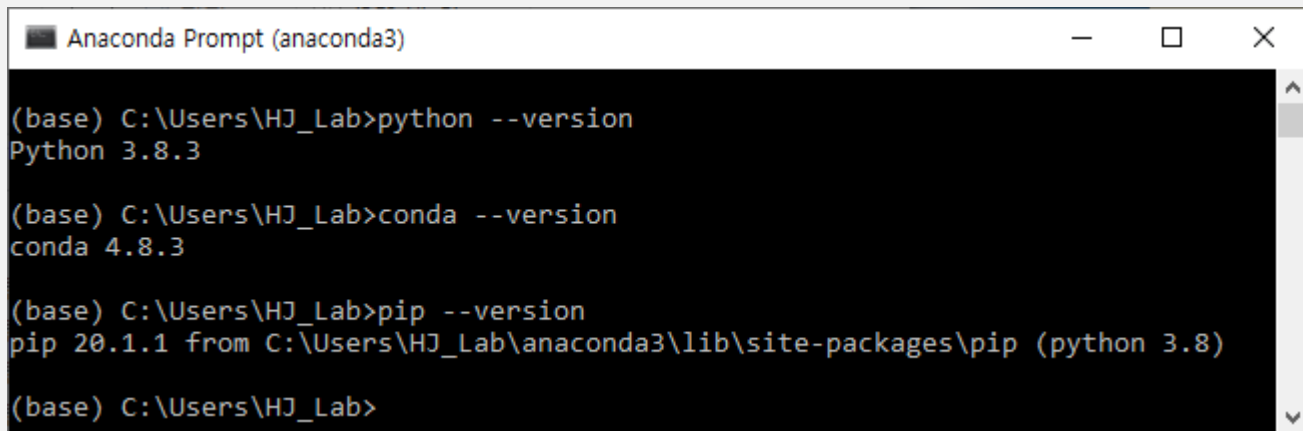
Windows 	MacOS 	Linux 
Python 3.8 64-Bit Graphical Installer (466 MB) 32-Bit Graphical Installer (397 MB)	Python 3.8 64-Bit Graphical Installer (462 MB) 64-Bit Command Line Installer (454 MB)	Python 3.8 64-Bit (x86) Installer (550 MB) 64-Bit (Power8 and Power9) Installer (290 MB)

1 딥러닝과 파이썬

(4) 설치하기

➤ 설치 확인

- **Anaconda Prompt 실행**
- (base)>python --version ↵
- (base)>conda --version ↵
- (base)>pip --version ↵



```
Anaconda Prompt (anaconda3)

(base) C:\Users\HJ_Lab>python --version
Python 3.8.3

(base) C:\Users\HJ_Lab>conda --version
conda 4.8.3

(base) C:\Users\HJ_Lab>pip --version
pip 20.1.1 from C:\Users\HJ_Lab\anaconda3\lib\site-packages\pip (python 3.8)

(base) C:\Users\HJ_Lab>
```

Contents

1. 소개 및 환경 구축

- ① 기계학습과 파이썬
- ② 아이파이썬(IPython) 환경

2. 파이썬 실습

- ① 파이썬 기초
- ② 기계학습을 위한 파이썬

2 아이파이썬(IPython) 환경



(1) 아이파이썬(IPython)과 주피터(Jupyter)

