BÁO CÁO SƠ BỘ GIAI ĐOẠN 1 LẦN 1

Người báo cáo: Nguyễn Đức Quân

###################

###################

###################

CHƯƠNG I. TÌM HIỂU VỀ DATA ENGINEER

# 1.1 Data engineer là gì

Là các kỹ sư dữ liệu, sử dụng đầu vào là dữ liệu để phân tích nó hoặc tạo ra các hướng phát triển tiêp theo.

# 1.2 Dữ liệu là gì

Dữ liệu bao gồm các sự kiện, quan sát, nhận thức, số, ký tự, biểu tượng và hình ảnh có thể được giải thích để rút ra ý nghĩa.

# 1.3 Các loại data

Tóm lại, Dữ liệu có cấu trúc là dữ liệu được tổ chức tốt ở các định dạng có thể được lưu trữ trong cơ sở dữ liệu và phù hợp với các phương pháp và công cụ phân tích dữ liệu tiêu chuẩn; Dữ liệu bán cấu trúc là dữ liệu được tổ chức phần nào và dựa vào thẻ meta để nhóm và phân cấp; và Dữ liệu phi cấu trúc là dữ liệu không được tổ chức theo quy ước dưới dạng các hàng và cột theo một định dạng cụ thể. Trong video tiếp theo, chúng ta sẽ tìm hiểu về các loại cấu trúc tệp khác nhau.

# 1.4 Các công cụ

Course 1 tuần 2 video thứ 5

Spark streaming

RSS(Really simple syndication)

Python

Các thư viện hữu ích trong python: Numpy và scipy, pandas Beatifulsoup và Scrapy, Matplotlib và seaborn, Oprncy

Ngôn ngữ R

Java

Unix/Linux

Ngôn ngữ truy vấn, chẳng hạn như SQL, được sử dụng để truy cập và thao tác dữ liệu từ cơ sở dữ liệu.

Các ngôn ngữ lập trình như Python, R và Java, để phát triển các ứng dụng và kiểm soát hành vi ứng dụng.

Các ngôn ngữ Shell và Scripting, chẳng hạn như Unix / Linux Shell và PowerShell, để tự động hóa các tác vụ hoạt động lặp đi lặp lại.

# 1.5 Các loại database

Cơ sở dữ liệu quan hệ, còn được gọi là RDBMSes, được xây dựng dựa trên các nguyên tắc tổ chức của các tệp phẳng, với dữ liệu được tổ chức thành định dạng bảng với các hàng và cột tuân theo cấu trúc và lược đồ được xác định rõ. Tuy nhiên, không giống như các tệp phẳng, RDBMS được tối ưu hóa cho các hoạt động dữ liệu và truy vấn liên quan đến nhiều bảng và khối lượng dữ liệu lớn hơn nhiều. Ngôn ngữ truy vấn có cấu trúc, hoặc SQL, là ngôn ngữ truy vấn tiêu chuẩn cho cơ sở dữ liệu quan hệ.

chúng tôi có cơ sở dữ liệu phi quan hệ, còn được gọi là NoSQL hoặc "Không chỉ SQL". Cơ sở dữ liệu phi quan hệ xuất hiện để đáp ứng với khối lượng, sự đa dạng và tốc độ mà dữ liệu đang được tạo ra ngày nay, chủ yếu bị ảnh hưởng bởi những tiến bộ trong điện toán đám mây, Internet of Things và sự phổ biến của phương tiện truyền thông xã hội. Được xây dựng cho tốc độ, tính linh hoạt và quy mô, cơ sở dữ liệu phi quan hệ giúp lưu trữ dữ liệu theo kiểu không có lược đồ hoặc dạng tự do. NoSQL được sử dụng rộng rãi để xử lý dữ liệu lớn. Kho dữ liệu hoạt động như một kho lưu trữ trung tâm kết hợp thông tin đến từ disphân phối các nguồn và hợp nhất nó thông qua quá trình trích xuất, chuyển đổi và tải, còn được gọi là quy trình ETL, thành một cơ sở dữ liệu toàn diện để phân tích và kinh doanh thông minh. Ở cấp độ rất cao, quy trình ETL giúp bạn trích xuất dữ liệu từ các nguồn dữ liệu khác nhau, chuyển đổi dữ liệu thành trạng thái sạch và có thể sử dụng được, đồng thời tải dữ liệu vào kho dữ liệu của doanh nghiệp

