

파이썬과 자료구조

이준원
SKT Data Scientist

A diagram with 'AI' in a box at the top, connected by a line to '학습 목표' (Learning Objectives). From '학습 목표', three lines branch out to three numbered learning objectives. The background of the slide features a city skyline at night, binary code (0s and 1s), and various tech icons like a globe, Wi-Fi, and a location pin.

AI

학습 목표

1. 데이터 분석과 파이썬에 대해 설명할 수 있다.
2. IPython과 노트북을 사용할 수 있다.
3. 파이썬 자료구조를 다룰 수 있다.



I _ 데이터 분석과 Python

- 데이터 분석이란
- 파이썬을 통한 데이터 분석

II _ IPython과 노트북

- IPython
- 주피터 노트북

III _ 파이썬 자료구조

- 튜플 (tuple)
- 리스트 (list)
- 딕셔너리 (dictionary)

I

데이터 분석과 파이썬

CHAPTER

- 데이터 분석이란
- 파이썬을 통한 데이터 분석

Institute for
K-Digital Training
미래를 향한
인재를 양성합니다



빅데이터와 데이터 과학

- 빠른 속도로 쌓이는 방대한 데이터 (빅데이터)
- 높은 컴퓨팅 파워와 저렴한 클라우드 비용
- 데이터 과학의 대중화 (분석, 학습, 연구)
- 데이터 과학자는 어떤 일을 하는가?
 - 데이터를 수집하고 정리하는 일
 - 알고리즘을 선택하고 데이터를 모델링
 - 결과를 분석하여 유용한 정보를 추출

데이터 분석

- 구조화된 데이터가 필요
 - 행과 열로 이루어진 표 or 스프레드시트
 - ex) 문자열 데이터, 시계열 데이터, 숫자 데이터
- 데이터 분석을 하기 위한 도구
 - 파이썬 프로그래밍
 - 데이터 분석 라이브러리
 - 데이터 모델링 기법

파이썬을 사용하는 이유?

- 가장 직관적이고 배우기 쉬우며 사용하기에도 편함
 - 프로그래밍에 익숙하지 않은 사람도 쉽게 배울 수 있는 하이레벨 언어
 - 작은 프로그램이나 자동화된 스크립트를 만들기 쉬움
- 대용량의 데이터를 빠르게 처리할 수 있음
 - 마이크로소프트 엑셀과 비교
- 다양한 라이브러리를 지원함
 - numpy, pandas (데이터)
 - matplotlib, seaborn (시각화)
 - sklearn, tensorflow (머신러닝 모델링)

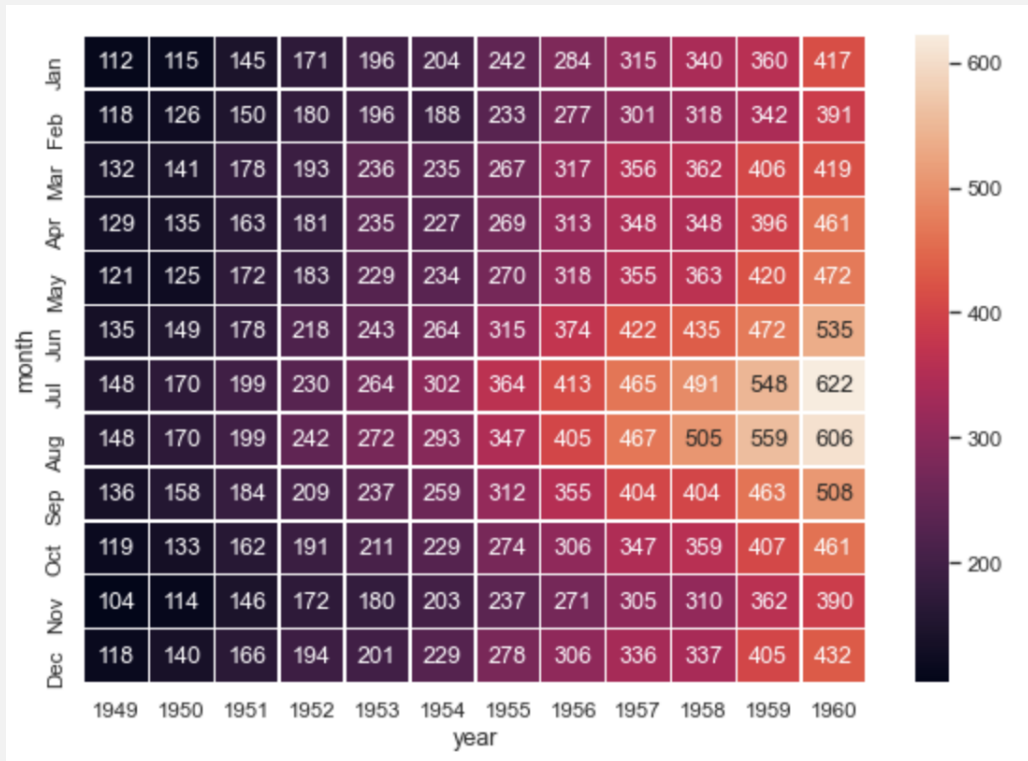
데이터 라이브러리

- Numpy
 - Numerical Python, 파이썬 산술 연산의 기본이 되는 라이브러리
 - 다차원 배열을 다룰 수 있으며, 배열간의 수학 계산을 수행할 수 있음
 - 선형 대수, 난수 생성 등의 고급 수학 기능도 제공함
- Pandas
 - 구조화된 데이터(행/열)를 빠르고 쉽게 다룰 수 있는 라이브러리
 - 다양한 형태의 데이터를 다룰 수 있음 (숫자, 문자열, 시계열, 범주형)
 - 누락/중복 데이터 처리 등의 데이터 전처리 기능을 제공함
 - 관계형 데이터베이스처럼 데이터를 합치거나 관계 연산을 수행할 수 있음