➤ 대화형 인터페이스 개발환경

- Interactive Python
- 시행착오(trial and error) 방법론

➤ 웹브라우저에서 실행

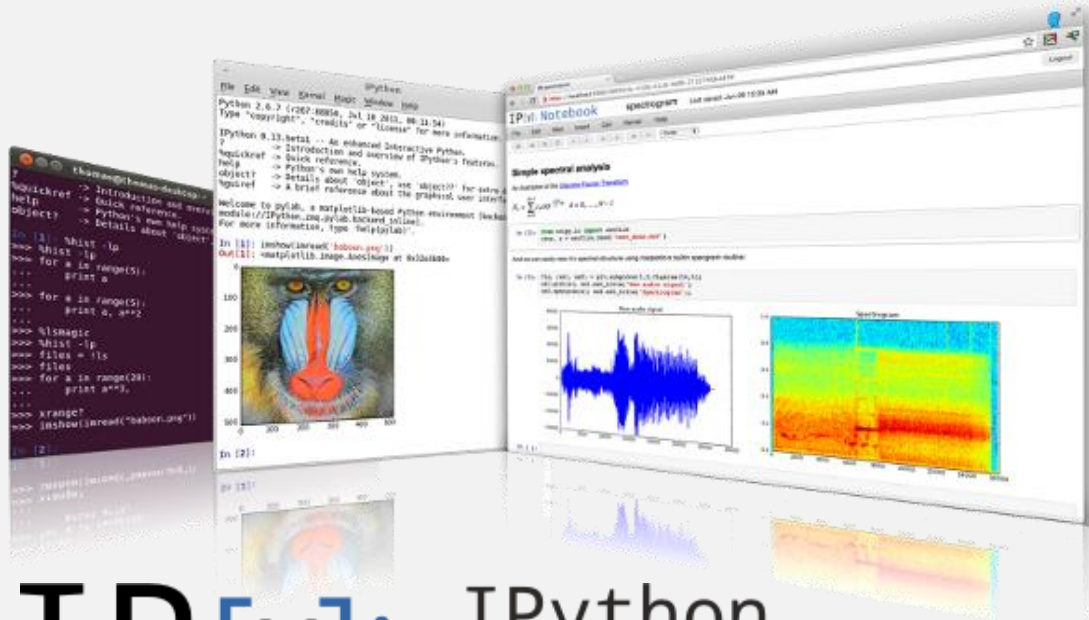
- 수학적 식, 도표, 그래프 등을 표현 가능

IP[y]: IPython
Interactive Computing

[출처] <https://ipython.org>

2 아이파이썬(IPython) 환경

(1) 아이파이썬(IPython)과 주피터(Jupyter)



IP[y]:

IPython
Interactive Computing

[출처] <https://ipython.org>

2 아이파이썬(IPython) 환경

(2) 가상환경 설정

➤ 가상환경 만들기

- >conda create -n [가상환경명] python=[버전] ↵
- >conda create -n iab_1 python=3.7 ↵

➤ 가상환경 활성화

- >activate [가상환경명] ↵
- >activate iab_1 ↵

```
C:\Users\hyekjun.choe>activate tensorflow3.6.5
```

```
(tensorflow3.6.5) C:\Users\hyekjun.choe>
```

커맨드라인 앞에 (가상환경명) 이 붙으면 활성화 상태

➤ 가상환경 비활성화

- >deactivate ↵

② 아이파이썬(IPython) 환경

(2) pip 를 이용하여 패키지 설치

➤ jupyter 설치

- (가상환경명) > pip install jupyter ↵

```
명령 프롬프트 - pip install jupyter

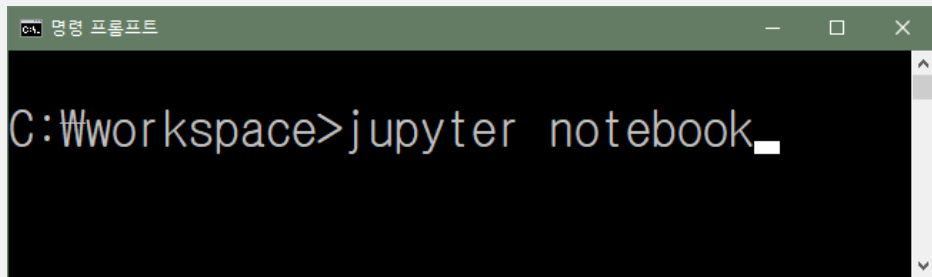
(tensorflow3.6.5) C:\Users\WhyeokJun.choe>pip install jupyter
Looking in indexes: http://ftp.daumkakao.com/pypi/simple
Collecting jupyter
  Downloading http://mirror.kakao.com/pypi/packages/83/df/0f5dd132200728a86190397e1ea87cd76244e42d39ec5e88efd25b2abd7e/j
  upyter-1.0.0-py2.py3-none-any.whl
Collecting ipywidgets (from jupyter)
  Downloading http://mirror.kakao.com/pypi/packages/ea/c5/0482342559f0fd24909572fe00bb59b2bae98b22d90aac7950f51a98f555/i
  pywidgets-7.4.1-py2.py3-none-any.whl (110kB)
  100% |#####| 112kB 9.7MB/s
Collecting jupyter-console (from jupyter)
  Downloading http://mirror.kakao.com/pypi/packages/77/82/6469cd7fccf7958cbe5dce2e623f1e3c5e27f1bb1ad36d90519bc2d5d370/j
  upyter_console-5.2.0-py2.py3-none-any.whl
Collecting ipykernel (from jupyter)
  Downloading http://mirror.kakao.com/pypi/packages/7a/de/a03a5c1f8b743675add3c98f1eb877c67bb29c5196ee6ce54e9c839d23cc/i
  pykernel-4.9.0-py3-none-any.whl (110kB)
  100% |#####| 112kB 9.0MB/s
Collecting qtconsole (from jupyter)
  Downloading http://mirror.kakao.com/pypi/packages/ff/1f/b340d52dee46fbb8a097dce76d1197258bb599692159d94c80921fef9eb/c
  tconsole-4.4.1-py2.py3-none-any.whl (112kB)
  100% |#####| 112kB 9.4MB/s
Collecting nbconvert (from jupyter)
  Downloading http://mirror.kakao.com/pypi/packages/b5/bb/94c493051d60e5b9c0f7f9a368b324201818c1b1c4cae85d1e49a41846c7/n
  bconvert-5.4.0-py2.py3-none-any.whl (405kB)
  100% |#####| 409kB 12.1MB/s
```

