Task 9:

[I] Numpy For machine Learning

Numpy is a kind of arrays.
We need this when we start to work with massive amount of data so we use numpy array

→ all data in numpy array must have The Same datatype
it can be nd dimentional array.

```
np1 = numpy.array([ ])    ← data
np1.shape   →  numbes of items
```

→ `numpy.arange(10)` ← have numbers froms 0 → 9 (10 items)

→ `numpy.arange(0, 10, 2)`   [0-2-4-6-8]
(start, end, step)

→ `numpy.zeros(10)` → have 10 zeros [0.] [0. 0. 0. 0.]

Multi dimentional zeros

→ `numpy.zeros((2, 10))`   `numpy.ones((2,3))`
(2 → n dimentional, 10 → عدد العناصر في كل dimentional)

Full

→ `numpy.full((10), 6)`   [6 6 6 6 6 ...]
(10 → عدد العناصر, 6 → الرقم اللي هتحطه)

Multidimentional Full
→ `numpy.Full((2, 10), 6)`

to get number in array
`np1[0]`

Convert from Python List to numpy
→ `numpy.array(my_list)`

② Slicing Numpy Arrays

np1 = np.array([1,2,3,4,5,6,7,8,9])

How to return only 2,3,4,5

→ Print(np1[1:5]) من العنصر 1 إلى العنصر 4
doesn't include

From Something to the end

→ Print(np1[3:]) Print 4-9

Negative Slices

Print(np1[-3:-1]) "7,8"
تبدأ من اليمين

Steps

Print(np1[1:5:2]) [2,4]
Step

Steps with entire array

np[::2] [1 3 5 7 9]
Step

Slice 2D array

np2 = np.array([[1,2,3,4,5],[6,7,8,9,10]])

np2[1,2] → 8

np2[0:1,1:3] → [[2,3]]

np[0:2,1:3] [[2,3]
[7,8]]

We have documentation to help called (numpy universal Functions) SciPy.org

[3] ~~Numpy universal Functions~~

```
np1 = np.arange(10)
```

① # square root of each element

```
print(np.sqrt(np1))
```

② # Absolute value

```
np.absolute(np1)
```

③ # Exponential

```
np.exp(np1)
```

④ # Min/Max

```
np.max(np1)
np.min(np1)
```

⑤ # Sign

```
np.sign(np1)
```

it returns -1 → for negative, 1 → for positive, 0 → for zero

Trig sin / cos / log

```
np.sin(np1)
np.log(np1)
```

→ returns nan for not a number

[4] Numpy copy vs view

```
np1 = np.arange(5)
np2 = np1.view()
print(np1)
print(np2)
```

prints the same output

if use `np1[0] = 41` هيغير np1 ، np2

original changing in the viewed array.

Create a copy ((copy not connected to the original))

```
np3 = np1.copy()
print(np1)
print(np2)
np1[0] = 41
```

نفس الحاجه ← print

على عكس view الاتنين مرتبطين ببعض لو غيرت في اي واحد التاني هيتغير

ده هنا هيغير في np1 بس ← np1[0] = 41

np3 لا لانها مستقله بنفسها

3

5 Shape and reshape

```
#create 1-D and Get Shape.
np1 = np.array([1,2,3,4,5,6,7,8,9,10,11,12])
np1.Shap  → (12,)

#Create 2-D and get Shape.
np2 = np.array([[1,2,3,4,5,6],[7,8,9,10,11,12]])
np2.Shape → (2,6)

#Reshape 2-D
np3 = np1.reshape(3,4)   [[1, 2  3 4]
                          [5 6   7 8]
                          [9 10  11 12]]

#Reshape 3-D
np4 = np1.reshape(2,3,2)  [[[1  2]
                           [3  4]
                           [5  6]]
                          [[7  8]
                           [9  10]
                           [11 12]]]

# Flatten to 1-D
np5 = np4.reshape(-1)
```