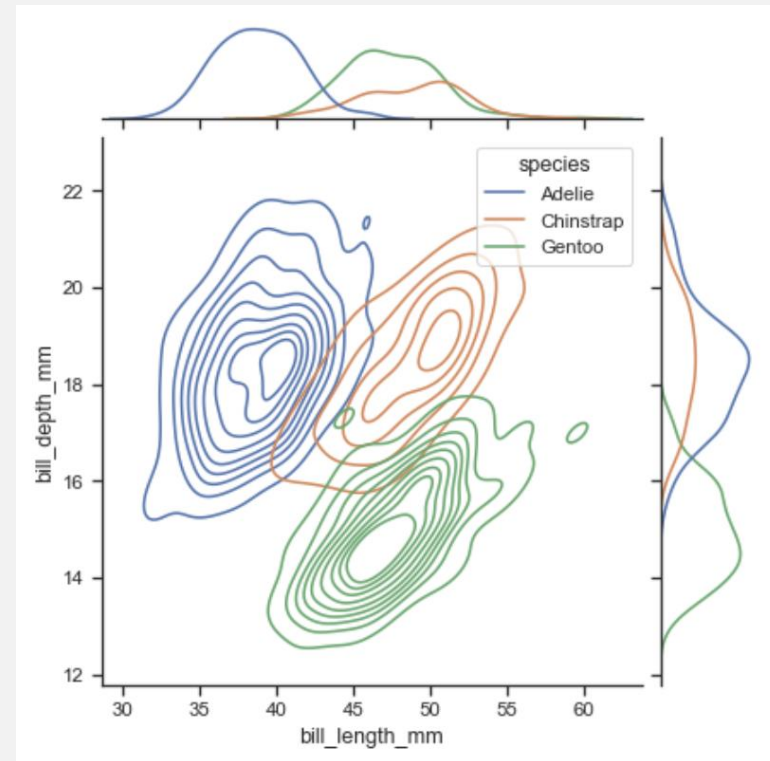
시각화 라이브러리

- Matplotlib
 - 2차원 평면 그래프를 생성하는 시각화 라이브러리
 - 가장 기본이 되는 시각화 라이브러리로 다른 라이브러리와 연동이 잘 됨
 - 객체지향 프로그래밍을 지원하며 그래프의 다양한 요소를 customize할 수 있음
- Seaborn
 - Matplotlib의 기능과 스타일을 확장한 고급 버전
 - 단순한 인터페이스를 제공하며 다양한 통계 차트를 제공함
 - [Seaborn Gallery](#)