2 아이파이썬(IPython) 환경

(3) 실행하기

➤ 터미널의 명령줄 입력을 통해 실행

- 작업하고자 하는 폴더로 이동
- (가상환경명)>jupyter notebook ↵
- 웹브라우저가 뜨면서 실행
 - 해당 터미널을 유지해야 작업 가능
 - 웹브라우저 주소창에 `http://localhost:8888`
 - Ctrl+C 로 터미널 종료 가능



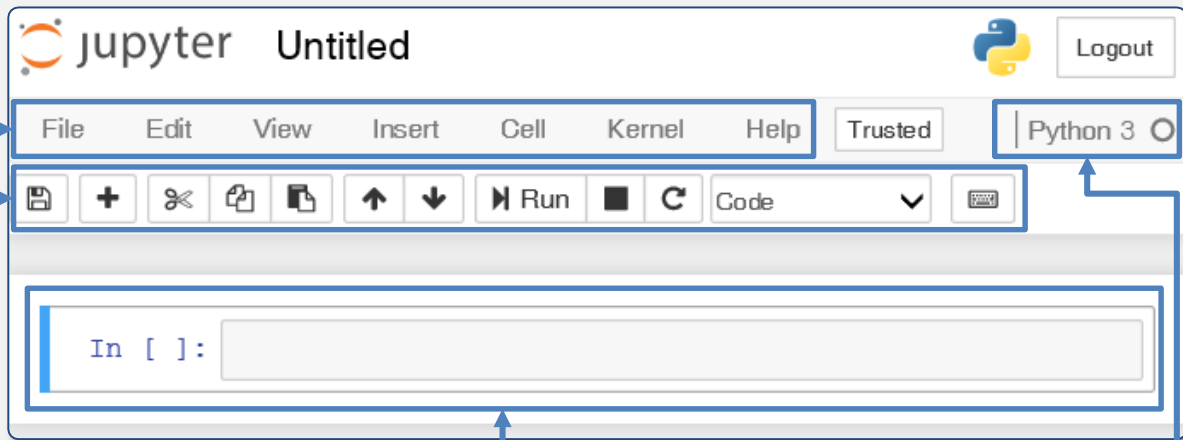
```
C:\workspace>jupyter notebook
```


2 아이파이썬(IPython) 환경

(3) Jupyter 실행하기

➤ 웹 브라우저를 통해 코드 작성 및 결과 확인

- New 버튼을 열어서 Notebook(Python 3) 선택



메뉴바

툴바

코드셀

커널

② 아이파이썬(IPython) 환경



(3) Jupyter 실행하기

➤ 명령 모드(Command mode)

- *ESC* 키를 눌러서 명령 모드 진입
- 선택 셀이 회색 경계선으로 표시
- 주요 명령어
 - *h* : 명령 목록과 도움말 표시
 - *↑* / *↓* : 이전 / 다음 셀 선택
 - *a* / *b* : 위 / 아래에 새 셀 추가
 - *x* / *c* / *b* : 잘라내기 / 복사하기 / 붙여넣기
 - *y* / *m* : 코드 / 마크다운 셀 전환
 - *dd* : 선택 셀 삭제
 - *z* : 실행 취소

② 아이파이썬(IPython) 환경



(3) Jupyter 실행하기

➤ 편집 모드(Edit mode)

