🐣 Essential Python for Data Analyst

- OOP
- Pandas / Numpy
- Requests

## 🎯 OOP: object oriented programming

```
## OOP
## object oriented programming
```

```
## create a new class (cookie cutter)
class Human:
    ## initialazation (create)
    def __init__(self, name, age, country):
        self.name = name
        self.age = age
        self.country = country

    ## string method
    def __str__(self):
        return "I am Human!"

    ## your own method
    def greeting(self):
        print(f"Hi! my name is {self.name}")

    ## age + 1
    def get_older(self, year):
        self.age += year
        print(f"Age increases by {year}.")
```

```
## create a new instance
user1 = Human("jay", 25, "Thailand")
user2 = Human("jenny", 23, "USA")
user3 = Human("jesoo", 21, "UK")
user4 = Human("lisa", 20, "Taiwan")
```

```
print(user1.name, user2.name, user3.name, user4.name)
```

```
jay jenny jesoo lisa
```

```
print(user1)
```

```
I am Human!
```

```
user3.greeting()
```

```
Hi! my name is jesoo
```

```
user1.name
```

```
'jay'
```

```
user1.greeting()
```

```
Hi! my name is jay
```

```
user1.age
```

```
25
```

```
user1.get_older(5)
```

```
Age increases by 5.
```

```
user1.age
```

```
30
```

💳Create: Class ATM

```
## ATM
class ATM:
    def __init__(self, name, bank, balance):
        self.name = name
        self.bank = bank
        self.balance = balance

    def check_balance(self):
        print(f"Your balance is {self.balance} $")

    def deposit(self, amount):
        self.balance += amount
        print(f"You just deposit {amount} $")

    def withdraw(self, amount):
        if self.balance >= amount:
            self.balance -= amount
            print(f"You just withdraw {amount} $")
        else:
            print("Insufficient balance!")

    def transfer(self, amount, receiver):
        if self.balance >= amount:
            self.balance -= amount
            receiver.balance += amount
            print(f"You just transfer {amount} $ to {receiver.name}")
        else:
            print("Insufficient balance!")
```

```
binnie = ATM("binnie", "KTB", 5000)
bonnie = ATM("bonnie", "KTB", 10000)
```

```
binnie.check_balance()
```

```
Your balance is 5000 $
```

```
binnie.deposit(1000)
```

```
You just deposit 1000 $
```

```
binnie.check_balance()
```

```
Your balance is 6000 $
```

```
binnie.withdraw(85000)
```

```
Insufficient balance!
```

```
binnie.withdraw(3000)
```

```
You just withdraw 3000 $
```

```
binnie.check_balance()
```

```
Your balance is 3000 $
```

```
bonnie.check_balance()
```

```
Your balance is 10000 $
```

```
bonnie.transfer(1000, binnie)
```

```
You just transfer 1000 $ to binnie
```

```
binnie.check_balance()
```

```
Your balance is 4000 $
```

```
bonnie.check_balance()
```

```
Your balance is 9000 $
```

🎯 API 101 in Python

```
## standard module (library)
import requests
```

```
url = "https://swapi.info/api/people/1"
```

```
res = requests.get(url)
```

```
res.status_code
```

```
200
```

```
if res.status_code == 200:
    print("Success")
else:
    print("Please check the path again!")
```

```
Success
```

```
res.json()
```

```
{'name': 'Luke Skywalker',
 'height': '172',
 'mass': '77',
 'hair_color': 'blond',
 'skin_color': 'fair',
 'eye_color': 'blue',
 'birth_year': '19BBY',
 'gender': 'male',
 'homeworld': 'https://swapi.info/api/planets/1',
 'films': ['https://swapi.info/api/films/1',
  'https://swapi.info/api/films/2',
  'https://swapi.info/api/films/3',
  'https://swapi.info/api/films/6'],
 'species': [],
 'vehicles': ['https://swapi.info/api/vehicles/14',
  'https://swapi.info/api/vehicles/30'],
 'starships': ['https://swapi.info/api/starships/12',
  'https://swapi.info/api/starships/22'],
 'created': '2014-12-09T13:50:51.644000Z',
 'edited': '2014-12-20T21:17:56.891000Z',
 'url': 'https://swapi.info/api/people/1'}
```

```
## loop API
import requests
import time


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

names = []
heights = []
masses = []

for i in range(1, 6):
    response = requests.get(url + str(i))
    name = response.json()["name"]
    height = response.json()["height"]
    mass = response.json()["mass"]
    names.append(name)
    heights.append(height)
    masses.append(mass)
    print(name)
    time.sleep(2)
```

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

```
import pandas as pd
```

```
df = pd.DataFrame({
    "name": names,
    "height": heights,
    "mass": masses
})
```

```
df
```

	name	height	mass
0	Luke Skywalker	172	77
1	C-3PO	167	75
2	R2-D2	96	32
3	Darth Vader	202	136
4	Leia Organa	150	49

🎯 Pandas, Numpy

The most common for data analyst

- Numpy: numerical pythom (fast computation)
- Pandas

```
import numpy as np
import pandas as pd
```

```
list_a = [1, 2, 3, 4, 5 , 6, 7, 8, 9, 10]
sum(list_a)
```

```
55
```

```
 def sum_seq(lst):
    result = (lst[0] + lst[-1]) * (lst[-1] / 2)
    return result
```

```
sum_seq(list_a)
```

```
55.0
```

```
import numpy as np
```

```
np_a = np.array(list_a)
```

```
np_a
```

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

```
print(
    np.sum(np_a),
    np.mean(np_a),
    np.std(np_a),
    np.min(np_a),
    np.max(np_a),
    np.median(np_a)
)
```

```
55 5.5 2.8722813232690143 1 10 5.5
```