Seaborn Gallery



heatmap



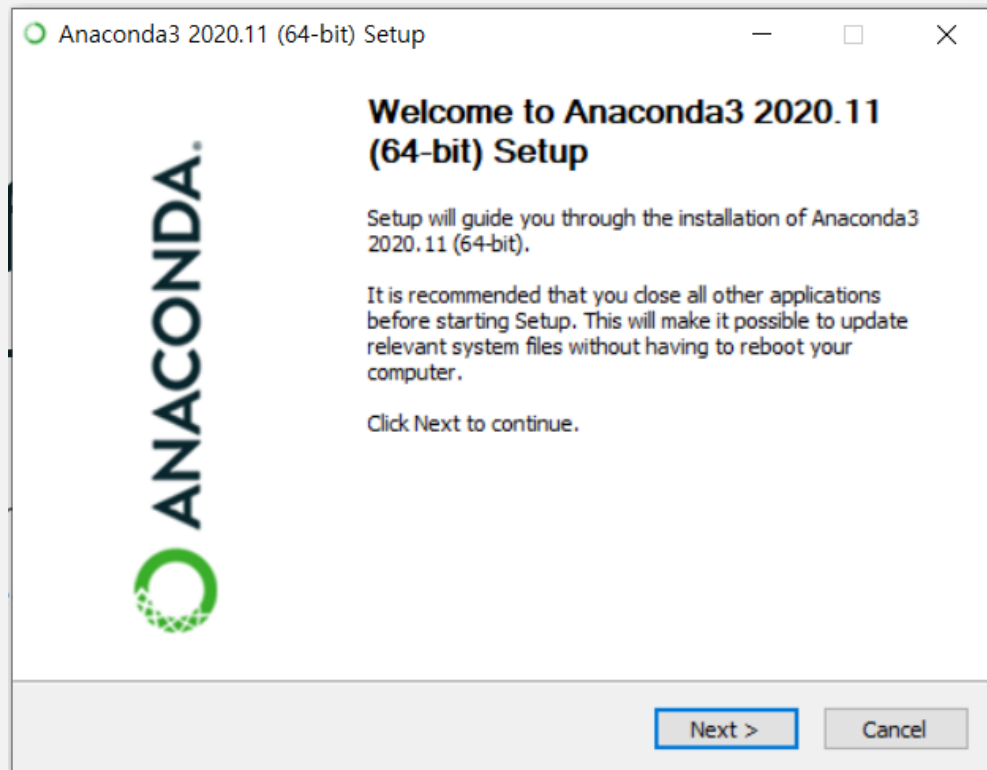
jointplot

Scikit-learn

- 범용 머신러닝 도구로 사용되며 다양한 종류의 알고리즘을 지원함
- 분류: SVM, 최근접 이웃, 랜덤 포레스트, 로지스틱 회귀
- 클러스터링: k-평균, 스펙트럴 클러스터링, DBSCAN
- 차원축소: 주성분분석(PCA), 행렬 인수분해(MF), 잠재 디리클레 할당(LDA)
- 기타: 모델 선택, 교차 검증, 변수 정규화, 성능 평가

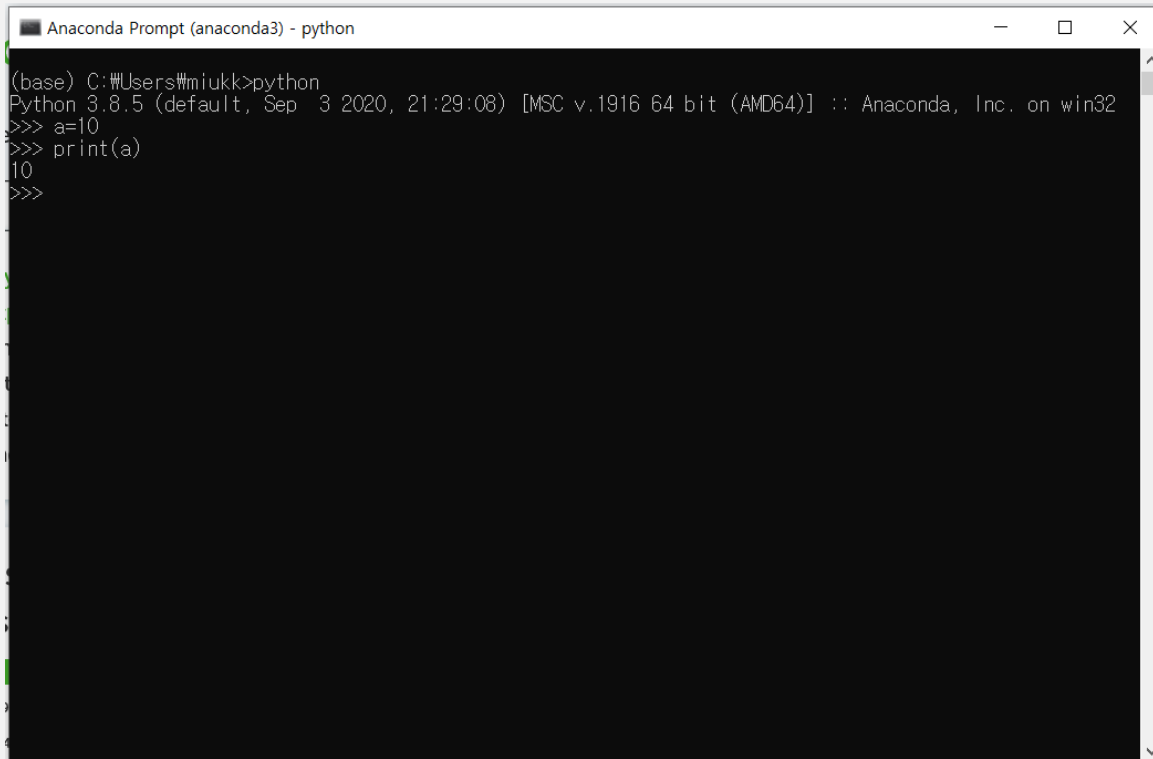
파이썬 설치 (윈도우)

- 윈도우용 아나콘다 인스톨러 설치
 - ex) Anaconda3-2020.11-Windows-x86_64.exe



파이썬 설치 (윈도우)

