Anaconda

Day-1-2-3-4

22-07-2024
23-07-2024
24-07-2024
25-07-2024

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.

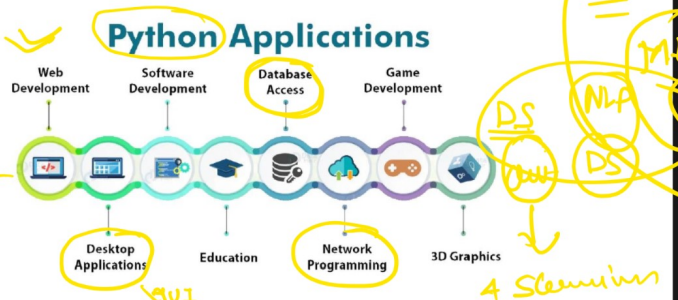
It offers efficient high-level data structures and an object-oriented programming style that is simple yet effective.

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.



AI IoT
web
ML
NLP
DL
DS
DS
SKlearn
ANN
Images
Video
Audio
Django
Flask
tkinter
PyQt
4 Selenium
BSA
testing
Pytest
Robert

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.

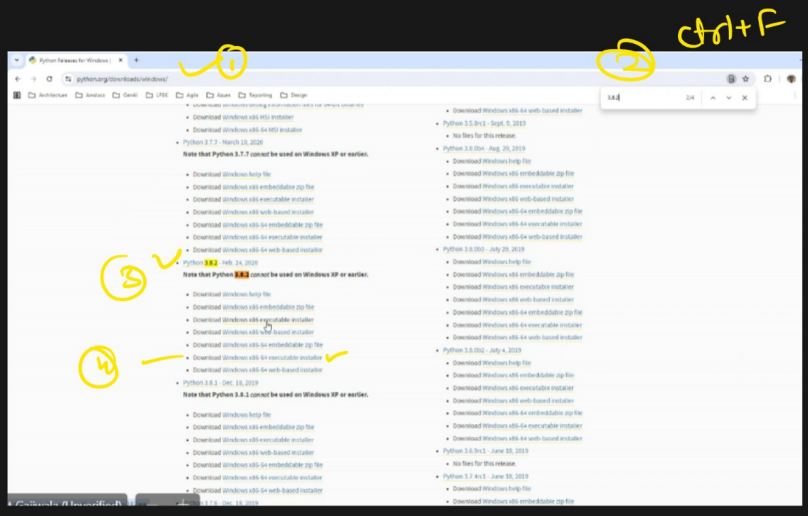
Who Developed Python Programming Language?

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.

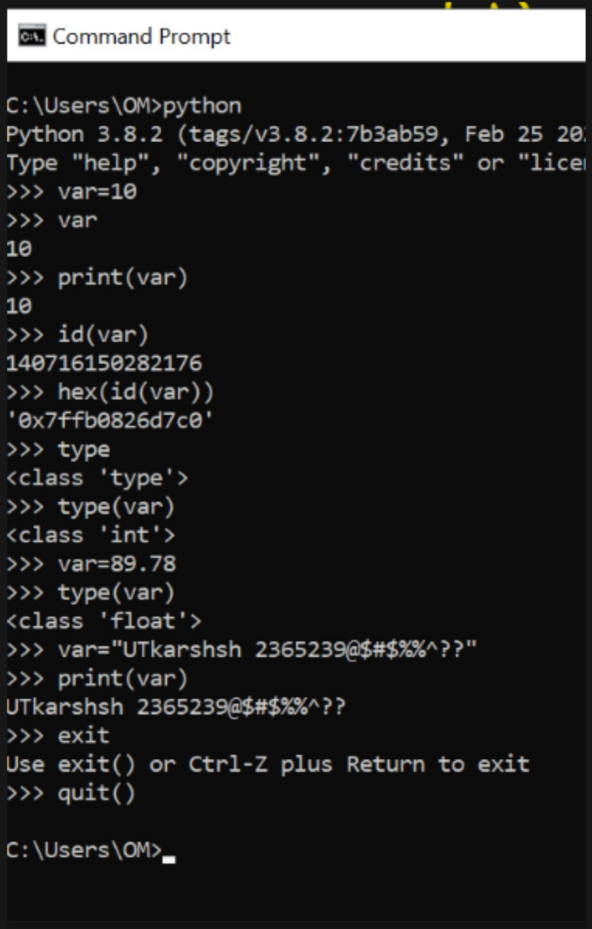
He began working on Python in the late 1980s while working at the Centrum Wiskunde & Informatica (CWI) in the Netherlands.



