

파이썬을 이용한 데이터수집 및 스마트공장 견학

Python의 기초 II

2021년 1월 12일
안재관

금일 목표

- 예외처리, File I/O, 모듈 호출에 대해서 배운다.
- 판다스와 데이터 프레임에 대해 배운다.

예외 처리

● try-except 구조

구체적인 내용을 as로 활용하여 변수로 받을 수도 있음

```
try:
    myNum = int('two')
    print(myNum)
except ValueError:
    print('convert error')
```

convert error

```
try:
    myNum = int('two')
    print(myNum)
except ValueError as e:
    print(e)
```

invalid literal for int() with base 10: 'two'

예외가 발생해도 별다른 처리 없이 회피해야 할 경우 pass 사용

● 예외 발생 여부에 관계 없이 항상 실행되게 할 때는 finally문 사용

```
try:
    myNum = int('two')
    print(myNum)
except ValueError as e:
    print(e)
finally:
    print("finally")
```

invalid literal for int() with base 10: 'two'
finally

● except문을 여러 번 사용 가능하기도, 하나의 except로 받을 수도 있음

```
line = None
try:
    with open('a.txt', 'r') as f:
        line = f.readline()
        myNum = int(line)
        myResult = 200 / myNum
except (IOError, ValueError) as e:
    print('IOError or ValueError', e)
except ZeroDivisionError as e:
    print('ZeroDivisionError', e)
```

IOError or ValueError [Errno 2] No such file or directory: 'a.txt'

File I/O

● open(filename, mode, encoding)

txt, csv와 같은 텍스트 형태의 파일 입출력은 open() 함수 사용

Filename: 상대경로 혹은 절대경로로 표기 가능

Mode: 파일에 대한 권한 부여

- 'r' - 읽기, 'w' - 쓰기, 'a' - 이어쓰기

Encoding: 따로 명시하지 않아도 기본적인 처리 가능

- 알파벳, 숫자, 기본 문장 기호
- 다른 문자 포함시 'utf-8'

● 읽기

read(): 모든 문자를 하나의 문자열로 반환

readlines(): 줄넘김으로 구분되어 있는 문장의 리스트 반환

readline(): 줄넘김으로 구분되어 있는 문장 하나 반환
반복해서 호출 시 다음 문장

한번 읽은 부분은 읽은 이후에 다시 읽을 수 없음

```
f = open("Intro.txt", "r")
f.read()
'Seoul National University\nDepartment of Engineering\nIndustrial Engineering\nInformation Management Lab\nSeongeun Lee'

f.readlines()
[]

f.close()

f = open("Intro.txt", "r")
f.readlines()
['Seoul National University\n',
'Department of Engineering\n',
'Industrial Engineering\n',
'Information Management Lab\n',
'Seongeun Lee']

f.read()
''

f.close()
```

Image 관련

- Image 관련 task를 위한 이미지를 불러올 때, 보통 OpenCV와 PIL 두 라이브러리 중 하나를 이용

OpenCV는 “pip install opencv-python”으로 설치하며,
PIL은 “pip install pillow”를 이용

- OpenCV

- BGR로 이미지 읽어옴
- Video capture와 같은 기능 지원이 잘 되어있음
- PIL에는 없는 다양한 함수들 지원
- numpy array와의 호환성이 좋음. numpy array 인덱싱을 이용해 이미지에 대한 전처리가 가능하기에 좀 더 자유롭게 이미지 전처리 가능
- 반대로 torchvision과의 호환성이 안좋음, 그렇지만 opencv와 호환되는 torchvision 버전도 존재.

- PIL

- RGB로 이미지 읽어옴
- torchvision에서의 지원이 잘 되어있음
- numpy array와의 암묵적 casting이 안되고, numpy array를 Image array로 바꿔줘야함(은근 귀찮음)

```
1 from PIL import Image
2 import numpy as np
3 import cv2
4 from matplotlib import pyplot as plt
```

```
1 #path=r'C:\Users\User\Desktop\test_model.jpg'
2 path='./sample.jpg'
3 image = cv2.imread(path)
4 # print(img_test)
5 image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
6 fig=plt.imshow(image)
7 fig.axes.get_xaxis().set_visible(False)
8 fig.axes.get_yaxis().set_visible(False)
9 print(image.shape)
```

(639, 957, 3)



```
1 im = Image.fromarray(image)
2 im.save("sample_altered.jpeg")
```

