

Andocs

Day 1-2-3

22-07-2024

23-07-2024

24-07-2024

## Full History of Python Language (Founder, Release Date, All Versions)

### What is Python Programming Language?

Python is a high-level programming computer language that provides instructions to teach the computer how to perform a task.

It offers efficient high-level data structures and an object-oriented programming style that is simple yet effective.

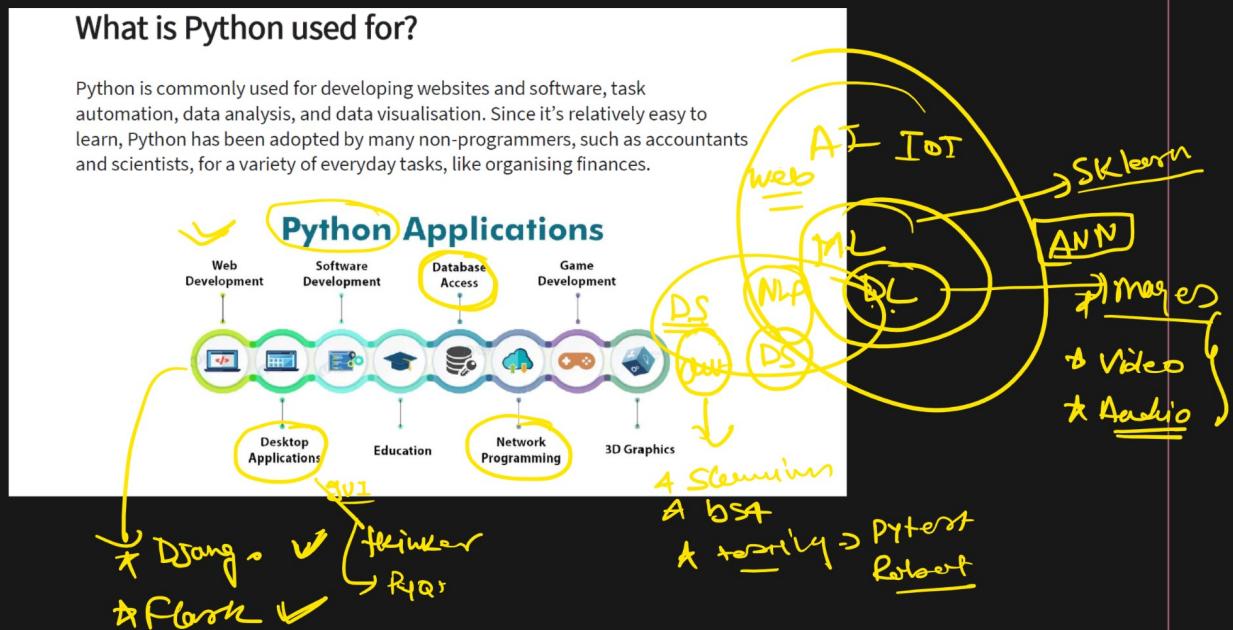
Python, a high-grade language, is a computer programming language that is meant to represent the needs of a problem and mimics natural language or mathematical notation.

It is a free and open-source language.



### What is Python used for?

Python is commonly used for developing websites and software, task automation, data analysis, and data visualisation. Since it's relatively easy to learn, Python has been adopted by many non-programmers, such as accountants and scientists, for a variety of everyday tasks, like organising finances.



### Fun Fact:

Guido van Rossum was reading the BBC's Monty Python's Flying Circus when developing Python. He named this language after Python, where he thinks the length is just right with a subtle mystery.

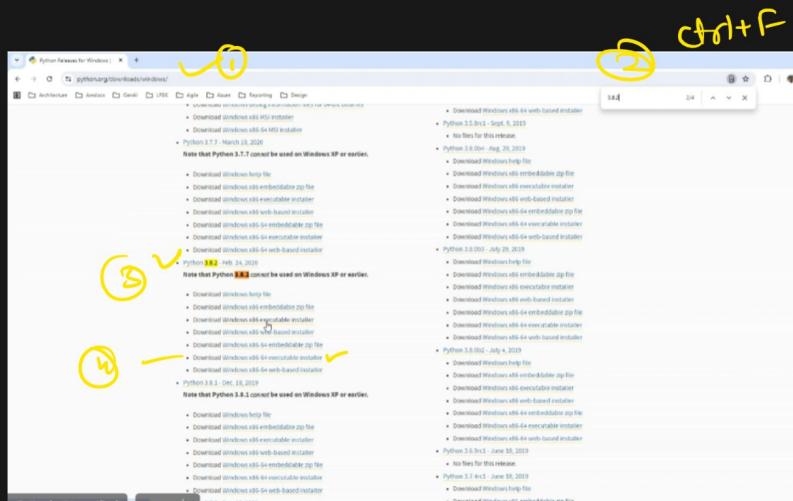
### Who Developed Python Programming Language?

Python was created by Guido van Rossum, a Dutch programmer. He started working on Python in the late 1980s, and the first official release, Python 0.9.0, came out in February 1991.

Obviously, when talking about the history of Python language, the first question that arises is who developed Python. And, Guido van Rossum is known as the founder of Python.

Guido studied mathematics and computer science at the University of Amsterdam. His early exposure to programming languages like ABC influenced the development of Python.

He began working on Python in the late 1980s while working at the Centrum Wiskunde & Informatica (CWI) in the Netherlands.



### Env. Path's

① C:\Python38



Python.exe

② C:\Python38\Scripts



Pip.exe

L3.P.P

own file

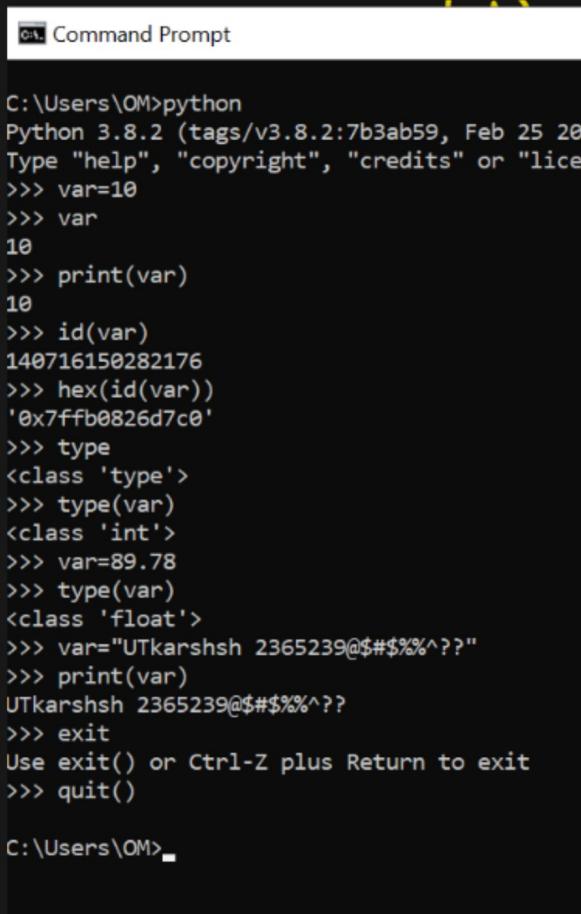
online or web

Pythonhosted.org

easy\_install.exe

offline

- ① int → 69
- ② float → 69.8
- ③ string → 'Abc 123'  
or  
"Nishant"