Env. Path's

1. C:\Python38 → Python.exe
2. C:\Python38\Scripts →
   - pip.exe
     - 43.P.P (online or web)
     - Pythonhosted.org
     - .whl file
   - easy_install.exe
     - ↳ offline

1. Int → 69
2. Float → 69.8
3. String → 'AbC 123'
   or
   "Nishat"

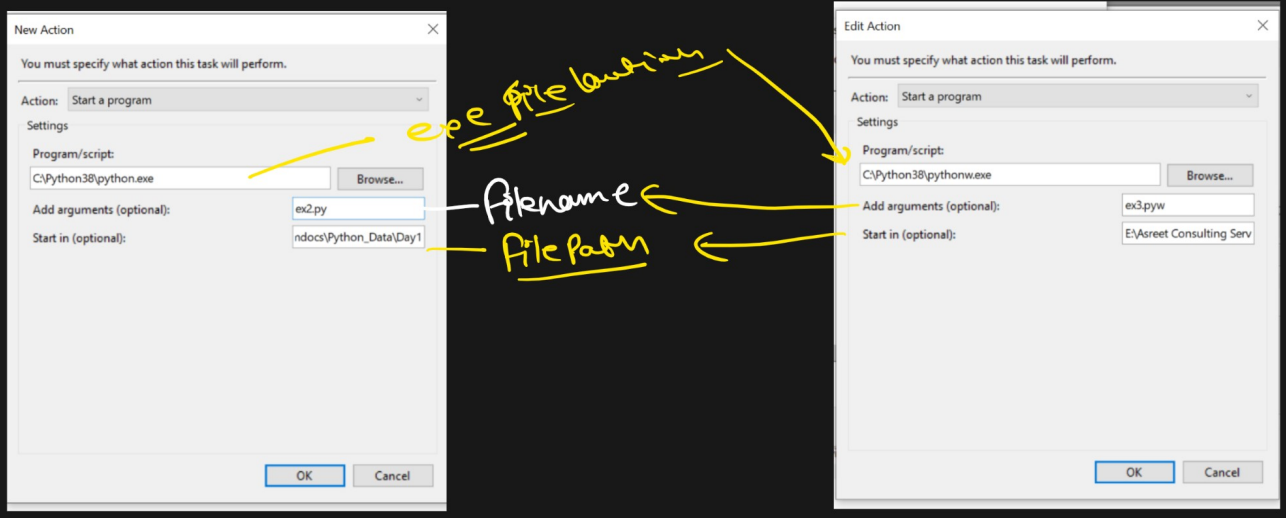


Create A Python Script ?