Module 설치 및 호출

- 함수나 클래스를 모아둔 .py 파일로 다른 프로그램에서 사용 가능하게 함
Package는 모듈을 모아둔 디렉토리
- 설치
보통 cmd 혹은 terminal에 `pip install [package name]` 혹은
`conda install [package name]` 으로 진행
- 불러오기
`import 모듈명 as 새모듈명`
`from 모듈명 import 컴포넌트 as alias`
`from 패키지 import 모듈명`
- 다른 사람들이 만들어 놓은 package를 설치해서 사용 가능하기도 하며,
본인이 module을 정의해서 사용 역시 가능
보통 `[package name] + pypl` 로 구글 검색
Github에서 `requirement.txt`로 한번에 가져올 수 있음

Pandas: Series & DataFrame

- Pandas는 고수준의 자료 구조 라이브러리
- 이 역시 매우 많은 기능을 제공하므로 documentation 참고 추천
 - <https://pandas.pydata.org/docs/>
- 장점
 - 시계열 데이터 저장에 적합
 - RDBMS와 유사하게 사용 가능
 - SQL의 join과 같은 기능 제공 (merge)
 - SQL의 집계 함수와 같은 기능 제공 (agg, groupby)
 - 메타데이터
 - Missing data 처리 용이
 - 행렬 데이터의 축에 따른 분석 기능
 - Numpy와 함께 사용하는 것이 편리함
- 가상환경 실행 후 pip install pandas 를 통해 설치
- 주로 import pandas as pd로 pd라는 별칭 사용

Pandas: Series & DataFrame

● Pandas의 자료 구조

- Series(시계열 데이터), Dataframe(RDBMS와 유사)
- Index라는 구분자 존재
- Dataframe: column name이 존재
- Numpy function을 접목시킬 수 있으며, Numpy 변수를 Dataframe이나 Series로 변화 가능

```
obj = pd.Series([4,7,-5,3])  
print(obj)
```

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

```
obj.values
```

```
array([ 4,  7, -5,  3], dtype=int64)
```

```
obj.index
```

```
RangeIndex(start=0, stop=4, step=1)
```

```
data = {  
    'state': ['Ohio', 'Ohio', 'Ohio', 'Nevada', 'Nevada'],  
    'year': [2000, 2001, 2002, 2001, 2002],  
    'pop': [1.5, 1.7, 3.6, 2.4, 2.9]  
}  
frame = pd.DataFrame(data)  
print(frame)
```

```
   state  year  pop  
0   Ohio  2000  1.5  
1   Ohio  2001  1.7  
2   Ohio  2002  3.6  
3  Nevada  2001  2.4  
4  Nevada  2002  2.9
```

```
frame.index = ['one', 'two', 'three', 'four', 'five']  
print(frame)
```

```
   state  year  pop  
one   Ohio  2000  1.5  
two   Ohio  2001  1.7  
three Ohio  2002  3.6  
four  Nevada  2001  2.4  
five  Nevada  2002  2.9
```


Pandas: Series & DataFrame

● 현재 로드되어 있는 데이터를 Dataframe으로 바꾸기

- 2차원 ndarray, 어떤 객체의 사전 혹은 리스트
- Index나 column name을 주지 않을 경우 자동으로 생성

● 주로 file이나 db를 통해 생성

- pd.read_csv: text file을 읽어서 Dataframe을 생성하는 함수
- pd.read_sql: sql문을 사용해서 Dataframe을 생성하는 함수

● 색인

- Index를 활용한 데이터 색인(행)
 - frame.loc: index의 이름으로 색인
 - frame.iloc: index의 순서로 색인
- Column을 이용한 색인
 - frame[list of column names]
- Numpy와 마찬가지로 boolean으로 색인 가능

```
frame.loc[['one', 'two']]
```

	state	year	pop
one	Ohio	2000	1.5
two	Ohio	2001	1.7

```
frame.iloc[0:2]
```

	state	year	pop
one	Ohio	2000	1.5
two	Ohio	2001	1.7

```
frame[['year', 'state']]
```

	year	state
one	2000	Ohio
two	2001	Ohio
three	2002	Ohio
four	2001	Nevada
five	2002	Nevada

