* Pandas *

↳ is a PowerFull tool For data analysis and Manipulation

⇐ أداة قوية لتحليل البيانات ومعالجتها

⇐ مبنية على numpy لا مهي سريعة وفعالة

⇒ two main data structure in Pandas ⇐

Pandas series | Pandas dataFrame

[1] Pandas series

```
import Pandas as Pd   → alias
```

⇐ انشاء Pandas series

```
nameof object = Pd.Series()
```

كائنات بيانية مصفوفة احادية البعد يمكن تخزين العديد من انواع البيانات

⇐ لاحظ انها عكس مصفوفات numpy تخزن نوع واحد فقط

```
nameof object = Pd.series (data=[ , , ...], index=[ , ...])
```

output ⇒ index data

⇐ خصائص ال Pandas series

```
nameofobject.shape    (row , col)
            .ndim     (number of dimension)
            .size     (number of elements)
            .index    (index)
            .value    (data value)
```

```
index in nameof object ⇒ True → لو ال index موجود
                         False → ال " مش موجود
```

⇒ Accessing data

① nameofobject[index]
② " [numberofpendex]
③ " " [index1, index2, ...]
④ " " [[list of index]]
⑤ " " [-1] ⇝ آخر عنصر
⑥ " " .loc[numberof index] ⇒ مثلاً رقم
⑦ " " .iloc[numberof index] ⇒ ارقام

⇐ لاحظ انا نقدر نغير ونعدل في الـ data اللي انشأناها بكل انواع الـ accessing

⇐ نقدر نحذف عناصر منها nameofobject.drop(index)

لاحظ يحذف من السلسلة الأصلية لو عايزة يحذف هناك صلبة
nameofobject.drop(index, inplace = True)

⇐ يمكنا كمل عمليات حسابية في Pandas series

⇒ Arithmetic operation between Pandas series and single number

nameofobject + − * / number

⇐ يتم تطبيقها على جميع عناصر الـ Pandas series

import numpy as np

⇐ نقدر نستخدم دوال numpy

np.sqrt(nameof object)
.exp( )
.Power(, number)

nameof object[index] + − * / number

⇐ لو عايز تطبق العملية على عنصر واحد نستخدم الـ index

⇐ لاحظ انه اجراء العمليات الحسابية في انواع مختلفة من البيانات لو مش نوع العدد الحسابية دي هيطلع error

[2] data structure in Pandas "Dataframe"

ـ تمكانك تنائي الابعاد مع تسميات للصفوف والاعمدة

ـ يمكن تخزين انواع مختلفة ومعقدة ، ممكن نتخيلها انها جدول بيانات قوي للغاية

ـ يمكن انشاء dataframe او تحميل البيانات من ملف

[1] Creating a data frame manually from a dictionary

ex

```
import pandas as pd
items = { 'Bob': pd.series([245, 25, 55], index=['bike',
         'pants', 'watches']),
          "Alice": pd.series([40, 110, 500, 45], index=['book',
          'glasses', 'bike', 'pants']) }
shoping_frame = pd.DataFrame(items)
```

⟸ يتم عرضها في شكل جدول الـ index هو الصفوف والـ key بتاعة الـ dict هي الاعمدة والـ value بتاعة الـ pd هي الـ data

⟸ ملاحظة يتم ترتيب الاعمدة ابجديا

⟸ ملاحظة ان لو معملناش index هو هيعمل لهم كده لوحده

NaN ⟸ عنصر ليس له قيمة

nameofobject.index → row

key . columns → value of key "pd.series → data"

.values

.shape → (row, columns)

.ndim → number of dimension

.size → " elements

⟸ لقص جزء من dataframe عناصر محددة

```
bob_shopping_cart = pd.Dataframe(items, columns=['Bob'])
                  = pd.DataFrame( " , index = ['pants', 'book'])
pd.Dataframe(item, index=[, ...], columns=[ , , ])
```

ex

```
data = { 'integers' : [1,2,3] , 'float' : [4.5, 8.1, 9.6] }
df = pd.DataFrame(data, index=['label 1', 'label 2', 'label 3'])
```

