

Andocs

Day 1-2-3-4-5-6-7-8

22-07-2024

23-07-2024

24-07-2024

25-07-2024

26-07-2024

29-07-2024

30-07-2024

31-07-2024

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.

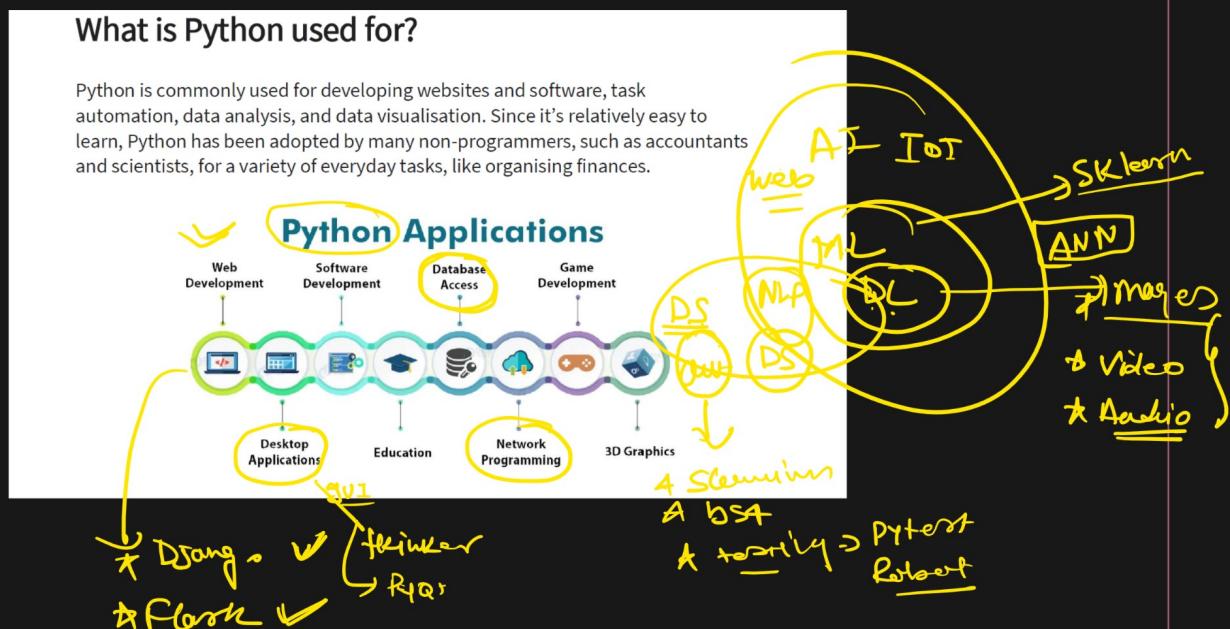
Python, a high-grade language, is a computer programming language that is meant to represent the needs of a problem and mimics natural language or mathematical notation.

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.

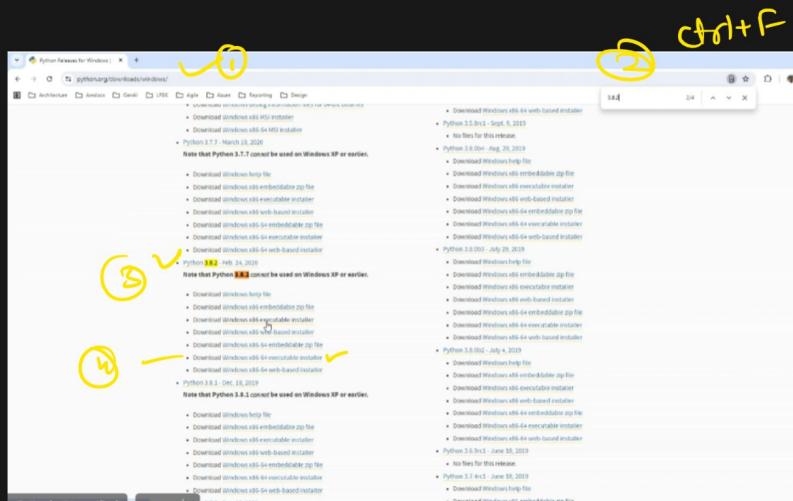
Who Developed Python Programming Language?

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

43.P.P

own file

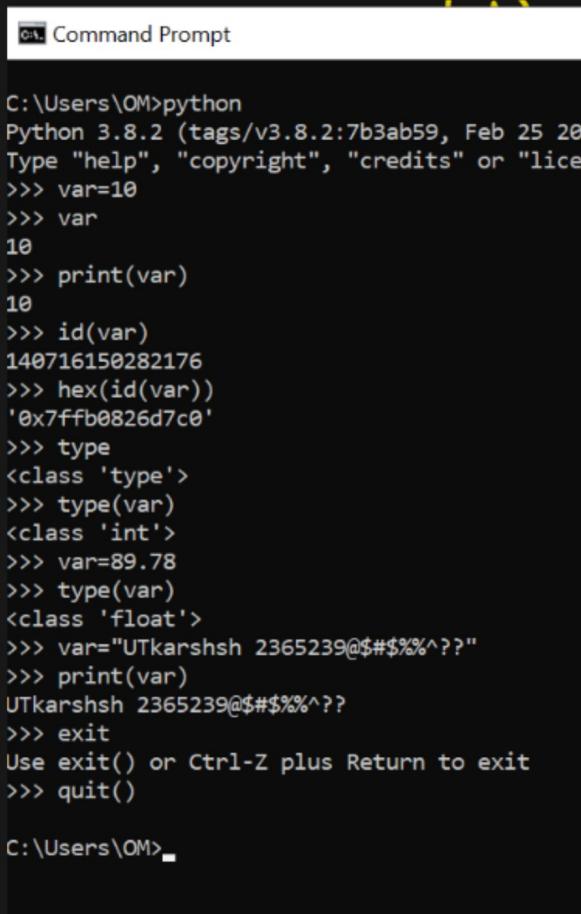
online or web

Pythonhosted.org

easy_install.exe

offline

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



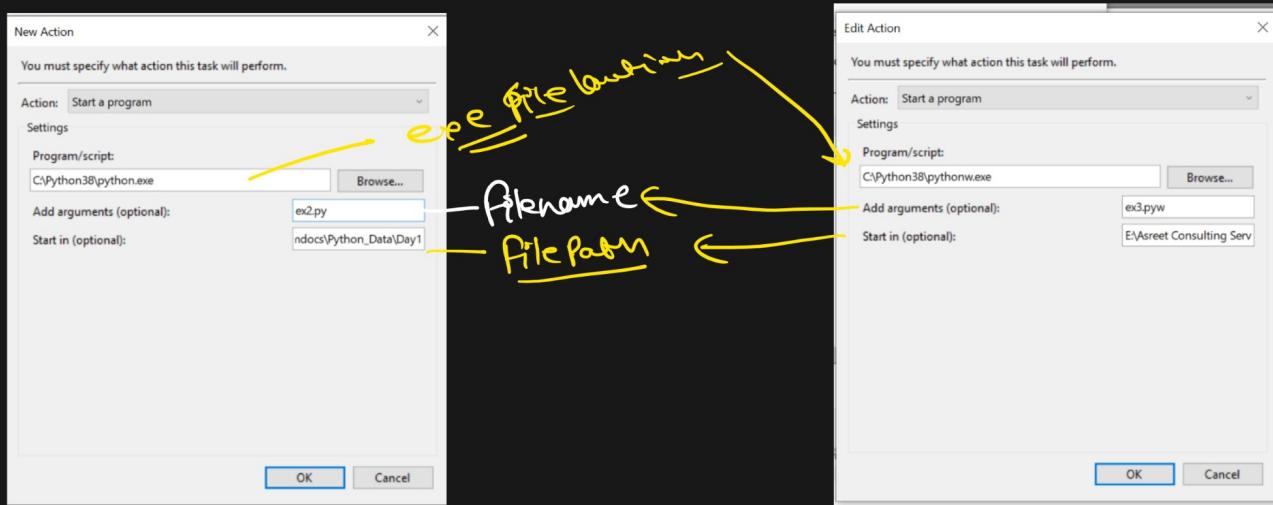
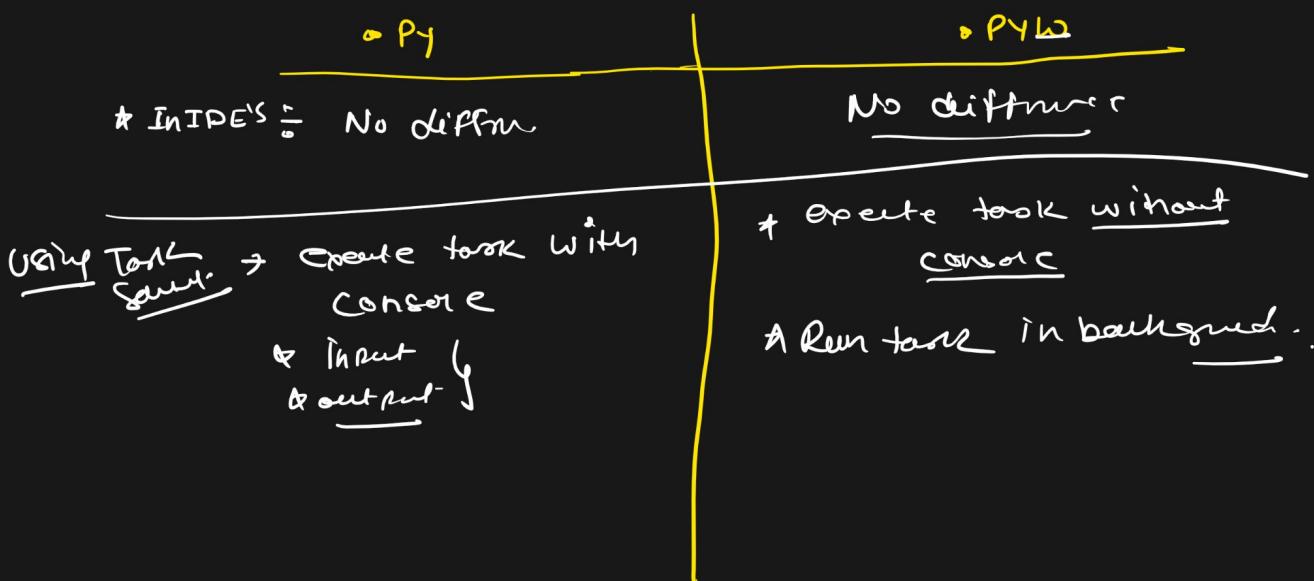
```
C:\Users\OM>python
Python 3.8.2 (tags/v3.8.2:7b3ab59, Feb 25 2020, 22:46:08)
Type "help", "copyright", "credits" or "license" for more information
>>> var=10
>>> var
10
>>> print(var)
10
>>> id(var)
140716150282176
>>> hex(id(var))
'0x7ffb0826d7c0'
>>> type
<class 'type'>
>>> type(var)
<class 'int'>
>>> 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>
```