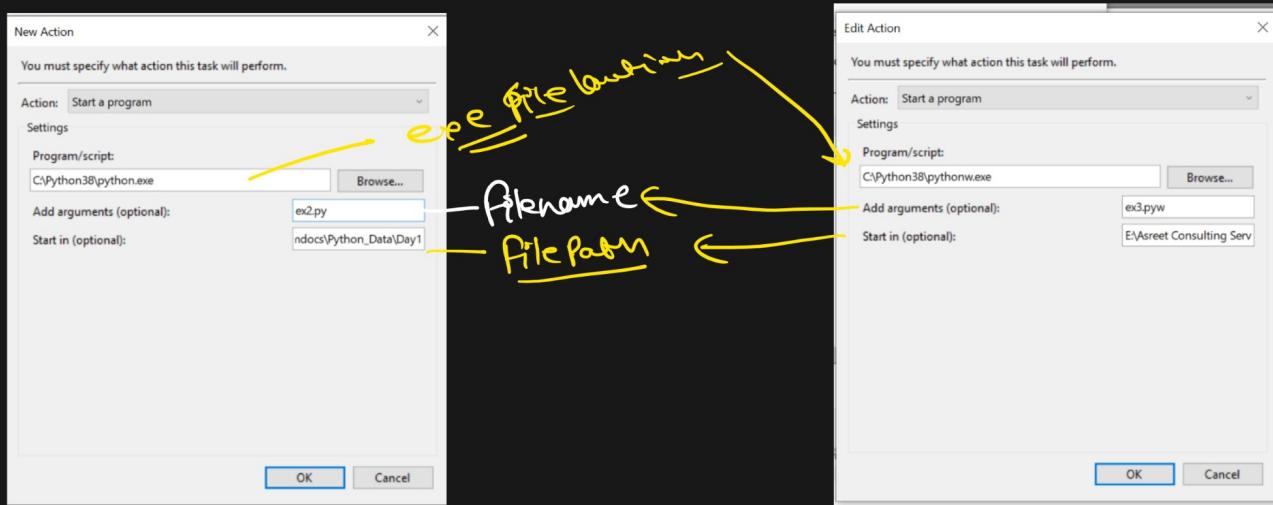
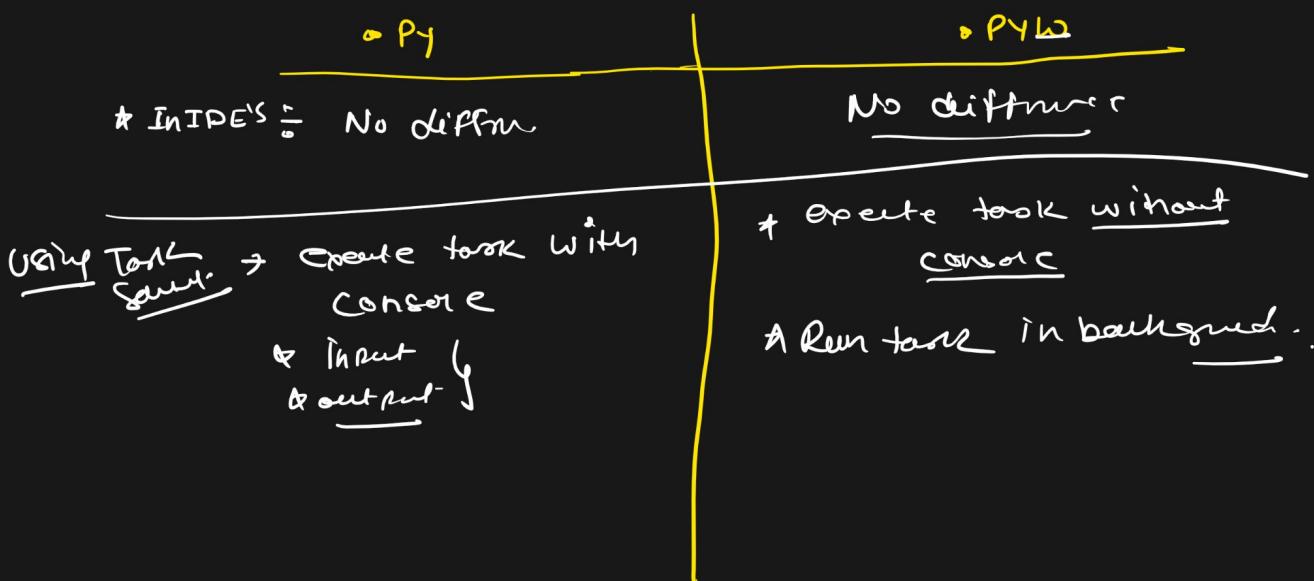
```
C:\Users\OM>python
Python 3.8.2 (tags/v3.8.2:7b3ab59, Feb 25 2020, 22:46:08)
Type "help", "copyright", "credits" or "license" for more information
>>> var=10
>>> var
10
>>> print(var)
10
>>> id(var)
140716150282176
>>> hex(id(var))
'0x7ffb0826d7c0'
>>> type
<class 'type'>
>>> type(var)
<class 'int'>
>>> var=89.78
>>> type(var)
<class 'float'>
>>> var="UTkarshsh 2365239@$$%%^??"
>>> print(var)
UTkarshsh 2365239@$$%%^??
>>> exit
Use exit() or Ctrl-Z plus Return to exit
>>> quit()

C:\Users\OM>
```

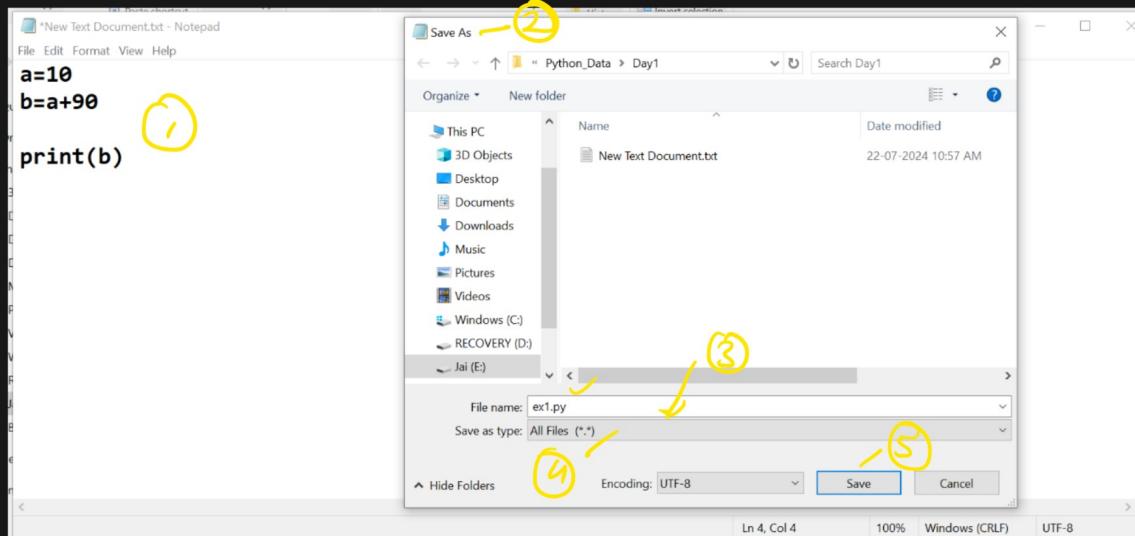
# Create A Python Script?