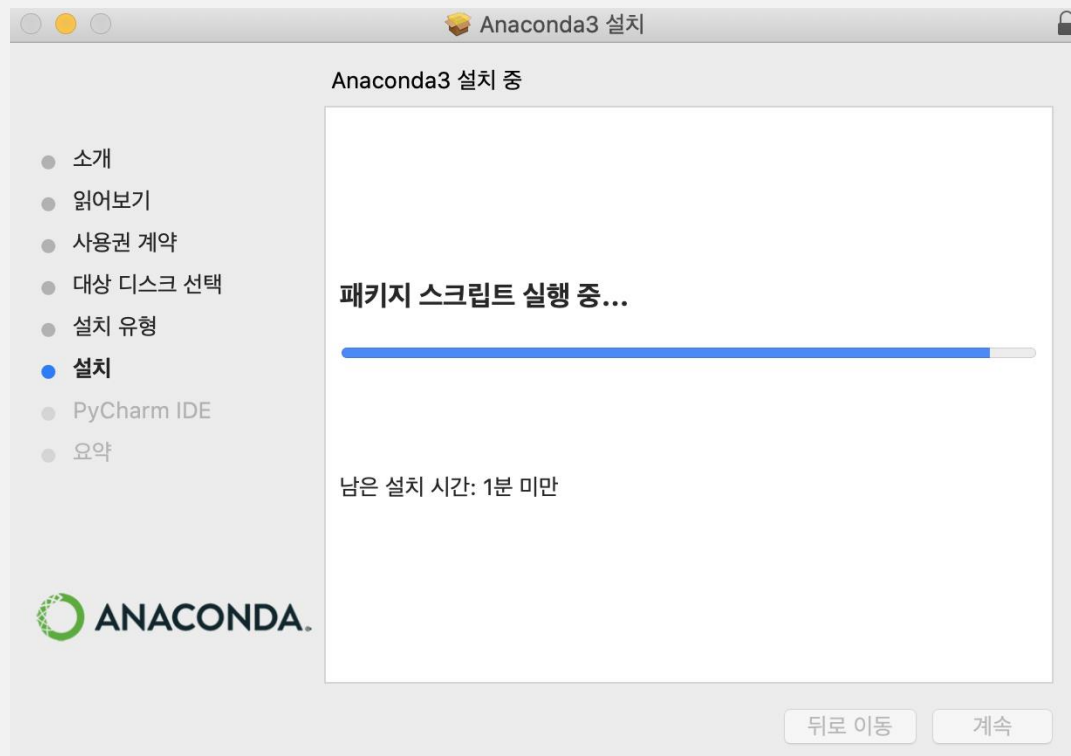
- 명령 프롬프트를 열고 python을 입력하면 파이썬 인터프리터가 실행됨
- 명령 프롬프트 대신에 Anaconda Prompt를 실행하는 것도 가능

A screenshot of the Anaconda Prompt window. The title bar reads "Anaconda Prompt (anaconda3) - python". The command prompt shows the following text:

```
(base) C:\Users\#miukk>python
Python 3.8.5 (default, Sep 3 2020, 21:29:08) [MSC v.1916 64 bit (AMD64)] :: Anaconda, Inc. on win32
>>> a=10
>>> print(a)
10
>>>
```

파이썬 설치 (macOS)

- macOS용 아나콘다 인스톨러 설치
 - ex) Anaconda3-2020.11-MacOSX-x86_64.pkg



파이썬 설치 (macOS)

- Terminal 앱을 열고 python을 입력하면 설치된 파이썬 인터프리터가 실행됨

```
~ python 일 5/ 9 01:08:36 2021
Python 3.7.6 (default, Jan 8 2020, 13:42:34)
[Clang 4.0.1 (tags/RELEASE_401/final)] :: Anaconda, Inc. on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> █
```

파이썬 패키지 설치 및 업데이트

- 패키지 설치

```
conda install package_name (ex. numpy, pandas)
```

```
pip install package_name
```

- 패키지 업데이트

```
conda update package_name
```

```
pip install --upgrade package_name
```


II

IPython과 노트북

CHAPTER

- IPython
- 주피터 노트북

Institute for
K-Digital Training
미래를 향한
인재를 양성합니다



IPython이란?

- Interactive Python의 약어로 대화형(interactive) 파이썬 인터프리터를 의미함
 - IPython 커널에서 편리하게 파이썬 코드를 작성하고 테스트하고 디버깅 할 수 있음
 - 데이터 분석 과정에서 필요한 탐색적이고 반복적인 코딩을 하기에 편리함
- 전통적 프로그래밍
 - 코드 편집 → 컴파일 → 실행
- 데이터 분석 프로그래밍
 - 실행 → 탐색 방식

IPython 셀 실행

- IPython은 파이썬 인터프리터를 실행하듯이 ipython 명령어를 통해 실행

```
~ ipython 48.7s < 일 5/ 9 00:38:36 2021
Python 3.7.6 (default, Jan 8 2020, 13:42:34)
Type 'copyright', 'credits' or 'license' for more information
IPython 7.13.0 -- An enhanced Interactive Python. Type '?' for help.

In [1]: var = 100

In [2]: print(var)
100

In [3]: var = var + 100

In [4]: var
Out[4]: 200

In [5]: █
```

IPython 셀 실행

- interactive하게 코드를 한 줄씩 입력하면서 변수를 생성하고 출력할 수 있음

```
In [5]: import numpy as np
In [6]: data = [np.random.randint(0, 100) for i in range(10)]
In [7]: data
Out[7]: [9, 33, 97, 39, 1, 12, 20, 91, 41, 77]
In [8]: new_data = [num + 10 for num in data]
In [9]: new_data
Out[9]: [19, 43, 107, 49, 11, 22, 30, 101, 51, 87]
In [10]: █
```