- *ENTER* 키를 누르거나 마우스 클릭으로 진입
- 선택 셀이 녹색 경계선으로 표시
- 일반적인 편집기처럼 사용

➤ 자주 사용하는 단축키

- Ctrl+*ENTER* : 선택 셀 실행
- Shift+*ENTER* : 선택 셀 실행 후 다음 셀 선택
- Alt+*ENTER* : 선택 셀 실행 후 아래에 새 셀 추가
- Ctrl+s : notebook 저장
 - 확장자 ipynb으로 저장됨

② 아이파이썬(IPython) 환경



(3) Jupyter 실행하기

➤ 편집 모드(Edit mode)

- *ENTER* 키를 누르거나 마우스 클릭으로 진입
- 선택 셀이 녹색 경계선으로 표시
- 일반적인 편집기처럼 사용

➤ 자주 사용하는 단축키

- Ctrl+*ENTER* : 선택 셀 실행
- Shift+*ENTER* : 선택 셀 실행 후 다음 셀 선택
- Alt+*ENTER* : 선택 셀 실행 후 아래에 새 셀 추가
- Ctrl+s : notebook 저장
 - 확장자 ipynb으로 저장됨

Contents

1. 소개 및 환경 구축

- ① 기계학습과 파이썬
- ② 아이파이썬(IPython) 환경

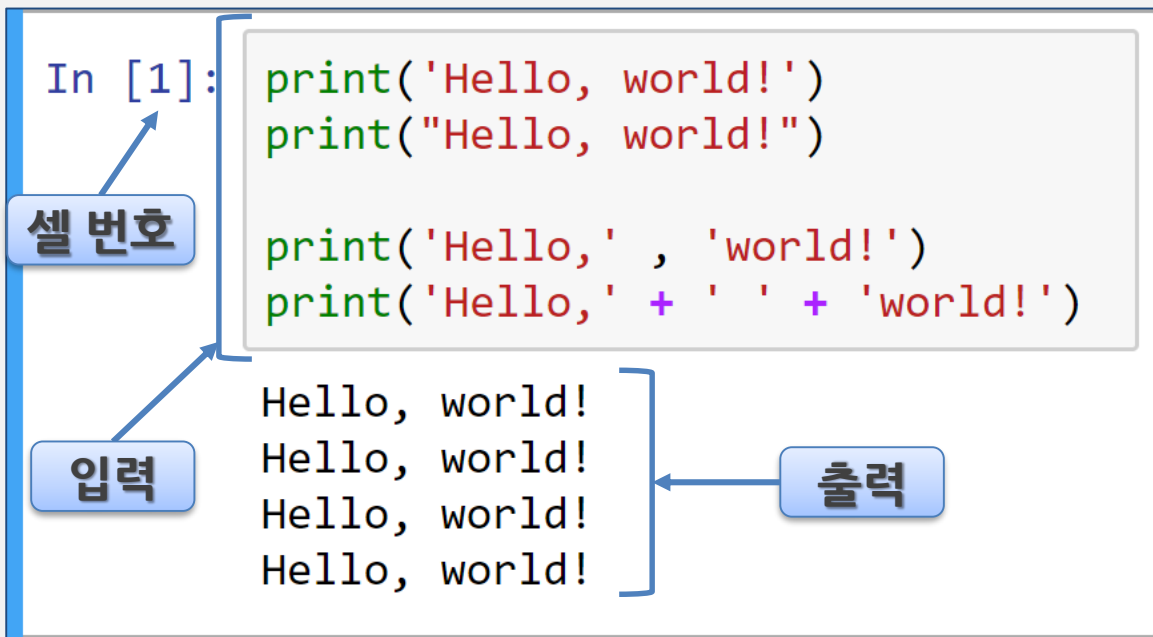
2. 파이썬 실습

- ① 파이썬 기초
- ② 기계학습을 위한 파이썬

1 파이썬 기초

(1) Hello, world

➤ 프로그래밍 언어 학습의 출발점



1 파이썬 기초

(2) 주석 작성하기

➤ 두 가지 방법이 존재

In [2]:

```
"""  
Comments  
주석문  
"""
```

둘러싼 전체

이후 줄 끝까지

#또다른 방법

```
print("Hello world") # Hello, world
```

Hello world

1 파이썬 기초

(3) 리스트(List)

```
myList_1 = [1,2,3,4,5]
myList_2 = 'A string is also a list'
# Empty lists
myList_3 = []
myList_4 = list()
```

[] 을 통해 선언

, 로 원소 구분

문자열도 리스트

자료형을 명시할
필요 없음

```
print(myList_1)
print(myList_2)
print(myList_3)
print(len(myList_1))
```