- ✓ ① Text file (Notepad)
- ✓ ② IDLE (GUI of Python)
- ✓ ③ IDE's ( PyCharm, Anaconda, Eclipse etc)
- ✓ ④ text editor ( VS code, Sublime etc)

→ .py & .pyw



## # Create Python file using Notepad.



## # Execute A Python Script

① Install Python ✓

② Set the Path ✓

③ cd to file location.

④ Python filename.py

```
C:\Users\QM>cd E:\Asreet Consulting Services\Agenda_amdocs\Python_Data\Day1  
C:\Users\QM>e:  
E:\Asreet Consulting Services\Agenda_amdocs\Python_Data\Day1>dir  
Volume in drive E is Jai  
Volume Serial Number is 0C69-B921  
  
Directory of E:\Asreet Consulting Services\Agenda_amdocs\Python_Data\Day1  
  
22-07-2024 10:58 <DIR> .  
22-07-2024 10:58 <DIR> ..  
22-07-2024 10:58 24 ex1.py  
22-07-2024 10:57 0 New Text Document.txt  
2 File(s) 24 bytes  
2 Dir(s) 63,631,749,128 bytes free  
  
E:\Asreet Consulting Services\Agenda_amdocs\Python_Data\Day1>python ex1.py  
100
```

A bracket on the right side groups the steps ③ and ④, with an arrow pointing to the 'List of dirs' in the terminal output. Another arrow points from the 'output' label to the terminal window.

## Multiple Assignment:

Multiple assignment can be done in Python at a time.  
There are two ways to assign values in Python:

### 1. Assigning single value to multiple variables:

x=y=z=50 — one line  
**print**(x)  
**print**(y)  
**print**(z)

## 2. Assigning multiple values to multiple variables:

a, b, c = 10, 10.9, 'Hello'  
One Array → Tuple

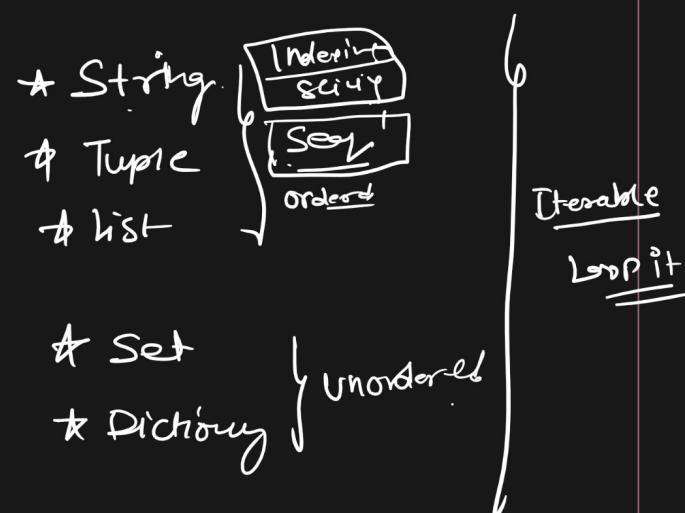
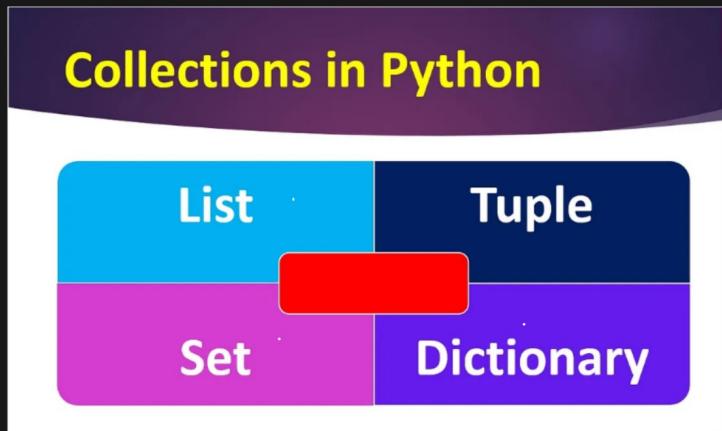
Unpack the values

\* len(var) = len(values)

exception

One var = more than one val

# Python Collections



## Tuple

- ① Tuple object can store diff. kind of Data Types
- ② Tuple objects are ordered (Seq's) → Indexing, slicing.
- ③ Tuple objects are Iterable (Loop it)
- ④ Tuple objects are Immutable → C U R D  
    ✓ X ✓ X
- ⑤ ↑ C → empty.

↑ C → empty.

```
#create a one value Tuple Object of UTKARSH ?  
a=("UTKARSH",)
```

```
1 a=()#empty  
2 a=(10,10.8,"Test",(1,1.9,("hello",89)))  
3 print(a)  
4 print(id(a),type(a),len(a))
```

## Tuple

C

\* C  
\* C('Hello')

\* (1, 2, 3, 4)

U

X

R  
★ Indexing.  
a[3]

D  
X

\* Slicing.  
a[: :-1]

\* Iteration (Loop it)  
for i in X:

≡

(Replaces  
one value)

+ve Indexing.  
Left - Right  
Starts with → 0

-ve Indexing.  
Right - Left  
Starts with → -1

var = 'NISHANT'      -ve      List  
L      R  
-7 -6 -5 -4 -3 -2 -1  
N | I | S | (H) | A | (N) | T  
0    1    2    3    4    5    6

L      → +ve  
↑      User  
R

Var[-2] = var[5] or var[-2]

↓      'N'  
-ve Indexing

var[3] or var[-4]  
\ /  
  H

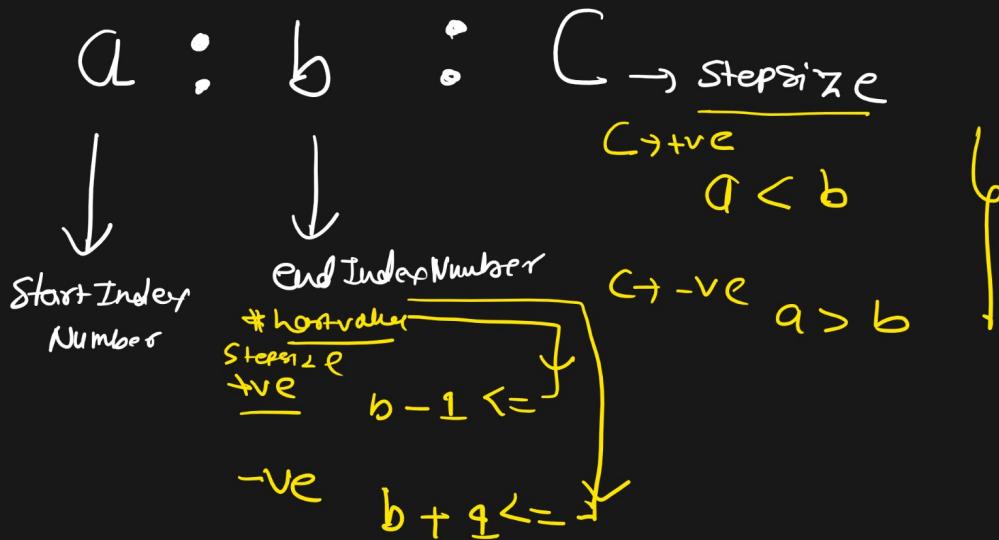
Middle char

+ve Index

Var[1], or var[-6]

\ /  
  I  
+ve Index

\* A.P  $\Rightarrow a_1 \left\{ a_1, a_1+d, a_1+2d, \dots, a_1+nd \right\}$ . Slicing + Seq's  $\rightarrow$  List + Tuple.



