머신러닝응용 제01강

Data handling with Python

첨단공학부 김동하교수

제01강 Data handling with Python

1	Python의 변수 및 자료의 형태에 대해 학습한다.
2	데이터의 입출력에 대해 학습한다.
3	주어진 데이터의 병합, 추출, 정렬 등의 핸들링 기법에 대해 학습한다.

핵심 단어

- 데이터 입력 및 출력
- 데이터 프레임
- 데이터 병합, 추출, 정렬

01강. Data handling with Python

01. Python 프로그램

1) Python이란

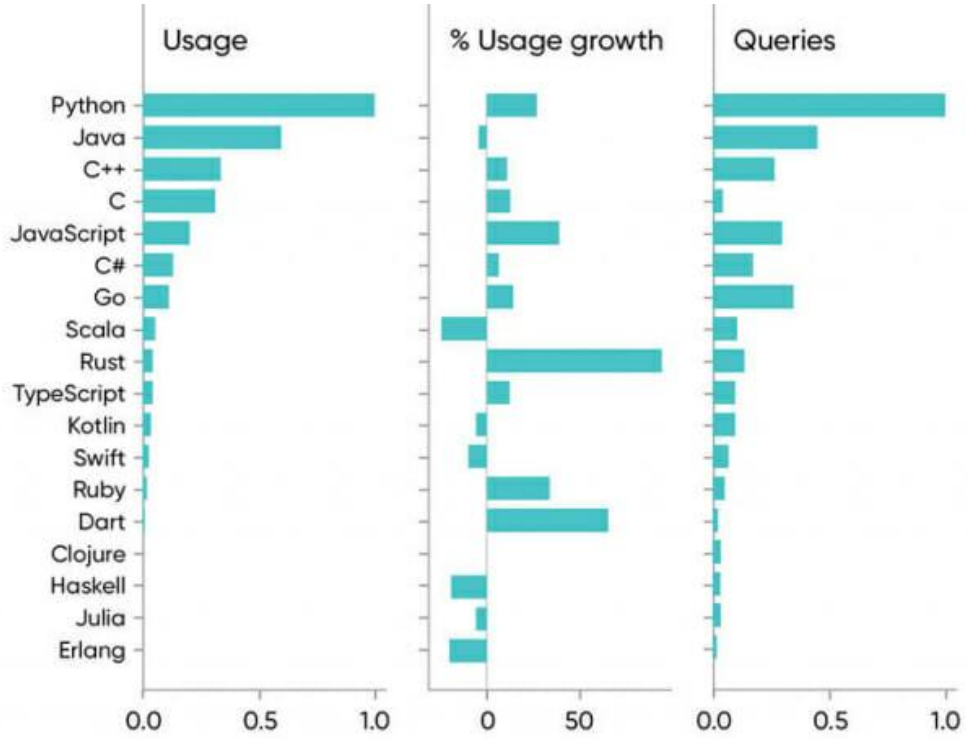
Python 프로그래밍

- 네덜란드 프로그래머인 귀도 반 로섬이 발표한 고급 프로그래밍 언어.
- 비영리의 파이썬 소프트웨어 재단이 관리하는 개방형, 공동체 기반 개발 모델. (오픈 소스)
- 윈도우, 리눅스, Mac OS X 등 다양한 시스템에서 사용 가능.

1) Python이란

Python 프로그래밍의 높은 인기

- 가장 많은 프로그램 사용자들 사용.
- 인공지능(AI) 산업의 성장과 더불어 높은 사용 증가율.



오라일리 미디어에서 공개한 프로그래밍 언어 사용률 그래프(이미지=오라일리 미디어)

출처: https://zdnet.co.kr/view/?no=20210126152435

2) Python의 장단점

장점

- 쉬운 문법, 높은 가독성, 풍부한 라이브러리 보유, 다양한 플랫폼에서 사용 가능, 메모리 자동 관리 등

단점

- 빠른 속도로 처리 불가 (C > JAVA > Python, R,…)
- 하드웨어를 직접 건드려야 하는 일에는 부적합

3) Python 개발활경

파이썬 개발환경 만들기

- Anaconda 홈페이지에서 다운로드 가능 (https://www.anaconda.com/products/distribution)
- 구글 Colab을 통해 설치 없이 파이썬 프로그램을 사용할 수 있음.

4) Python 패키지

파이썬 패키지 설치하기

- 파이썬 패키지는 cmd창에서 “pip install 패키지명” 의 명령어를 입력하여 설치할 수 있음.
- 파이썬 프로그램을 실행 후 “import 패키지명”을 입력할 때 오류가 나지 않으면 해당 패키지가 잘 설치된 것.