리스트의 길이를
출력하는 함수

```
[1, 2, 3, 4, 5]
A string is also a list
[]
5
```


1 파이썬 기초

(3) 리스트(List)

```
myList_3 = myList_1 + [6,7,8,9,10]  
myList_4 = myList_2[4:8]
```

+ 로 합치기

부분리스트

```
print(myList_3)  
print(myList_3[0])  
print(myList_3[-1])  
print(myList_4)
```

첫 항목이 0번

뒤에서 첫 번째 항목

```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]  
1  
10  
ring
```

1 파이썬 기초

(4) 딕셔너리(Dictionary)

```
myDict_1 = {'key_a':'value_1', 'key_b':'value_2'}  
myDict_2 = {'a':1, 'b':2, 'c':3, 'd':4, 'e':5}  
# Empty Dictionaries  
myDict_3 = {}  
myDict_4 = dict()
```

key : value

- key : value 쌍을 원소로 갖는 구조
- 리스트와는 달리 원소들간 순서가 없음

1 파이썬 기초

(4) 딕셔너리(Dictionary)

```
myDict_1 = {'key_a': 'value_1', 'key_b': 'value_2'}  
myDict_2 = {'a': 1, 'b': 2, 'c': 3, 'd': 4, 'e': 5}  
myDict_1['key_c'] = 'value_3'  
print(myDict_1.keys())  
print(myDict_2.values())  
print(myDict_1['key_c'])  
print(myDict_2['c'])
```

새로운 key:value 추가

```
dict_keys(['key_a', 'key_b', 'key_c'])  
dict_values([1, 2, 3, 4, 5])  
value_3  
3
```

- [] 안에 인덱스 대신 key로써 value에 접근

1 파이썬 기초

(5) 반복문과 조건문

➤ for 반복문

갱신되는 변수

: 으로 끝

```
for eachone in myList_1:
    print(eachone, end=' ')
    print()
for i in range(len(myList_1)):
    print(myList_1[i], end='.')
    print()
```

1 2 3 4 5
1.2.3.4.5.

키워드 인자 전달

들여쓰기

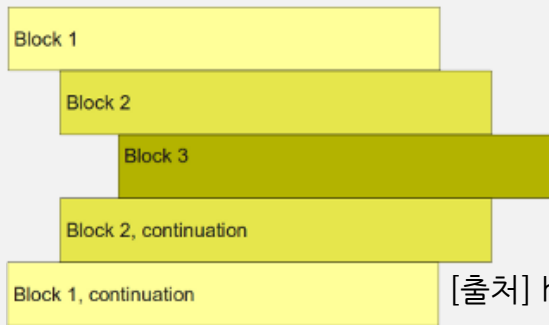
인덱스 접근 반복문

1 파이썬 기초

(5) 반복문과 조건문

➤ 스위트(Suite)와 들여쓰기

- 스위트: for, if, def 등의 키워드 선언 후 따라붙는 같은 맥락의 코드 묶음(block)
- 하나의 스위트 내에서 반드시 같은 들여쓰기
- 중첩된(nested) 스위트는 추가 들여쓰기
- 단일 공백(space)과 탭공간(tab)이 구분됨
- 관례적으로 스위트 당 4칸 공백 들여쓰기

