Essential Python for Data Analyst

Contents

- review class (OOP)
- review loop + dictionary
- try except
- assert
- csv. json, context manager
- web scraping
- numpy pandas sqlite sklearn

```
# super classs
class Person:
    def __init__(self, name):
        self.name = name
    def hey (self):
        print("hello world!")
    def sayhi(self):
        print(f"Hi my name is {self.name}!")

class Employee(Person):
    def __init__(self, name, company):
        super(). __init__(name)
        self.company = company
    def potato(self):
        print("Potato is very good!")
    def sayhi(self):
        print(f"Hi my name is {self.name} and I work at {self.company}.")

```

```
plume = Person("plume")
john = Employee("John", "Google")
plume.sayhi()
john.sayhi()
```

```
Hi my name is plume!
Hi my name is John and I work at Google.
```

```
plume.hey()
```

```
hello world!
```

```
john.potato()
```

```
Potato is very good!
```

```
#try-except block
try:
    1/0
except :
    print("this is an error message")
```

```
this is an error message
```

```
try:
    print("hello")
except ZeroDivisionError:
    print("Cannot divide a number with zero")
except NameError:
    print("Variable is not found")
except ValueError:
    print("Please check your data type again!")
else:
    print("Done!")
finally :
    print("This is the last line")
```

```
hello
Done!
This is the last line
```

```python
assert 1+1 ==5, "1+1 must be equal 2"
```

```
---------------------------------------------------------------------------
AssertionError                            Traceback (most recent call last)
<ipython-input-31-044f29da9dba> in <cell line: 1>()
----> 1 assert 1+1 ==5, "1+1 must be equal 2"

AssertionError: 1+1 must be equal 2
```

SEARCH STACK OVERFLOW

```python
#type hint
def check_this(a: int, b: int) -> int:
    assert a+b > 10, "a+b should be more than 10"
    return a+b
```

```python
check_this(10,4)
```

```
14
```

```python
# lamda function(เหมาะกับ simple function)
```

```python
def greeting(name):
    print("hello! "+ name)
```

```python
greeting = lambda name: print("hello! "+ name)
```

```python
greeting("pasit")
```

```
hello! pasit
```

```python
# แบบมาตรฐาน
def odd_or_even(num):
    if num% 2 == 0:
        print("Even")
    else:
        print("Odd")
```

```python
# แบบสั้น (เหมาะกับ simple function)
odd_or_even = lambda num: "even" if num %2 == 0 else "odd"
```

```python
odd_or_even(4)
```

```
'even'
```

```python
# review class
# try except
# assert 1+1==2, "error message"
# lambda function
```

```python
hello = lambda : print("hello!")
```

```python
hello()
```

```
hello!
```

```python
# list of colored balls
balls = ["red", "red", "greed", "blue", "blue"]
```

```python
def summary(lst_items):
    result = {}
    for item in lst_items :
        if item in result:
            result[item] += 1
        else:
            result[item] = 1
    return(result)

```

```
summary(balls)
```

```
{'red': 2, 'greed': 1, 'blue': 2}
```

```
# comma separated values (CSV file)
import csv

try:
    file = open("hotel.csv")
    data = csv.reader(file)
    file.close()
except FileNotFoundError:
    print("File not found.")
else:
    print("Load data successfully.")
```

```
Load data successfully.
```

```
# context manager
import csv

try:
    with open ("hotel.csv", "r") as file:
        data = csv.reader(file)
        for row in data:
            print(row)
except:
    print("Found some errors, please check again.")
```

```
['id', 'hotel ', 'location', 'price']
['1', 'Accor', 'Thailand', '50']
['2', 'Ibis', 'Italy', '60']
['3', 'London', 'UK', '35']
['4', 'Seoul', 'Korea', '30']
['5', 'le concord', 'USA', '25']
```

```
csv_data = []

with open ("hotel.csv", "r") as file:
    data = csv.reader(file)
    for row in data:
        csv_data.append(row)

print(csv_data)
```

```
[['id', 'hotel ', 'location', 'price'], ['1', 'Accor', 'Thailand', '50'], ['2', 'Ibis', 'Italy', '60'], ['3', 'London', 'UK', '35'], ['4
```

```
import pandas as pd
df= pd.read_csv("hotel.csv")

df["price_double"] = df["price"]*2

df.to_csv("hotel_modifiled.csv")
```

```
# write csv files
# "w" write mode
import csv

headers = ["id", "name", "age"]
data = [
    [1, "pasit", 27],
    [2, "jisoo", 26],
    [3, "john", 32]
]


with open("newfile.csv", "w") as file:
    writer = csv.writer(file)
    writer.writerow(headers)
    writer.writerows(data)
```

```
!cat newfile.csv
```

```
id,name,age
1,pasit,27
2,jisoo,26
3,john,32
```

```
result=[]
with open("newfile.csv", "r") as file:
    data = csv.reader(file)
    for row in data:
        result.append(row)
print(result)
```

```
[['id', 'name', 'age'], ['1', 'pasit', '27'], ['2', 'jisoo', '26'], ['3', 'john', '32']]
```

```
#regular expression

import re

#from re import search

text = "a tiger walks into bar"

result = re.search("duck", text)

if result:
    print("Duck Found")
else:
    print("Duck not found")
```

```
Duck not found
```

```
# list comprehension
numbers = [1, 2, 3, 4, 5]
```

```
result = [num*2 for num in numbers]
print(result)
```

```
[2, 4, 6, 8, 10]
```

```
oddNums = [num for num in numbers if num%2 == 1]
evenNums = [num for num in numbers if num%2 == 0]

print(oddNums, evenNums)
```