(6) iterating Trough numpy Arrays

```
np1 = np.array([1,2,3,4,5,6,7,8,9,10])
  for x in np1:
    print(x)

# 2-D
np2 = np.array([[1,2,3,4,5],[6,7,8,9,10]])
  for x in np2:
  #print rows:        [1 2 3 4 5]
    print(x)  ->      [6 7 8 9 10]
      for y in x:
        print(y)  ->  1
                      2
                      3
                      :

#3D Array
np3 = np.array([[[1,2,3],[4,5,6]],[[7,8,9],[10,11,12]]])
  for x in np3:
    print(x)  <- print every row
    for y in x:
      print(x)        print every row
      for z in y:
        print(z)  -> each item

# use np.nditer()                 بدل ما اكتر العدد loops كتير
  for x in np.nditer(np3):
     print(x)  -> each item
```

⑦ Sorting numpy Array

```
np1 = np.array([6,7,4,9,0,2,1])
np.sort(np1) → هيرتب
```

ALphabetical.

```
np2 = np.array(["John", "Tina", "Aaron", "Zed"])
                  2       3       1       4
np.sort(np2)
```

#Booleans T/F

```
np3 = np.array([True, False, False, True])
np.sort(np3) ← From False to Trues
               0           1
```

↑ when using np.sort it doesn't chang the origiond

2D Sort كل صف هيرتب لوحده مش هيتخلطوا ببعض

⑧ Searching Numpy

```
np1 = np.array[1,2,3,4,5,6,7,8,9,10])

x = np.where(np1=3)  return index has 3 item 3
     ~~(2)~~
```

واحد او اكتر من واحد

```
X → (array[2], dtype=int64),)
```

ده بيوتر Tuple.

```
X[0] → [2]
```

we call first item of Tuple.

```
np1[X[0]] ← return [3]
```

return even items

```
y = np.where(np1 % 2 == 0) ← return even numbers!
```

numbers have even index

~~not even~~ n

even number يعني ارقام

8) Filtring numpy arrays

```
npi=np.array([1,2,3,4,5,6,7,8,9,10])
x=[True, True, False, False, False, False, False, False, False, False]
```

Print(npi[x]) ⟶ [1 2]

```
#Filtered list
Filterd = [ ]
For thing in npi:
    if thing%2==0:
      Filterd.appead(True)
    else:
      Filterd.append(False)
```

npi[Filtered] → [2, 4 6 8 10]

```
# Shurtcut!!

Filtered = npi%2==0
npi[Filtered] → return
```

→ to get dimentional → np1.ndim

→ To Get Type → np1.dtype → 'int64'

→ Get Size → a.itemsize → 8 ← byte.

→ Get total size → np1. Size → سالعناصر.

→ To get a single row a[0, :]

→ To get a specific Coloumn a[:, 2]

np.full_like (a, 4) نفس شكل a ملانه بـ 4

#Random decimal.
np.random.rand (4,2)
np.random.randint(7, size =(3,3))

→ #repeat array.
np.repeat(np1,3)

Mathematics
np a + 2 ← يضيع ٢ على كل العناصر

#Linear Algebra.
a = np.full((2,3), 1)
b = np.full(3,2), 2)

np.matmul(a,b)

#determinant of Matrix
C = np.identity(3)
np.Linalg.det(C) → 1.0

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

~~statistics~~

stats = np.array([[1, 2, 3], [4, 5, 6]])

np.min(stats)   np.sum(stats, axis=1)

np.max(stats)   هيجمع كل صف لوحده

np.vstack([np1, np2])   هيطلع عندي 2D array

كل array من اللي معايا تمثل صف

np.hstack([np1, np2])   زي بالضبط

كل واحده يا اما عمود

Load data from file

file = np.genfromtxt('data.txt', delimiter=',')

file.astype('int32')

Advanced indexing. Boolean Masking.

~ ← not

(File date > 50) → هيطلع نفس حجم array Filedate

كلها False ما عدا الارقام اللي اكبر من 50 هتبقى True.