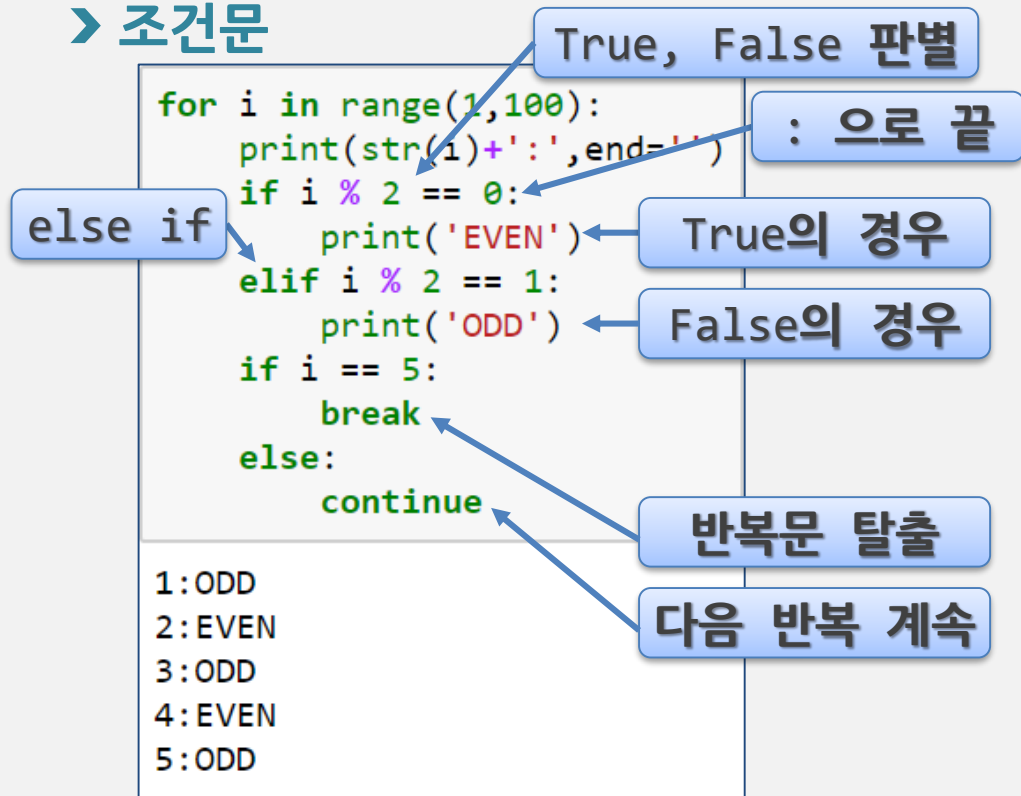


[출처] <https://www.python-course.eu>

1 파이썬 기초

(5) 반복문과 조건문

➤ 조건문



1 파이썬 기초

(5) 반복문과 조건문

➤ 리스트 컴프리헨션

처리 방법

```
squreOfEachone = [one*one for one in myList_1]
for eachone in squreOfEachone:
    print(eachone, end=' ')
print()
squreOfEven = [one*one for one in myList_1 if one % 2 == 0]
for eachone in squreOfEven:
    print(eachone, end=' ')
print()
isEven = [True if one%2==0 else False for one in myList_1]
for eachone in isEven:
    print(eachone, end=' ')
print()
```

if문과 함께 사용

if-else문과 함께 사용

1 4 9 16 25

4 16

False True False True False

- 각각의 원소를 동일한 방법으로 처리하여 새로운 리스트를 생성

1 파이썬 기초

(6) 함수(Function)

➤ 정의(definition)

```
def printList(theList):  
    for one in theList:  
        print(one, end=' ')  
    print()
```

함수 이름

```
def mySqr(n):  
    return n*n
```

함수 인자

```
def sqrList(l):  
    return [i*i for i in l]
```

함수 리턴

def와 :

- 여러 개의 인자는 ,로 구분

1 파이썬 기초

(6) 함수(Function)

➤ 기본 인자와 키워드 인자

```
def plusplus(a,b=1,c=2):  
    return a+b+c
```

기본 인자

```
print(plusplus(10))  
print(plusplus(1,2))  
print(plusplus(10,b=20))
```

키워드 인자

```
13  
5  
32
```

- 기본 인자가 정의된 경우 호출 시 생략 가능
- 호출 시 인자의 이름을 명시할 수 있음

1 파이썬 기초

(7) 클래스(class)

➤ 객체를 생성하기 위한 일종의 설계도

- 객체가 갖는 데이터와 객체의 기능에 대해 기술
 - 객체가 갖는 데이터 : 객체 변수
 - 객체의 기능 : 멤버 함수
- 객체에 관한 변수와 함수의 묶음

➤ 객체 (Object)

- 클래스라는 설계도로 실체화된 인스턴스
- 객체 변수는 각각 객체별로 독립적

1 파이썬 기초

(7) 클래스(class)

➤ 정의

클래스 이름

클래스 변수는 객체들간 공유

```
class Circle :
```

```
    pi = 3.141592
```

```
    def __init__(self, radius):
```

```
        self.radius = radius
```

```
    def setRadius(self, r):
```

```
        self.radius = r
```

```
    def getRadius(self):
```

```
        return self.radius
```

```
    def getArea(self):
```

```
        area = self.radius*self.radius*Circle.pi
```

```
        return area
```

생성자 정의

객체 변수

멤버 함수 정의

- 객체 변수는 멤버 함수 내에서 선언
- 멤버 함수의 첫 번째 인자는 self

1 파이썬 기초

(7) 클래스(class)

