🤟 Outline Essential Python for Data Analyst

sunsun-datateathyme

- Review data structure
- OOP: Object Oriented Programming
- Request API
- Read SQLite
- Library: numpy

```
## Review data sturcture
## list, tuple, dictionary, set
friend_list = ["jay", "john", "joe", 42, 50, 99, [1, 2, 3], {"orange", "banana"}]
```

```
len(friend_list)
```

```
8
```

```
friend_list[7]
```

```
{'banana', 'orange'}
```

```
## handle the error
try:
    1/0
except:
    print("There is an error")
```

```
There is an error
```

```
try:
    1/1
    print("Okay!")
except:
    print("There is an error")
```

```
Okay!
```

```
try:
    name = "sunsun"
    print(sunsun)
except:
    print("This is an error")
finally:
    print("This is the completed")
```

```
This is an error
This is the completed
```

```
## dictionary
## key-value pair

jay = {
    "name": "jay",
    "age": 25,
    "gender": "male",
    "movie_fav": ["Titanic", "Superman"]
}
```

```
jay["movie_fav"][1]
```

```
'Superman'
```

```
## example loop in dictionary
fruits = ["orange", "banana", "orange", "orange", "banana"]

result = {} # empty dict

for fruit in fruits:
```

```
    if fruit in result:
        result[fruit] += 1
    else:
        result[fruit] = 1

print(result)
```

```
{'orange': 3, 'banana': 2}
```

OOP: Object Oriented Programming

```
class MooDeng():
    def __init__(self, name, age, species):
        self.name = name
        self.age = age
        self.species = species

    def hello(self):
        print("I'm MooDeng!")

    def sleep(self):
        print("I am going to bed now.")
```

```
md = MooDeng("moodeng", 1, "hippo")
```

```
jay = MooDeng("jay", 2, "hippo")
```

```
print(md.name, md.age, md.species)
print(jay.name, jay.age, jay.species)
```

```
moodeng 1 hippo
jay 2 hippo
```

```
# call method
md.hello()
md.sleep()
```

```
I'm MooDeng!
I am going to bed now.
```

```
class User():
    ## attribute
    def __init__(self, name, age, gender, city):
        self.name = name
        self.age = age
        self.gender = gender
        self.city = city

    ## method
    def upload_image(self):
        ## take image from a user
        print("Upload image successfully!")

    def add_age(self):
        self.age += 1

    def minus_age(self):
        self.age -= 1

    ## string representation
    def __str__(self):
        text = f"{self.name} is a {self.gender}, {self.age} years old. lives in {self.city}"
        return text
```

```
user1 = User("jay", 25, "male", "Taipei")
user1.name
user1.upload_image()
```

```
Upload image successfully!
```

```
print(user1.age)
user1.add_age()
print(user1.age)
```

```
25
26
```

```
print(user1)
```

```
jay is a male, 26 years old. lives in Taipei
```

```
jenny = User("Jenny", 22, "female", "London")
print(jenny)
```

```
Jenny is a female, 22 years old. lives in London
```

```
## Homework
# OOP create ATM
# create 4-5 method
class ATM:
    def __init__(self, name, bank, balance):
        self.name = name
        self.bank = bank
        self.balance = balance

    def check_balance(self):
        print(f"Current Balance: {self.balance} $")

    def deposit(self, amount):
        self.balance += amount

    def withdraw(self, amount):
        if amount > self.balance:
            raise ValueError("Insufficient funds")
        self.balance -= amount
        print(f"Withdrawal successful. Please collect your cash ({amount})$.")

    def transfer(self, recipient_atm, amount):
        if amount > self.balance:
            raise ValueError("Insufficient funds")
        self.balance -= amount
        recipient_atm.balance += amount
        print(f"Transfer successful. {amount} $ has been sent to {recipient_atm.name}.")
```

```
account_jay = ATM("jay", "TaipeiBank", 10000)
```

```
account_jenny = ATM("jenny", "TaipeiBank", 5000)
```

```
account_jay.check_balance()
```

```
Current Balance: 10000 $
```

```
account_jay.deposit(5000)
```

```
account_jay.check_balance()
```

```
Current Balance: 15000 $
```

```
account_jay.withdraw(10000)
```

```
Withdrawal successful. Please collect your cash (10000)$.
```

```
account_jay.check_balance()
```

```
Current Balance: 5000 $
```

```
account_jay.transfer(account_jenny, 5000)
```

```
Transfer successful. 5000 $ has been sent to jenny.
```

```
account_jenny.check_balance()
```

```
Current Balance: 10000 $
```

```
account_jay.check_balance()
```

```
Current Balance: 0 $
```