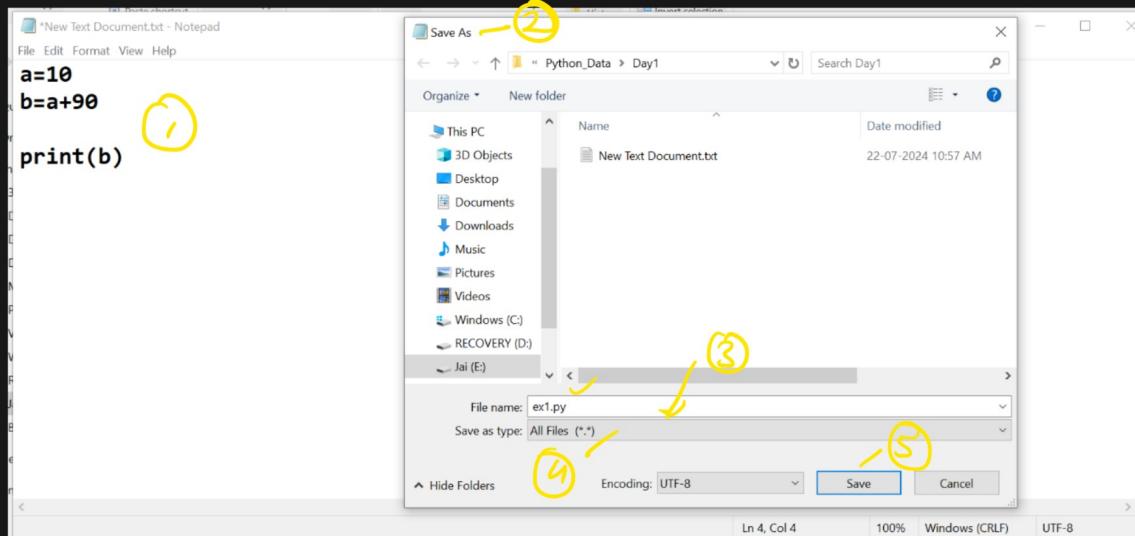
Create A Python Script?