➤ 객체의 선언과 참조

```
a = Circle(10)
b = Circle(1)
print(a.getRadius())
print(a.getArea())
print(b.getRadius())
print(b.getArea())
```

10
314.1592
1
3.141592

객체

클래스

멤버 함수는 .을 이용해 참조

- 멤버 함수 호출 시 첫번째 인자 self 생략

Contents

1. 소개 및 환경 구축

- ① 기계학습과 파이썬
- ② 아이파이썬(IPython) 환경

2. 파이썬 실습

- ① 파이썬 기초
- ② 기계학습을 위한 파이썬

2 기계학습을 위한 파이썬



(1) 모듈과 패키지

➤ 모듈(Module)

- 변수, 함수, 클래스의 집합

➤ 패키지(Package)

- 모듈들의 집합

➤ import 명령어를 통해 코드에 포함

- `import name1, name2`
- `import name1.name1-1`
- `from name1 import name1-1, name1-2`
- `import name as newname`

2 기계학습을 위한 파이썬

(2) NumPy 패키지

➤ 대부분의 과학/수치 연산 패키지의 기초

- SciPy, scikit-learn, ...

➤ 다차원 배열 데이터 처리에 최적화

➤ 배열 브로드캐스팅, 선형대수, 푸리에 변환, 무작위 숫자 처리 기능 등



[출처] <http://www.numpy.org>

2 기계학습을 위한 파이썬

(2) NumPy 패키지

> 1차원 배열

패키지 импорт

자료형 명시

정수와 실수

역슬래쉬

```
import numpy as np
print('ones(5):', np.ones(5))
print('zeros(5):', np.zeros(5))
print('zeros(5, dtype=np.int32):', \
      np.zeros(5, dtype=np.int32))
print('arange(5):', np.arange(5))
print('arange(5.):', np.arange(5.), \
      'Type:', np.arange(5.).dtype)
```

```
ones(5): [1. 1. 1. 1. 1.]
zeros(5): [0. 0. 0. 0. 0.]
zeros(5, dtype=np.int32): [0 0 0 0 0]
arange(5): [0 1 2 3 4]
arange(5.): [0. 1. 2. 3. 4.] Type: float64
```


② 기계학습을 위한 파이썬

(2) NumPy 패키지

> 1차원 배열

```
import numpy as np
print('linspace(0,1,5):', np.linspace(0,1,5))
print('random.uniform(size=3):', \
      np.random.uniform(size=3))
print('myList_1:', np.array(myList_1), \
      'Type:', np.array(myList_1).dtype)
```

무작위 실수

```
linspace(0,1,5): [0.   0.25 0.5   0.75 1.   ]
random.uniform(size=3): [0.68548091 0.40360446 0.4261804 ]
myList_1: [1 2 3 4 5] Type: int64
```

리스트를 통한 생성

2 기계학습을 위한 파이썬



(2) NumPy 패키지

➤ 다차원 배열(ndarray)

- ndim : 차원 수
- shape : 각 차원의 길이
- dtype : 데이터 타입

[0,0]	[0,1]	[0,2]
[1,1]	[1,1]	[1,2]
[2,1]	[2,1]	[2,2]

- ndim : 2
- shape : 3,3

2 기계학습을 위한 파이썬

(2) NumPy 패키지

➤ 다차원 배열(ndarray)

```
x = np.arange(1,11)
print('dim. of x:',x.ndim)
xx = x.reshape(1,-1)
print('dim. of xx:', xx.ndim)
print('-----')
print('xx:')
print(xx)
print('Transposed xx:')
print(xx.T)
```

(1행, 맞춤) 로 변환

전치 행렬

1차원 배열

2차원 배열

dim. of x: 1
dim. of xx: 2

xx:

[[1 2 3 4 5 6 7 8 9 10]]

Transposed xx:

[[1]
 [2]
 [3]

2 기계학습을 위한 파이썬



(2) NumPy 패키지

➤ 다차원 배열(ndarray)

- 1차원 배열의 전치행렬은 무의미

```
print('x:')  
print(x)  
print('Transposed x:')  
print(x.T)
```

```
x:  
[ 1  2  3  4  5  6  7  8  9 10]  
Transposed x:  
[ 1  2  3  4  5  6  7  8  9 10]
```

② 기계학습을 위한 파이썬

(2) NumPy 패키지

➤ 다차원 배열(ndarray)