Body ①

① Stepsize -ve

$$-10 : -2 : -3$$

$a > b \rightarrow \text{true}$

$-10 > -2 \rightarrow \text{False}$

No Execution

②

Step-1 Stepsize +ve

$$10 : 22 : 2$$

$$a < b \rightarrow T$$

$$10 < 22 \rightarrow T$$

Step-2 hostvalue

Stepsize +ve

$$b-1 \leq \text{hostvalue}$$

$$21 \leq$$

$$21 \leq$$

$$a_1 \ a_1+d \ a_1+2d.$$

$$\boxed{10, 12, 14, 16, 18, 20} \times$$

③

①

$$-2^{\circ} - 26^{\circ} - 4$$

$$a > b \rightarrow \text{True}$$

$$-2 > -26 \rightarrow \underline{\text{True}}$$

② Step-2  
last

Stepsize-ve

$$b+1 \leq \text{lastval}$$

$$-26+1 \leq$$

$$-25 \leq$$

$$q_1, q_1+d$$

$$\boxed{-2, -6, -10, -14, -18, -22} - \cancel{26}$$

a = 'NISHANT'  
0 1 2 3 4 5 6

$$b = a \left[ \frac{2 : \text{len}(a) : 2}{\downarrow} \right]$$

$$2 : 7 : 2 = 2, 4, 6$$

b = 'SAT'

Var='NISHANT'  
Var[:3]

C = 3

a = 0  
b = len(var)

+resizer  
Var [ : ]

a = 0  
b = len(a)  
c = 1

Var[0:10]

a = 0

b = 10

c = 1

-ve stepsize

Var[x:y:-1]

-~~✓~~

X = -1

Y = -(len(var)+1)

Day-2

23-07-2024



## Python Keywords

Keywords are special reserved words which convey a special meaning to the compiler/interpreter. Each keyword have a special meaning and a specific operation. List of Keywords used in Python are:

True	False	None	and	as
asset	def	class	continue	break
else	finally	elif	del	except
global	for	if	from	import
raise	try	or	return	pass
nonlocal	in	not	is	lambda

These can be variables ,class ,object ,functions , lists , dictionaries etc  
There are certain rules defined for naming i.e., Identifiers.

- ✓ I. An identifier is a long sequence of characters and numbers.
  - ✓ II. No special character except underscore ( \_) can be used as an identifier.
  - ✓ III. Keyword should not be used as an identifier name.  
*1. Python module - Name*
  - ✓ IV. Python is case sensitive. So using case is significant.
  - ✓ V. First character of an identifier can be character, underscore ( \_) but not digit.

Var  
object  
ClassName  
func Name  
Python FileName  
DirName (folder)  
A-Z a-z 0-9 -

$\left[ \begin{matrix} A & -Z \\ -A & Z \end{matrix} \right] \left[ \begin{matrix} A & -Z \\ -A & Z \end{matrix} \right] = \dots$   
 1st char  
 $\downarrow$

Var  $\rightarrow X$   
 Var 1234  $\rightarrow \checkmark$   
 Var 123w  $\rightarrow \checkmark$   
 Var  $\rightarrow X$

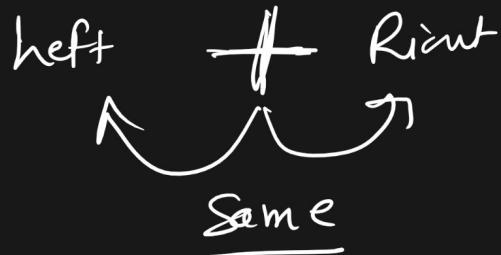
## I. String literals:

String literals can be formed by enclosing a text in the quotes. We can use both single as well as double quotes for a String.

+ triple quotes

```
# 'I am Niswet'          # " " " "
# "I am Niswet" .       # ' '
# \n      newline        W
# It      tab space.    ,,
# \      → escape char's
```

## Concatenation In Seq's



$\text{Str} + \text{Str} \rightarrow \text{Str}$   
 $\text{tuple} + \text{tuple} \rightarrow \text{tuple}$   
 $\text{list} + \text{list} \rightarrow \text{tuple}$

#  $\text{Int}( ) \rightarrow \text{Int} \quad 6 \rightarrow 6$

float  $6.2 \rightarrow 6$

String  $\rightarrow \checkmark 11436' \rightarrow \text{only digits}$   
 $\hookrightarrow 11436 \in \text{int}$