4) Python 패키지

파이썬 패키지 불러오기

- **import “package name”: 패키지 불러옴**
  - Ex: import numpy
- **import “package name” as “abbr.”: 축약된 이름을 사용**
  - Ex: import numpy as np
- **from: 하부 모듈을 불러오고 싶을 때 사용**
  - Ex: from sklearn.neighbors import KNeighborsClassifier

01강. Data handling with Python

02. 데이터 입출력하기

1) 데이터의 입력

파일 로딩 함수

- 주로 pandas 패키지의 read_csv, read_table, read_excel을 사용.
- read_csv
  - 파일 혹은 URL 등으로부터 데이터를 읽어오는 함수.
  - 데이터 구분자는 쉼표(,)를 기본으로 함.
- read_table
  - read_csv와 같은 역할.
  - 단, 데이터 구분자를 탭(₩t)으로 한다는 차이가 있음.

1) 데이터의 입력

파일 로딩 함수

- **read_excel**
  - 엑셀 파일 (.xls, .xlsx)의 데이터를 읽어오는 함수.

2) CSV 파일 불러오기

- 데이터가 존재하는 디렉토리 설정
- read_csv 또는 read_table을 이용하여 데이터 불러오기.

[1]:
```
import pandas as pd                                    # pandas 패키지 불러오기
import os                                              # os 패키지 불러오기
```

[2]:
```
os.getcwd()
```

[2]:
```
'/home/dongha0718/KNOU_Machine_Learning/chap1'
```

[3]:
```
data_path = './data'                                   # Data 경로
os.chdir(data_path)                                    # 작업 디렉토리 변경
```

[4]:
```
os.getcwd()
```

[4]:
```
'/home/dongha0718/KNOU_Machine_Learning/chap1/data'
```

2) CSV 파일 불러오기

- read_csv 또는 read_table을 이용하여 데이터 불러오기.

[5]:
```
df1 = pd.read_csv('ex1.csv')
```

[6]:
```
df1
```

[6]:

	a	b	c	d	message
0	1	2	3	4	hello
1	5	6	7	8	world
2	9	10	11	12	foo

2) CSV 파일 불러오기

- 칼럼명을 없애고 불러오기.

[9]:
```
pd.read_csv('ex2.csv', header=None)
```

[9]:

	0	1	2	3	4
0	1	2	3	4	hello
1	5	6	7	8	world
2	9	10	11	12	foo

2) CSV 파일 불러오기

- 칼럼명이 없는 데이터 불러오기.

[10]:
```
pd.read_csv('ex2.csv')
```

[10]:

	1	2	3	4	hello
0	5	6	7	8	world
1	9	10	11	12	foo

[11]:
```
pd.read_csv('ex2.csv', names=['a','b','c','d','message'])
```

칼럼명 추가할 경우

[11]:

	a	b	c	d	message
0	1	2	3	4	hello
1	5	6	7	8	world
2	9	10	11	12	foo

3) 엑셀 파일 불러오기

- **openpyxl, xlrd를 설치해야 한다.**
  - pip install openpyxl xlrd
- **첫번째 sheet에 있는 데이터 불러오기.**

[17]:
```
xls_filename = "ex1.xlsx"
pd.read_excel(xls_filename, sheet_name = "Sheet1", engine='openpyxl')
```

[17]:

	Unnamed: 0	a	b	c	d	message
0	0	1	2	3	4	hello
1	1	5	6	7	8	world
2	2	9	10	11	12	foo

4) 결측값 다루기

- 특정 값에 대해서 결측값으로 처리하기.

[18]:
```
result = pd.read_csv('ex3.csv')
result
```

[18]:

	something	a	b	c	d	message
0	one	1	2	3.0	4	NaN
1	two	5	6	NaN	8	world
2	three	9	10	11.0	12	foo

원본 데이터

[20]:
```
result = pd.read_csv('ex3.csv', na_values=4)
result
```

[20]:

	something	a	b	c	d	message
0	one	1	2	3.0	NaN	NaN
1	two	5	6	NaN	8.0	world
2	three	9	10	11.0	12.0	foo

결측치 처리 후

4) 결측값 다루기

- 특정 칼럼을 선택하여 결측치 처리를 할 수 있다.