IBM DB2, Microsoft SQL Server, MySQL, Oracle Database và PostgreSQL là một số cơ sở dữ liệu quan hệ phổ biến.Cơ sở dữ liệu quan hệ dựa trên đám mây, còn được gọi là Cơ sở dữ liệu dưới dạng dịch vụ, đang được sử dụng rộng rãi vì chúng có quyền truy cập vào khả năng tính toán và lưu trữ vô hạn do đám mây cung cấp. Một số cơ sở dữ liệu quan hệ đám mây phổ biến bao gồm Amazon Relational Database Service (RDS), Google Cloud SQL, IBM DB2 on Cloud, Oracle Cloud và SQL Azure

Điều quan trọng cần nhấn mạnh là "Không" trong "NoSQL" là chữ viết tắt của "không chỉ" và không phải là  từ thực tế "Không". Cơ sở dữ liệu NoSQL được xây dựng cho các mô hình dữ liệu cụ thể và có các lược đồ linh hoạt cho phép ác lập trình viên tạo và quản lý các ứng dụng hiện đại.

Ưu điểm chính của NoSQL là khả năng xử lý khối lượng lớn dữ liệu có cấu trúc, bán cấu trúc và phi cấu trúc. Một số ưu điểm khác của nó bao gồm: Khả năng chạy như các hệ thống phân tán được mở rộng trên nhiều trung tâm dữ liệu, cho phép chúng tận dụng cơ sở hạ tầng điện toán đám mây; Một kiến trúc mở rộng quy mô hiệu quả và tiết kiệm chi phí, cung cấp thêm dung lượng và hiệu suất với việc bổ sung các nút mới; và Thiết kế đơn giản hơn, kiểm soát tốt hơn tính khả dụng và khả năng mở rộng được cải thiện cho phép bạn nhanh nhẹn hơn, linh hoạt hơn và lặp lại nhanh hơn Để tóm tắt sự khác biệt chính giữa cơ sở dữ liệu quan hệ và phi quan hệ

Tất cả các kho lưu trữ khai thác dữ liệu đều có một mục tiêu tương tự, đó là lưu trữ dữ liệu để báo cáo, phân tích và rút ra thông tin chi tiết. Nhưng mục đích của chúng, các loại dữ liệu được lưu trữ và cách dữ liệu được truy cập khác nhau.