- ✓ 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 ← List of dirs  
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 ← output
```

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

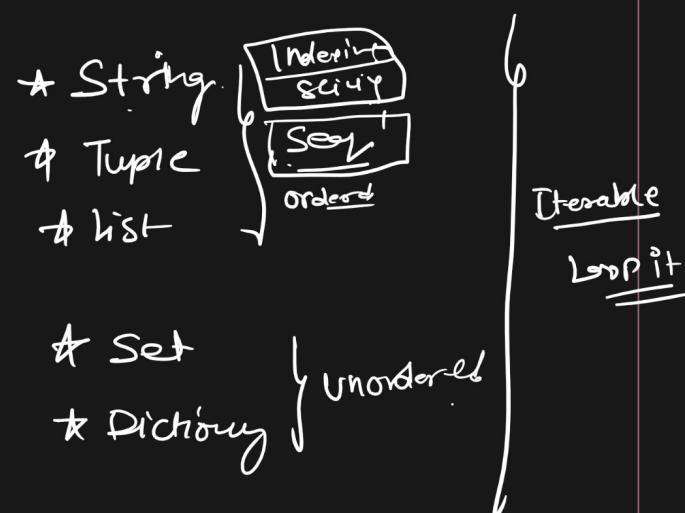
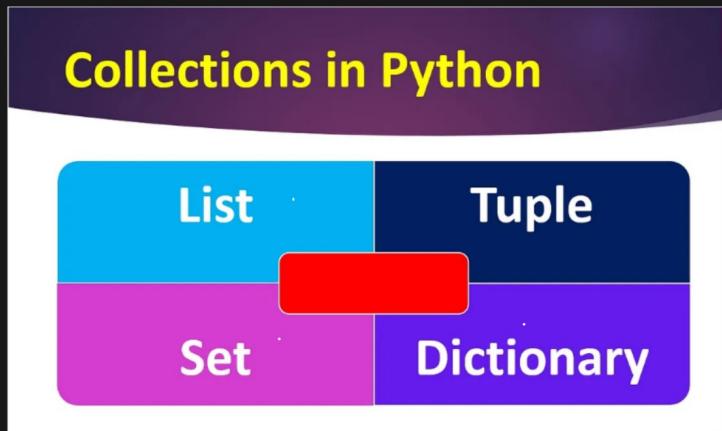
Unpack the values

* 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.
[0, 3]

D
X

* Slicing.
[0 : :-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
[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

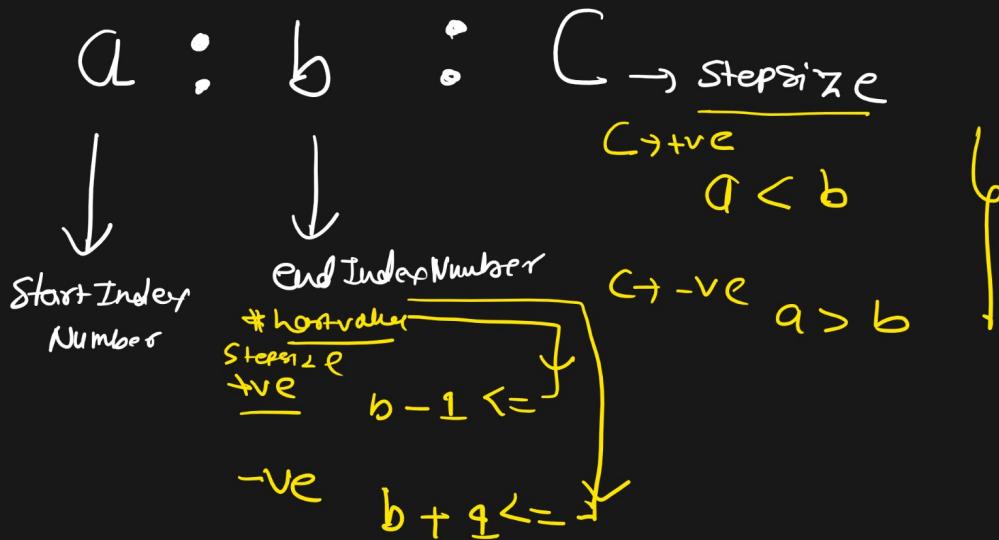
Middle char

+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 Expansion

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.
Only module name
 - ✓ III. Keyword should not be used as an identifier 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 -

$\left[\underline{A-Z} \underline{a-z} - \right]$ $\left[A-Z a-z \circ-\circ \right] \dots$
 1st char 

 $1Var \rightarrow X$ $-1234 \rightarrow \checkmark$
 $Var12345 \rightarrow \checkmark$ \checkmark
 $(Var \rightarrow X)$

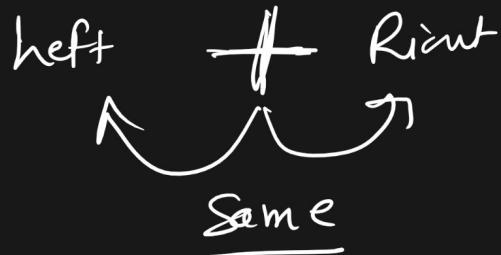
I. String literals:

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 # " " " " " " " "
- # 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

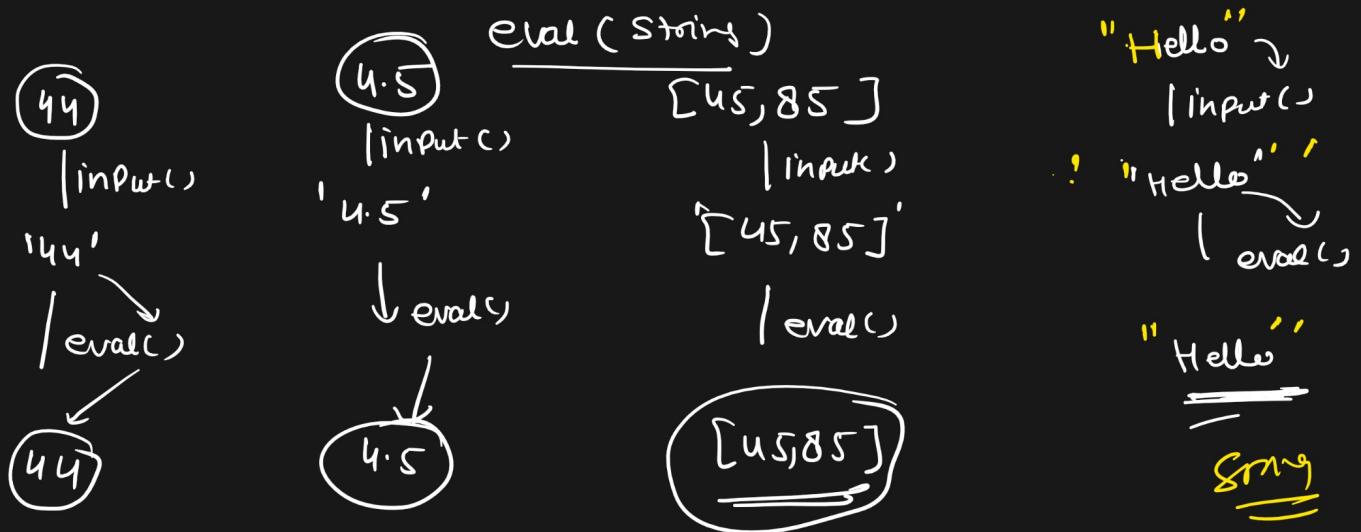
(ii) 6.2 → 6.2

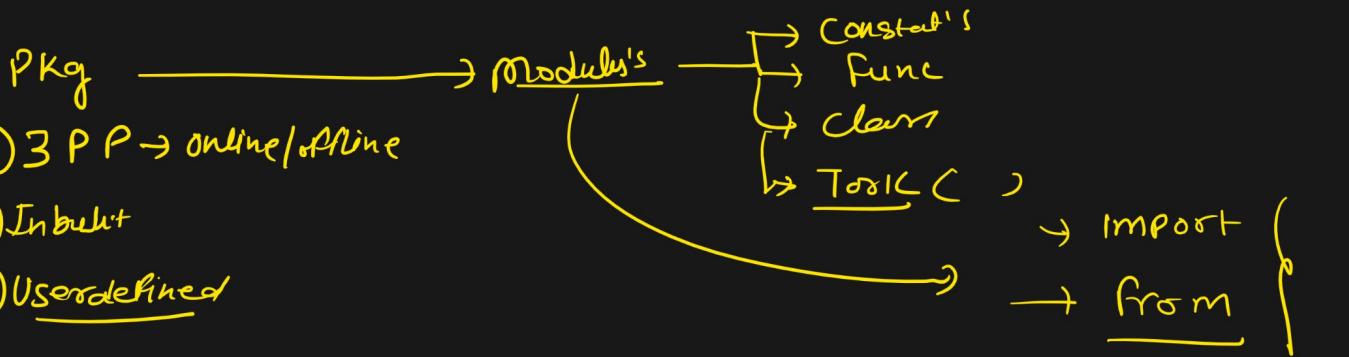
(iii) String $\rightarrow \cancel{1} 1345^{\prime} \rightarrow$ only digits-
 $\rightarrow \underline{\underline{1345.0}}$

\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

(1) 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`

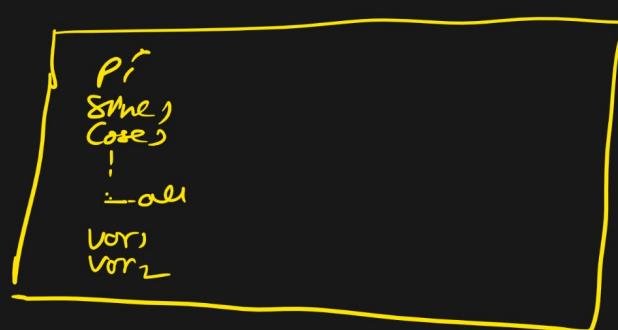
`m.sin()`

* Memory wasted .

(B) `from`

(i) `from moduleName import *`

`from math import *`

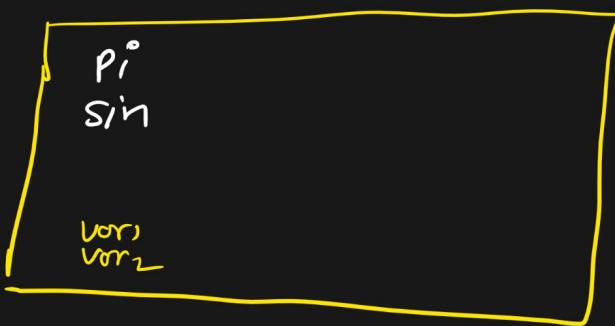


* No directory created .

* No refers required

* Memory still wasted

(B) from ~~AAA~~



from math import pi, sin

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

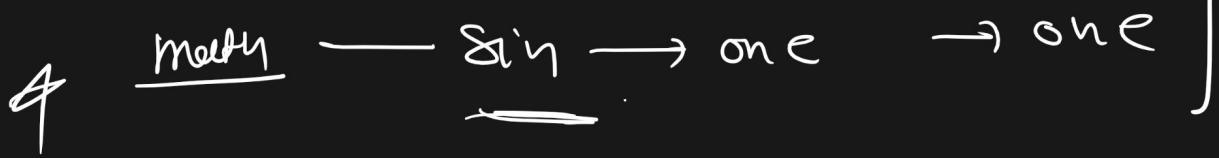
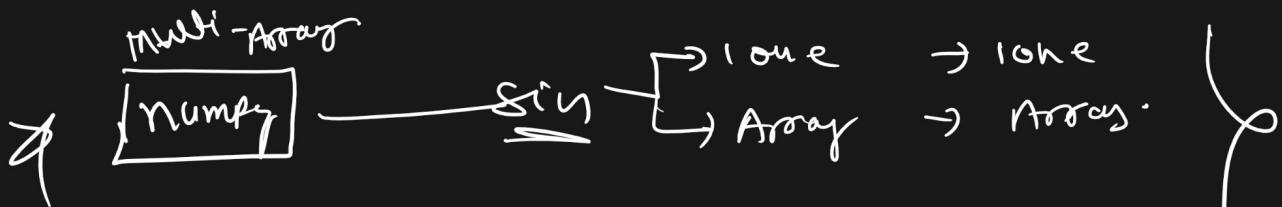
Import

- * Import math
- * Module get imported
 - ↳ Need return to call
 - ↳ Use constants, functions etc.

math.pi
math.sin
- * Everything imported
so memory wasted

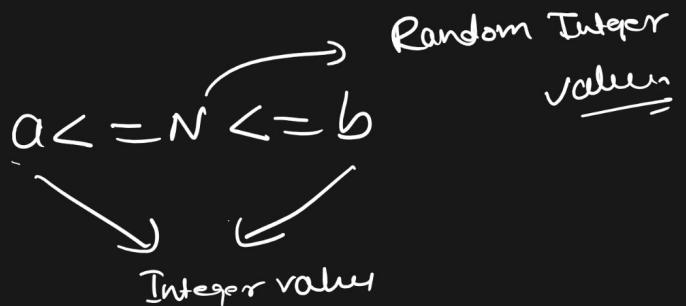
from

- * from math import X1, X2
- * Importing fun's, class's, const.
Do no need return
- * pi
sin()
- * We imported as per requirement
so memory wasted.



Random

① randint(a, b)

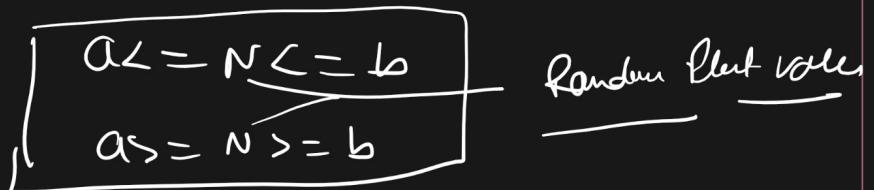


② random()

$0.0 \leq N \leq 1.0$

↳ Random float value.

③ uniform(a, b)



Random data generation	
<code>import random #package</code>	
<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. $\rightarrow \boxed{1.2 \text{ GB}}$

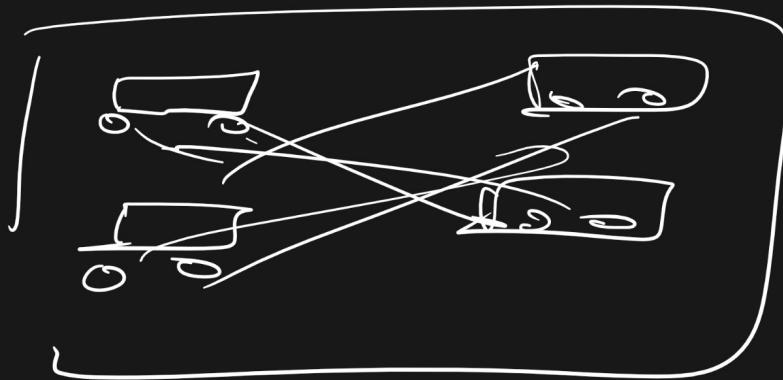
 Yes
 No ✓

Song List

$\left['C:\New\Delta\01.mp3', 'D:\Tera\02.mp3', 'E:\03.mp3' \right]$

$\left['E:\03.mp3', 'C:\New\Delta\01.mp3', 'D:\Tera\02.mp3' \right]$

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.

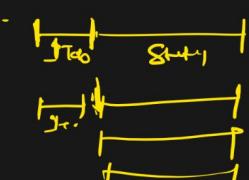
} DISPLAY output(4)
```

