Python basics:

number 1 = 20
number 2 = 30
print(number1 * number2)

N.B: number1 ** number2 → 10^3

List:

numbers = [ ]
numbers.append(1) ⟵ what i add to the list
" .append("String")

input:

number1 = int(input("Enter the first nb"))
if(Number1 > Number2):
print(Number1, "is greater than", number2)

Loop:

For i in range(5): → 0→5
print(i)

For i in range(1,6): → 1→5
print(i)

raw_input:

takes what is typed from the Keyboard ⟶ string.

g = raw_input("Enter your name")
print(g)

multiple input:

x, y = input("Enter 2 values:").split() the number is entered as a string
print(x)
print(y)

⟹ x = input('Enter nb')
S = 1 + int(x)

Multiple input at a time:

add type casting list() function

x = list(map(int, input("Enter multiple values").split()))
print(x)

end parameter:

print(x, end=" ")

1 ␣ 2
end space

```
x, y, z = [1, 2, 3]
print(x)  #1
print(y)  #2
print(z)  #3
```

Unpack collection:

```
S = [1, 2, 3]
x, y, z = S   <== btsewa ha
```

```
x = S[0]
y = S[1]
z = S[2]
```

```
print(x)
print(y)
print(z)
```

Length:

```
a = "Hello, world"
print(len(a))
print(a[1])   ← STRINGS are ARRAYS
print(a[1:5])
For i in "Hello, world":
    print(i)
```

```
print(a.upper())
print(a.replace("H", "J"))
```

```
print(a.split(","))
#['Hello', ' world']
print(a.strip())  → remove spacij
```

Formal:

```
name = Input("What is your name?")
print("Hello my Friend {}".format(name))
or print(f"hello my friend {name}")
```

Dictionaries:

```
Dict= {
"name": "Sarah",
'age' : 30, 3
print(Dict.get('age"))
print(Dict)
```

~~custor~~

ubers = I can import math and i'll get many F(.

```
2.9
int(round(x))

y = -1
print(abs(y)) # absolute value
```

if Statement:

```
if :
elif :
else :

if has_high_income and has_good_credit:
    print("he can buy it")

"  "  Or  "  "  :

if has_high_income and not has good credit :
```

For Loop:

```
                from  to  scale 2
                  ↓   ↓   ↓
for i in range(5, 10, 2):

for i in range(len(list)): # list = ["apple", "banana", "cherry"]
    print(list[i])

OR  while (i < len(list))
      print(list[i])
      i++
```

Nested loop:

```
for x in range(4):
    for y in range(3):
        print(f'({x}, {y})')
```

(0,1)
(0,2)
(0,3)

(1,1)
(1,2)
(1,3)

(2,1)
(2,2)
(2,3)

(3,1)
(3,2)
(3,3)

(4,1)
(4,2)
(4,3)

Lists:

```
List = [1, 2, 3]
List . append (20)
List . insert (0, 10)
List . remove (8)
List . clear ( )
List . index (3)
print( 15 in list )   # return false
List . sort ( )                    List . sort (reverse = True)
print ( list )                                  ↑ sort bl elab
```

(tuples are similar to lists bout we cannot use methods on the)

Add numbers from a list to another

```
nb = [2, 2, 4, 6, 3, 4, 6, 1]
add = [ ]
for n in nb :
    if n not in add :
        add . append ( n )
```

or `add = [n for n not in add]`

Unpacking: `newlist = [x for x in range (10)]`

def __init__ (self, max_speed , mileage, capacity):

Self . max_speed = max_speed
Self . mileage = mileage

V_1 = Vehicle (5, 8)

def

class bus (Vehicle):

def __init__ (self, max_speed, mileage, capacity=50)

Super ().__init__ (max_speed , mileage)

Self . capacity = capacity.

b1 = bus (23, 26, 52)

print (b1 . max_speed, b1 . mileage, b1 . capacity)

def fare(self): is currently used object

rehurn self . capacity * 100

to call a function from superclass

def fare (self): ← makes it self fct

amount = super (). fare ()

amount = amount + amount * 0.1

return Amount.

to check to which type an object belongs

print (type (b1))

is also an instance of Vehicle : print (isinstance (b1 , vehicle))

```
import numpy as np
arr = np.array([1,2,3,4,5])
print(arr)
print(np._version_.)
print(type(arr))   # numpy.ndarray
```

to create an nd array we can use any list, tuple ... and Covert it into an ndarray:

```
arr = np.array((1,2,3,4,5))
```

Nested arrays : arrays that have arrays as element.