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

```
# json (JavaScript Object Notation)
import json

with open("pasit.json", "r") as file:
    data = json.load(file)
```

```
print(data, type(data))
```

```
{'name': 'Pasit', 'age': 27, 'movie_lover': True, 'movies': ['batman', 'superman']} <class 'dict'>
```

```
data["name"]
```

```
'Pasit'
```

```
# create new .jason file
school = {
    "name" : "bootcamp",
    "batch": 7,
    "duration": "4 months"
}

with open("newjsonfile.json", "w") as file:
    json.dump(school, file)
```

```
!cat newjsonfile.json
```

```
{"name": "bootcamp", "batch": 7, "duration": "4 months"}
```

```
with open("newjsonfile.json", "r") as file:
    data2 = json.load(file)

data2
```

```
{'name': 'bootcamp', 'batch': 7, 'duration': '4 months'}
```

API

```
# API
# HW01 - find public API
from requests import get

nums = list(range(1, 6))
result = []

for i in nums:
    url= f"https://swapi.dev/api/people/{i}"
    response = get(url)
    data = response.json()
    row = [
        data["name"],
        data["height"],
        data["mass"],
        data["gender"]
    ]
    result.append(row)
    print(result)

```

```
[['Luke Skywalker', '172', '77', 'male']]
[['Luke Skywalker', '172', '77', 'male'], ['C-3PO', '167', '75', 'n/a']]
[['Luke Skywalker', '172', '77', 'male'], ['C-3PO', '167', '75', 'n/a'], ['R2-D2', '96', '32', 'n/a']]
[['Luke Skywalker', '172', '77', 'male'], ['C-3PO', '167', '75', 'n/a'], ['R2-D2', '96', '32', 'n/a'], ['Darth Vader', '202', '136', 'ma
[['Luke Skywalker', '172', '77', 'male'], ['C-3PO', '167', '75', 'n/a'], ['R2-D2', '96', '32', 'n/a'], ['Darth Vader', '202', '136', 'ma
```

```
import pandas as pd

df = pd.DataFrame(result, columns=["name", "height", "mass", "gender"])

df.to_csv("starwarapi.csv")
```

Web scraping

```
#web scraping
#gazpacho

!pip install gazpacho
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Collecting gazpacho
  Downloading gazpacho-1.1.tar.gz (7.9 kB)
  Installing build dependencies ... done
  Getting requirements to build wheel ... done
  Preparing metadata (pyproject.toml) ... done
Building wheels for collected packages: gazpacho
  Building wheel for gazpacho (pyproject.toml) ... done
  Created wheel for gazpacho: filename=gazpacho-1.1-py3-none-any.whl size=7480 sha256=cae997e716e59e0a6c4f9eba3fccc6afd4f52f7df9f8bfe2b3
  Stored in directory: /root/.cache/pip/wheels/7b/82/22/0f73f5a5fa5abb29aac7a06941ff561e0bb415e558bb64d467
Successfully built gazpacho
Installing collected packages: gazpacho
Successfully installed gazpacho-1.1
```

```python
from gazpacho import Soup
from requests import get

url = "https://www.imdb.com/search/title/?groups=top_100&sort=user_rating%2Cdesc"

response = get(url)

imdb = Soup(response.text)
```

```python
imdb.find("h3")[9].strip()
```

```
'10. The Lord of the Rings: The Fellowship of the Ring (2001)'
```

```python
title_list = imdb.find("h3", {"class": "lister-item-header"})
```

```python
titles = [title.strip() for title in title_list]
print(titles)
```

```
['1. The Shawshank Redemption (1994)', '2. The Godfather (1972)', '3. The Dark Knight (2008)', "4. Schindler's List (1993)", '5. The Lor
```

```python
runtimes = imdb.find("span", {"class": "runtime"})
runtimes = [runtime.strip() for runtime in runtimes]
print(runtimes)
```

```
['142 min', '175 min', '152 min', '195 min', '201 min', '96 min', '202 min', '154 min', '148 min', '178 min', '139 min', '142 min', '179
```

```python
import pandas as pd

df = pd.DataFrame({
    "title" : titles,
    "runtime": runtimes
})

df.head()
```

	title	runtime
0	1. The Shawshank Redemption (1994)	142 min
1	2. The Godfather (1972)	175 min
2	3. The Dark Knight (2008)	152 min
3	4. Schindler's List (1993)	195 min
4	5. The Lord of the Rings: The Return of the Ki...	201 min

```python
df.to_csv("imdb2.csv")
```

