

Python 기초

1. Python 기본 문법



개요

Jupyter Notebook

Python Basics

개요

- ✓ 과정 개요
- ✓ 환경 설정

과정 개요

- ✓ 실습 위주로 진행 됩니다.
- ✓ 목표 : 분석하기 좋은 형태의 데이터를 만들어 낼 수 있다.
 - ① 데이터를 불러와서
 - ② 데이터를 탐색하고
 - ③ 데이터를 분석하기 알맞은 형태(dataframe)으로 만들 수 있다.

1일차

✓ Python 개요

- * 환경설정
- * Jupyter Notebook
- * Python 기본문법

2일차

✓ Numpy

- * Array 생성
- * Array 조회
- * Array 변환 및 연산

3일차

✓ Pandas

- * Pandas Series
- * Pandas Dataframe
- * Using SQL in Pandas

환경설정

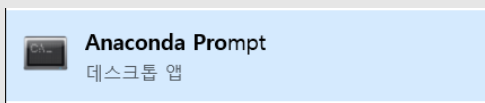
✓ Anaconda 설치

✓ 작업 폴더 생성

- 가급적 C:\Temp\Python

✓ Anaconda Prompt 실행

- 작업 폴더로 이동하여
- Jupyter Notebook 실행

A screenshot of the Anaconda Prompt terminal window. The title bar is white with the text "Anaconda Prompt". The terminal area has a black background with white text. The text shows the user navigating to the C:\Temp\python directory and running the jupyter notebook command.

```

Anaconda Prompt

(base) C:\Users\Gilbert Han>cd c:\temp\python

(base) c:\Temp\Python>jupyter notebook
```

Jupyter Notebook


- ✓ 과정 개요
- ✓ 환경 설정

Jupyter Notebook (1/2)

- ✓ Jupyter Notebook은 웹기반 개발도구로, Python, R 등 약 40가지의 개발언어를 지원
- ✓ 소스코드 셀과 결과, 그리고 설명(마크다운) 셀을 구분하여 작성 및 관리 가능
- ✓ 실행하면 백그라운드에서 커널이 실행
- ✓ 웹브라우저에서 노트북 파일을 실행/중지 등 관리

```
(base) c:\Temp\Python>jupyter notebook
[I 11:45:33.827 NotebookApp] JupyterLab extension loaded from C:\Users\Gilbert Han\
b
[I 11:45:33.828 NotebookApp] JupyterLab application directory is C:\Users\Gilbert H
[I 11:45:33.849 NotebookApp] Serving notebooks from local directory: c:\Temp\Python
[I 11:45:33.849 NotebookApp] The Jupyter Notebook is running at:
[I 11:45:33.849 NotebookApp] http://localhost:8888/?token=308a1a75ed6e5d0caa02f7895
[I 11:45:33.849 NotebookApp] Use Control-C to stop this server and shut down all ke
[C 11:45:33.949 NotebookApp]

Copy/paste this URL into your browser when you connect for the first time,
to login with a token:
http://localhost:8888/?token=308a1a75ed6e5d0caa02f78954adefa0e0fa79b8d18507
[I 11:45:34.208 NotebookApp] Accepting one-time-token-authenticated connection from
[W 11:45:45.899 NotebookApp] Notebook 00_Colab_Basics.ipynb is not trusted.
```



The screenshot shows the Jupyter Notebook web interface. At the top, there are 'Quit' and 'Logout' buttons. Below them are tabs for 'Files', 'Running', and 'Clusters'. The 'Running' tab is selected, showing a list of active notebooks. The interface includes a search bar and buttons for 'Upload', 'New', and a refresh icon. The list of notebooks is as follows:

	Name	Last Modified	File size
<input type="checkbox"/>	0		
<input type="checkbox"/>	00_Colab_Basics.ipynb	Running 8분 전	15.7 kB
<input type="checkbox"/>	01.Python_basics.ipynb	9시간 전	31.7 kB
<input type="checkbox"/>	71.Numpy.ipynb	9시간 전	29.9 kB
<input type="checkbox"/>	72.Pandas.ipynb	9시간 전	88.4 kB

Jupyter Notebook (2/2)

1. 셀 선택, 편집 모드 진입/선택모드

화살표, Enter, Esc

2. 셀을 실행하는 방법들

Alt / Ctrl / Shift + Enter

3. 마크다운 셀 / 코드 셀 전환

M, Y

2. 코드 Line number

L

5. 셀 생성, 셀 삭제

A,B / DD

5. 링크 삽입 / 이미지 삽입

[**링크이름**](URL) / ![**이미지이름**](URL)