Ques

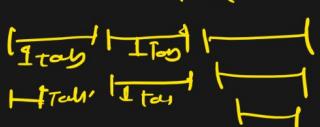
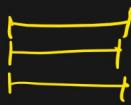
Day-3

conditional statements in python

if exp:



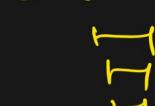
① if exp:



② if exp:



else:



③ if exp:



elif exp2:



elif exp3:

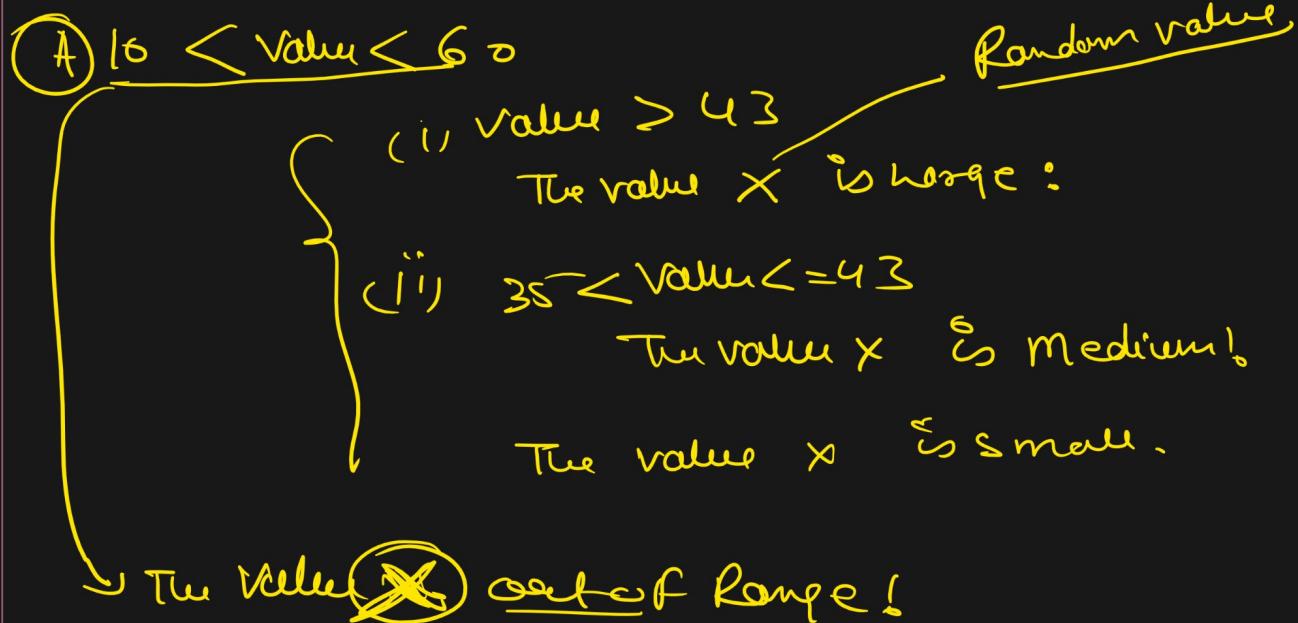


:



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)

a = 0	0 : b : 1
b = b	0 : 5 : 1
c = 1	0, 1, 2, 3, 4

for i in range(5) :

print(i) 5 times

range(a, b, c)

a = 10 10, 22, 2

b = 22 (10, 12, 14, 16, 18, 20)

c = 2

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

print(i) 6 times

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

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

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

$$!1 \rightarrow 1$$

$$!0 \rightarrow 1$$

+/- → 0 or any other value.

*// → Except 0, any other value

Membership Operators:

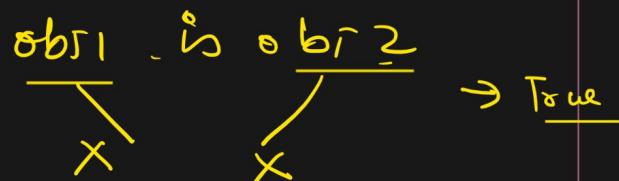
Operators	Description
in	Returns true if a variable is in sequence of another variable, else false.
not in	Returns true if a variable is not in sequence of another variable, else false.

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

09 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.

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

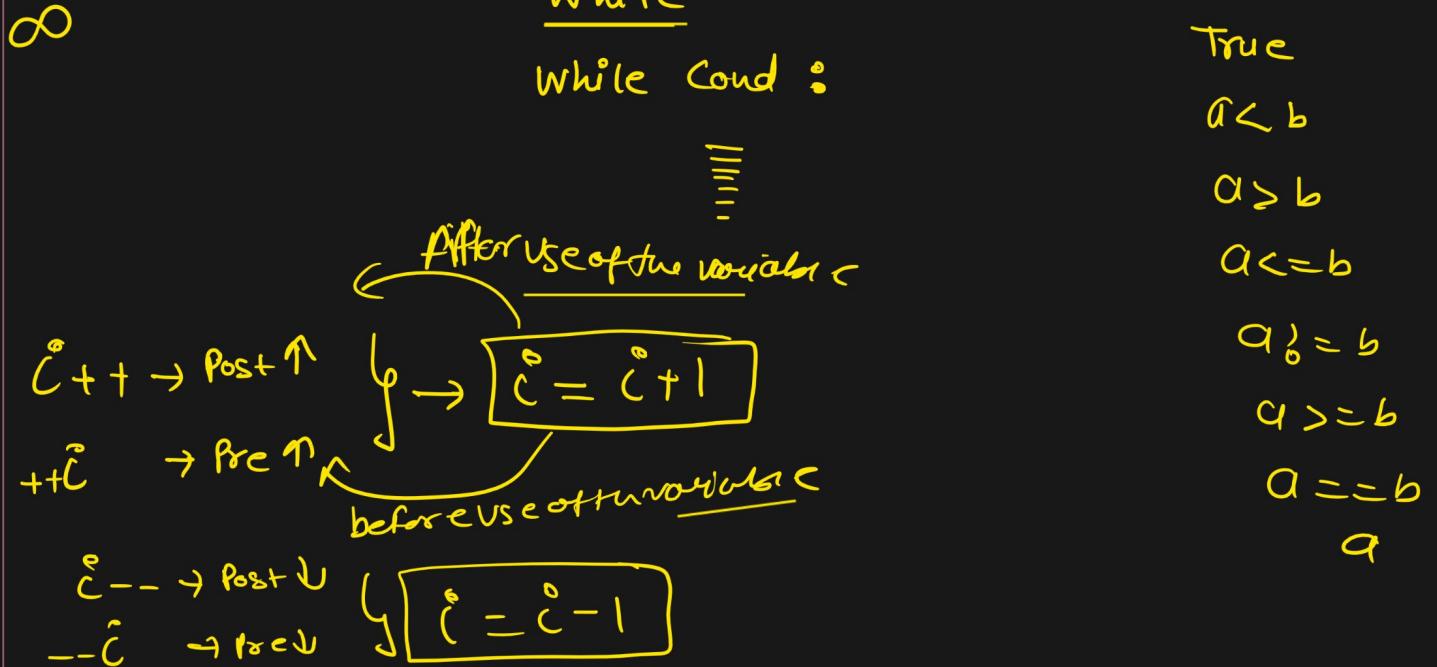
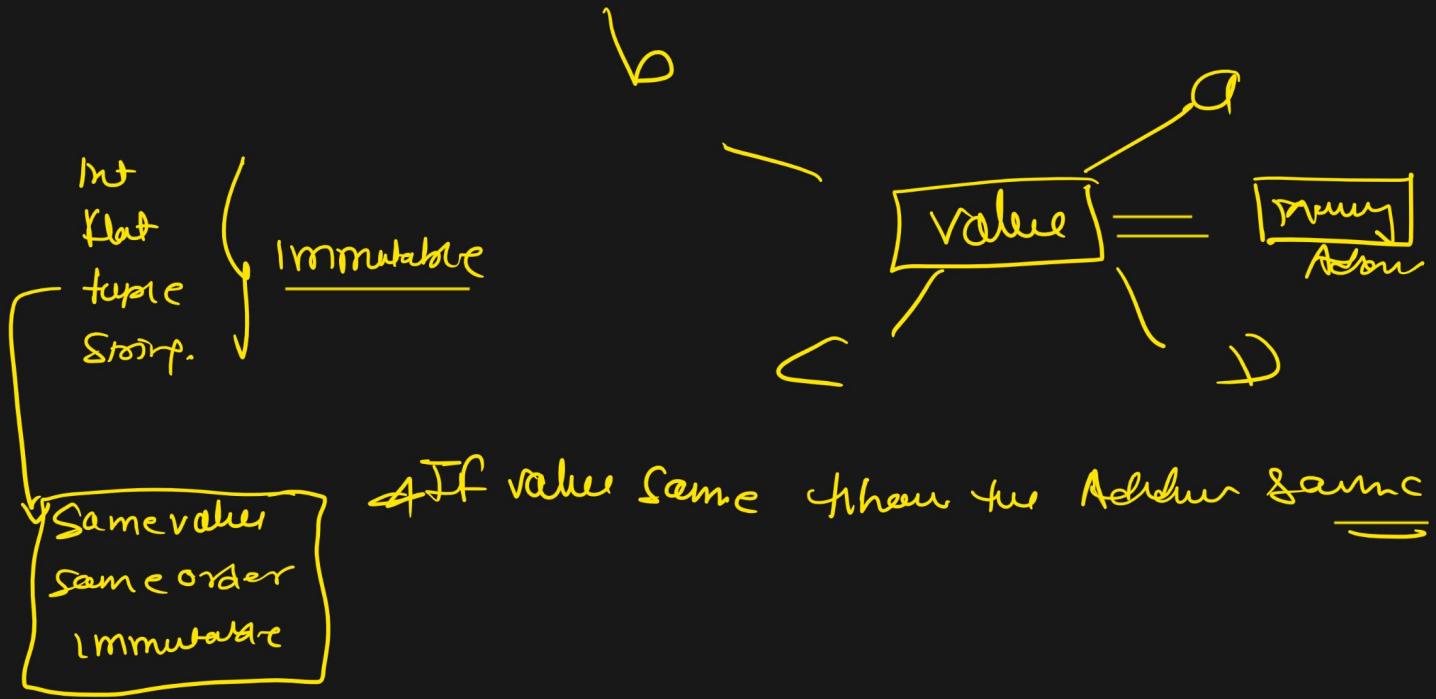
$$b = (1, 1.2, (1, 2))$$