## 1.5.1 Data warehouse

Kho dữ liệu là kho lưu trữ dữ liệu trung tâm được tích hợp từ nhiều nguồn. Kho dữ liệu đóng vai trò là nguồn sự thật duy nhất — lưu trữ dữ liệu hiện tại và lịch sử đã được làm sạch, tuân thủ và phân loại. Khi dữ liệu được tải vào kho dữ liệu, nó đã được mô hình hóa và cấu trúc cho một mục đích cụ thể, có nghĩa là nó đã sẵn sàng phân tích. Theo truyền thống, kho dữ liệu được biết là lưu trữ dữ liệu quan hệ từ các hệ thống giao dịch và cơ sở dữ liệu hoạt động như các ứng dụng CRM, ERP, HR và Tài chính. Nhưng với sự xuất hiện của các công nghệ NoSQL và các nguồn dữ liệu mới, các kho dữ liệu phi quan hệ cũng đang được sử dụng để lưu trữ dữ liệu. Thông thường, kho dữ liệu có kiến trúc ba tầng: Tầng dưới cùng của kiến trúc bao gồm các máy chủ cơ sở dữ liệu, có thể là quan hệ, phi quan hệ hoặc cả hai, trích xuất dữ liệu từ các nguồn khác nhau. Tầng giữa của kiến trúc bao gồm OLAP Server, một loại phần mềm cho phép người dùng xử lý và phân tích thông tin đến từ nhiều máy chủ cơ sở dữ liệu. Và tầng trên cùng của kiến trúc bao gồm lớp front-end của máy khách. Bậc này bao gồm tất cả các công cụ và ứng dụng được sử dụng để truy vấn, báo cáo và phân tích dữ liệu. Để đáp ứng với sự tăng trưởng dữ liệu nhanh chóng và các công cụ phân tích tinh vi ngày nay, các kho dữ liệu từng nằm trong các trung tâm dữ liệu tại chỗ đang chuyển sang đám mây. So với các phiên bản tại chỗ, một số lợi ích được cung cấp bởi kho dữ liệu dựa trên đám mây bao gồm: Chi phí thấp hơn, Khả năng tính toán và lưu trữ không giới hạn, Mở rộng quy mô trên cơ sở thanh toán theo mức sử dụng; và Phục hồi sau thảm họa nhanh hơn. Là một tổ chức, bạn sẽ chọn kho dữ liệu khi bạn có một lượng lớn dữ liệu từ các hệ thống hoạt động của mình cần có sẵn để báo cáo và phân tích. Một số kho dữ liệu được sử dụng phổ biến bao gồm nền tảng Teradata Enterprise Data Warehouse, Oracle Exadata, IBM Db2 Warehouse on Cloud, IBM Netezza Performance Server, Amazon RedShift, BigQuery by Google Cloudera's Enterprise Data Hub và Snowflake Cloud Data Warehouse

## 1.5.2 Data mart

Data mart là một phần phụ của kho dữ liệu, được xây dựng đặc biệt cho một chức năng kinh doanh, mục đích hoặc cộng đồng người dùng cụ thể.Đối với kỳ thile, các nhóm bán hàng hoặc tài chính trong một tổ chức truy cập dữ liệu cho báo cáo và dự báo hàng quý của họ.Có ba loại siêu thị dữ liệu cơ bản — siêu thị dữ liệu phụ thuộc, độc lập và lai. Mart dữ liệu phụ thuộc là một phần phụ của kho dữ liệu doanh nghiệp. Vì một siêu thị dữ liệu phụ thuộc cung cấp khả năng phân tích cho một khu vực hạn chế của kho dữ liệu, nó cũng cung cấp bảo mật biệt lập và hiệu suất bị cô lập. Các siêu thị dữ liệu độc lập được tạo ra từ các nguồn khác ngoài kho dữ liệu doanh nghiệp, chẳng hạn như hệ thống hoạt động nội bộ hoặc dữ liệu ngoài. Các siêu thị dữ liệu lai kết hợp đầu vào từ kho dữ liệu, hệ thống vận hành và hệ thống bên ngoài. Sự khác biệt cũng nằm ở cách dữ liệu được trích xuất từ các hệ thống nguồn, các biến đổi cần được áp dụng và cách dữ liệu được vận chuyển vào siêu thị. Ví dụ: các siêu thị dữ liệu phụ thuộc lấy dữ liệu từ kho dữ liệu doanh nghiệp, nơi dữ liệu đã được làm sạch và chuyển đổi. Các siêu thị dữ liệu độc lập cần thực hiện quá trình chuyển đổi trên dữ liệu nguồn vì nó đến trực tiếp từ các hệ thống hoạt động và các nguồn bên ngoài. Dù là loại nào, mục đích của siêu thị dữ liệu là: cung cấp dữ liệu của người dùng phù hợp nhất với họ khi họ cần, tăng tốc quy trình kinh doanh bằng cách cung cấp thời gian phản hồi hiệu quả, cung cấp cách hiệu quả về chi phí và thời gian để đưa ra quyết định dựa trên dữ liệu, cải thiện thời gian phản hồi của người dùng cuối; và cung cấp quyền truy cập và kiểm soát an toàn

1.5.3 Data lake

Là nơi tập hợp tất cả các định dạng dữ liệu, gọi là dữ liệu thô, các dữ liệu vào đây sẽ được phân loại, sắp xếp và sẵn sàng để chuyển sang data warehouse để phân tích.