주피터 노트북은

- IPython을 웹 기반 인터페이스에서 사용할 수 있게 해줌
 - 웹 브라우저 내에서 코드를 입력하고 실행하면 결과를 바로 확인할 수 있음
 - 주피터 커널은 IPython 시스템을 이용하여 동작함
- 노트북의 내용을 마크다운이나 HTML로 저장할 수 있음
 - 텍스트와 코드의 실행 결과를 포함하는 문서를 생성할 수 있음
- 기존에 IPython 노트북으로 불렸으나 Python 외에 Go, R 등의 다양한 언어를 지원하면서 주피터 노트북으로 이름을 바꿈

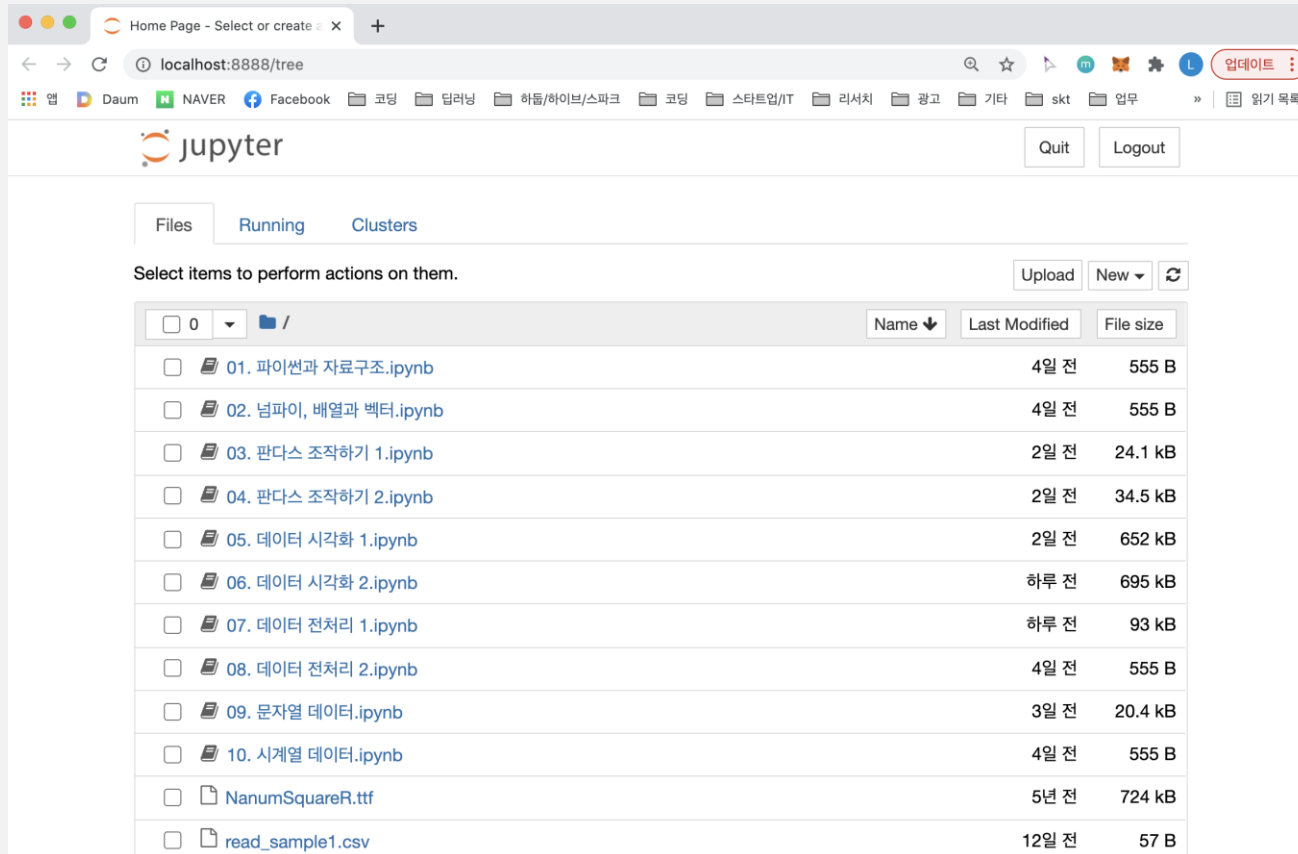
주피터 노트북 실행

- 명령 프롬프트 혹은 터미널에서 jupyter notebook 실행

```
~/W/t/data_analysis > jupyter notebook 일 5/ 9 00:43:43 2021
jupyter_http_over_ws extension initialized. Listening on /http_over_websocket
[I 00:43:54.807 NotebookApp] Serving notebooks from local directory: /Users/1112922/Work/tmp/data_
analysis
[I 00:43:54.807 NotebookApp] The Jupyter Notebook is running at:
[I 00:43:54.807 NotebookApp] http://localhost:8888/
[I 00:43:54.807 NotebookApp] Use Control-C to stop this server and shut down all kernels (twice to
skip confirmation).
```