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

$$a = 5$$

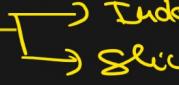
$$b = 5$$

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



List

✓ * Diff kind of Data types can store

* ordered → Seq's →  Indexing.

✓ * Iterable object

* Mutable → C U R D

* [] → empty.

['Hello'] → one 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]

slice.
a[0:2]

Iteration
for i in a:
print(i)



* del a[3]

* a.remove(value)

* m=a.pop()

* m=a.pop(index)

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])

[1, 2, 3, 45, 86, 77] → ⑥

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,...]
#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
(1) {} → Key should be
Identifier.
Immutability
→ Str
→ Int
→ Real
→ Tuple

+ d[key]=NewValue.
+ d[NewKey]=NewValue.
+ d.update(d1)
+ {k1d1, k2d2}

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

* d.keys()
* d.values()
* d.items()

D
del d[key]
* d.pop(key, default)

```

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-Z A-Z O-9 -]

[A-Z A-Z -] [A-Z A-Z O-9 -] - - -
A B

* Named Key X

* Keys as Array X



A - B

$A \cap B$



$A \cup B - A \cap B$

$B - A$

* Set()
* List()
* tuple()
* dict()

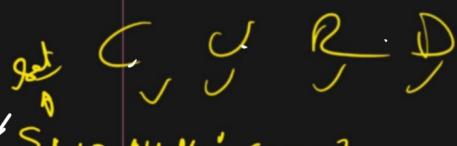


$A \cup B$

Set

Dictionary

- ✓ # Set() ← empty.
- ✓ # set contains only unique value.
- ✓ # set Contains only immutable values.
- ✓ # set is unordered.
- ✓ # Set object are mutable.



✓ {1, 2, 'Hello', (1, 2)} ✓

- # {} → empty.
- # {Pair1, Pair2, Pair3, ...}.
- # {Key1: Value1, Key2: Value2, ...}.

* Dict & Its sub objects are iterable
but can't subscriptable (indexing)

* Dict object is mutable



✓ { 'Name': 'Manish', 123: 'xyz' } ✓

Part - 1

W.A.P to create dict. of five family member Ages

Name	Age - Value
Name → Key.	

for

→ Plans Note: Two or more family Members belongs to same Age.

- ① Enter the 1. Name → Raj
Enter the Raj Age → 35
- ② Enter the 2. Name → Donylu
Enter the Donylu's Age → 35

Display → Data created

Part - 2 Searching W.A.P to Search Name By Age In dict.

while True

Search Name By Age In dict.

Age - value → Name → Key

Enter the Age to get Name / Names (Press q for quit) :- 35

Age

- ① Correct Data / Datatype
* Matched → with 1 Key →
* More than 1 Key →
- ② Incorect Data / Datatype → Not Matched! Try Again!
- ③ Press Q or q → Exit → Thankyou

The Functions

What a function is?

A function is a part of your code for doing some specific task and whenever you want to perform that task you can use that function. It provides your code a better abstraction. You can design your program using one function at a time. Let's say you want to detect the speed of a car. For that, you can create a function, say `getSpeedOfCar()`. Now, whenever you want to detect the car speed you just call this function and your task will be done. Once you have created your function you can use it any number of times. You are no longer required to write the same lines again and again.

FUNCTIONS:

Function is a sub program which consists of set of instructions used to perform a specific task. A large program is divided into basic building blocks called function.

Need For Function:

- ✓ When the program is too complex and large they are divided into parts. Each part is separately coded and combined into single program. Each subprogram is called as function.
- ✓ Debugging, Testing and maintenance becomes easy when the program is divided into subprograms.
- 1 Functions are used to avoid rewriting same code again and again in a program.

Modularity imposed

* Memory optimised →

- Script variable → Scope → global
- Function variable → Scope → local

def XYZ(
):

—
—
—

positional arguments

We have the following position of the Arguments -

P₁, P₂, P₃, P_n - - -

keyword arguments

: Allows to bypass the Argument order

Combo

P₁, P₂, P₃, K₈, K₆, K₇, K₉, K₅ - - -



non-default argument

ND₁, ND₂, ND₃ - - -

default argument

D₁, D₂, D₃ - - -

Combo

ND₁, ND₂, D₃, D₄, D₅ - - -

Ques - 1

```
Shapes:
1.Triangle
2.Rectangle
3.Circle
4.Square
Enter you choice: square
Enter side of square: 67
The area of square is 4489
```

```
Shapes:
1.Triangle
2.Rectangle
3.Circle
4.Square
Enter you choice: gdxvs
Invalid option!!!
The area of None is None
```

(1) def Area():

≡
f ← return Choice, Area
()

```
Shapes:
1.Triangle
2.Rectangle
3.Circle
4.Square
Enter you choice: 3
Enter radius of circle: 6
The area of Circle is 113.09733552923255
```

(2) def odd-even (n=5) :

```
Enter a number: 6
The value 6 is Even and factorial is 720.
```

math

return n, out, fact → factorial
odd Even

```
Enter a number: vci7809
Data Type:- <class 'str'> Is Invalid!..
The value vci7809 is None and factorial is None.
```

file

* doc string → """

≡
'''
def Area(,):
def odd-even():
 - calling
 -
 -

* Constants

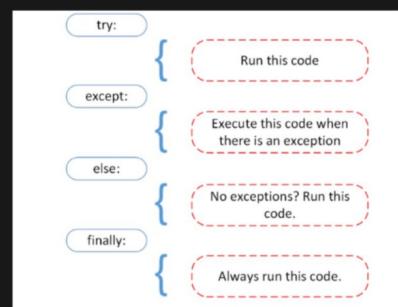
* definition (functions)

* definition (class)

script

{} >> calling of two class / func
{ } >>

Python Exceptions



dongli - 1. py

①

② -

③ -

④ -

```
1"""
2 This is User define modules with
3 1.Area
4 2.Odd_even
5
6 """
7 import time as t
8
9 def Area():
10
11 def odd_even(number='5'):
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
```

"The Python file Name use
as A module Name."

Calling - 1. py

✓ * import dongli_1
✓ * from dongli_1 import *

Python file as a module

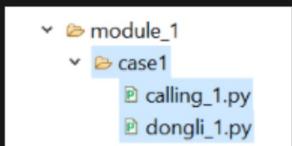
doc String.
Constants.
Func
Class
Task C)

if __name__ == '__main__':

Python file code (script)

Case - 2

When Both (calling - 1. py & module file) with In same directory



```
1# import dongli_1
2# print(dir(dongli_1))
3# print(dongli_1.__doc__)
4# print(dongli_1.t.sleep(2))
5# print(dongli_1.__file__)
6from dongli_1 import odd_even
7f=odd_even('8')
8print(f)
```

Namespace



```
file1.py
1 m = __name__
2 print(m)
```

When execute/run
By Its main file
then Its return
a string value.