Và rất nhiều lần các data lake này có thể trở thành đầm lầy dữ liệu.Điều này xảy ra khi có nhiều dữ liệu trùng lặp, không chính xác hoặc không đầy đủ, gây khó khăn cho việc theo dõi và quản lý tài sản. Vì vậy, nếu bạn nghĩ về nó, điều gì sẽ xảy ra khi dữ liệu đó trở nên cũ? Chà, nó sẽ mất giá trị trong việc tạo ra những hiểu biết sâu sắc, giống như cách các thành phần bị hỏng theo thời gian trong nhà hàng của chúng tôi nếu chúng tôi không sử dụng chúng. Vì vậy, Data lake cũng có những thách thức với hiệu suất truy vấn. Vì chúng không được xây dựng và tối ưu hóa để xử lý các truy vấn phân tích phức tạp, đôi khi có thể khó hiểu trực tiếp từ các data lake

CHƯƠNG II. PYTHON TRONG DATA ENGINEER

# 2.1 Các kiểu dữ liệu

String: là kiểu dữ liệu chứ các ký tự

Ví dụ: my\_string= “Hello World”

Các toán tử liên quan đến string:

my\_string.upper(): trả về các ký tự viết hoa, len(my\_string): trả về độ dài của chuỗi my\_string, my\_string.find(‘o’): trả về vị trí xuất hiện của kí tự cần tìm, nếu không có thì kết quả là -1

Integer: kiểu dữ liệu là số nguyên

Ví dụ: my\_integer= 1234, lưu ý khi cộng chuỗi với số nguyên thì sẽ xảy ra lỗi nên ta cần chuyển về string rồi mới cộng.

Ví dụ: print(“Đây là” + my\_integer +”kí tự”) sẽ xẩy ra lỗi, ta nên làm như sau: print(“Đây là” +str(my\_integer) +”ký tự”)

Float: kiểu dữ liệu là số phẩy

Ví dụ: my\_float: 2.34

Để chuyển từ số nguyên sang số phẩy hay ngược lại ta dùng ép kiểu. Ví dụ: int(3,99)=3, không thể làm tròn

Boolean: Là kiểu dữ liệu đúng sai, true/false

Ví dụ: a=true

List: là kiểu dữ liệu danh sách

Ví dụ: list\_a=[“10”,”b”,”c”]

Thao tác với list tương tự với mảng, list\_a[1]=b, len(list\_a)=3, ta có thể thay đổi giá trị từng phần tử trong list. Ví dụ: list\_a[0]=”d” thì list\_a=[“d”,”b”,”c”]. Lưu ý khi ta gán list\_b=list\_a thì đó là gán địa chỉ của a sang b, nên nếu ta thay đổi các giá trị của list\_b thì list\_a cũng thay đổi theo. Để copy một list thì ta sử dụng lệnh sau: list\_b=list\_a[:], và có các lệnh sau: # returns the length of a list

len(my\_collection)

# Add multiple items to a list

my\_collection.extend([ More", "Items"])

# Add a single item to a list

my\_collection.append("Single")

# Delete the object of a list at a specified index

del(my\_collection[2]

# Concatenate two lists

my\_collection\_2 = ["a", "b", "c"]

my\_collection\_3 = my\_collection + my\_collection\_2

# Calculate the sum of a list of ints or floats

number\_collection = [1,2,3,4.5]

sum(number\_collection)

# Check if an item is in a list, returns Boolean

item in my\_collection

#Check if an item is not in a list, returns Boolean

item not in my\_collection

Tuple: là kiểu dữ liệu giống với list nhưng nó là bất biến, không thể thay dổi giá trị khi đã khai báo