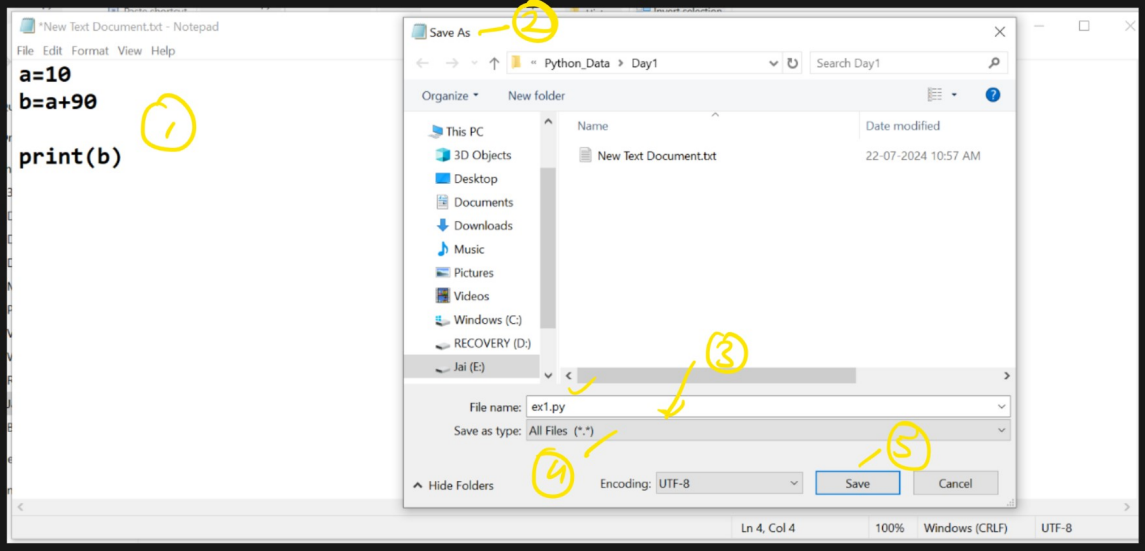
1. Text file (NotePad)
2. IDLE (GUI of Python)
3. IDE's (PyCharm, Anaconda, Eclipse etc)
4. text editors (VScode, Sublime etc)

→ .py → .pyw

.py	.pyw
* In IDE's :- No diff	No diff
Using Task Sch. → Create task with console * Input * output	* Create task without console * Run task in background.



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\OM>cd E:\Asreet Consulting Services\Agenda_amdocs\Python_Data\Day1

C:\Users\OM>e:

E:\Asreet Consulting Services\Agenda_amdocs\Python_Data\Day1>dir   ← list of dirs
 Volume in drive E is Jai
 Volume Serial Number is 0C69-0921

 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)  63,631,749,120 bytes free

E:\Asreet Consulting Services\Agenda_amdocs\Python_Data\Day1>python ex1.py
100   ← output
```

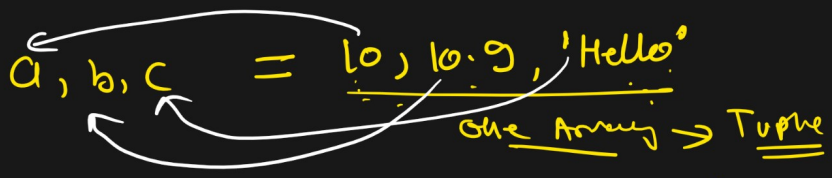
Multiple Assignment:

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:



One Array → Tuple

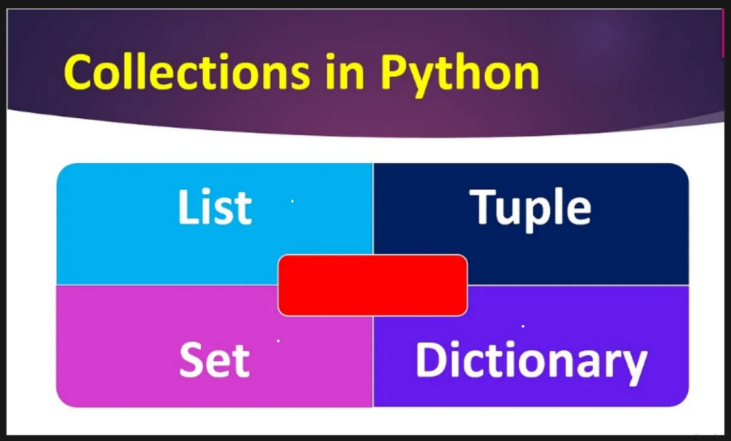
Unpack the values.

* len(var) = len(values)

Exception

one var = more than one val

Python Collections



* String
* Tuple
* list

} Indexing, Slicing, Seq's, ordered

* Set
* Dictionary

} unordered

Iterable
Loop it

Tuple

1. Tuple object can store diff. kind of DataTypes.
2. Tuple objects are ordered (Seq's) → Indexing, Slicing.
3. Tuple object are Iterable (Loop it).
4. Tuple object are Immutable → C ✓ U ✗ R ✓ D ✗
5. () → empty.

* (1, 1.2, 'Hello', (1,2))