↓ لازم الحجم نفس عدد العناصر

ex

```
items = [{ 'bike' : 20 , 'pants' : 30 , 'watches' : 35 },
         { 'watches' : 10, 'glasses' : 50, 'bikes' : 15, 'pants' : 5 }]

store_items = pd.DataFrame(items)
```

⇒ output

	bikes	glasses	pants	watches
0	20	NaN	30	35
1	15	50.0	5	10

or

```
store_items = pd.DataFrame(items, index=['store 1', 'store 2'])
```

⇒ output

	bikes	glasses	pants	watches
store 1	20	NaN	30	35
store 2	15	50.0	5	10

⇒ Accessing element in DataFrame

nameofobject [name of column] ⇐ تقدر توصل للعمود كامل

" [[name of col1 , ———]]

nameofobject.loc [[name of row , ...]] ⇐ تقدر توصل للصف كامل

⇐ تقدر توصل لعنصر معين باستخدام الصف والعمود

nameofobject [name of col] [name of row]

⇐ إضافة عمود جديد

nameofobject [name of new column] = [value of row1 , value of row2 , ———]

⇐ تقدر تعمل عمود جديد ناتج من عملية حسابية من أعمدة أخرى

nameof object [name of new column] = nameof object [name of col] + - * / nameof object [name of col]

⟸ لتقدر تضيف صف في dataframe

ex new_items = [{'bikes': 20, 'pants': 30, 'watches': 35, 'glasses': 4}]

new_store = pd.DataFrame(new_items, index = ['store 3'])

store_items = store_items.append(new_store)

⟸ لو عايز هنا يتم فرز الاعمدة الجديدة

nameofobject.insert(location "index number" , nameof Column "data" , value of each row)

" .pop(nameof Column) ⟹ لحذف عمود كامل

" .drop([Col,]), axis = n)

0 → لو هيحذف صف

1 → لو هيحذف عمود

⟸ لتغير اسم العمود nameofobject.rename(Columns = { 'nameof current' : 'new name' })

⟸ لتغير اسم الصف " " (index = { 'nameof current' : 'new name' })

⟸ قبل العمل في تحليل البيانات او تدريب خوارزمية التعلم للبيانات، نحتاج الى مرحلة لتنظيف البيانات اي عملية ايجاد طريقة لمعالجة الاخطاء في البيانات وتصحيحها

nameofobject.isnull() → NaN → True ; else → False

" .isnull().sum().sum() → عدد الـ NaN

" .Count() ⟹ عدد الكل عشان اعرف كل عمود "كم عنصر في كل عمود"

⟸ لمعالجة الـ NaN لحذف اي اي سيناريو

nameofobject.dropna(axis = n)

0 → هيحذف اي صف فيه NaN

1 → هيحذف اي عمود فيه NaN

⟸ ممكن نعالج NON بالاستبدال

nameofobject . fillna (n)

↳ للاستبدال برقم "n"

" . fillna (method = " " , axis = n)

ffill — يستبدلها بقيمة التي قبلها
backfill — يستبدلها بالتي بعدها
linear

0 — الفحص العمود
1 — الفحص الصف

⟸ عند العمل مع البيانات مرجح ان تستخدم قواعد بيانات من مصادر مختلفة
⟸ تسمح لنا Pandos بتحميل قواعد البيانات بتنسيقات مختلفة في DataFrame
⟸ تنسيق البيانات الاكثر شيوعا هو CSV ، هو تنسيق مقبول لغاية
⟸ نقدر نفتح ملف CSV باستخدام read_CSV

nameofobject = pd . read_CSV(relative path)

" . head(n) → عدد الصفوف من الاول
default = 5

" . tail(n) → عدد الصفوف من اسفل
default = 5

" . isnull() . any() → هل العمود فيه NON

" . describe() → يرجع احصائيات وصفية لكل عمود من ال dataFrame

" [nameofcolumn] . describe()
↳ نطبقها على عمود واحد او اكثر من عمود

" . max()
. min()
. corr() → ارتباط كل عمود
" . groupby([nameofcolumn]) [nameof column] . sum()
mean()
↳ نقوم بعمل عملية على البيانات تبعا لـ