Basic Data Tools

- numpy
- pandas
- sklearn

```python
# numpy => numerical python
import numpy as np

num = [1, 2, 3, 4, 5]

num_arr = np.array(num)

print(num_arr*2, num_arr+2, num_arr-2, num_arr/5)
```

```
[ 2  4  6  8 10] [3 4 5 6 7] [-1  0  1  2  3] [0.2 0.4 0.6 0.8 1. ]
```

```
# statistical basic calculation
np.sum(num_arr)
np.mean(num_arr)
np.std(num_arr)
np.min(num_arr)
np.max(num_arr)
len(num_arr)
```

```
5
```

```
num_arr.mean()
```

```
3.0
```

```
# matrix 2D-array
m1 = np.array([
    [1, 2],
    [3,4]
])

m2 = np.array([
    [5, 5],
    [4,2]
])
```

```
print(m1*2, "\n", m2)
```

```
[[2 4]
 [6 8]]
 [[5 5]
 [4 2]]
```

```
#intro to pandas
import pandas as pd
import numpy as np
```

```
hotel = pd.read_csv("hotel.csv")
```

```
hotel.shape
```

```
(5, 4)
```

```
hotel.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5 entries, 0 to 4
Data columns (total 4 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   id        5 non-null      int64
 1   hotel     5 non-null      object
 2   location  5 non-null      object
 3   price     5 non-null      int64
dtypes: int64(2), object(2)
memory usage: 288.0+ bytes
```

```
hotel.describe()
```

	id	price
count	5.000000	5.00000
mean	3.000000	40.00000
std	1.581139	14.57738
min	1.000000	25.00000
25%	2.000000	30.00000
50%	3.000000	35.00000
75%	4.000000	50.00000
max	5.000000	60.00000

```python
hotel["price"]
```

```
0    50
1    60
2    35
3    30
4    25
Name: price, dtype: int64
```

```python
hotel.price
```

```
0    50
1    60
2    35
3    30
4    25
Name: price, dtype: int64
```

```python
hotel["vat"]= hotel["price"] *0.07

hotel
```

	id	hotel	location	price	vat
0	1	Accor	Thailand	50	3.50
1	2	Ibis	Italy	60	4.20
2	3	London	UK	35	2.45
3	4	Seoul	Korea	30	2.10
4	5	le concord	USA	25	1.75

```python
# query(filter rows)
value_hotel = hotel.query("price <= 40")
```

```python
#drop column
new_hotel = hotel.drop(["id","vat"], axis=1)
```

```python
hotel.query("price < 40 and location == 'UK' ")
```

	id	hotel	location	price	vat
2	3	London	UK	35	2.45

```python
hotel.query("price > 30 or location == 'UK' ")
```

	id	hotel	location	price	vat
0	1	Accor	Thailand	50	3.50
1	2	Ibis	Italy	60	4.20
2	3	London	UK	35	2.45

```python
#sklearn => machine learning most popular python
# template in sklearn to train_test_split
```

```python
from sklearn.linear_model import LinearRegression
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import train_test_split
import pandas as pd
import numpy as np
```

```python
mtcars = pd.read_csv("https://gist.githubusercontent.com/seankross/a412dfbd88b3db70b74b/raw/5f23f993cd87c283ce766e7ac6b329ee7cc2e1d1/mtcar
```

```python
mtcars.head()
```

	model	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
0	Mazda RX4	21.0	6	160.0	110	3.90	2.620	16.46	0	1	4	4
1	Mazda RX4 Wag	21.0	6	160.0	110	3.90	2.875	17.02	0	1	4	4
2	Datsun 710	22.8	4	108.0	93	3.85	2.320	18.61	1	1	4	1
3	Hornet 4 Drive	21.4	6	258.0	110	3.08	3.215	19.44	1	0	3	1

```
#prepare
X = mtcars[ ["hp", "wt", "am"] ]
y = mtcars["mpg"]
```

```
#split data

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42 )
```

```
#train model
#model = LinearRegression()
#model = DecisionTreeRegressor()
model = RandomForestRegressor()
model.fit(X_train, y_train)

#test model/ scoring
pred = model.predict(X_test)

# MAE mean absolute error
mae = np.mean(np.absolute((y_test - pred)))

#MSE
mse = np.mean((y_test - pred)**2)

print(mae, mse)
#evaluate model

```

```
2.0610000000000013 8.637949571428555
```