\* Rest  $\rightarrow$  Error

otherwise  
 $\hookrightarrow \underline{\text{Error}}$

\* float( ) (i) 6 → 6.0

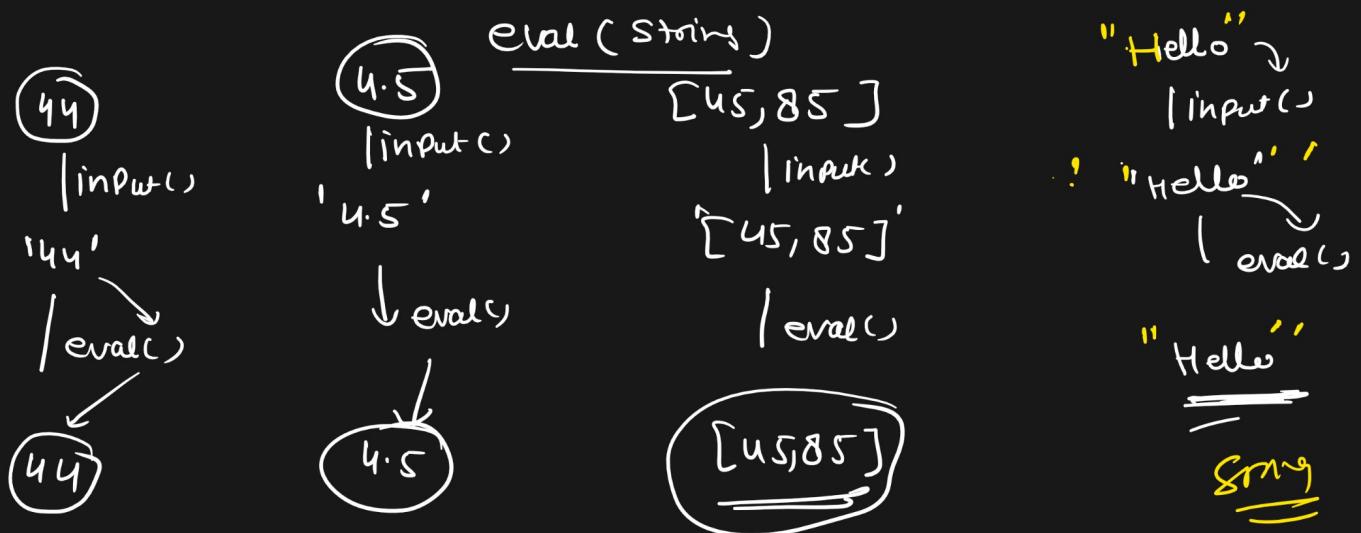
(ii) 6.2 → 6.2

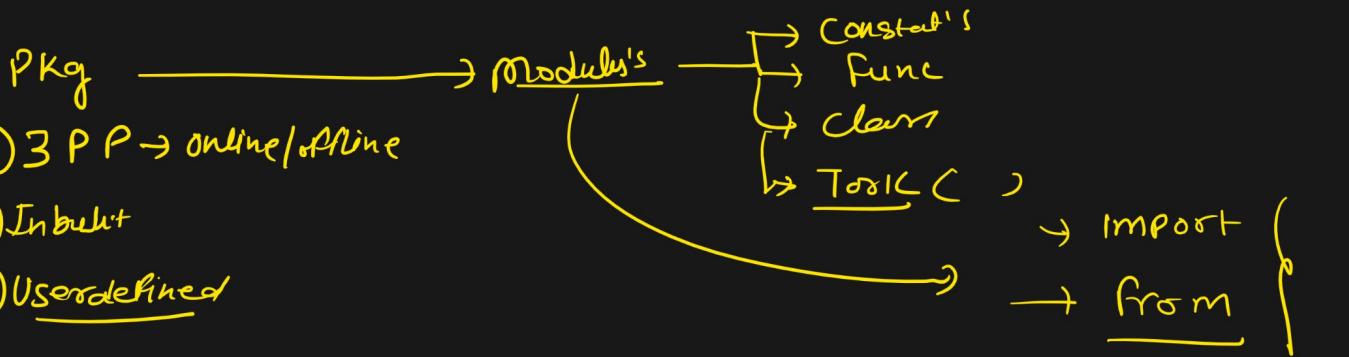
(iii) String  $\rightarrow \star^{11345^+} \rightarrow$  only digits-  
 $\rightarrow \underline{\underline{1345.0}}$

$\checkmark$  \* 11345.45' → 1345.45

Rest  $\rightarrow$  Error

#### (iv) Other data types





- \* numpy → numpy
- \* regex → re
- \* pandas → pandas → function name
- \* datetime → datetime → datetime

### (A) import

(i) import moduleName

>> import math



→ module import as a directory with same Name

→ Need Refers to Call them.

math.pi ✓  
math.sin() ✓

\* Memory wasted .

(A) `import`

(i) `import moduleName as newname`

> `import math as m`



\* A module import as a dictionary with new name

\* Need Refers to Call them -

`m.pi`

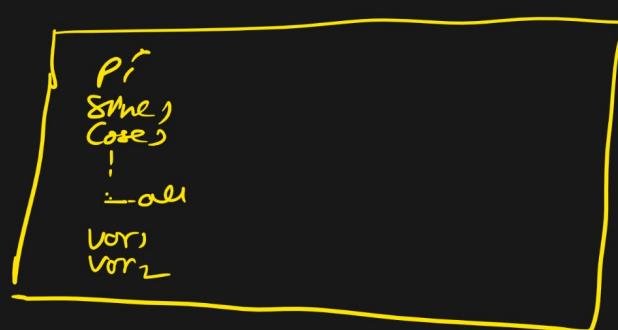
`m.sin()`

\* Memory wasted .

(B) `from`

(i) `from moduleName import *`

`from math import *`



\* No directory created .

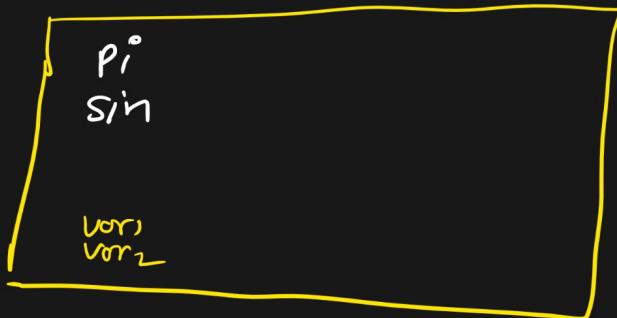
\* No refers required

\* Memory still wasted

(B) from ~~AAA~~

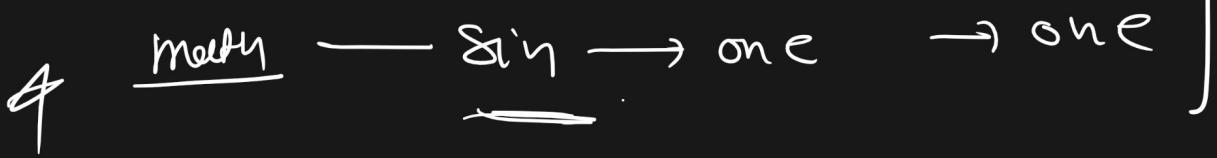
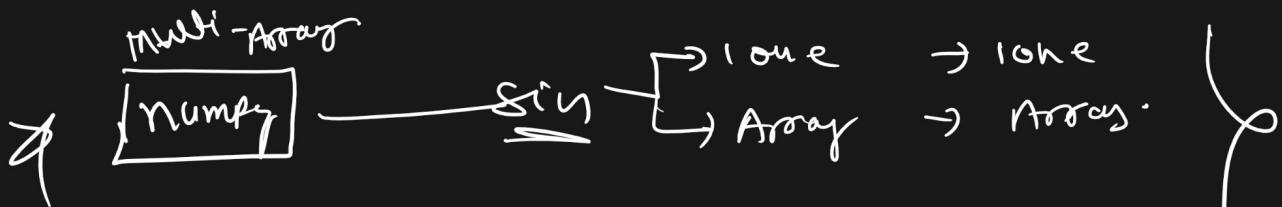
(ii) from moduleName import x<sub>1</sub>, x<sub>2</sub>

from math import pi, sin



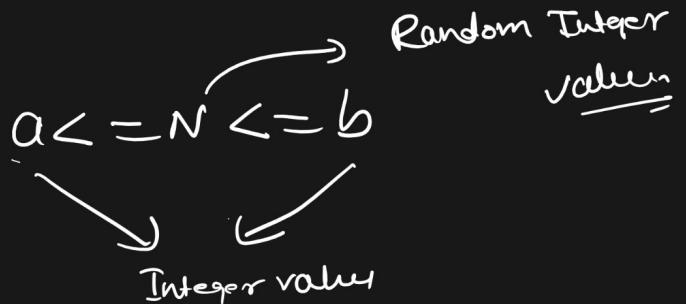
- \* No object created.
- \* No return required.
- \* No memory wasted.

import	from
* import math	from math import x <sub>1</sub> , x <sub>2</sub>
* <u>Module get imported</u> So Need return to call use constants, functions etc.	* Importing fun's, class's, const. Do no need return
math.pi math.sin	pi sin()
* <u>Everything imported</u> so memory wasted	* <u>We imported as per requirement</u> so memory <u>not wasted</u> .