[18]:
```
sentinels = {'message':['foo', 'NA'], 'something':['two']}
pd.read_csv('ex3.csv', na_values=sentinels)
```

[18]:

	something	a	b	c	d	message
0	one	1	2	3.0	4	NaN
1	NaN	5	6	NaN	8	world
2	three	9	10	11.0	12	NaN

5) 데이터 출력

파일 출력 함수

- **to 함수**
  - 주어진 데이터를 원하는 형태, 원하는 이름의 파일로 내보내는 함수.

6) CSV 파일 내보내기

- to 함수를 이용하여 파일 저장하기.

[31]:
```
data = pd.read_csv('ex1.csv')
data.to_csv('out.csv')
```

[33]:
```
data
```

[33]:

	a	b	c	d	message
0	1	2	3	4	hello
1	5	6	7	8	world
2	9	10	11	12	foo

[32]:
```
pd.read_csv('out.csv')
```

[32]:

	Unnamed: 0	a	b	c	d	message
0	0	1	2	3	4	hello
1	1	5	6	7	8	world
2	2	9	10	11	12	foo

6) CSV 파일 내보내기

- Index 없이 저장하고 싶으면 index=False 옵션을 추가하면 된다.

[27]:
```
data.to_csv('out2.csv', index=False)

pd.read_csv('out2.csv')
```

[27]:

	a	b	c	d	message
0	1	2	3	4	hello
1	5	6	7	8	world
2	9	10	11	12	foo

6) CSV 파일 내보내기

- 칼럼의 일부분만을 저장하거나 순서를 직접 지정할 수도 있다.

[30]:
```
data.to_csv('out3.csv',index=False, columns=['b','a','c'])

pd.read_csv('out3.csv')
```

[30]:

	b	a	c
0	2	1	3
1	6	5	7
2	10	9	11

01강. Data handling with Python

03. 데이터 핸들링하기

1) Pandas의 데이터 구조

Series

- 모든 데이터 유형 (정수, 문자 등)을 저장할 수 있는 1차원 배열.

DataFrame

- 잠재적으로 다른 유형의 열이 있는 2차원 데이터 구조.
- 스프레드 시트나 SQL 테이블 등으로 생각할 수 있음.
- R의 data.frame()과 같은 기능.

2) Pandas Series

- Series 생성하기.

[2]:
```
import pandas as pd
obj = pd.Series([4,7,-5,3])
obj
```

[2]:
```
0    4
1    7
2   -5
3    3
dtype: int64
```

[3]:
```
obj.values
```

[3]:
```
array([ 4,  7, -5,  3])
```

2) Pandas Series

- Index를 지정할 수도 있다.

[4]:
```
obj2 = pd.Series([4,7,-5,3], index=['d', 'b', 'a', 'c'])
obj2
```

[4]:
```
d    4
b    7
a   -5
c    3
dtype: int64
```

3) Pandas DataFrame

- 3개의 DataFrame (user1~3)을 생성해보자.
- 칼럼의 이름을 다양한 방식으로 입력할 수 있다.

[24]:
```
columns = ['name', 'age', 'gender', 'job']

user1 = pd.DataFrame([['alice', 19, "F", "student"],
                      ['john', 26, "M", "student"]],
                     columns=columns)
user1
```

[24]:

	name	age	gender	job
0	alice	19	F	student
1	john	26	M	student

3) Pandas DataFrame

[43]:
```
user2 = pd.DataFrame([['eric', 22, "M", "student"],
                      ['paul', 58, "F", "manager"]],
                     columns=columns)
user2
```

[43]:

	name	age	gender	job
0	eric	22	M	student
1	paul	58	F	manager

[25]:
```
user3 = pd.DataFrame(dict(name=['peter', 'julie'],
                          age=[33, 44],
                          gender=['M', 'F'],
                          job=['engineer', 'scientist']))
user3
```

[25]:

	name	age	gender	job
0	peter	33	M	engineer
1	julie	44	F	scientist

4) 데이터 합치기: combine

- Pandas의 append와 concat 함수를 이용하여 데이터를 합칠 수 있다.

[8]:
```
# Combining DataFrames
user1.append(user2)
```

[8]:

	name	age	gender	job
0	alice	19	F	student
1	john	26	M	student
0	eric	22	M	student
1	paul	58	F	manager

[16]:
```
users = pd.concat([user1, user2, user3])
users
```

[16]:

	name	age	gender	job
0	alice	19	F	student
1	john	26	M	student
0	eric	22	M	student
1	paul	58	F	manager
0	peter	33	M	engineer
1	julie	44	F	scientist