Ví dụ: tuple\_a=(2,3) hoặc tuple\_a=2,3

Khi muốn thay đổi một giá trị trong tuple thì ta phải thay đổi cả tuple, ví dụ thay đổi giá trị đầu tiên trong list thì sẽ là: list\_a[0] = “a”, còn tuple là tuple\_a=(“a”,3), nhưng khi làm như thế dẫn đến hiệu năng kém bởi vì tuple ko thay đổi giá trị mà là sẽ tạo mới một biến tuple khác với địa chỉ khác, nếu hang nghìn lần như thế thì dẫn đến không đủ bộ nhớ

Dictionary: là kiểu dữ liệu gồm key và value

Ví dụ; my\_dictionary = {'banana': 1, 12: 'laptop', (0,0):'center'} các câu lệnh cơ bản trong dictionary là:

# Access value using key

my\_dictionary['banana']

# Get all keys in a dictionary as a list

my\_dictionary.keys()

# Get all values in a dictionary as a list

my\_dictionary.values()

Set: Là kiểu dữ liệu tương tự list nhưng khác với list là không lặp lại các phần tử, có nghĩa là các phần tử trong set là duy nhất, không được trùng nhau

Ví dụ: set\_a={3,4,5,3,6,4,8}, set\_a sẽ tự động laoij bỏ các giá trị trùng nhau, nên set\_a={3,4,5,6,8}, các câu lệnh cơ bản trong set là:

# Convert a list to a set

my\_set = set([1,1,2,3])

# Add an item to the set

Set\_a.add(“hello”)

# Remove an item from a set

Set\_a.remove(4)

# Returns set a minus b, lấy ra các giá trị set a khác với set b

Set\_a.difference(b)

# Returns intersection of set a and b

Set\_a.intersection(b)

# Returns the union of set a and b

Set\_a.union(b)

# Returns True if a is a subset of b, false otherwise

Set\_a.issubset(b)

# Returns True if a is a superset of b, false otherwise

Set\_a.issuperset(b)

# 2.2 Kỹ thuật indexing và slicing

## 2.2.1 Indexing

Cách truy cập vào các, string, mảng list, tuple, set,…

A[0]: phần tử đầu, a[-1]: phần tử cuối, a[0,0] hoặc a[0][0] truy cập phần tử mảng

## 2.2.2 slicing

Cú pháp thông thường cho mảng một chiều như sau: A[start:stop:step], ví dụ:

S = np.array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])

S[2:5] = [2 3 4] (từ index 2 ->4)

S[:4] = [0 1 2 3] (từ index 0 ->3)

S[6:] = [6 7 8 9] (từ index 6 ->hết)

S[:] = [0 1 2 3 4 5 6 7 8 9] (tất cả)

Đối với mảng nhiều chiều, cú pháp sẽ là A[start:stop:step, start:stop:step, start:stop:step], dấu phẩy chính là ngăn cách định nghĩa range của phép slicing.

Ví dụ:

A = np.array([[11,12,13,14,15],

[21,22,23,24,25],

[31,32,33,34,35],

[41,42,43,44,45],

[51,52,53,54,55]])

A[:3,2:] = [[13 14 15] (hàng 0 đến hàng 2, cột 2 đến hết)

[23 24 25]

[33 34 35]]

Làm sao để có được kết quả của toàn bộ 2 hàng cuối cùng của mảng phía trên? Đó là A[-2:,:] hoặc A[3:,:].

# 2.3 Các hàm trong Python

+ Vòng lặp For

Khi biết trước số lần lặp thì nên sử dụng vòng lặp for

for x in range(x):

# lặp x lần x, hoặc range(a,b) thì lặp với x có giá trị từ a đến b-1

Có thể dung else cho các vòng lặp, để áp dụng cho điều kiện vòng lặp đó sai thf chạy lệnh else, ví dụ:

for x in range(x):

{lệnh}

Else:

{lệnh}

for x in iterable:

# lặp lần lượt với x có giá trị là từng phần tử của iterable( string, set, list, tuple,…)

Ví dụ:

# Ví dụ lặp một chuỗi

for letter in "Python":

print ("Chữ cái hiện tại:", letter);