### Random

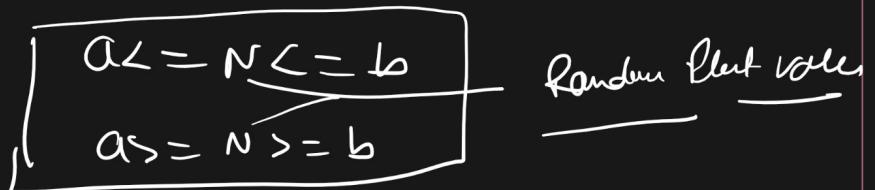
① `randint(a, b)`



② `random()`

$0.0 \leq N \leq 1.0$   
↳ Random float value.

③ `uniform(a, b)`



Random data generation	
<pre>import random #package</pre>	
<code>random.randint(a,b)</code>	Return a random integer N such that $a \leq N \leq b$ .
<code>random.random()</code>	Return the next random floating point number in the range $[0.0, 1.0]$ .
<code>random.uniform(a, b)</code>	Return a random floating point number N such that $a \leq N \leq b$ for $a \leq b$ and $b \leq N \leq a$ .
<code>random.randrange(start, stop[, step])</code>	
<code>random.choice('abcdefghijklmnopqrstuvwxyz')</code>	Choose a random element
<code>random.shuffle(items)</code>	
<code>random.sample([1, 2, 3, 4, 5], n)</code>	Choose n random sample

`randrange(a, b, c)`

$a : b : c$

$10 : 22 : 4$

$\boxed{10, 14, 18, 22}$  ~~2~~

10

14

18

22

1.2 GB Songs.

$\rightarrow \boxed{1.2 \text{ GB}}$

$\Rightarrow \ll \text{ } \gg$   
Shuffle

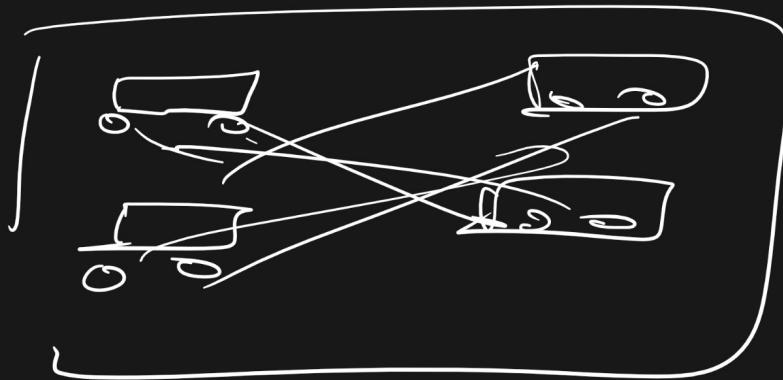
Yes  
 No

Song List

$\left[ 'C:\text{New}\text{\Delta}\text{ta}\text{\Delta}01\text{-Kehn.mp3}', 'D:\text{1Tera}\text{\Delta}02\text{-mp3}', 'E:\text{103.mp3}' \right]$

$\left[ 'E:\text{103.mp3}', 'C:\text{New}\text{\Delta}\text{ta}\text{\Delta}01\text{-mp3}', 'D:\text{1Tera}\text{\Delta}02\text{-mp3}' \right]$

Song List  
Vox 1  
Vox 2  
None  
Song (Vox)  
Itself  
VPintered



**python**  
powered

**Write First Python Script  
By Today Learning Only**

Not to use %  
If else  
def()

**P. 3**

W.A.P for following

Amount per item

Burger = 30

Pizza = 100

Tomato Soup = 40

sgst = 2.5%

cgst = 2.5% *(Handwritten note: gtax)*

**Input Window**

```
Command Window
Watch this Video, see Examples, or read Getting Started.

Food Menu :
1.Burger
2.Pizza
3.Tomato Soup
Enter your choice : 2
Enter quantity : 5

} DISPLAY(1)
} Input(2)
} Calculation(3)
```

**Output Window**

```
Command Window
Watch this Video, see Examples, or read Getting Started.

Tax : 50.000000
Total amount is : 550.000000
Thank you Visit again.

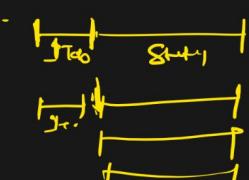
} DISPLAY output(4)
```

Ques

Day-3

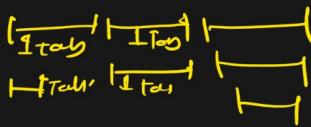
### conditional statements in python

# if exp:



① if exp:

if exp2:



else:



② if exp:

else:



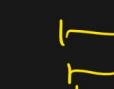
③ if exp1:



elif exp2:



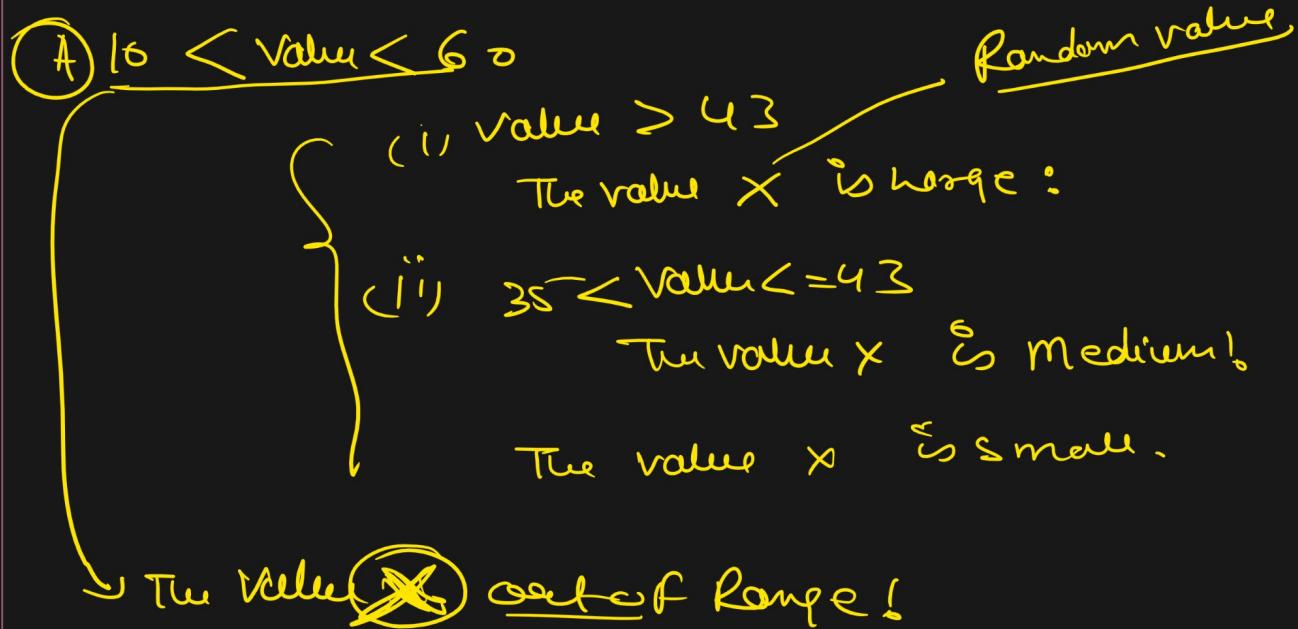
elif exp3:



:

else:

Ques W.A.P to generate Random Integer No b/w 0-100 and follows the below conditions.