```
#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))
5
```

Tuple

C U R D

C
- ☆ ()
- ☆ ('Hello',)
- ☆ (1,2,3,4)

U ✗

R
- ☆ Indexing. a[3]
- ☆ Slicing. a[::-1]
- ☆ Iteration (Loop it) for i in X:

D ✗

Indexing – Seq's → Storing Tuple

for only one value

+ve Indexing.
Left – Right
Starts with → 0

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

var = 'NISHANT'

	N	I	S	H	A	N	T
-ve	-7	-6	-5	-4	-3	-2	-1
+ve	0	1	2	3	4	5	6

L → R (+ve)

Cursor

var-2 = var[5] or var[-2]
→ 'N'
-ve Indexing

var[3] or var[-4]
→ 'H'
Middle chars
+ve Indexing

var[1] or var[-6]
→ 'I'
+ve Indexing

* AP → a1 and d { a1, a1+d, a1+2d, ... a1+nd. Slicing + Seq's → Str, List, Tuple. }

a : b : C → Stepsize

C → +ve a < b

C → -ve a > b

a → Start Index Number

b → End Index Number

* Last value

Stepsize +ve b - 1 <=

-ve b + 1 <=

Prob ①

-10 : -2 : -3

① Stepsize -ve

a > b → true.

-10 > -2 → False } No Execution

②

10 : 22 : 2

Step-1 Stepsize +ve

a < b → T

10 < 22 → T

Step-2 Last value

Stepsize +ve

b - 2 <= Last value

22 - 1 <=

21 <=

a1 a1+d a1+2d.

[10 12 14 16, 18, 20) 22

③ $-2 : -26 : -4$

① Step-1 Stepsize -ve

$a > b \rightarrow$ True.

$-2 > -26 \rightarrow$ True

② Step-2 Last

Stepsize -ve

$b+1 <= Lastval$

$-26+1 <=$

$-25 <=$

$a_1 \quad a_1+d$

$-2, -6, -10, -14, -18, -22$ ~~-26~~