# Ví dụ lặp một mảng

fruits = ['chuối', 'táo', 'xoài']

for fruit in fruits:

print ("Bạn có thích ăn:", fruit);

+ Vòng lặp While:

While( điều kiện):

Câu lệnh

+ Hàm If/elif/else

Ví dụ:

if dieu\_kien1:

Khối lệnh của if

elif dieu\_kien2:

Khối lệnh của elif

else:

Khối lệnh của else

Có thể không cần dung elif hoặc else

+ Try/Except

Ví dụ sử dụng else trong try except, nếu chương trình không có lỗi thì nó sẽ in ra lệnh trong else

x = 5

try:

print(x)

except: # Neu trong Try gay ra loi

print("Ton tai loi chuong trinh!")

else: # Truong hop chuong trinh khong ton tai loi

print("Chuong trinh khong ton tai loi!")

Ví dụ sử dụng finally, nó sẽ chạy lệnh trong finally dù chương trình có xảy ra lỗi hay không:

try:

print(x)

except: # Neu trong Try co loi

print("Bien chua duoc dinh nghia")

finally: # Thuc thi cau lenh du co loi hay khong co loi

print("Hello World")

ví dụ in ra tên lỗi để biết nó là thuộc dạng lỗi gì:

try:

print(x)

except Exception as e:

print("e")

ví dụ sử dụng chính xác lỗi mình đã biết:

try:

# code

except exceptionName:

# code

Exception Name bao gồm: ZeroDivisionError, RuntimeError, OverflowError, FloatingPointError, AttributeError, ImportError, …

# 2.4 Tạo hàm trong Python

def function\_name(optional\_parameter\_1, optional\_prameter\_2):

# code to execute

return optional\_output

ví dụ:

def chao(ten):

"""Hàm này dùng để chào một người,

tên được truyền vào như một tham số"""

print("Chào bạn " + ten + ". Chúc một ngày vui vẻ!")

# 2.5 Đọc file trong Python

Cấu trúc đọc file trong Python như sau:

File1=open(“url\_file”, “mode”)

Trong đó url\_file là đường dẫn đến file, nếu file nằm cùng thư mục trong file code thì chỉ cần tên\_file.đuôi\_file

Mode là các chức năng thao tác với file, “r” là read, “w” là viết đè lên file, “a” là them vào file, không ghi đè lên các data cũ như “w”, “x” là tạo 1 file mới, và một số mode khác

Luôn đóng file khi đã sử dụng xong file, dung câu lệnh sau để đóng file: file1.close

Để tránh dính lỗi quên đóng file sau khi sử dụng thì khuyến khích dùng câu lệnh:

with open(“url\_file”, “mode”) as File1

file\_content=file1.readlines()

print(file\_content)

print(file1.closed) #kiểm tra lại file đã đóng chưa

file1.read(): đọc và lưu data trên 1 dòng

file1.readlines(): đọc data theo từng dòng và lưu từng dòng

file1\_readlines(x): đọc data theo x dòng và lưu x dòng

file\_content[0] :lấy dữ liệu dòng đầu tiên

Cách sao chép 1 file vào file khác:

With open(“url\_file1”,”r”) as file\_read:

With open("url\_file2”,”w”) as file\_write:

For line in file\_read :

File\_write.write(line)

Cách đọc file tiếng Việt:

File= open(“url\_file”,mode=”r”, encoding=”utf-8”)

# 2.6 Các thư viện phổ biến trong Python

+ Beautifulsoup

Web crawling là quá trình tự động trích xuất các thông tin từ các trang web và lưu trữ nó dưới một định dạng phù hợp. Chương trình mà thực hiện công việc này gọi là web crawler. Thư viện BeautifulSoup là một thư viện của Python cho phép chúng ta lấy dữ liệu từ HTML đơn giản và hiệu quả.

