Essential Python for DA

```python
print("hello world")
```

```
hello world
```

OOP

```python
class ATM:
    pass
```

```python
from random import randint

class ATM:
    def __init__(self, account_name, bank_name, initial):
        self.account_name = account_name
        self.bank_name = bank_name
        self.balance = initial

    ## string representation
    def __str__(self):
        return f"This is an account of {self.account_name}, bank: {self.bank_name}"

    ## method => function
    def check_balance(self):
        print(f"Balance: {self.balance} THB")

    def deposit(self, money):
        self.balance += money
        print(f"Deposit successfully: your new balance: {self.balance} THB")

    def withdraw(self, money):
        self.balance -= money
        print(f"Withdraw successfully: your new balance: {self.balance} THB")

    def get_OTP(self):
        otp = randint(1000, 9999)
        print(f"Your OTP: {otp} This OTP will be available in the next 2 minutes.")
```

```python
# create an instance from ATM class
acc1 = ATM("toy", "scb", 500)
```

```python
# code is for human
print(acc1)
```

```
This is an account of toy, bank: scb
```

```
acc1.check_balance()
```

```
Balance: 500 THB
```

```
acc1.deposit(1000)
```

```
Deposit successfully: your new balance: 1500 THB
```

```
acc1.withdraw(300)
```

```
Withdraw successfully: your new balance: 1200 THB
```

```
acc1.get_OTP()
```

```
Your OTP: 7204 This OTP will be available in the next 2 minutes.
```

```
## OK
```

```

```

Try Except Block

```
try:
    1/0
except ZeroDivisionError:
    print("cannot divide by zero")
except NameError:
    print("variable not defined")
else:
    print("Done")
finally:
    print("Complete!")
```

```
cannot divide by zero
Complete!
```

```python
## import csv
import csv
```

```python
try:
    file = open("fasdasdriends.csv")
    data = csv.reader(file)
    for row in data:
        print(row)
    file.close()
except FileNotFoundError:
    print("File not found.")
```

```
File not found.
```

```python
## context manager
result = []

## open and close file automatically
try:
    with open("friends.csv", "r") as file:
        data = csv.reader(file)
        for row in data:
            result.append(row)
except:
    print("file not found")
else:
    print("load data successfully!")
finally:
    print(result)
```

```
load data successfully!
[['id', 'name', 'age', 'city'], ['1', 'toy', '35', 'bangkok'], ['2', 'john', '32', 'london'], ['3', 'mary', '28', 'seou
```

```
import pandas
```

```
try:
    df = pandas.read_csv("friends.csv")
except:
    print("a little error.")
```

```
## write csv file using pandas
df.to_csv("newCSVFile.csv")
```

```
# write csv using csv module
import csv

col_names = ["food_id", "food", "price"]

data = [
    [1, "pizza", 200],
    [2, "french fried", 50],
    [3, "coke", 10]
]

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

```
!cat food.csv
```

```
food_id,food,price
1,pizza,200
2,french fried,50
3,coke,10
```

```
# JSON
```

```
## json = dictionary in python

import json

with open("data.json") as file:
    result = json.load(file)

print(result)
```

```
{'id': 1, 'name': 'toy', 'favorite_food': ['coke', 'pizza']}
```

```
result["favorite_food"].append("hamburger")
result
```

```
{'id': 1, 'name': 'toy', 'favorite_food': ['coke', 'pizza', 'hamburger']}
```

```
result["name"] = "John Wick"
```

```
result["city"] = "New York"
```

```
result
```

```
{'id': 1,
 'name': 'John Wick',
 'favorite_food': ['coke', 'pizza', 'hamburger'],
 'city': 'New York'}
```

```
from json import load, dump

## with =??
with open("JohnWick.json", "w") as file:
    json.dump(result, file, indent=6)
    print("successfully dump a new json file.")
```

```
successfully dump a new json file.
```

```
!cat JohnWick.json
```

```
{
      "id": 1,
      "name": "John Wick",
      "favorite_food": [
            "coke",
            "pizza",
            "hamburger"
      ],
      "city": "New York"
}
```

```
<Response [200]>
```

API

Application Programming Interface

Numpy & Pandas

numerical python

pandas dataframe

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

```python
nums = [1, 20, 25, 30, 100] # vector c(1,20,25,30,100)
```

```python
## numpy array
arr_nums = np.array(nums)
```

```python
np.sum(arr_nums)
```

```
176
```

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

```
176 35.2 25.0 1 100 33.85498486190771
```

```python
arr_nums.std()
```

```
33.85498486190771
```

```python
## vector in R
m1 = np.array([
    [1,2],
    [3,4]
])
```

```python
# element wise computation
# broadcasting
m1 + 100
```

```
    array([[101, 102],
           [103, 104]])
```

```
np.ones((3,3))
```

```
    array([[1., 1., 1.],
           [1., 1., 1.],
           [1., 1., 1.]])
```