```
a='NISHANT'
  0 1 2 3 4 5 6
b= a[2:len(a):2]
```

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

b='SAT'

var='NISHANT'

var[: :3]

c = 3

a = 0

b = len(var)

+ve stepsize

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

Python Identifiers

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.

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
Class Name
Func Name
Python File Name
Directory Name (folder)

[A-Z a-z 0-9 _]

[A-Z a-z _] [A-Z a-z 0-9 _] ....

1st char

1Var → X

Var1234 → ✓

_1234 → ✓

CVar → 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 Nisut'

"I am Nisut"

\n newline

\t tab space.

\ → escape Chars

""" """

''' '''

Concatination In Seq's

Str + Str → Str

tuple + tuple → tuple

List + List → tuple

int() → int 6 → 6

float 6.2 → 6

String → ✓ '1436' → only digits.

↳ 1436 ∈ int

* Rest → Error

Otherwise

↳ Error

* float( ) (i) 6 → 6.0

(ii) 6.2 → 6.2

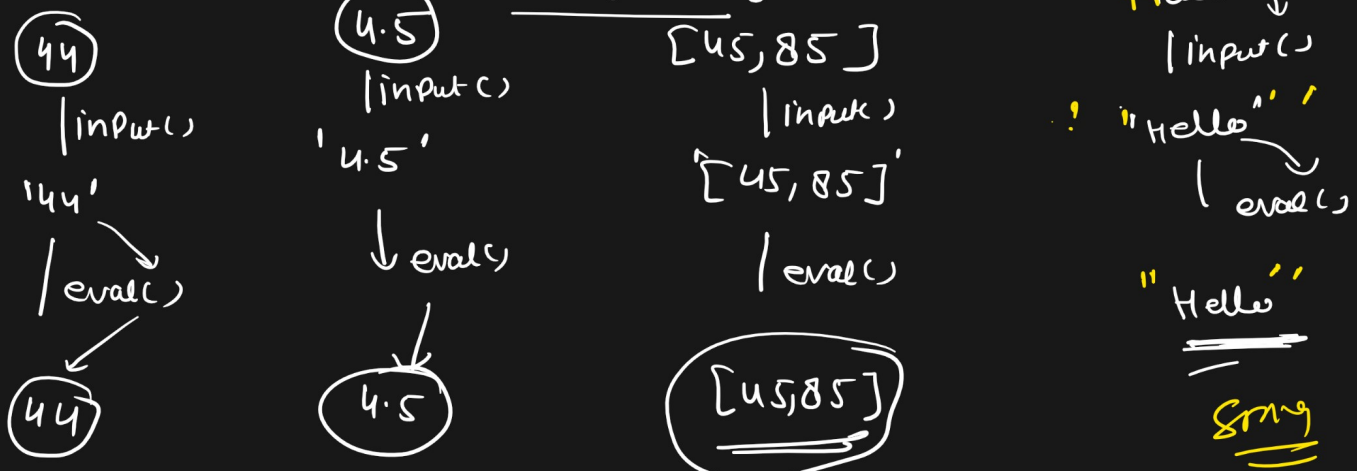
(iii) String → * '1345' → only digits → 1345.0

* '1345.45' → 1345.45

Rest → Error

(iv) Other data types → Error.

eval(String)



Pkg → Modules → Constant's, Func, class, ToolC( )

① 3PP → online/offline
② Inbuilt
③ Userdefined

→ import
→ from

* numpy → numpy
* regex → re
* Pandas → Pandas.
* datetime → datetime → datetime

Function name

(A) import

(1) import moduleName

>> import math

* A module import as a directory with same Name
* Need Refrem to call them.

math. Pi
math. sin( )

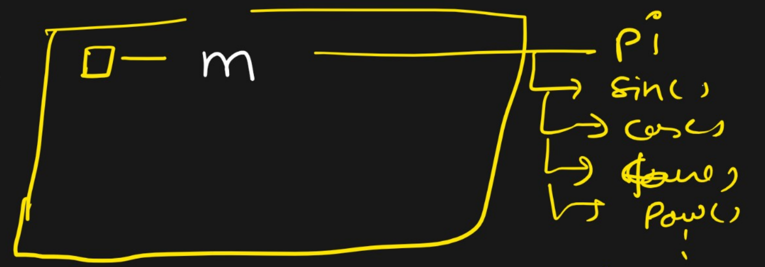
* Memory wasted.

(A) import

* rename
* short name

(i) import moduleName as newname

> import math as m



* module import as a directory with New Name

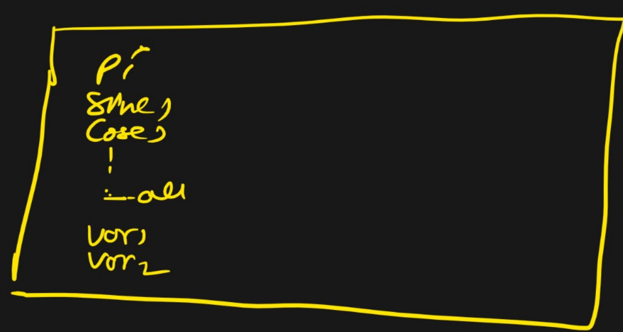
* Need Refrence to call them.
  m.pi
  m.sin()

* Memory wasted.

(B) from

Everything
↓
(i) from moduleName import *

from math import *

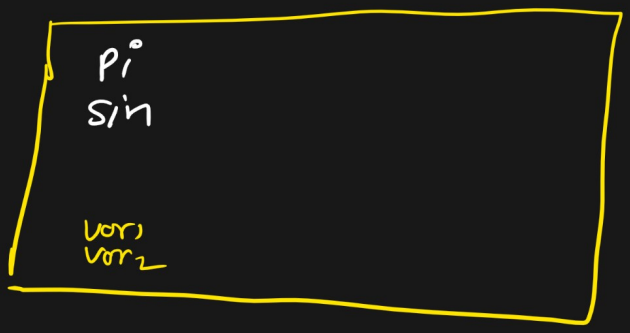


* No directory created.
* No refrence required
* Memory still wasted

(B) from AAAA

(ii) from moduleName import x1, x2

from math import pi, sin



* No directly created.
* No return required
* No memory wasted.

import	from
* import math	* from math import x1, x2