```
file1.py
1 m = __name__
2 print(m)
```

File2
Its File Name as
string.

```
Console
<terminated> file2.py [C:\Python37123\python.exe]
file1
File Name :- file2.py
```

`'__main__'`

```
file1.py
1 m = __file__.split("\\\\")[-1]
2 print(f'This is {m} and namespace return:- ', __name__)
```

```
file1.py
1 import file1
2
3 m = __file__.split("\\\\")[-1]
4 print(f'This is {m} and namespace return:- ', __name__)
```



```
file1.py
1 import file2
2
3 m = __file__.split("\\\\")[-1]
4 print(f'This is {m} and namespace return:- ', __name__)
```

← file 3 executed

Output:

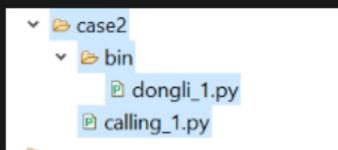
This is file1 and namespace return :- file1

This is file2 and namespace return :- file2
This is file3 and namespace return :- file3

: file2
: file3
: __main__

Case 2

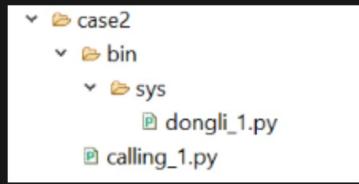
when the module file with in sub dir of calling file.



```

calling_1.py
1 from bin.dongli_1 import odd_even,Area
2 from time import sleep
3
4 print("=====calling_1.py File is")
5 f=odd_even('8')
6 print(f)
7 sleep(3)
8 a=Area()
9 print(a)

```



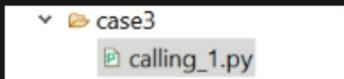
```

calling_1.py
1 from bin.sys.dongli_1 import odd_even,Area
2 from time import sleep
3
4 print("=====calling_1.py File is executed")
5 f=odd_even('8')
6 print(f)
7 sleep(3)
8 a=Area()
9 print(a)

```

Case 3

when module In some other direct drive etc.



```

calling_1.py
1 import sys
2 path1=r"C:\Users\OM\eclipse-workspace\01_22July_AmDocs\Day5\module_1\case2\bin\sys"
3 sys.path.append(path1)
4
5 from dongli_1 import odd_even,Area
6 from time import sleep
7
8 print("=====calling_1.py File is executed=====")
9 f=odd_even('8')
10 print(f)
11 sleep(3)
12 a=Area()
13 print(a)

```



While normal functions are defined using the def keyword, in Python anonymous functions are defined using the lambda keyword.

Syntax of Lambda Function in python

lambda arguments: expression

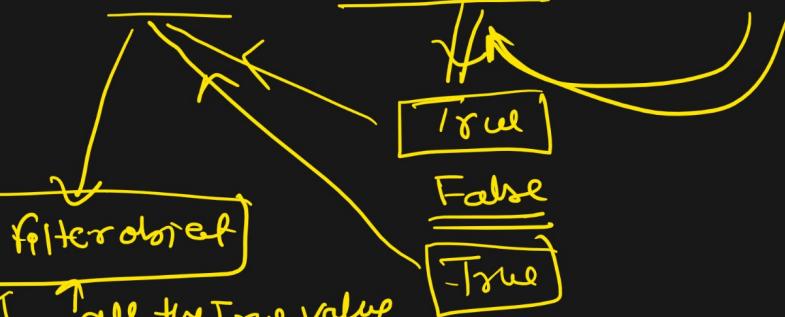
Lambda functions can have any number of arguments but only one expression. The expression is evaluated and returned. Lambda functions can be used wherever function objects are required.

One line function object
Anonymous

$$f(x,y) = x^2 e^{-y^2} + 2xy$$

Filter

>> filter(func-object, Seq)



Prob-1

W.A.P remove numbers from string using lambda() or filter() method .

Input Str:- I122N33D343I33@333 D454E34r3L444H67734I

Filtered Chars:- ['I', 'N', 'D', 'I', '@', ' ', ' ', ' ', ' ', ' ', 'D', 'E', 'r', 'L', 'H', 'I']

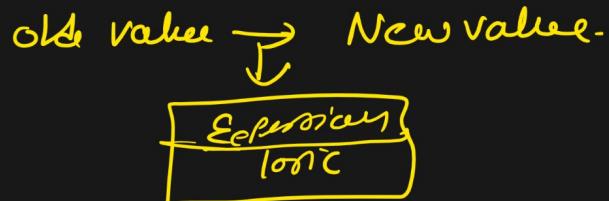
Final output:- INDI@ DErLHI

Prob-2

W.A.P remove any letter from string using lambda() or filter() method .