Pandas: Series & DataFrame. DataFrame 합치기

● Merge

- RDBMS의 join과 유사
- `pd.merge(df1, df2, on, how)`
- `how`는 'inner', 'left', 'right', 'outer' 중 하나

```
left = pd.DataFrame({'key': ['K0', 'K1', 'K2', 'K3'],
                    'A': ['A0', 'A1', 'A2', 'A3'],
                    'B': ['B0', 'B1', 'B2', 'B3']})

right = pd.DataFrame({'key': ['K0', 'K1', 'K2', 'K3'],
                    'C': ['C0', 'C1', 'C2', 'C3'],
                    'D': ['D0', 'D1', 'D2', 'D3']})

pd.merge(left, right, on='key', how='left')
```

	key	A	B	C	D
0	K0	A0	B0	C0	D0
1	K1	A1	B1	C1	D1
2	K2	A2	B2	C2	D2
3	K3	A3	B3	C3	D3

● Concat

- 하나의 축을 따라 여러 개의 DataFrame을 이어 붙이기
- `pd.concat([dataframes])`

```
df1 = pd.DataFrame({'A': ['A0', 'A1', 'A2', 'A3'],
                    'B': ['B0', 'B1', 'B2', 'B3'],
                    'C': ['C0', 'C1', 'C2', 'C3'],
                    'D': ['D0', 'D1', 'D2', 'D3'],
                    index=[0, 1, 2, 3])

df2 = pd.DataFrame({'A': ['A4', 'A5', 'A6', 'A7'],
                    'B': ['B4', 'B5', 'B6', 'B7'],
                    'C': ['C4', 'C5', 'C6', 'C7'],
                    'D': ['D4', 'D5', 'D6', 'D7'],
                    index=[4, 5, 6, 7])

pd.concat([df1, df2])
```

	A	B	C	D
0	A0	B0	C0	D0
1	A1	B1	C1	D1
2	A2	B2	C2	D2
3	A3	B3	C3	D3
4	A4	B4	C4	D4
5	A5	B5	C5	D5
6	A6	B6	C6	D6
7	A7	B7	C7	D7

Pandas: Series & DataFrame. 데이터 불러오기/내보내기

● Pd.read_csv()/pd.read_excel()

- '경로/filename.csv'
- index_col = '인덱스로 지정할 column명'
- header : 열 이름(헤더)으로 사용할 행 지정
첫 행이 헤더가 아닌 경우 *header = None*

```
1 import pandas as pd
2 df=pd.read_csv('./titanic.csv')
```

● To_csv()/to_excel()

- 데이터프레임명.to_csv('경로/저장할 파일명.csv')
- index = False : 인덱스 안 나타나게

```
1 df1.to_excel('./titanic_extracted.xlsx', index = False)
```

● Tip:

- 1) ('./Practice/socre.csv')라 쓰여 있는데, .(dot)은 상대 경로를 사용을 의미.
- 2) 상대경로를 이용할 경우, 현재 작업중인 폴더 기준, .(dot)의 의미는 현재 작업중인 폴더 위치를 말함.
- 3) ..(dotdot)은 참고로 현재 위치에서 한 단계 상위 폴더를 지칭
- 4) 절대경로 사용 시, r'절대경로' 사용

Jupyter Notebook에서의 DataFrame

- 아래와 같이 깔끔하게 표시하기 때문에 자료 capture에 용이함.

```
1 import seaborn as sns
2
3 titanic_2 = sns.load_dataset("titanic")
```

```
1 titanic_2
```

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	deck	embark_town	alive	alone
0	0	3	male	22.0	1	0	7.2500	S	Third	man	True	NaN	Southampton	no	False
1	1	1	female	38.0	1	0	71.2833	C	First	woman	False	C	Cherbourg	yes	False
2	1	3	female	26.0	0	0	7.9250	S	Third	woman	False	NaN	Southampton	yes	True
3	1	1	female	35.0	1	0	53.1000	S	First	woman	False	C	Southampton	yes	False
4	0	3	male	35.0	0	0	8.0500	S	Third	man	True	NaN	Southampton	no	True
...
886	0	2	male	27.0	0	0	13.0000	S	Second	man	True	NaN	Southampton	no	True
887	1	1	female	19.0	0	0	30.0000	S	First	woman	False	B	Southampton	yes	True
888	0	3	female	NaN	1	2	23.4500	S	Third	woman	False	NaN	Southampton	no	False
889	1	1	male	26.0	0	0	30.0000	C	First	man	True	C	Cherbourg	yes	True
890	0	3	male	32.0	0	0	7.7500	Q	Third	man	True	NaN	Queenstown	no	True

891 rows × 15 columns