Đầu tiên cần import thư viện:

from bs4 import BeautifulSoup

import urllib.request

url = 'https://vnexpress.net' #trang web cần lấy dữ liệu

page = urllib.request.urlopen(url)

soup = BeautifulSoup(page, 'html.parser')

soup.prettify() : trả về kiểu data html

soup.find(tag): tìm phần tử đầu tiên xuất hiện có tag như yêu cầu

soup.find\_all(tag): tìm tất cả phần tử có tag như yêu cầu

….

+ Request

Requests module là một thư viện hỗ trợ chúng ta có thể gửi bất kỳ một loại request HTTP nào một cách đơn giản nhất.

Cú pháp:

requests.method(url, params, data, json, headers, cookies, files, auth, timeout, allow\_redirects, proxies, verify, stream, cert)

trong đó method là 1 trong các method sau: get, post, put, patch, delete, options

các tham số có thể bỏ trống, trừ URL bắt buộc có.

Ví dụ:

import requests

r= requests.get("https://toidicode.com/")

#các lệnh dưới dây là in ra kết quả nếu hàm requests thành công, nên sử dụng try except để kiểm tra các except có thể xảy ra trong quá trình request

print(r.apparent\_encoding), print(r.content), print(r.encoding), print(r.cookies), print(r.elapsed), print(r.encoding), print(r.headers), print(r.history), print(r.is\_permanent\_redirect), print(r.iter\_content()), print(r.links), print(r.raise\_for\_status()), print(r.json())

+ Pandas

Là thư viện dùng để thao tác với nhiều loại file khác nhau như csv, xls, dữ lieuj ma trận, dữ liệu thời gian cố định, không cố định

Để sử dụng thư viện pandas ta cần cài đặt và import

Pip install pandas

import pandas as pd

Khi đọc file thì mặc định thư viện pandas(pd) sẽ trả về kiểu datafram (df) là kiểu dữ liệu theo hang và cột.

Đoc file csv: peoples\_df = pd.read\_csv('./people.csv')

In ra n dòng đầu tiên của file, măc định để trống thì sẽ là 5: peoples\_df.head(n)

# Xem chiều dài của df, tương đương shape[0]

print('Len:', len(peoples\_df))

# Xem thông tin dataframe vừa đọc được

peoples\_df.info()

# Xem kích thước của dataframe

print('Shape:', peoples\_df.shape)

# Lấy thông tin của 1 cột hoặc nhiều cột

peoples\_df['name'] hoặc peoples\_df[['name', 'age']].head(5)

#Lấy thông tin theo điều kiện

professor = peoples\_df[peoples\_df.title == 'Professor']

professor.head(5)

#Lấy thông tin theo theo kiểu list chỉ cần them .values

peoples\_df['name'].values

#Tạo cột mới với dữ liệu birthday được tạo random từ hàm

df\_len = len(peoples\_df)

birthday = [random.randrange(1980, 2000, 1) for i in range(df\_len)]

peoples\_df['birthday'] = birthday

#Tạo cột mới với dữ liệu age đã có sẵn và kết quả của cột mới này là true/false

peoples\_df['is\_young'] = peoples\_df['age'] < 25

#Tạo cột mới với dữ liệu bỏ trống

peoples\_df['new\_column'] = None

#Xóa cột trong df

peoples\_df.drop('tên cột cần xóa', axis=1) # Xóa 1 cột

peoples\_df.drop(['cột 1', 'cột 2'], axis=1) # Xóa nhiều cột

df.drop(columns=['B', 'C']) # Xóa các cột có tên là B và C

#Tạo dataframe mới bằng dictionary với key là tên cột, value là các giá trị của hàng, lưu ý các kích thước của key và values phải hợp lý với nhau

peoples = {'name': ['Nguyễn Văn Hiếu', 'Hiếu Nguyễn Văn'], 'age': [28, 28], 'website': ['https://blog.luyencode.net', None]}

df = pd.DataFrame(peoples)

print(df)

#Sắp xếp các dataframe tăng dần theo cột “age”

df = pd.DataFrame({'name': ['Nam', 'Hiếu', 'Mai', 'Hoa'], 'age': [18,18,17,19]})

print('Before sort\n', df)

df = df.sort\_values('age', ascending=True)

print('After sort\n', df)