```
Enter string:-welcome to capital of india
Enter Letter to remove:-a
['w', 'e', 'l', 'c', 'o', 'm', 'e', ' ', 't', 'o', ' ', 'c', 'p', 'i', 't', 'i'
welcome to cpitl of indi
```

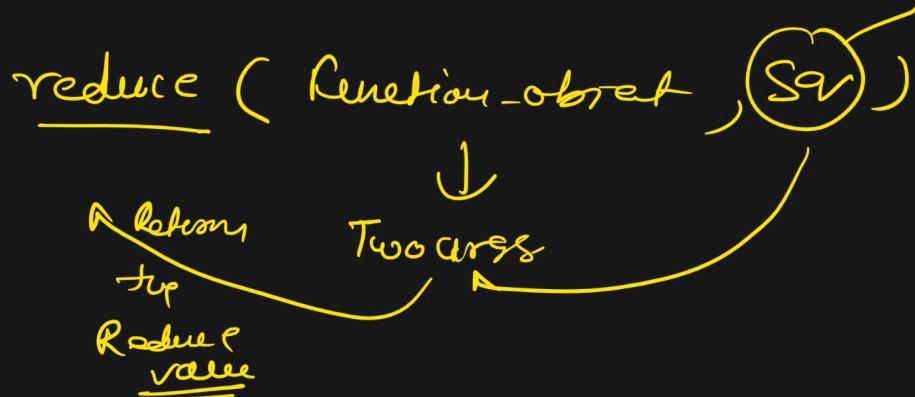
③ Map



map (func-object, Seq)



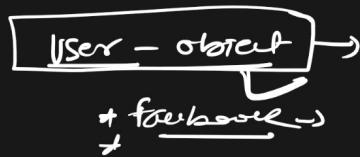
③ reduce



one value
→ +
→ 1
→ mean

Clears demo :
① var = 30
② def - -

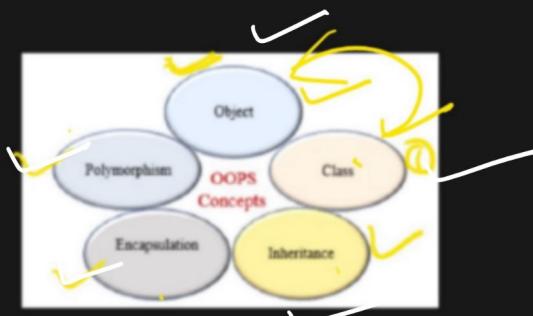
Day - 7
OOP's



What is Object Oriented Programming in Python

Object-oriented programming (OOP) is a programming paradigm based on the concept of "objects". The object contains both data and code: Data in the form of properties (often known as attributes), and code, in the form of methods (actions the object can perform).

An object-oriented paradigm is to design the program using classes and objects. Python programming language supports different programming approaches like functional programming, modular programming. One of the popular approaches is object-oriented programming (OOP) to solve a programming problem is by creating objects



Class and Objects



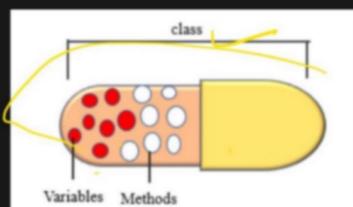
In Python, everything is an object. A class is a blueprint for the object. To create an object we require a model or plan or blueprint which is nothing but class.

Attribute Method

A class contains the properties (attribute) and action (behavior) of the object. Properties represent variables, and the methods represent actions. Hence class includes both variables and methods.

Encapsulation in Python

In Python, encapsulation is a method of wrapping data and functions into a single entity. For example, A class encapsulates all the data (methods and variables). Encapsulation means the internal representation of an object is generally hidden from outside of the object's definition.



Need of Encapsulation

Encapsulation acts as a protective layer. We can restrict access to methods and variables from outside, and It can prevent the data from being modified by accidental or unauthorized modification. Encapsulation provides security by hiding the data from the outside world.

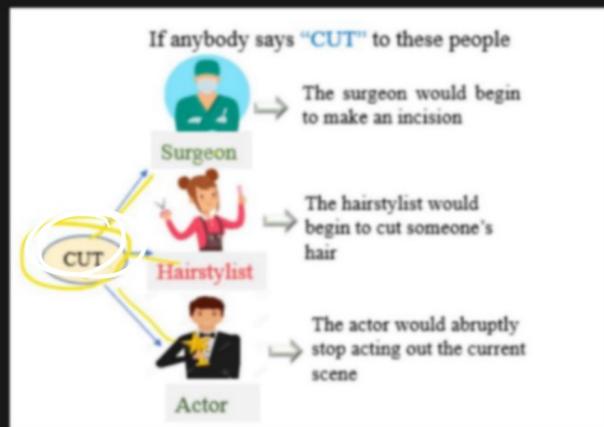
Polymorphism in Python

Polymorphism in OOP is the ability of an object to take many forms. In simple words, polymorphism allows us to perform the same action in many different ways.

Polymorphism is taken from the Greek words Poly (many) and morphism (forms). Polymorphism defines the ability to take different forms.

For example, The student can act as a student in college, act as a player on the ground, and as a daughter/brother in the home. Another example in the programming language, the + operator, acts as a concatenation and arithmetic addition.

Python Polymorphism



Class execute without calling/
creating any object.

Class demo :

* Class Demo :-

demo.var → 90

Class demo

var = 90

& Object Demo

obj1.var → 90

Obj1

var = 90 + 10 = 100

obj1.var = 10

Obj2

obj1.var = 900

var = 90

class Menu

Inside class | outside class
 $\Rightarrow \underline{\text{demo.var}}$ $\underline{\text{demo.var}}$

Class

+ var = 90
+ gold()

Obj

$\rightarrow \text{var} = \underline{100}$

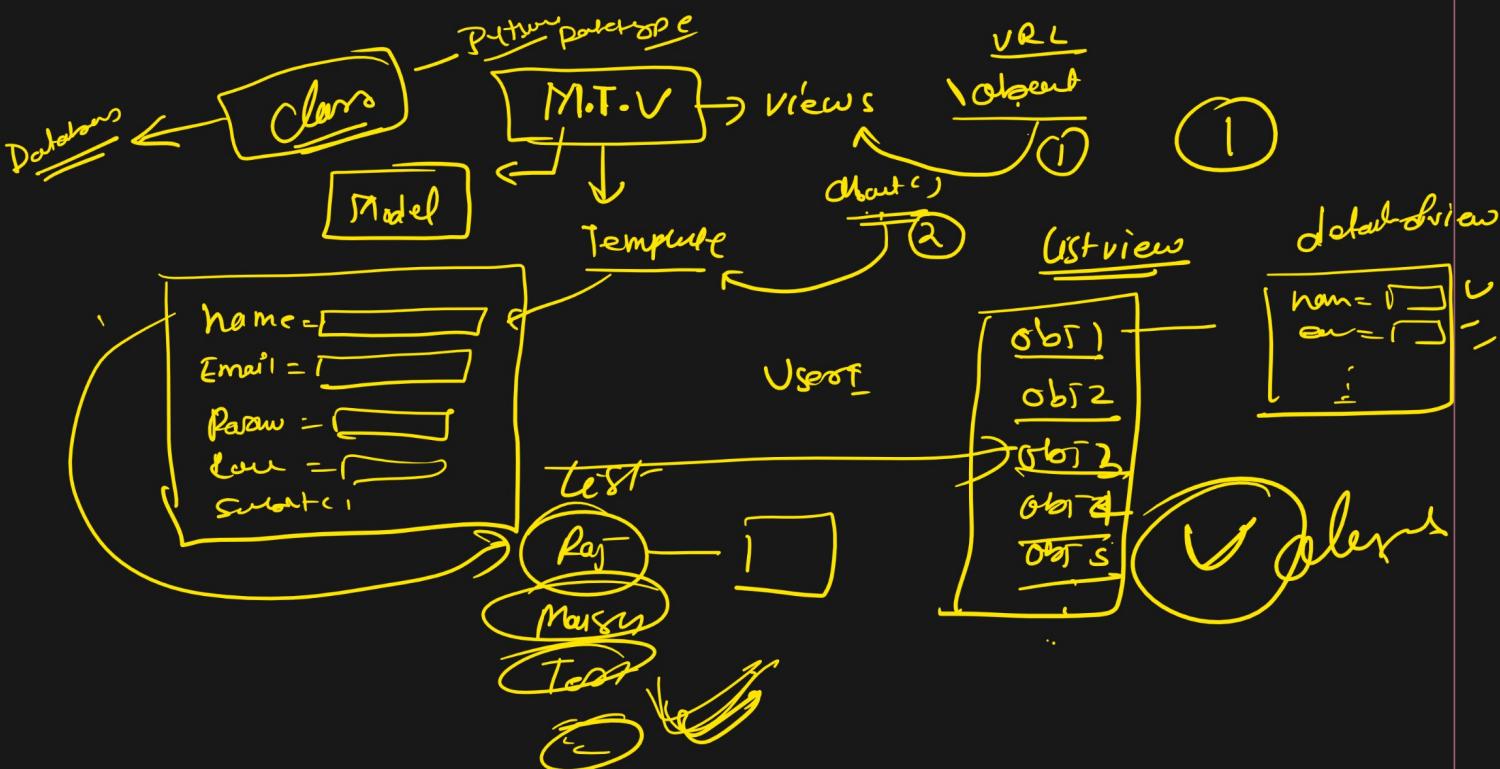
def gold()

Self.var = 100

demo.var = 90

object Menu(self)

inside class	outside class
$\star \underline{\text{self.var}}$	• $\underline{\text{obj1.var}}$
$\star \underline{\text{self.gold()}}$	• $\underline{\text{obj2.gold()}}$
	• $\underline{\text{obj3.add()}}$



0005-1

← 5

Exercise

We have a class defined for vehicles. Create two new vehicles called car1 and car2. Set car1 to be a red convertible worth \$60,000.00 with a name of Fer, and car2 to be a blue van named Jump worth \$10,000.00.

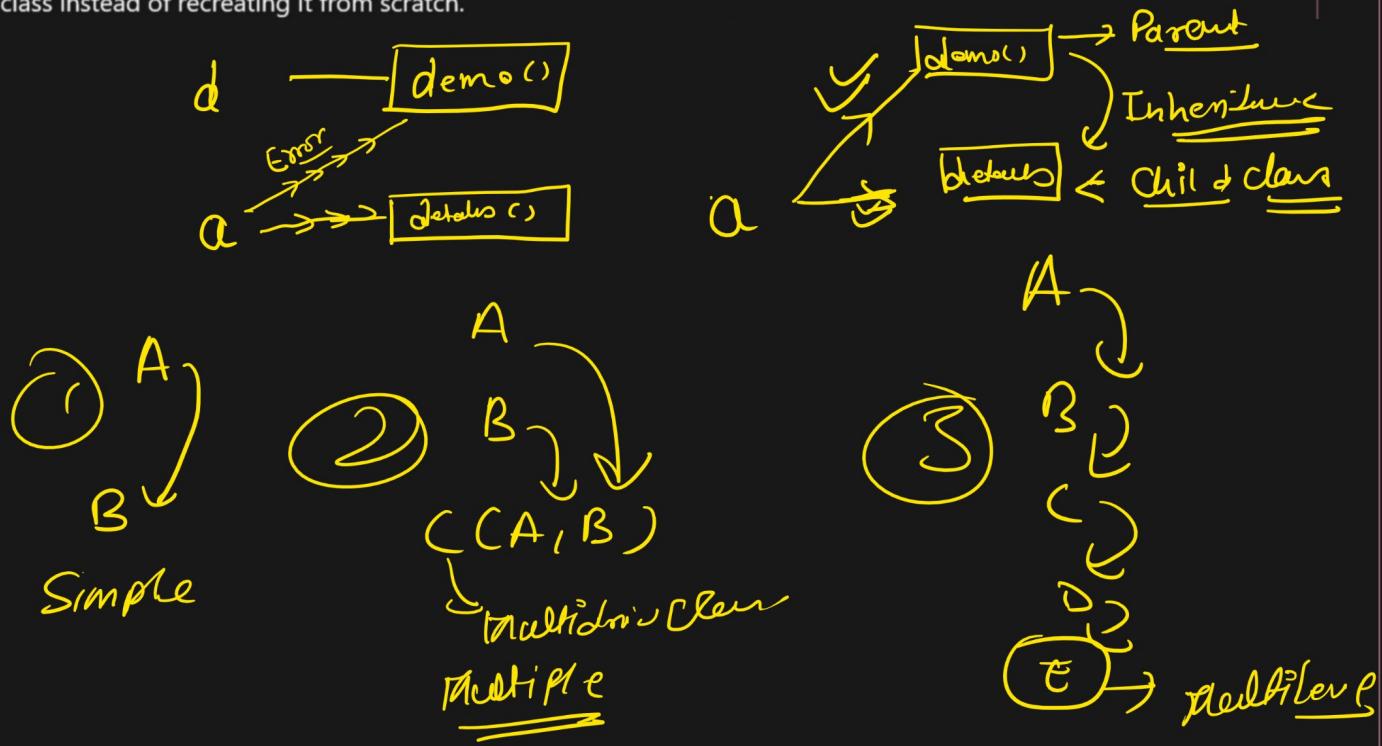
output