7. 파이썬 스크립트 저장, 불러오기

%magic command

실습 #1 : Jupyter Notebook

Python Basics

- ✓ 목표
- ✓ data types
- ✓ Control
- ✓ 자료형(Container) Lists
- ✓ 자료형(Container) Dictionaries
- ✓ 함수
- ✓ Packages

목표

1. 데이터 타입을 알고 있다. - 숫자, 문자, boolean
2. 출력문(print)을 이용하여 출력할 수 있다.
3. 조건문(if-else)이 무엇인지 알고 조건문이 사용된 소스코드를 이해할 수 있다.
4. 반복문(loop)이 무엇인지 알고 반복문이 사용된 소스코드를 이해할 수 있다.
5. 자료형이 무엇인지 알고 자료형이 사용된 소스코드를 이해할 수 있다.
6. List, Dictionary 등 주요 자료형의 특징과 차이점을 안다.
7. 함수(Function)가 무엇인지 알고, 함수가 사용된 소스코드를 이해할 수 있다. 간단한 함수를 작성할 수 있다.
8. Python으로 작성된 소스코드를 이해하고, 간단한 코드를 작성할 수 있다.

Help 이용하기

✓ ? 함수명

- 함수에 대한 설명

```
In [1]: # help '?' 이용,  
        ?print
```

Docstring:
print(value, ..., sep=' ', end='\n', file=sys.stdout, flush=False)

Prints the values to a stream, or to sys.stdout by default.
Optional keyword arguments:

✓ () 안에 Shift+Tab

- Tooltip 처럼 설명이 나타남

```
# () 안에 Shift + Tab  
sum()
```

Signature: sum(iterable, start=0, /)

Docstring:
Return the sum of a 'start' value (default: 0) plus an iterable of numbers

Python 연산자

비교

연산자	설명
>	큼
<	작음
>=	크거나 같음
<=	작거나 같음
!=	같지 않음
==	같음

계산

연산자	설명
+	덧셈
-	뺄셈
*	곱셈
/	나눗셈
//	나눗셈의 몫
%	모듈로 (나눗셈의 나머지)
**	지수 연산자

Basic data types

✓변수

- 변수 이름 : 대소문자 구분
- 각 변수는 하나의 단일 값을 저장

Python

```
height = 1.73  
tall = True
```

✓Types

- Type() : 변수의 타입을 확인
- float - real numbers
- int - integer numbers
- str - string, text
- bool - True, False

Basic data types

✓문자열을 다루는 함수들

- `len()` : 문자열의 길이를 반환
- `.format()` : 문자열 안의 특정 위치에 값을 포함시킴.
- `.capitalize()` : 알파벳 첫문자를 대문자로.
- `.upper()` : 알파벳 전체를 대문자로.
- `.rjust()` : 오른쪽 정렬
- `.replace('o', 'go')` : 'o'를 'go'로 변경
- `.strip()` : 앞 뒤 공백 제거

Basic data types

.format

String

Old

```
'%s %s' % ('one', 'two')
```

New

```
'{} {}'.format('one', 'two')
```

Output

```
o n e   t w o
```

Number

```
'%d %d' % (1, 2)
```

```
'{} {}'.format(1, 2)
```

```
1 2
```


실습 #2 : Python_basics ①

Control 1/3

✓조건문

if 조건1:	조건1이 True이면
처리1	처리1
elif 조건2:	조건1이 아니고 and 조건2이면
처리2	처리2
else:	아니면
처리3	처리3

- 제어문 블록은 ':' 와 (들여쓰기) 로 구분

Python

```
my_score = 73

if my_score >= 90:
    my_grade = 'A'
elif my_score >= 80:
    my_grade = 'B'
elif my_score >= 70:
    my_grade = 'C'
elif my_score >= 60:
    my_grade = 'E'
else:
    my_grade = 'F'

print(my_grade)
```

Control 2/3

✓반복문 : for

for i in 리스트 :
처리

첫번째부터 마지막
값까지 하나씩 가져다
처리

- range(5) : 1~5까지 자연수
- [.....,,] : 리스트를 나타냄
- enumerate() 순서가 있는 자료형을 입력받아, 인덱스와 값을 포함하는 오브젝트로 리턴.