* Module get imported. So Need return to call / use constants, functions etc.	* Imports only Fun's, Class's, Constants. Do No Need return
math.pi	pi
math.sin	sin()
* Everything imported so memory wasted	* We imported only required. So memory imported.

Multi-Array

Numpy → Sin → one → one
→ Array → Array.

Maths → Sin → one → one

Random

① randint(a, b)

$a <= N <= b$ → Random Integer values

Integer value

② random()

$0.0 <= N <= 1.0$

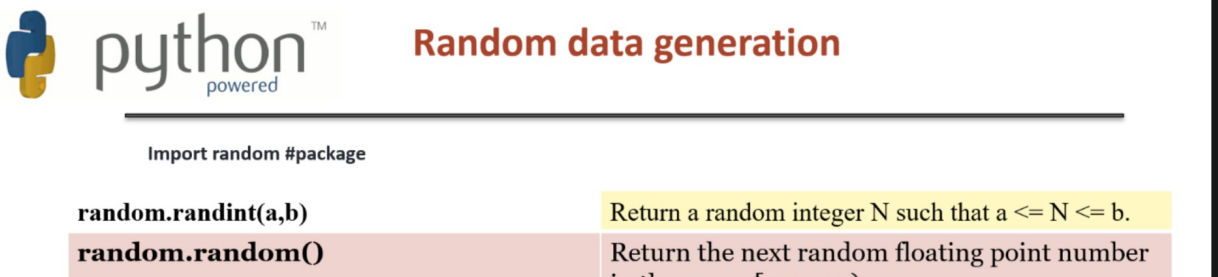
↳ Random float value.

③ Uniform(a, b)

$a <= N <= b$

$a >= N >= b$

Random float values



random.randint(a,b)	Return a random integer N such that a <= N <= b.
random.random()	Return the next random floating point number in the range [0.0, 1.0).
random.uniform(a, b)	Return a random floating point number N such that a <= N <= b for a <= b and b <= N <= a
random.randrange(start, stop[, step])	
random.choice('abcdefghij')	Choose a random element
random.shuffle(items)	
random.sample([1, 2, 3, 4, 5], n)	Choose n random sample

randrange(a, b, c)

a : b : c

10 : 22 : 4

[10, 14, 18, 22]

10
14
18
10

1.2GB Songs. → 1.2GB

Yes
No

Same list

['C:\New\Desktop\01.Kabir.mp3', 'D:\Test\02.mp3', 'G:\03.mp3']

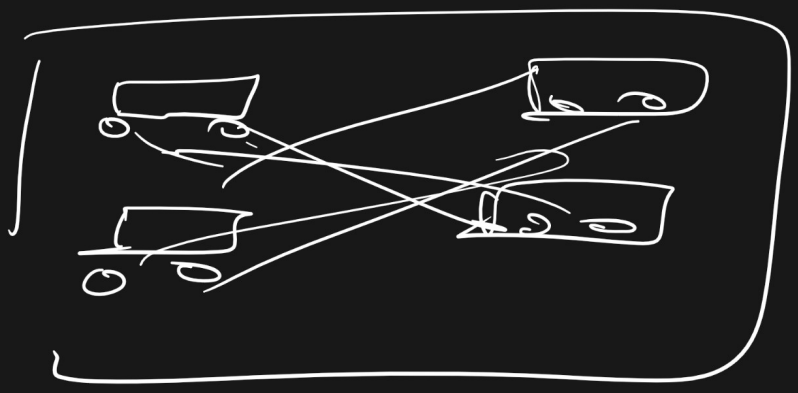
['E:\03.mp3', 'C:\New\Desktop\01.mp3', 'D:\Test.mp3']

Same list

Yes
None

Self (Yes)

It self updated



Not to Use 8

python powered

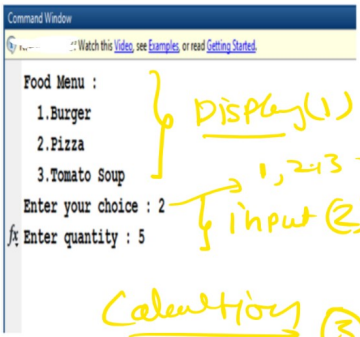
Write First Python Script By Today Learning Only

P. 3

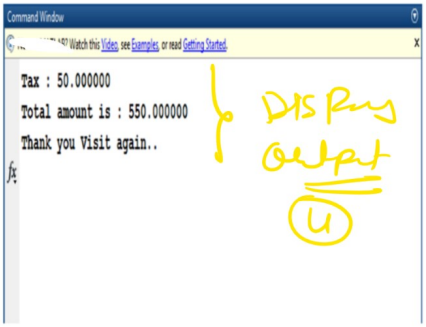
W.A.P for following

Amount per item
Burger =30
Pizza=100
Tomato Soup= 40
sgst= 2.5%
cgst= 2.5%

Input Window



Output Window

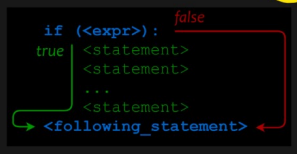


Day-3

conditional statements in python

0 indent

if EXP : ——



① if EXP :

② if EXP :

else :

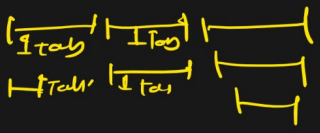
③ if EXP1 :

elif EXP2 :

elif EXP3 :

else :

if EXP2 :



else :

Prob 1 W.A.P to generate Random Integer No blw 0 - 100 and follow the below conditions.

(A) 10 < Value < 60

(i) Value > 43

The value X is large :

Random value

(ii) 35 < Value <= 43

The value X is Medium !

The value X is small.

The Value X out of Range !

Loops

C++