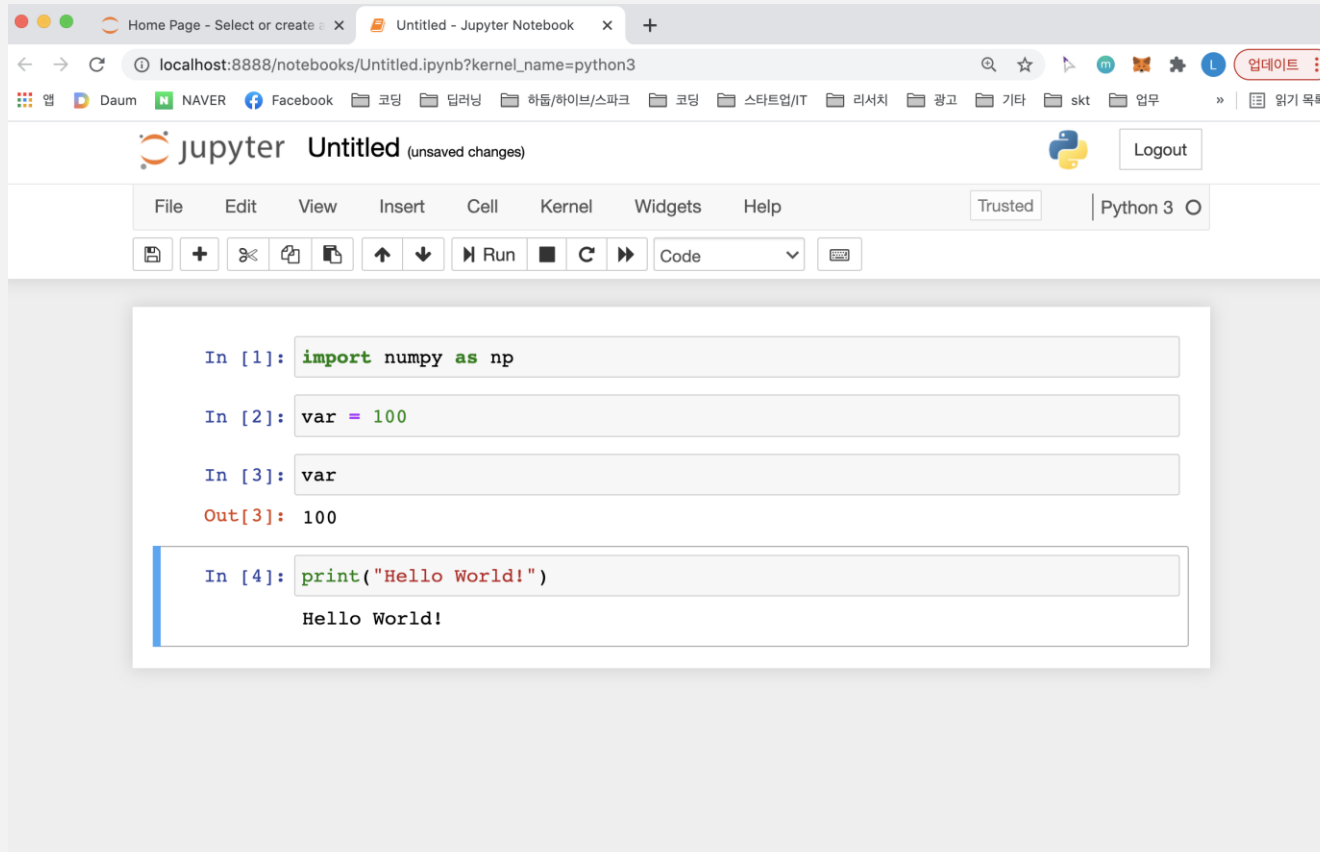
주피터 노트북 실행

- 웹 브라우저를 통해 주피터 노트북에 접속할 수 있음



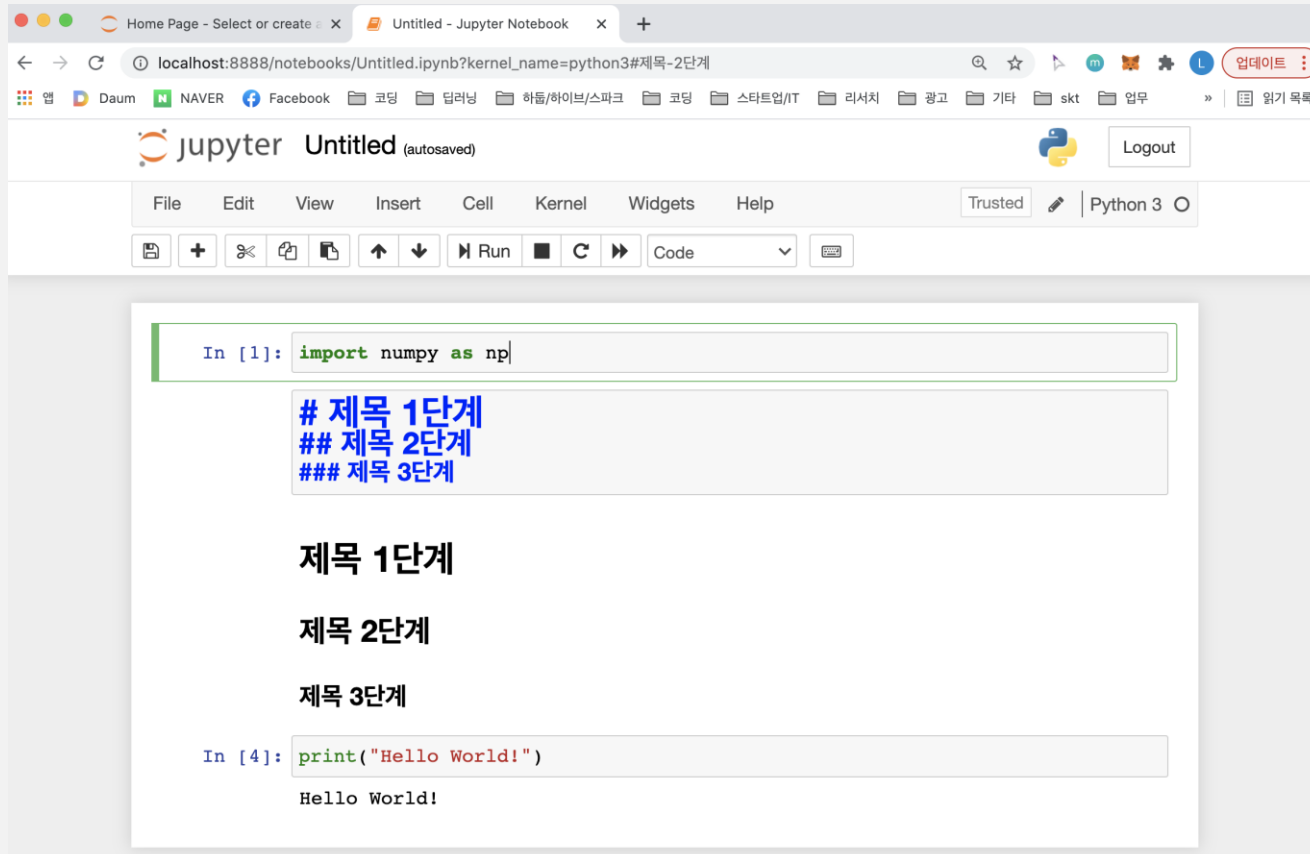
주피터 노트북 실행

- .ipynb 파일로 생성된 노트북 파일 내에서 IPython처럼 파이썬을 사용할 수 있음



주피터 노트북 실행

- 마크다운 문법 사용 가능





파이썬 자료구조

CHAPTER

- 튜플 (tuple)
- 리스트 (list)
- 딕셔너리 (dictionary)

Institute for
K-Digital Training
미래를 향한
인재를 양성합니다



단순하고 강력한 자료구조

- 데이터를 담는 가장 기초가 되는 것은 파이썬 내장 자료구조
- numpy, pandas 등의 라이브러리를 다루기 전에 기본적인 파이썬 자료구조를 알아야 함
 - 파이썬 내장 자료 처리 도구와 함께 애드온 라이브러리를 사용함
- 파이썬 자료구조 종류
 - 튜플 (tuple)
 - 리스트 (list)
 - 사전 (dictionary)
 - 집합 (set)

튜플 (tuple)

```
tup = 1, 2, 3  
tup
```

```
(1, 2, 3)
```

```
tup[2]
```

```
3
```

```
nested_tup = (1, 2, 3), (4, 5)  
nested_tup
```

```
((1, 2, 3), (4, 5))
```

```
nested_tup[1][1]
```

```
5
```

```
string_tup = tuple('string')  
string_tup
```

```
('s', 't', 'r', 'i', 'n', 'g')
```

```
string_tup[5] = 'a'
```

```
-----  
TypeError                                Traceback (most recent call last)  