4) 데이터 합치기: combine

- **Numpy의 concatenate, vstack, hstack 등을 이용해서도 데이터를 합칠 수 있다.**
  - vstack: 행 합치기, hstack: 열 합치기

```
[30]: np.vstack([arr,arr])
```

```
[30]: array([[ 0,  1,  2,  3],
             [ 4,  5,  6,  7],
             [ 8,  9, 10, 11],
             [ 0,  1,  2,  3],
             [ 4,  5,  6,  7],
             [ 8,  9, 10, 11]])
```

```
[31]: np.hstack([arr,arr])
```

```
[31]: array([[ 0,  1,  2,  3,  0,  1,  2,  3],
             [ 4,  5,  6,  7,  4,  5,  6,  7],
             [ 8,  9, 10, 11,  8,  9, 10, 11]])
```

4) 데이터 합치기: combine

- **Numpy의 concatenate, vstack, hstack 등을 이용해서도 데이터를 합칠 수 있다.**
  - concatenate: vstack과 hstack 모두를 아우르는 함수 (axis 옵션 사용)

```
[28]: np.concatenate([arr, arr], axis=1)
```

```
[28]: array([[ 0,  1,  2,  3,  0,  1,  2,  3],
             [ 4,  5,  6,  7,  4,  5,  6,  7],
             [ 8,  9, 10, 11,  8,  9, 10, 11]])
```

5) 데이터 합치기: merge

- merge 함수는 기본적으로 내부조인 (inner join)을 수행하여 교집합인 결과를 반환한다.
- 조인할 때의 key값은 on 옵션을 통해 설정할 수 있다.

Users

	name	age	gender	job
0	alice	19	F	student
1	john	26	M	student
0	eric	22	M	student
1	paul	58	F	manager
0	peter	33	M	engineer
1	julie	44	F	scientist

User4

	name	height
0	alice	165
1	john	180
2	eric	175
3	julie	171

5) 데이터 합치기: merge

- 조인할 때의 key값은 on 옵션을 통해 설정할 수 있다.

[19]:
```
# Use union of keys from both frames
users2 = pd.merge(users, user4, on="name")
users2
```

[19]:

	name	age	gender	job	height
0	alice	19	F	student	165
1	john	26	M	student	180
2	eric	22	M	student	175
3	julie	44	F	scientist	171

5) 데이터 합치기: merge

- how 옵션에 'left', 'right', 'outer' 등을 입력하여 각각 왼쪽 우선 외부조인, 오른쪽 우선 외부조인, 완전 외부조인 등을 수행할 수 있다.

[20]:
```
users3 = pd.merge(users, user4, on="name", how='outer')
users3
```

[20]:

	name	age	gender	job	height
0	alice	19	F	student	165.0
1	john	26	M	student	180.0
2	eric	22	M	student	175.0
3	paul	58	F	manager	NaN
4	peter	33	M	engineer	NaN
5	julie	44	F	scientist	171.0

5) 데이터 합치기: merge

- merge하려는 두 개의 데이터에서 key값의 칼럼명이 다른 경우 left_on, right_on을 이용할 수 있다.

[21]:
```
pd.merge(users, user4, left_on = 'name', right_on = 'name')
```

[21]:

	name	age	gender	job	height
0	alice	19	F	student	165
1	john	26	M	student	180
2	eric	22	M	student	175
3	julie	44	F	scientist	171

6) 칼럼 및 로우 선택하기

- 주어진 data frame에서 원하는 칼럼 및 로우를 선택할 수 있다.
- 칼럼 선택하기

Users

	name	age	gender	job
0	alice	19	F	student
1	john	26	M	student
0	eric	22	M	student
1	paul	58	F	manager
0	peter	33	M	engineer
1	julie	44	F	scientist

```
[32]: users['gender'] # select one column

[32]: 0    F
      1    M
      0    M
      1    F
      0    M
      1    F
      Name: gender, dtype: object
```

6) 칼럼 및 로우 선택하기

- 두 개 이상의 칼럼을 선택할 수도 있다.

[34]:
```
# select multiple columns
users[['age', 'gender']] # select two columns
```

[34]:

	age	gender
0	19	F
1	26	M
0	22	M
1	58	F
0	33	M
1	44	F

6) 칼럼 및 로우 선택하기

- iloc와 loc를 이용해서도 칼럼을 선택할 수 있다.

```
[66]: users.iloc[:,1:3]
```

[66]:

	age	gender
0	19	F
1	26	M
0	22	M
1	58	F
0	33	M
1	44	F

```
[67]: users.loc[:,['age','gender']]
```

[67]:

	age	gender
0	19	F
1	26	M
0	22	M
1	58	F
0	33	M
1	44	F

6) 칼럼 및 로우 선택하기

- iloc 및 loc를 이용하여 로우를 선택할 수도 있으며, 원하는 원소만 뽑을 수도 있다.