```
for (i=0; i<=10; i++)
{
}
```

i = loop_variable

Datatype = Static = int

Total_No_Iterations = 11

Python

Iter_object :- Str, tuple, list, zip, Dict, set, filter, map, etc.

```
for loop_var in Iter_object :
```

ADatatype = Dynamic = Any type

Total_No_Iterations = len(Iter_object)

range(b)

a = 0

b = b

c = 1

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 → 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 the odd Input's → Abc123

for loop ②

5! → 1×2×3×4×5 → 120

L5 → 5×4×3×2×1 → 120

while loop

1! → 1

0! → 1

+/- → 0 or any other value.

#/1 → 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.

var = [48, 09, 61, 49]

9.9 in var. = True

61 not in var → False

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.

Memory Addrs of Two objects

obj1 is obj2 → True

a = 5

b = 5

a is b → True

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

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

a is b → True

b

a

Int
Float
tuple
Strings.
} Immutable

value == memory Address

< >

Same value
same order
Immutable

If value same then the Address same

∞

While

while Cond :

After use of the variable c

c++ → Post ↑

++c → Pre ↑

→ c = c + 1

Before use of the variable c

c-- → Post ↓

--c → Pre ↓

→ c = c - 1

True

a < b

a > b

a <= b

a >= b

a == b

a

List

- Diff Kind of Data types Can store
- Ordered → Seq's → Indexing. / Slicing.
- Iterable Object
- Mutable → C U R D

[ ] → empty.

['Hello'] → one value List

['Hello', 1, 1.2, 'Test']

---

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

List

C
- [ ]
- ['Hello']
- [1, 2, 'Hello']

U
- a[3] = NewValue.
- a.append(Object)
- a.insert(Index, Obj)
- a.extend(Iter-obj)

R
Indexing
- a[3]

Slicing.
- a[0::2]

Iteration
- for i in a: print(i)

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

`a.append(object)`

Object → int, float, Iterobject → str, list, tuple, map, set, dict, zip, filter 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]] — 4

`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] — 4

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

`a.extend(iterable)`

Str, List, Set, Tuple, Dict, Zip, Filter, Map etc.

a = [1,2,3] — 3

a.extend((45,))

→ [1,2,3,45] — 4

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))
```

```
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))
print("==========delete========")
#create a list of Value "Test" index numbers . ? Dynamically → Grp
#[0,2,4....]
#Delete all the "Test" Values from given list ?
```