Print

for (i=0; i<=10; i++)



i = loop\_variable

Datatype = static = int

Total-No-Iteration = 11

loop S

Python

for loop-var in Iter-object :



<u>Iter-object</u> -	Str	dict
	tuple	set
	list	filter
	zip	map
		else

\* Datatype = Dynamic = Any type

\* Total-No-Iteration = len(Iter-object)

range(b)

$$\begin{cases} a = 0 \\ b = b \\ c = 1 \end{cases}$$

0 : b : 1

0 : 5 : 1

0, 1, 2, 3, 4

for i in range(5) :

print(i) ↳ 5 times

range(a, b, c)

a = 10

b = 22

c = 2

for i in range(10, 22, 2) :

print(i)

10, 12, 14, 16, 18, 20

Prob: 2 W.A.P to calculate factorial b/w 0-20 values. By taking input from the user.

① Note: take care due to odd inputs → ABC123

$$!5 \rightarrow 1 \times 2 \times 3 \times 4 \times 5 \rightarrow 120$$

$$!5 \rightarrow 5 \times 4 \times 3 \times 2 \times 1 \rightarrow 120$$

$$!9 \rightarrow 9$$

$$!0 \rightarrow 1$$

+/- → 0 or any other value.

\*// → Except 0, any other value

#### Membership Operators:

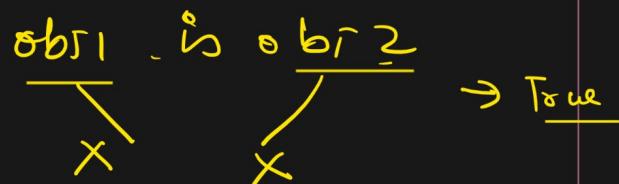
Operators	Description
in	Returns true if a variable is in sequence of another variable, else false.
not in	Returns true if a variable is not in sequence of another variable, else false.

$$\text{var} = [48, 09, 61, 40]$$

9 in var = True

61 not in var → False

#### Memory Address of Two objects



#### Identity Operators:

Operators	Description
is	Returns true if identity of two operands are same, else false.
is not	Returns true if identity of two operands are not same, else false.

$$a = (1, 1.2, (1, 2))$$

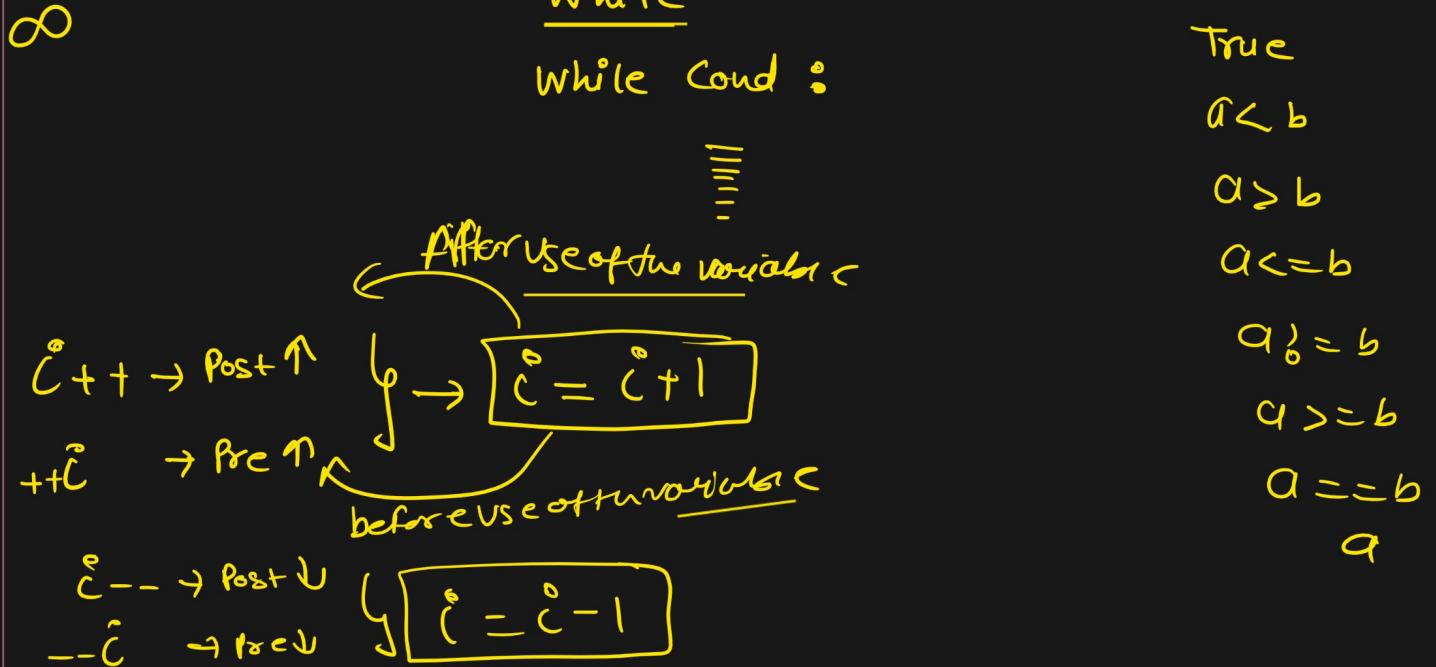
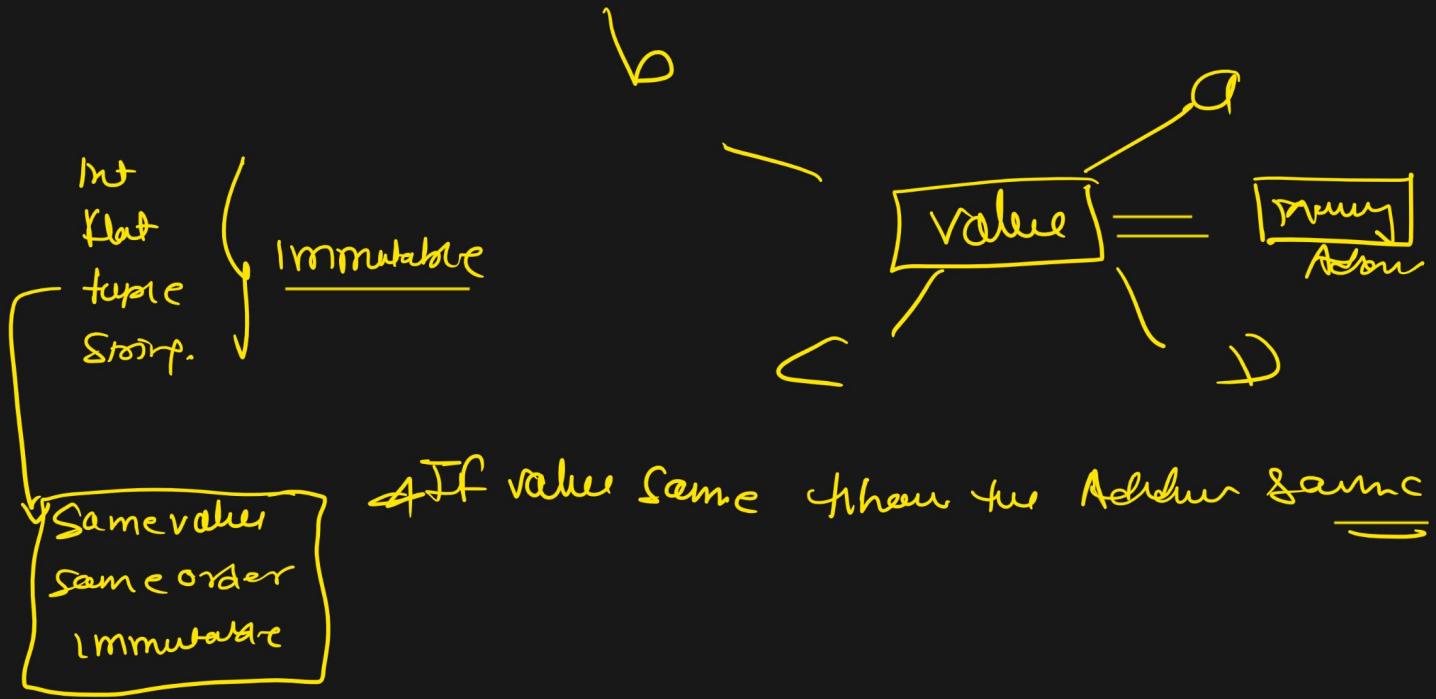
$$b = (1, 1.2, (1, 2))$$