0 Δ : array

```
arr = np.array(42)
[42]
```

1 Δ

```
arr = np.array([1,2,3])
[1 2 3]
```

2 Δ

```
arr = np.array([1,2,3],[4,5,6
[ [1 2 3]
  [4 5 6] ]
```

3 Δ

```
arr = np.array([1,2,3],[45,6];[7,8,1])
[[ 1  2  3]
 [ 4  5  6]
 [ 7  8  9]]
```

```
print(arr.ndim) #dimension of the array
arr = np.array([1,2,3,4], ndmin=5)  # ndmin: precise the dimensio
print(arr.ndim)
```

[[[[[1,2,3,4]]]]]

print (arr[0])
print (arr[1] + arr[2])
print (arr[0, 1]) # In 2 d array ([1,2,3], [4,5,6]) 'npu mber

0 1 2 0 1 2

print (arr[0, 1, 2]) # array([[1,2,3], [4,5,6]],[[7,8,9],[10,11,12]]

6 0 1

print (arr[-1]) # last element in 1D array
print (arr[1:5]) # from 1 to 5 (5 not included)
" (" [4:]) # From 4 to the end
" (" [-3:-1]) # " -3 to -1 (-1 not included)
" (" [1:5:2]) # from 1 to 5 return every other element
" (" [::2]) # every other element (1 yes 1 no)
" (" [1, 1:4]) # 2D array second array from 1 to 4
array element
" (" [0:2, 2]) # From both arrays print element of index 2
arrays element
" (" [0:2, 1:4]) # from both arrays print " from 1 to 4.

Data types: i, b, u (unsigned integer), f, c, m (time delta),
M (datetime), O (object), S, U (unicode string), V (void)

arr = np.array ([1,2,3])
print(arr.dtype)

3 bytes integer

arr = np. array ([1,2,3], dtype = 'i3')
new arr = arr. astype ('f') # ('i') or (int)

arr= np.array ([1,2,3])

x = arr. copy ()
arr[0]=42 #arr changes
print(x) #x does not changes

y = arr.view ()
arr[0]=42
print (arr)
print(y) # y changes. y[0]=42

(y.base) # does not return none

print(arr.shape) # (2, 4)
↓ ↓
rows cols

x: arr = np.array([1,2,3,4], ndimin=5)
print(arr.shape) # (1,1,1,1,4)

arr = np.array([1,2,3,4,5,6,7,8,9,10,11,12])

newarr = arr.reshape(4,3)
↓ ↓
rows cols

```
[1 2 3]
[4 5 6]
[7 8 9]
[10 11 12]
```

newarr2 = arr.reshape(2,3,2)
↓ ↓ ↓
2arr each 3arr 2 elements

```
[ [1,2]
  [3,4]
  [5,6] ]

[ [7,8]
  [9,10]
  [11,12] ]
```

arr = np.array([1,2,3,4,5,6,7,8])
print(arr.reshape(2,4).base) # view
↳ to check if the reshaped array is a copy or a view

If (arr.reshape(3,3)) ⇒ error.

Unknown dimension:

arr = np.array([1,2,3,4,5,6,7,8])
new arr = arr.reshape(2,2,-1)
↑
Numpy calculate the number for you.

```
[ [ [1 2]
    [3 4] ]

  [ [5 6]
    [7 8] ] ]
```

multidimensional → 1D ⇒ arr.reshape(-1)

```
arr = np.array([1,2,3])
for x in arr:
print(x)
```

2d
```
for x in arr:
    for y in x:
        print(y)
```
] Ek ghotele yehon ka elements nahet ba3ed mnh ka 2 arrays fyon elements.

3d
```
for x in arr:
    for y in x:
        for z in y:
print(z)
```

OR

```
For x in np.nditer(arr):
print(x)
```

```
For x in np.nditer(arr, flags=['buffered'], op_types=['S'])
    print x
```
↑ extra space

b'1'
b'2'
b'3'

```
Arr = np.array([1,2,3,4],[5,6,7,8])
For x in np.nditer(arr[:, ::2]);
  print(x)
```
both arrays

1 3 5 7

= np.array([1,2,3])

or idx, x in np.ndenumerate(arr):
    print(idx, x)

(0) 1
(1) 2
(2) 3

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

For idx, x in np.ndenumerate(arr):
    print(idx, x)

Join 2 arrays:

arr = np.concatenate((arr1, arr2)) [1 2 3 4 5 6]

arr = " ((" , ") axis = 1)

↑ Join 2D arrays along rows,6])

ex arr1 = np.array([[1,2],[3,4]])
arr2 = " ([[5,6],[7,8]])

[1,2,5,6] ~~[3,4,7,8]~~
[3 4 7 8]

Concatenate

ex: arr1 = np.array([1,2,3]) axis = 1
arr2 = " ([4,5,6])

[1 4]
[2 5]
[3 6]

arr = hp.hstack((arr1,arr2))
print(arr)

↳ stack along rows
[1 2 3 4 5 6]

arr = VStack((arr1, arr2))

↳ stack along columns

arr = np.dstack(arr1,arr2)
↑ along height

[[1 2 3]
 [4 5 6]]

[1 4]
[2 5]
[3 6]

```
arr= np.array ([1,2,3,4,5,6])
newarr = np.array_split (arr,3)
  print( newarr)
```

number of splits

if it was 4

1 2
3 4
5
6

```
arr= np.array ([[1,2,3],[4,5,6],[7,8,9],[10,11,12],[13,14,15],[16,1
newarr = np.array_split ( arr, 3, axis=1)
```

array [[1] [4] [7] [10] [13] [16]]

array [[2] [5] [8] [11] [14] [17]]

array [[3] [6] [9] [12] [15] [18]]