0 2 4 6 8 9 11

output

(1) [0,2,4,6,8,9,11]

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, - - - - }

Pair1 Pair2 Pair3

dict → Its subobject doesn't support Indexing & Slicing.

dict → Its objects are Iterable - Loop it

dict object are Mutable

C U R D

Dict {}

C U R D

d={'name':'Manish',('dept','other'):[["L&D"],[1,2,3]
4576563447.78:'%*#%(*(sghdh$&98855JBJ8^%$)'}

(1) {} → Key should be Immutable → Str, Int, Float, Bool, Tuple

(2) dict → key should be Identifier.

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

* d[Key]=NewValue.
* d[NewKey]=NewValue.
* d.Update(d1)
  → {kkd, dtkd1}

v= d[Key]

v= d.get(Key, default)

# Iterate: for i in d: # Keys
print(i)

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

del d[Key]

* d.pop(Key, default)

```
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))
```

name → str

float

→ Immutable

int, float, str, tuple → Immutable

Key as An Array?

* Tuple
* List

[A-Z a-z 0-9 -]

[A-Z a-z .] [A-Z a-z 0-9 -] - - -

* Numeric Key ✓
* Key as Array ✗

* set()
* list()
* tuple()
* dict()

A∩B

A - B

B - A

A∪B - A∩B

Set	Dictionary

set() ← empty.

set contains only Unique value.

set Contains only Immutable values.

set is using unordered.

* set object are mutable

set: C U R D

{1, 1.2, 'Hello', (1,2)}

{} → empty.

* {pair1, pair2, pair3, - - - -}.
* {key1 : value1, key2 : value2, - - - -}

* Dict → Its sub object are iterable but can't subscriptable – Indexing slicing
* Dict object is mutable

C U R D

{'name' : 'Manish', 123 : 'XYZ'}

Prob - Dict

W.A.P to create dict. of five family member Age & Name

PART-1

Age – Value

Name → Key.

For

→ Flow

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 → Donru

Enter the Donru's Age → 35

Display → Data created !

PART-2

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

While True

Cater 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 →

→ with more than 1 Key →

* Not matched →

② Incorrect Data / Datatype → Handle It → 'Not Matched ! Try Again !'

③ Press 'q' or 'Q' → Exit → Thank you

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.
- Functions are used to avoid rewriting same code again and again in a program.

Modularity improved.

* Memory optimised →

→ Script variable → Scope → global

→ function variable → Scope → local

def XYZ_1( ):
—
—
—

positional arguments We have the following the position of the Arguments.

$P_1, P_2, P_3, P_4 - - -$

keyword arguments : Allow the Bypass the Arguments orders

K_4 K_3 K_2, K_1 - - -

Combo

P_1, P_2, P_3, K_8, K_6, K_7, K_4, K_5 - - - -

non-default argument

ND_1, ND_2 ND_3 - - -

default argument

D_1, D_2, D_3 - - -

Combo

ND_1, ND_2, D_3, D_4, D_5 - - -

Prob-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
```

(1) def 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
```

return Choice, area

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

(2) def add_even (n='5'):

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

return n, out, fact → factorial

math

odd Even

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

file

- docstring → ''' '''
- Constants
- definition (functions)
- definition (Class)

'''
- def Area( ):
- def add_even( ):
- Calling

Script

>>
>>
Calling of the class / Fun