Read CSV file

import csv

```
import csv

data = []

try:
    file = open("customers_arpu.csv", "r")
    reader = csv.reader(file)
    for row in reader:
        data.append(row)
    print(data)
    file.close()
except:
    print("file not found, please check the filename again.")
```

```
[['\ufeffid', 'name', 'arpu', 'city'], ['1', 'john', '500', 'BKK'], ['2', 'toy', '250', 'BKK'], ['3', 'anne', '300', 'BKK'], ['4',
```

```
import csv

data = []

try:
    ## context manager - with ปิดไฟล์ให้อัตโนมัติ
    with open("customers_arpu.csv", "r") as file:
        reader = csv.reader(file)
        for row in reader:
            data.append(row)
        print(data)
except:
    print("file not found, please check the filename again.")
```

```
[['\ufeffid', 'name', 'arpu', 'city'], ['1', 'john', '500', 'BKK'], ['2', 'toy', '250', 'BKK'], ['3', 'anne', '300', 'BKK'], ['4',
```

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

	id	name	arpu	city
0	1	john	500	BKK
1	2	toy	250	BKK
2	3	anne	300	BKK
3	4	jessica	400	Lon
4	5	joy	800	Lon

```
df["arpu"].sum()
```

```
np.int64(2250)
```

```
!pip install gazpacho
```

Show hidden output

How to write a csv file

```
## how to write a csv file
## csv.writer()

import csv

## nested list
header = ["Name", "Age", "City"]

data = [
    ["Alice", 25, "New York"],
    ["Bob", 30, "London"],
    ["Charlie", 22, "Paris"]
]

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

```
!cat example_data.csv
```

```
Name,Age,City
Alice,25,New York
Bob,30,London
Charlie,22,Paris
```

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

	Name	Age	City
0	Alice	25	New York
1	Bob	30	London
2	Charlie	22	Paris

```
df["Age"] += 1
df["Country"] = ["USA", "UK", "France"]
df
```

	Name	Age	City	Country
0	Alice	26	New York	USA
1	Bob	31	London	UK
2	Charlie	23	Paris	France

```
# write csv
df.to_csv("update_example_data.csv")
```

```
!cat update_example_data.csv
```

```
,Name,Age,City,Country
0,Alice,26,New York,USA
1,Bob,31,London,UK
2,Charlie,23,Paris,France
```

```
import csv

with open("customers_arpu.csv", "r") as file:
    ## a reader
    reader = csv.reader(file)
    for row in reader:
        print(row)
```

```
['\ufeffid', 'name', 'arpu', 'city']
['1', 'john', '500', 'BKK']
['2', 'toy', '250', 'BKK']
['3', 'anne', '300', 'BKK']
['4', 'jessica', '400', 'Lon']
['5', 'joy', '800', 'Lon']
```

🤚 json: javascript object notation => API

import `json`

- json.dump( ) # write file
- json.load( ) # read file

json == **dict in python**

```
## json => API
jay_dict = {
    "name": "jay",
    "age": 25,
    "fav_move": ["Superman", "loki"]
}

jay_dict
```

```
{'name': 'jay', 'age': 25, 'fav_move': ['Superman', 'loki']}
```

```
import json

# write dict to json file
with open("jay_data.json", "w") as file:
    json.dump(jay_dict, file)
```

```
!cat jay_data.json
```

```
{"name": "jay", "age": 25, "fav_move": ["Superman", "loki"]}
```

```
## read json to dict in python
with open("jay_data.json", "r") as file:
    data = json.load(file)

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

```
{'name': 'jay', 'age': 25, 'fav_move': ['Superman', 'loki']} <class 'dict'>
```

📩 API

```
# import requests
from requests import get
```

```
response = get("https://swapi.info/api/people/1")

response.status_code
```

```
200
```

```
response.json()["name"]
```

```
'Luke Skywalker'
```

```
## get data from id 1-5
for i in range(1, 6):
    response = get(f"https://swapi.info/api/people/{i}")
    print(response.json()["name"])
```

```
Luke Skywalker
C-3PO
R2-D2
Darth Vader
Leia Organa
```

```
## get data from id 1-5
from requests import get
from time import sleep

base_url = "https://swapi.info/api/people/"

for i in range(1, 6):
```

```
    response = get(base_url + str(i))
    print(response.json()["name"])
    # best practice
    sleep(2) ## break 2 seconds
```

```
Luke Skywalker
C-3PO
R2-D2
Darth Vader
Leia Organa
```

```
## get data from id 1-5
from requests import get
from time import sleep

base_url = "https://swapi.info/api/people/"

characters = []

for i in range(1, 6):
    response = get(base_url + str(i))
    response_js = response.json()
    name = response_js["name"]
    height = response_js["height"]
    mass = response_js["mass"]
    result = [name, height, mass]
    characters.append(result)
    sleep(2) ## break 2 seconds

print(characters)
```

```
[['Luke Skywalker', '172', '77'], ['C-3PO', '167', '75'], ['R2-D2', '96', '32'], ['Darth Vader', '202', '136'], ['Leia Organa', '156
```

```
## write csv file
header = ["name", "height", "mass"]
with open("starwars.csv", "w") as file:
    writer = csv.writer(file)
    writer.writerow(header)
    writer.writerows(characters)
```

```
!cat starwars.csv
```