```
Console <terminated> D:\TechExperiential\python\Exercise Files\12 Classes\exercise1.py
Fer is a red convertible worth $60000.00.
Jump is a blue van worth $10000.00.
```

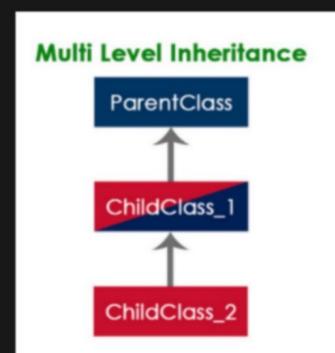
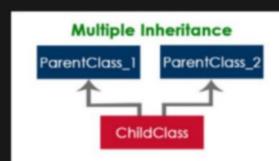
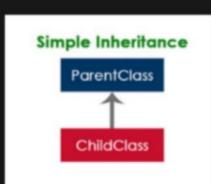
Inheritance In Python

In an Object-oriented programming language, inheritance is an important aspect. In Python, inheritance is the process of inheriting the properties of the parent class into a child class.

The primary purpose of inheritance is the reusability of code. Using inheritance, we can use the existing class to create a new class instead of recreating it from scratch.



types of inheritances



Access Modifiers in Python : Public, Private and Protected

Public

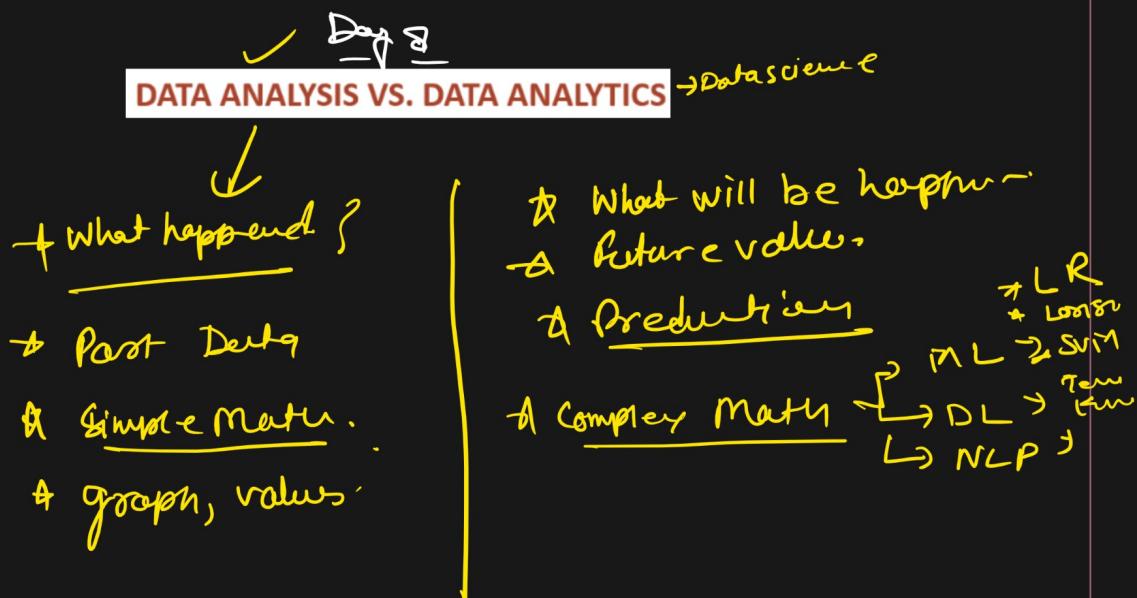
All members in a Python class are **public** by default. Any member can be accessed from outside the class environment.

Private

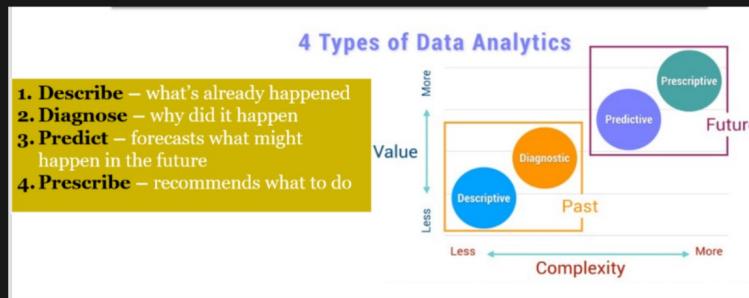
A **double underscore __ prefixed** to a variable makes it **private**. It gives a strong suggestion not to touch it from outside the class. Any attempt to do so will result in an `AttributeError`:

Protected

Python's convention to make an instance variable protected is to add a **prefix _ (single underscore)** to it. This effectively



No.	DATA ANALYSIS	DATA ANALYTICS
1	Data analysis is a process involving the collection, manipulation, and examination of data for getting a deep insight.	Data analytics is taking the analyzed data and working on it in a meaningful and useful way to make well-versed business decisions.
2	Data analysis helps design a strong business plan for businesses, using its historical data that tell about what worked, what did not, and what was expected from a product or service.	Data analytics helps businesses in utilizing the potential of the past data and in turn identifying new opportunities that would help them plan future strategies. It helps in business growth by reducing risks, costs, and making the right decisions.
3	In data analysis, experts explore past data, break down the macro elements into the micros with the help of statistical analysis, and draft a conclusion with deeper and significant insights	Data analytics utilizes different variables and creates predictive and productive models to challenge in a competitive marketplace.



Question:

The process of systematically applying techniques to evaluate data is known as ?

- A. Data Munging
- B. Data Analysis**
- C. Data Science
- D. Data Bases

- What is it?
 - Apply logical techniques to
 - Describe, condense, recap and evaluate Data and
 - Illustrate Information
- Goals of Data Analysis:
 - 1. Discover useful information
 - 2. Provide insights
 - 3. Suggest conclusions
 - 4. Support Decision Making

[Numpy](#) is the core library for scientific computing in Python. It provides a high-performance multidimensional array object, and tools for working with these arrays. If you are already familiar with MATLAB, you might find [this tutorial useful](#) to get started with Numpy.

```
pip install numpy
```

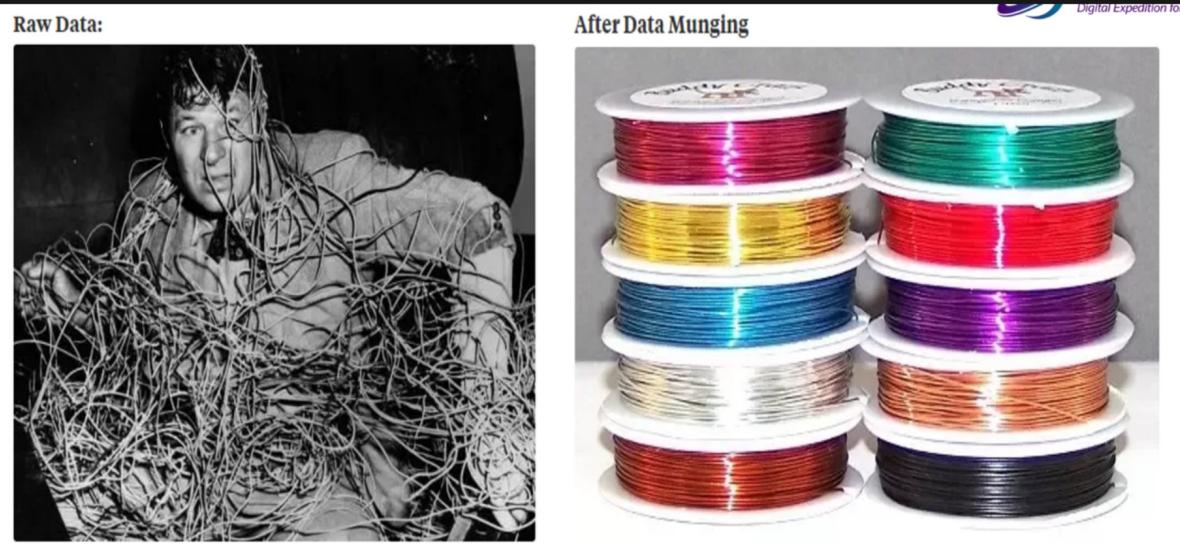
Arrays

A numpy array is a grid of values, all of the same type, and is indexed by a tuple of nonnegative integers. The number of dimensions is the *rank* of the array; the *shape* of an array is a tuple of integers giving the size of the array along each dimension.

Pandas is a python module that makes data science easy and effective

- ✓ Pandas is an open source, BSD-licensed library
- ✓ High-performance, easy-to-use data structures and data analysis tools
- ✓ Built for the Python programming language.
- ✓ Rich data structures and functions to make working with structured data fast, easy, and expressive.
- ✓ Built on top of Numpy with its high performance array-computing features.
- ✓ flexible data manipulation capabilities of spreadsheets and relational databases.
- ✓ Sophisticated indexing functionality slice, dice, perform aggregations, select subsets of data.

Process of cleaning messy data is called data munging or data wrangling



Series and DataFrames

The primary two components of pandas are the Series and DataFrame. A **Series** is essentially a column, and a **DataFrame** is a multi-dimensional table made up of a collection of Series.

Series $m \times 1$ **Series** $m \times 1$ **DataFrame** $m \times n$

	apples		oranges	
0	3		0	
1	2		3	
2	0		7	
3	1		2	

+

	apples		oranges	
0	3		0	
1	2		3	
2	0		7	
3	1		2	

=

	apples		oranges	
0	3		0	
1	2		3	
2	0		7	
3	1		2	

Practo-1



Conclusion Activity



Assignment for Today

Read 3 Excel Files data using pandas then write all three data frames into one new excel file in Three different sheets .