#Lưu file csv vào máy

df.to\_csv('comments.csv')

+Numpy

Là một thư viện toán học phổ biến và mạnh mẽ của Python. Cho phép làm việc hiệu quả với ma trận và mảng, đặc biệt là dữ liệu ma trận và mảng lớn với tốc độ xử lý nhanh hơn nhiều lần

Để cài đặt numpy ta thực hiện lệnh: pip install numpy

Để sử dụng numpy ta import vào chương trình

Import numpy as np

Khởi tạo mảng một chiều với kiểu dữ liệu mặc định hoặc mình có thể tự định nghĩa

arr = np.array([1,3,4,5,6], dtype = int)

arr = np.array([1,3,4,5,6])

Khởi tạo mảng 2 chiều

arr1 = np.array([(4,5,6), (1,2,3)], dtype = int)

Khởi tạo với các hàm có sẵn

np.zeros((3,4), dtype = int): Tạo mảng hai chiều các phần tử 0 với kích thước 3x4.

np.ones((2,3,4), dtype = int): Tạo mảng 3 chiều các phần tử 1 với kích thước 2x3x4.

np.arange(1,7,2): Tạo mảng với các phần tử từ 1 - 6 với bước nhảy là 2.

np.full((2,3),5): Tạo mảng 2 chiều các phần tử 5 với kích thước 2x3.

np.eye(4, dtype=int): Tạo ma trận đơn vị với kích thước là 4x4.

np.random.random((2,3)): Tạo ma trận các phần tử ngẫu nhiên với kích thước 2x3.

Thao tác với mảng

dtype: Kiểu dữ liệu của phần tử trong mảng.

shape: Kích thước của mảng.

size: Số phần tử trong mảng.

ndim: Số chiều của mảng.

ví dụ:

array.dtype

array.size

arr.max() hoặc np.max(arr): Lấy giá trị lớn nhất của mảng arr.

arr.min() hoặc np.min(arr): Lấy giá trị nhỏ nhất của mảng arr.

arr.sum() hoặc np.sum(arr): Tổng tất cả các phần tử trong mảng arr.

arr.mean() hoặc np.mean(arr): Trung bình cộng của tất cả các phần tử trong mảng arr.

np.median(arr): Trả về giá trị trung vị của mảng arr.

# 2.7 OOP trong Python

Cấu trúc của class:

class ClassName:

# tập lệnh

Ví dụ:

class Employee:

id = 10

name = "The Mac"

def display (self):

print(self.id, self.name)

self ở đây luôn là đối số đầu tiên trong hàm, dù nó có được sử dụng hay không, self sẽ trỏ đến đối tượng hiện tại

Một số hàm có sẵn cho class:

getattr(obj,name): trả về name của đối tượng obj

setattr(obj, name,value): gán value cho thuộc tính name của đối tượng obj

delattr(obj,name): xóa thuộc tính name của đối tượng obj

hasattr(obj, name): trả về true/false nếu thuộc tính name xuất hiện trong đối tượng obj

cách để khai báo public, protected, private cho các thuộc tính và hàm:

với public thì có thể truy cập được ở mọi nơi, nên cách khai báo như hàm bình thường

Với protected thì chỉ lớp con có thể truy cập được, cách khai báo bằng cách bắt đầu bằng một dấu gạch ngang "\_", VD: \_age.

Với private thì chỉ class đó có quyền truy cập, cách khai báo bằng cách bắt đầu bằng 2 dấu gạch ngang "\_\_", VD: \_\_age.

Tương tự có thể áp dụng cho hàm. Ví dụ def \_getName(self): …..

Kế thừa trong Python:

class Dog(Animal):

class Dog đã kế thừa lớp Animal

Đối với đa kế thừa ta làm như sau:

class Dog(Animal,Fly):

class Dog đã kế thừa 2 class đó là Animal và Fly

OOP trong Python cũng có các tính chất như bao ngôn ngữ lập trình khác

CHƯƠNG III. DATABASE