Python

```
for i in range(5):  
    print(i)
```

```
fruits = ['apple', 'banana', 'cherry']  
print(type(fruits))  
for i in fruits:  
    print(i)
```

```
for index, value in enumerate(fruits):  
    print(index, value)
```

Control 3/3

✓반복문 : while

while 조건 :

처리

조건변경문

조건이 참인 경우
처리

Python

```
i = 0
while i < len(fruits):
    print(fruits[i])
    i += 1
```

실습 #3 : Python_basics ②

자료형(Container) Lists

✓ 기존 변수의 한계

- 단일 값 저장 → 많은 데이터 처리 곤란

✓ List : [a, b, c]

- 값의 집합
- 하나의 리스트에 **여러 형식의** 데이터 저장 가능

Python

```
age1 = 44  
age2 = 44  
age3 = 14  
age4 = 12  
age5 = 8
```

```
Family_age = [44,44,14,12,8]
```

```
Family = ['Dad',44, 'Mom',44,  
          'Son1',14, 'Son2',12, 'Son3',8]
```

자료형(Container) Lists

✓리스트의 데이터 사용하기

- Index number

```
family = ['Dad', 44, 'Mom', 44, 'Son1', 14, 'Son2', 12, 'Son3', 8]
```

Index :	0	1	2	3	4	5	6	7	8
9									
	-10	-9	-8	-7	-6	-5	-4	-3	-2
-1									

Python

```
Family[0]  
'Dad'
```

```
Family[-3]  
12
```

자료형(Container) Lists

✓리스트의 데이터 사용하기

- Slicing : [Start Index(포함) : End Index(미포함)]

```
family = ['Dad', 44, 'Mom', 44, 'Son1', 14, 'Son2', 12, 'Son3', 8]
```

Index :	0	1	2	3	4	5	6	7	8	9
	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1

Python

```
Family[2:6]  
['Mom', 44, 'Son1', 14]
```

Python

```
Family[6:]  
['Son2', 12, 'Son3', 8]
```


자료형(Container) Lists

✓리스트의 데이터 변경하기

```
family = ['Dad', 44, 'Mom', 44, 'Son1', 14, 'Son2', 13, 'Son3', 8]
```

Index :	0	1	2	3	4	5	6	7	8	9
	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1

Python

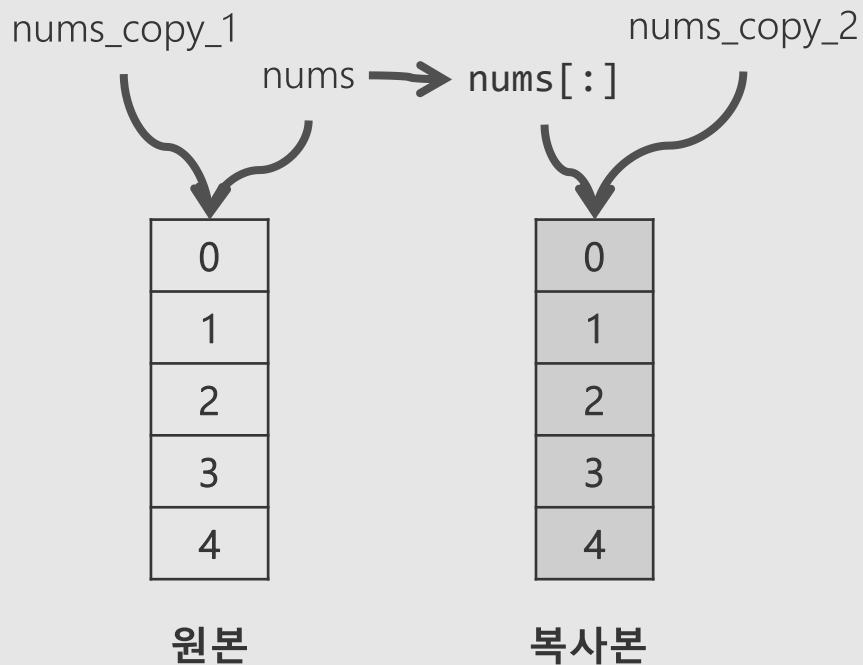
```
Family[7] = 13
```

Python

```
Family[8:] = ['Son4', 9]
```

자료형(Container) Lists

✓리스트 복사하기



Python

```
nums_copy_1 = nums
nums_copy_2 = nums[:]
print(nums_copy_1)    #[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
print(nums_copy_2)    #[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