$$a \stackrel{?}{=} b \rightarrow \underline{\text{True}}$$

$$a = 5$$

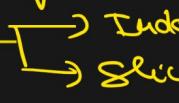
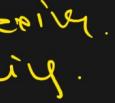
$$b = 5$$

$$a \stackrel{?}{=} b \rightarrow \underline{\text{True}}$$



## List

✓ \* Diff kind of Data types can store

\* ordered → Seq's →  Indexing.  
 Slicing.

✓ \* Iterable object

\* Mutable → 

\* [ ] → empty.

[ 'Hello' ] → the value list

\* [ 'Hello', 1, 2, 'Test' ]

# min() max() sum() len()

list



\* [ ]

\* [ 'Hello' ]

\* [ 1, 2, 'Hello' ]



- \* a[3] = NewValue
- \* a.append( Object )
- \* a.insert( index, obj )
- \* a.extend( Iter-Object )



- # a[3]
- Slicing.
- # a[0:2]



- \* del a[3]
- \* a.remove( value )
- \* m=a.pop()
- \* m=a.pop(index)

Iteration  
# for i in a:  
 print(i)

a.append(object)

Object → int  
float  
str  
list  
set  
dict  
tuple  
zip  
map  
etc.

a = [1, 2, 3] → ③

a.append(45)

[1, 2, 3, 45] → ④

a.append([45, 86, 77])

[1, 2, 3, [45, 86, 77]] → ⑤

a.insert(index, object)

a = [1, 2, 3] → ③  
Index → 0 ↑ 1 ↑ 2

a.insert(1, 45)

[1, 45, 2, 3] → ④

a.insert(2, (45, 86, 77))

[1, 2, (45, 86, 77), 3] → ⑤  
① ② ③ ④

a.extend(iterable)

Str List  
Set Tuple  
Dict Zip  
filter Map  
etc.

a = [1, 2, 3] → ③

a.extend([45, ]) → ④

[1, 2, 3, 45] → ⑤

a.extend([45, 86, 77]) → ⑥

[1, 2, 3, 45, 86, 77] → ⑦

List Prob

```
a=['Test',1,'Test',1.2,'Test',(1,2),'Test',[89,67],'Test','Test',56.6,'Test']
print(a)
print(id(a),type(a),len(a))
```

```
1 a=['Test',1,"Test",1.2,"Test",(1,2),"Test",[89,67],"Test","Test",56.6,"Test"]
2 print(a)           2   4   6   8   0   11
3 print(id(a),type(a),len(a))      5
4 print("=====Delete=====")      6
5 #create a list of Value "Test" index numbers . ? Parametrically → Group
6 #[0,2,4,...]
7 #Delete all the "Test" Values from given list ?
```

Output

① [0, 2, 4, 6, 8, 10, 11]

12 - 7 = 5

② {1, 1.2, (1, 2), [89, 67], 56.6}

## dictionary in python

{ # {} → empty.  
# { pair1, pair2, pair3, ... }  
# { key1 : value1, key2 : value2, key3 : value3, ... }  
# dict is Subobject of Subparts under a single.  
# dict's objects are Iterables - Loop it  
# dict objects are Tuples

S U R D

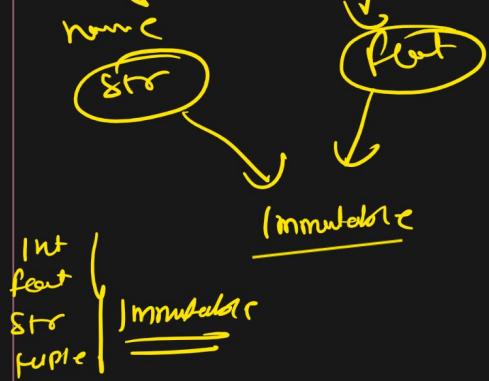
dict {}

C U R D  
d={'name': 'Manish', ('dept', 'other'): [[ "L&D"], [1, 2, 3], 4576563447.78: '%\*%\*(sghdh\$&98855JB38^%)'}]  
(1) {} → Key should be  
Identifier  
Identifier  
Identifier  
Identifier

# v = d[key]  
# v = d.get(key, default)  
# It returns  
for i in d:  
 print(i)

```
# d=dict(key1=value1,key2=Value2,...,...)  
d=dict(name='Manish',dept='IT')
```

```
ex1 ::  
1 d={'name': 'Manish', 'dept': "L&D", 'name': ["Manish", 'Rahul ', 'SitaRama'],  
2     4576563447.78: '%*#%(*sghdh$&98855JBJ8^%$)'}  
3 print(d)  
4 print(id(d), type(d), len(d))
```



Key as An Array?

\* Tuple

\* List

[A-2 a-2 o-9 - ]

[A-2 a-2 - ] [A-2a-2o-9-] - - -

\* Named Key X

\* Key as Array X