[57]:
```
df = users.copy()
df.iloc[0] # first row
```

[57]:
```
name        alice
age            19
gender          F
job       student
Name: 0, dtype: object
```

[47]:
```
df.iloc[2:4]
```

[47]:

	name	age	gender	job
0	eric	22	M	student
1	paul	58	F	manager

[86]:
```
print(df.iloc[0, 0]) # first item of first row
```

```
alice
```

[56]:
```
df.loc[0]
```

[56]:

	name	age	gender	job	1
0	alice	55	F	student	55.0
0	eric	22	M	student	55.0
0	peter	33	M	engineer	55.0

[55]:
```
print(df.loc[0, "age"])
```

```
0    55
0    22
0    33
Name: age, dtype: int64
```

7) 필터링하기

- 논리연산자를 이용하여 특정 조건을 만족하는 데이터를 뽑을 수도 있다.

[35]:
```
# simple logical filtering
users[users.age < 20]       # only show users with age < 20
```

[35]:

	name	age	gender	job
0	alice	19	F	student

[37]:
```
# Advanced logical filtering
users[users.age < 20][['age', 'job']] # select multiple columns
```

[37]:

	age	job
0	19	student

7) 필터링하기

- 두 개 이상의 논리연산자를 사용할 수도 있다.

[38]:
```
users[(users.age > 20) & (users.gender == 'M')] # use multiple conditions
```

[38]:

	name	age	gender	job
1	john	26	M	student
0	eric	22	M	student
0	peter	33	M	engineer

7) 필터링하기

- isin 명령어를 사용하여 원하는 조건에 부합하는 자료를 추출할 수 있다.

[39]:
```
users[users.job.isin(['student', 'engineer'])] # filter specific values
```

[39]:

	name	age	gender	job
0	alice	19	F	student
1	john	26	M	student
0	eric	22	M	student
0	peter	33	M	engineer

8) 정렬하기

- **특정 칼럼을 기준으로 오름차순 또는 내림차순으로 자료를 재정렬할 수 있다.**
  - 칼럼 선택은 by 옵션을, 내림차순 여부는 ascending 옵션을 사용.

[68]:
```
# sort rows by a specific column
df = users.copy()
df.sort_values(by='age')
```

[68]:

	name	age	gender	job
0	alice	19	F	student
0	eric	22	M	student
1	john	26	M	student
0	peter	33	M	engineer
1	julie	44	F	scientist
1	paul	58	F	manager

[69]:
```
# use descending order instead
df.sort_values(by='age', ascending=False)
```

[69]:

	name	age	gender	job
1	paul	58	F	manager
1	julie	44	F	scientist
0	peter	33	M	engineer
1	john	26	M	student
0	eric	22	M	student
0	alice	19	F	student

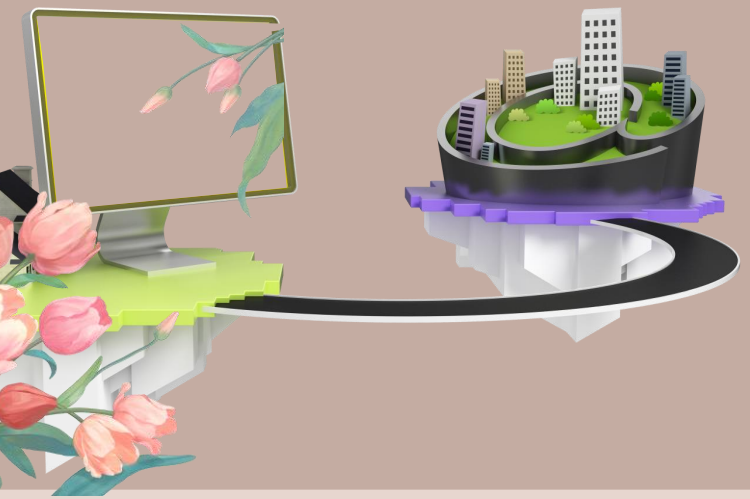
8) 정렬하기

- 두 개 이상의 칼럼을 동시에 고려하여 정렬할 수도 있다.

[42]:
```
df.sort_values(by=['job', 'age']) # sort by multiple columns
```

[42]:

	name	age	gender	job
0	peter	33	M	engineer
1	paul	58	F	manager
1	julie	44	F	scientist
0	alice	19	F	student
0	eric	22	M	student
1	john	26	M	student



다음시간안내

제02강

Basic Methods for Regression 1.