```
np.zeros((2,2))
```

```
    array([[0., 0.],
           [0., 0.]])
```

```
np.arange(1, 101, 10)
```

```
    array([ 1, 11, 21, 31, 41, 51, 61, 71, 81, 91])
```

```
np.linspace(1, 101, 10)
```

```
    array([  1.        ,  12.11111111,  23.22222222,  34.33333333,
            45.44444444,  56.55555556,  67.66666667,  78.77777778,
            89.88888889, 101.        ])
```

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

m2 = np.array([
    [5,5],
    [3,2]
])
```

```
np.dot(m1, m2)
```

```
array([[11,  9],
       [27, 23]])
```

```
m1.dot(m2)
```

```
array([[11,  9],
       [27, 23]])
```

```
import pandas as pd

## create dataframe from scratch
data = {
    "id": [1,2,3],
    "name": ["toy", "anna", "jessica"],
    "city": ["BKK", "JPN", "LON"]
}
```

```
df = pd.DataFrame(data)
```

```
df
```

	id	name	city
0	1	toy	BKK
1	2	anna	JPN
2	3	jessica	LON

```
df["age"] = [35, 28, 29]
df
```

	id	name	city	age
0	1	toy	BKK	35
1	2	anna	JPN	28
2	3	jessica	LON	29

```
df.drop("age", axis=1) # 1 is columns
```

	id	name	city
0	1	toy	BKK
1	2	anna	JPN
2	3	jessica	LON

```
# read csv file from pandas

df = pd.read_csv("store.csv")

df.head(2)
```

	Row ID	Order ID	Order Date	Ship Date	Ship Mode	Customer ID	Customer Name	Segment	Country	City	...	Postal Code	Region	Product ID	C
0	1	CA-2016-152156	2016-11-08	2016-11-11	Second Class	CG-12520	Claire Gute	Consumer	United States	Henderson	...	42420	South	FUR-BO-10001798	F
1	2	CA-2016-152156	2016-11-08	2016-11-11	Second Class	CG-12520	Claire Gute	Consumer	United States	Henderson	...	42420	South	FUR-CH-10000454	F

2 rows × 21 columns

```python
df.shape # attribute
```

```
(9994, 21)
```

```python
df.info() # method
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9994 entries, 0 to 9993
Data columns (total 21 columns):
 #   Column         Non-Null Count  Dtype 
---  ------         --------------  ----- 
 0   Row ID         9994 non-null   int64 
 1   Order ID       9994 non-null   object
 2   Order Date     9994 non-null   object
 3   Ship Date      9994 non-null   object
 4   Ship Mode      9994 non-null   object
 5   Customer ID    9994 non-null   object
 6   Customer Name  9994 non-null   object
 7   Segment        9994 non-null   object
 8   Country        9994 non-null   object
 9   City           9994 non-null   object
 10  State          9994 non-null   object
 11  Postal Code    9994 non-null   int64 
 12  Region          9994 non-null   object
 13  Product ID      9994 non-null   object
 14  Category        9994 non-null   object
 15  Sub-Category    9994 non-null   object
 16  Product Name    9994 non-null   object
 17  Sales           9994 non-null   float64
 18  Quantity        9994 non-null   int64
 19  Discount        9994 non-null   float64
 20  Profit          9994 non-null   float64
dtypes: float64(3), int64(3), object(15)
memory usage: 1.6+ MB
```

```python
## query() => filter rows with conditions

## clean dataframe column names

col_names = list(df.columns)

# list comprehension
clean_col_names = [name.lower().replace(" ", "_").replace("-", "_")
                   for name in col_names]

print(clean_col_names)
```

```
['row_id', 'order_id', 'order_date', 'ship_date', 'ship_mode', 'customer_id', 'customer_name', 'segment', 'country', 'c
```

```python
## assign clean col names to dataframe
df.columns = clean_col_names

df.head()
```

e	ship_date	ship_mode	customer_id	customer_name	segment	country	city	...	postal_code	region	product_id
8	2016-11-11	Second Class	CG-12520	Claire Gute	Consumer	United States	Henderson	...	42420	South	FUR-BO-10001798
8	2016-11-11	Second Class	CG-12520	Claire Gute	Consumer	United States	Henderson	...	42420	South	FUR-CH-10000454
2	2016-06-16	Second Class	DV-13045	Darrin Van Huff	Corporate	United States	Los Angeles	...	90036	West	OFF-LA-10000240
1	2015-10-18	Standard Class	SO-20335	Sean O'Donnell	Consumer	United States	Fort Lauderdale	...	33311	South	FUR-TA-10000577
1	2015-10-18	Standard Class	SO-20335	Sean O'Donnell	Consumer	United States	Fort Lauderdale	...	33311	South	OFF-ST-10000760

```
## data transformation (R dplyr)

df2 = df[ ["customer_id", "customer_name"] ].head(5)
```

```
df[ df["customer_name"] == "Claire Gute" ][["order_date", "customer_id", "customer_name"]]
```