<ipython-input-22-a568c2c39f80> in <module>  
----> 1 string_tup[5] = 'a'  
  
TypeError: 'tuple' object does not support item assignment
```

- 1차원의 고정된 크기를 가지는 변경 불가능한 순차 자료구조
- 쉼표로 구분된 값을 대입함

튜플 연산

```
tup = 1, 2, 3  
a, b, c = tup
```

```
print(a, b, c)
```

```
1 2 3
```

```
for num in tup:  
    print(num)
```

```
1  
2  
3
```

```
tup = (1, 2, 2, 2, 3, 4, 2, 4, 1)
```

```
tup.count(2)
```

```
4
```

```
tup.count(4)
```

```
2
```

- tuple의 원소를 변수로 할당할 수 있음
- 다양한 연산을 위한 자료형은 아님

리스트 (list)

```
a_list = [1, 2, 3, 4]
a_list
```

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

```
tup = ('a', 'b', 3, 4)
tup_list = list(tup)
tup_list
```

```
['a', 'b', 3, 4]
```

```
tup_list[2] = 10
tup_list
```

```
['a', 'b', 10, 4]
```

```
gen = range(10)
gen
```

```
range(0, 10)
```

```
gen_list = list(gen)
gen_list
```

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

- 여러 개의 변수를 순차적으로 담을 수 있는 자료구조
- []를 이용하거나 list()를 이용하여 선언
- 파이썬은 다양한 자료형을 하나의 리스트에 담을 수 있음