Show hidden output

Gazpacho

Basic web scraping

```
# !pip install gazpacho
from gazpacho import Soup
from requests import get
```

```
url = "https://datarockie.com"
```

```
web = get(url)

datarockie = Soup(web.text)

print(type(datarockie))
```

```
<class 'gazpacho.soup.Soup'>
```

```
## find information we from this Soup
datarockie.find("h2", mode="first")
```

```
<h2 class="wp-block-heading has-x-large-font-size" style="font-style:normal;font-weight:900">Learn For Free</h2>
```

```
## clean
datarockie.find("h2", mode="first").strip()
```

```
'Learn For Free'
```

```
for h2 in datarockie.find("h2"):
    print(h2.strip())
```

Show hidden output

Import sqlite3

```
import sqlite3
import pandas as pd ## dataframe
```

```
## create connection
con = sqlite3.connect("chinook.db")
```

```
df = pd.read_sql("select * from customers limit 5", con)
```

```
df.head()
```

	CustomerId	FirstName	LastName	Company	Address	City	State	Country	PostalCode	Phone	Fax	Er
0	1	Luís	Gonçalves	Embraer - Empresa Brasileira de Aeronáutica S.A.	Av. Brigadeiro Faria Lima, 2170	São José dos Campos	SP	Brazil	12227-000	+55 (12) 3923-5555	+55 (12) 3923-5566	luisg@embraer.co
1	2	Leonie	Köhler	None	Theodor-Heuss-Straße 34	Stuttgart	None	Germany	70174	+49 0711 2842222	None	leonekohler@surfe
2	3	François	Tremblay	None	1498 rue Bélanger	Montréal	QC	Canada	H2G 1A7	+1 (514) 721-4711	None	ftremblay@gmail.
3	4	Bjørn	Hansen	None	Ullevålsveien 14	Oslo	None	Norway	0171	+47 22 44 22 22	None	bjorn.hansen@yaho
4	5	František	Wichterlová	JetBrains s.r.o.	Klanova 9/506	Prague	None	Czech Republic	14700	+420 2 4172 5555	+420 2 4172 [illegible]	frantisekw@jetbrains.

Next steps: Generate code with df | View recommended plots | New interactive sheet

```
df[["FirstName", "LastName", "City"]]
```

	FirstName	LastName	City
0	Luís	Gonçalves	São José dos Campos
1	Leonie	Köhler	Stuttgart
2	François	Tremblay	Montréal
3	Bjørn	Hansen	Oslo
4	František	Wichterlová	Prague

```
## close connection
con.close()
```

concat

```
## concat

import pandas as pd

# Create the first DataFrame
df1 = pd.DataFrame({'A': ['A0', 'A1', 'A2'],
                    'B': ['B0', 'B1', 'B2']},
                   index=[0, 1, 2])

# Create the second DataFrame
```

```
df2 = pd.DataFrame({'A': ['A3', 'A4', 'A5'],
                    'B': ['B3', 'B4', 'B5']},
                   index=[3, 4, 5])
```

```
df1
```

	A	B
0	A0	B0
1	A1	B1
2	A2	B2

Next steps: Generate code with df1 | View recommended plots | New interactive sheet

```
df2
```

	A	B
3	A3	B3
4	A4	B4
5	A5	B5

Next steps: Generate code with df2 | View recommended plots | New interactive sheet

```
## append, select * from df1 union all select * from df2
df3 = pd.concat([df1, df2])
df3
```

	A	B
0	A0	B0
1	A1	B1
2	A2	B2
3	A3	B3
4	A4	B4
5	A5	B5

Next steps: Generate code with df3 | View recommended plots | New interactive sheet

numpy

numerical python

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

```
gpa = [3.4, 3.5, 4.00, 2.9] #python just have => sum min max
```

```
gpa = np.array(gpa)
```

```
type(gpa)
```

```
numpy.ndarray
```

```
np.mean(gpa)
```

```
np.float64(3.45)
```

```
gpa.mean()
```

```
np.float64(3.45)
```

```
np.median(gpa)
```

```
np.float64(3.45)
```

```
print(gpa.sum()) # method
print(np.sum(gpa)) # function
```

```
13.8
13.8
```

Start coding or generate with AI.