```
nums_copy_1[0] = 'a'
print(nums_copy_1)    #[a, 1, 2, 3, 4, 5, 6, 7, 8, 9]
print(nums_copy_2)    #[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
print(nums)           #[a, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

```
nums_copy_2[0] = 'b'
print(nums_copy_1)    #[a, 1, 2, 3, 4, 5, 6, 7, 8, 9]
print(nums_copy_2)    #[b, 1, 2, 3, 4, 5, 6, 7, 8, 9]
print(nums)           #[a, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

실습 : Python_basics ③

자료형(Container) Lists

✓리스트와 반복문

- 반복할 범위를 리스트로 지정하여 반복

Python

```
animals = ['cat', 'dog', 'monkey']  
for animal in animals:  
    print(animal)
```

- 반복문 확장문법

for x in nums:

squares.append(x ** 2)

→ squares = [x ** 2 for x in nums]

Python

```
nums = [0, 1, 2, 3, 4]  
squares = []
```

```
for x in nums:  
    squares.append(x ** 2)
```

```
print(squares)
```

```
squares = [x ** 2 for x in nums]
```

실습 : Python_basics ④

자료형(Container) Dictionary

✓ Dictionary 구조 : {key1 : value1, key2 : value2, ... }

Python

```
# dictionary 생성  
d = {'cat': 'cute', 'dog': 'furry'}  
print(d)
```

```
# ictionary 특정 값 조회  
print(d['cat'])      # "cute"
```

```
# key 값이 들어있는지 확인  
print('cat' in d)    # "True"  
print('sheep' in d)  # "False"
```

```
# 새로운 (key, value) 추가  
d['fish'] = 'wet'  
print(d['fish'])     # "wet"
```

```
# key가 없는 경우, default value 사용  
print(d.get('monkey', 'N/A')) # "N/A"  
print(d.get('fish', 'N/A'))   # "wet"
```

```
# 요소 삭제  
del d['fish']  
print(d.get('fish', 'N/A')) # "N/A"
```

자료형(Container) Dictionary

✓ Dictionary 와 반복문

- list와 비슷하나, key와 value로 제어

Python

```
d = {'person': 2, 'cat': 4, 'spider': 8}
for key in d:
    value = d[key]
    print('A {} has {} legs'.format(key, value))

# 딕셔너리에서 루프로 키와 값을 동시에 가져오려면 .items()
for key, value in d.items():
    print('A %s has %d legs' % (key, value))
```

- 확장문법

Python

```
animals = {'cat': 11, 'dog': 7, 'fish': 15, 'horse': 3}
animals10 = {k:v for k, v in animals.items() if v > 10}
print(animals_over_10) # '{"cat': 11, 'fish': 15}"
```

실습 : Python_basics ⑤

자료형(Container) Set, Tuple

✓ Set : { .., .. }

- 집합
- 중복된 값을 중첩되어 가지지 않는 리스트

Python

```
animals = {'cat', 'dog'}
print(animals)

animals.add('fish')    # 요소 추가

# set 요소 갯수
print(len(animals))    # "3"
animals.add('cat')     # 중복 값은 추가되지 않음.
print(len(animals))    # "3"
print(animals)
```

✓ Tuple : (.., ..)

- 숫자로 구성된 좌표처럼 사용

Python

```
# 튜플 생성
t = (5, 6)
print(type(t))        # Prints "<class 'tuple'>"
print(t)              # "(5, 6)"
print(t[0])           # "5"
```

함수 – Method

✓call functions *on* objects

Python

```
name = "hky"
```

```
age = 45
```

```
customer = ["hky", 44, "nkm", 44]
```

Type

Methods

str

.capitalize(), .replace()

float

.bit_length()

list

.index(), .count()

함수 – User Defined Function

```
def 함수명(input_param, ...) :  
    code...
```

Python

```
def sign(x):  
    if x > 0:  
        return 'positive'  
    elif x < 0:  
        return 'negative'  
    else:  
        return 'zero'  
  
for x in [-1, 0, 1]:  
    print(sign(x))
```

실습 : Python_basics ⑥

Packages

✓ 개요

- Directory of Python Scripts
- Each script = module
- Specify functions, methods, types
- Thousands of packages available

✓ 설치

- Download get-pip.py
- Terminal:
python3 get-pip.py
pip3 install numpy

Packages

✓ Import

- 설치된 패키지 사용하기 위해 코드 페이지에 로딩

✓ import numpy

Python

```
import numpy
```

```
array([1,2,3])
```

```
NameError: name 'array' is not defined
```

```
numpy.array([1,2,3])
```

```
import numpy as np
```

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

• from numpy import array

Python

```
from numpy import array
```

```
np_fam = array(fam_ext)
```