리스트 연산

```
s_list = ['apple', 'banana', 'cat']  
s_list.append('dog')
```

```
s_list  
['apple', 'banana', 'cat', 'dog']
```

```
s_list.insert(2, 'baby')
```

```
s_list  
['apple', 'banana', 'baby', 'cat', 'dog']
```

```
word = s_list.pop(3)  
print(word)  
print(s_list)
```

```
cat  
['apple', 'banana', 'baby', 'dog']
```

```
print(s_list)  
s_list.remove('baby')  
print(s_list)
```

```
['apple', 'banana', 'baby', 'dog']  
['apple', 'banana', 'dog']
```

```
s_list = ['apple', 'banana', 'dog']
```

```
'apple' in s_list
```

```
True
```

```
'cat' in s_list
```

```
False
```

- **append**: 리스트 맨 뒤에 원소를 삽입
- **insert**: 정해진 위치에 원소를 삽입
- **pop**: 정해진 위치의 원소를 추출
- **remove**: 특정 value 값을 가진 원소를 제거
- **in**: 리스트에 원소가 있는지를 참/거짓으로 반환

리스트 정렬

```
num_list = [7, 10, 3, 1, 2, 6]
num_list.sort()
print(num_list)
```

```
[1, 2, 3, 6, 7, 10]
```

```
num_list.sort(reverse=True)
print(num_list)
```

```
[10, 7, 6, 3, 2, 1]
```

```
word_list = ['see', 'Him', 'book', 'six', 'foxes', 'Small']
word_list.sort()
print(word_list)
```

```
['Him', 'Small', 'book', 'foxes', 'see', 'six']
```

```
word_list.sort(key=len)
print(word_list)
```

```
['Him', 'see', 'six', 'book', 'Small', 'foxes']
```

- sort 메소드를 사용해 정렬
- reverse=True는 내림차순으로 정렬
- 문자열도 정렬이 가능
- 정렬의 기준값을 key를 이용해 사용자가 정의할 수 있음

리스트 슬라이싱

```
num_list = [7, 10, 3, 1, 2, 6, 1, 2]
num_list[1:5]
```

```
[10, 3, 1, 2]
```

```
num_list = [7, 10, 3, 1, 2, 6, 1, 2]
num_list[3:4] = [9, 7]
num_list
```

```
[7, 10, 3, 9, 7, 2, 6, 1, 2]
```

```
num_list = [7, 10, 3, 1, 2, 6, 1, 2]
print(num_list[:4])
print(num_list[5:])
```

```
[7, 10, 3, 1]
[6, 1, 2]
```

```
num_list = [7, 10, 3, 1, 2, 6, 1, 2]
print(num_list[-2:])
print(num_list[-6:-2])
```

```
[1, 2]
[3, 1, 2, 6]
```

- 리스트의 위치값을 활용하여 특정 원소들을 잘라내서 추출하는 것
- : 을 사용하여 범위를 지정할 수 있음

딕셔너리 (dictionary)

```
data_dict = {'key1': 'value1', 'key2': 30}
print(data_dict)
print(data_dict['key1'])
print(data_dict['key2'])
```

```
{'key1': 'value1', 'key2': 30}
value1
30
```

```
data_dict['key3'] = 'value3'
data_dict
```

```
{'key1': 'value1', 'key2': 30, 'key3': 'value3'}
```

```
data_dict['key1'] = 20
data_dict
```

```
{'key1': 20, 'key2': 30, 'key3': 'value3'}
```

```
'key1' in data_dict
```

```
True
```

```
20 in data_dict
```

```
False
```

- key, value로 이루어진 자료구조
- {}를 사용하여 딕셔너리를 선언
- 딕셔너리의 key는 모두 고유값을 가지며, key를 활용해 value 값을 바로 추출

딕셔너리 key, value

```
# key, value 메소드
data_dict = {'key1': 20, 'key2': 30, 'key3': 'apple'}
key_list = list(data_dict.keys())
value_list = list(data_dict.values())
print(key_list)
print(value_list)
```

```
['key1', 'key2', 'key3']
[20, 30, 'apple']
```

```
for key, value in data_dict.items():
    print(key, value)
```

```
key1 20
key2 30
key3 apple
```

```
# zip을 이용한 dictionary 생성
```

```
map_dict = dict(zip(['a', 'b', 'c', 'd'], [10, 9, 8, 7]))
map_dict
```

```
{'a': 10, 'b': 9, 'c': 8, 'd': 7}
```

- key와 value를 list로 각각 추출할 수 있음
- key, value 쌍을 for 문을 통해 동시에 가져와서 사용할 수 있음
- zip을 이용해 두 개의 리스트를 하나의 딕셔너리로 만들 수 있음

내용 정리

- 파이썬을 사용하면 쉽고 빠르게 데이터 분석을 할 수 있음
 - 다양한 라이브러리 제공 (numpy, pandas, matplotlib, scikit-learn)
- IPython과 주피터 노트북을 활용하여 interactive하게 코딩과 분석을 수행
 - 탐색적이고 반복적인 코딩을 하기에 편리함
- 단순하고 강력한 파이썬 내장 구조에 대해서 살펴봄
 - 튜플(tuple), 리스트(list), 딕셔너리(dictionary), 집합(set)

Thank you



junwonee@gmail.com