	order_date	customer_id	customer_name
0	2016-11-08	CG-12520	Claire Gute
1	2016-11-08	CG-12520	Claire Gute
5491	2017-01-26	CG-12520	Claire Gute
6877	2015-10-15	CG-12520	Claire Gute
6878	2015-10-15	CG-12520	Claire Gute

```
## query
df.query(" city == 'Los Angeles' and category == 'Furniture' and sub_category == 'Tables' ")[["customer_name", "segment"
```

	customer_name	segment	city
10	Brosina Hoffman	Consumer	Los Angeles
282	Jas O'Carroll	Consumer	Los Angeles
557	Olvera Toch	Consumer	Los Angeles
1097	Noel Staavos	Corporate	Los Angeles
1505	Pauline Chand	Home Office	Los Angeles

```
## aggregate data
## region == "West"

res = df.query("region == 'West'")\
    .groupby(["segment", "region"])[["sales", "profit"]]\
    .agg(['sum', 'mean', 'count'])\
    .reset_index()

print(res)

res.to_csv("agg_data.csv")
```

```
       segment region        sales                          profit             \
                               sum        mean count          sum       mean   
0     Consumer   West  362880.7730  217.033955  1672  57450.6040  34.360409   
1    Corporate   West  225855.2745  235.265911   960  34437.4299  35.872323   
2  Home Office   West  136721.7770  239.442692   571  16530.4150  28.949939   


  count  
0  1672  
1   960  
2   571  
```

```
1
```

Double-click (or enter) to edit

Load data from SQL

```
import sqlite3
import pandas as pd

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

```
custs = pd.read_sql("select * from customers where country='USA'", con)

custs
```

1	17	Jack	Smith	Microsoft Corporation	1 Microsoft Way	Redmond	WA	USA	98052-8300	+1 (425) 882-8080	+1 (425) 882-8081	ja
2	18	Michelle	Brooks	None	627 Broadway	New York	NY	USA	10012-2612	+1 (212) 221-3546	+1 (212) 221-4679	
3	19	Tim	Goyer	Apple Inc.	1 Infinite Loop	Cupertino	CA	USA	95014	+1 (408) 996-1010	+1 (408) 996-1011	
4	20	Dan	Miller	None	541 Del Medio Avenue	Mountain View	CA	USA	94040-111	+1 (650) 644-3358	None	
5	21	Kathy	Chase	None	801 W 4th Street	Reno	NV	USA	89503	+1 (775) 223-7665	None	
6	22	Heather	Leacock	None	120 S Orange Ave	Orlando	FL	USA	32801	+1 (407) 999-7788	None	
7	23	John	Gordon	None	69 Salem Street	Boston	MA	USA	2113	+1 (617) 522-1333	None	joh
8	24	Frank	Ralston	None	162 E Superior Street	Chicago	IL	USA	60611	+1 (312) 332-3232	None	
9	25	Victor	Stevens	None	319 N. Frances Street	Madison	WI	USA	53703	+1 (608) 257-0597	None	

10	26	Richard	Cunningham	None	2211 W Berry Street	Fort Worth	TX	USA	76110	+1 (817) 924-7272	None	ricu
11	27	Patrick	Gray	None	1033 N Park Ave	Tucson	AZ	USA	85719	+1 (520) 622-4200	None	
12	28	Julia	Barnett	None	302 S 700 E	Salt Lake City	UT	USA	84102	+1 (801) 531-7272	None	

```
con.close()
```

Sklearn Foundation

Model: linear regression

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

```
## read csv data from github: mtcars

url = "https://gist.githubusercontent.com/seankross/a412dfbd88b3db70b74b/raw/5f23f993cd87c283ce766e7ac6b329ee7cc2e1d1/mt

mtcars = pd.read_csv(url)

mtcars.head(2)
```

	model	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
0	Mazda RX4	21.0	6	160.0	110	3.9	2.620	16.46	0	1	4	4
1	Mazda RX4 Wag	21.0	6	160.0	110	3.9	2.875	17.02	0	1	4	4

```
# ML Workflow
## 4 steps: split > train > score > evaluate

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

## 1. split data
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.20, random_state=19
)

## 2. train model
model = DecisionTreeRegressor()
model.fit(X_train, y_train) ## model fitting

## 3. score
train_error = model.score(X_train, y_train)
```