- 1차원 배열의 확장

원하는 형태로 반복

1차원으로 펼침

```
x2 = np.tile(x, (2,1))  
print(x2)  
print(np.ravel(x2))
```

```
[[ 1  2  3  4  5  6  7  8  9 10]  
 [ 1  2  3  4  5  6  7  8  9 10]]  
[ 1  2  3  4  5  6  7  8  9 10  1  2  3  4  5  6  7  8  9 10]
```

② 기계학습을 위한 파이썬

(2) NumPy 패키지

➤ 리스트 연산의 문제점

```
xList = [1,2,3,4,5]
```

```
print(xList)
```

```
print(xList+xList)
```

```
# print(xList*2) : ERROR !
```

```
# print(xList+10): ERROR !
```

+연산은 병합

문법 오류

```
[1, 2, 3, 4, 5]
```

```
[1, 2, 3, 4, 5, 1, 2, 3, 4, 5]
```

2 기계학습을 위한 파이썬



(2) NumPy 패키지

➤ 다차원 배열의 연산

```
print(xx)  
print(xx+xx)  
print(xx*2)  
print(xx+10)
```

```
[[ 1  2  3  4  5  6  7  8  9 10]]  
[[ 2  4  6  8 10 12 14 16 18 20]]  
[[ 2  4  6  8 10 12 14 16 18 20]]  
[[11 12 13 14 15 16 17 18 19 20]]
```

② 기계학습을 위한 파이썬

(2) NumPy 패키지

➤ 다차원 배열의 연산

```
print('xx:')  
print(xx)  
print('xx*xx.T:')  
print(xx*xx.T)
```

(1×10)

(10×1)

(10×10)

xx:

```
[[ 1  2  3  4  5  6  7  8  9 10]]
```

xx*xx.T:

```
[[ 1  2  3  4  5  6  7  8  9 10]  
 [ 2  4  6  8 10 12 14 16 18 20]  
 [ 3  6  9 12 15 18 21 24 27 30]  
 [ 4  8 12 16 20 24 28 32 36 40]  
 [ 5 10 15 20 25 30 35 40 45 50]  
 [ 6 12 18 24 30 36 42 48 54 60]  
 [ 7 14 21 28 35 42 49 56 63 70]  
 [ 8 16 24 32 40 48 56 64 72 80]  
 [ 9 18 27 36 45 54 63 72 81 90]  
 [10 20 30 40 50 60 70 80 90 100]]
```


2 기계학습을 위한 파이썬



(2) NumPy 패키지

➤ 다차원 배열의 연산

- 브로드캐스팅 연산되는 경우
 - $(m \times n)$ 배열과 $(1 \times n)$ 배열과 연산
 - $(m \times n)$ 배열과 $(m \times 1)$ 배열과 연산
- 그 외 shape이 다른 배열들간 연산은 불가능

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

2 기계학습을 위한 파이썬

(2) NumPy 패키지

➤ 다차원 배열의 연산

```
print('xx:')  
print(xx)  
print('np.dot:')  
print(np.dot(xx, xx.T))  
print('np.matmul:')  
print(np.matmul(xx, xx.T))  
print('@ operator:')  
print(xx@xx.T)
```

```
xx:  
[[ 1  2  3  4  5  6  7  8  9 10]]  
np.dot:  
[[385]]  
np.matmul:  
[[385]]  
@ operator:  
[[385]]
```

내적 함수

행렬곱 함수

행렬곱 연산자

▶ 다차원 배열의 병합

원소 별 비교

```
vstack:
[[ 1  2  3  4  5  6  7  8  9 10]
 [ 1  2  3  4  5  6  7  8  9 10]]
shape : (2, 10)
hstack:
[[ 1  2  3  4  5  6  7  8  9 10 11]
 [ 1  2  3  4  5  6  7  8  9 10 11]]
[[ True  True  True  True  True  True  True
  [ True  True  True  True  True  True  True
```

② 기계학습을 위한 파이썬

(2) NumPy 패키지

➤ 다차원 배열의 참조

```
mm = xx*xx.T  
print(mm[3,3])  
print(mm[3:7,3:7])  
print(mm[:, -1])  
print(mm[1,:])
```

리스트와 동일한 방식

```
16  
[[16 20 24 28]  
 [20 25 30 35]  
 [24 30 36 42]  
 [28 35 42 49]]  
[ 10  20  30  40  50  60  70  80  90 100]  
[ 2  4  6  8 10 12 14 16 18 20]
```