

PythonNotes

BasicsToAdvanced

UNSOLVED

2 References for all codes and CW notes : [Python Notes – OkItsNotOk \(unaux.com\)](https://unaux.com/python-notes/)

3

4

5 February 8, 2023

6 1.

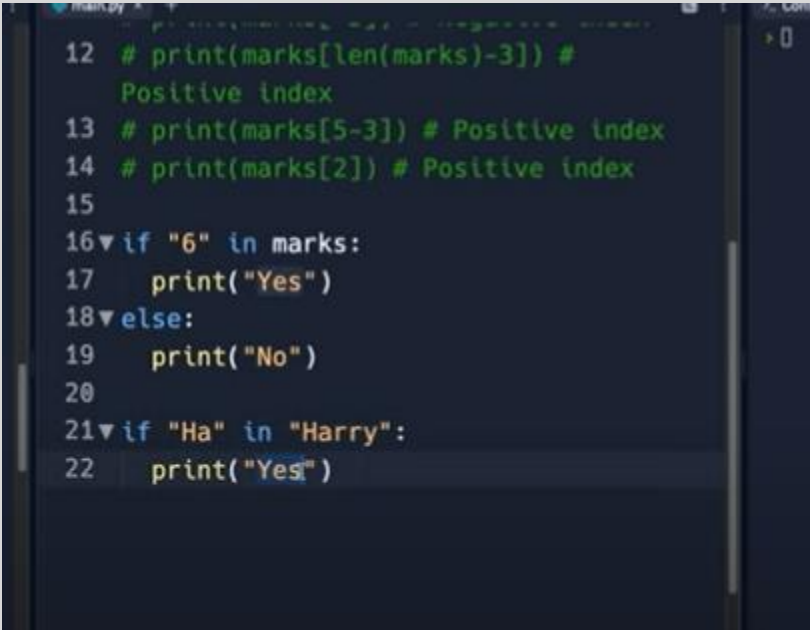
7 UNSOLVED — 02/08/2023 1:46 PM

8 length of list is simply no of elements in the list. elements can be numbers , strings, boolean or another list too

9 February 9, 2023

10 2.

11 UNSOLVED — 02/09/2023 1:35 AM



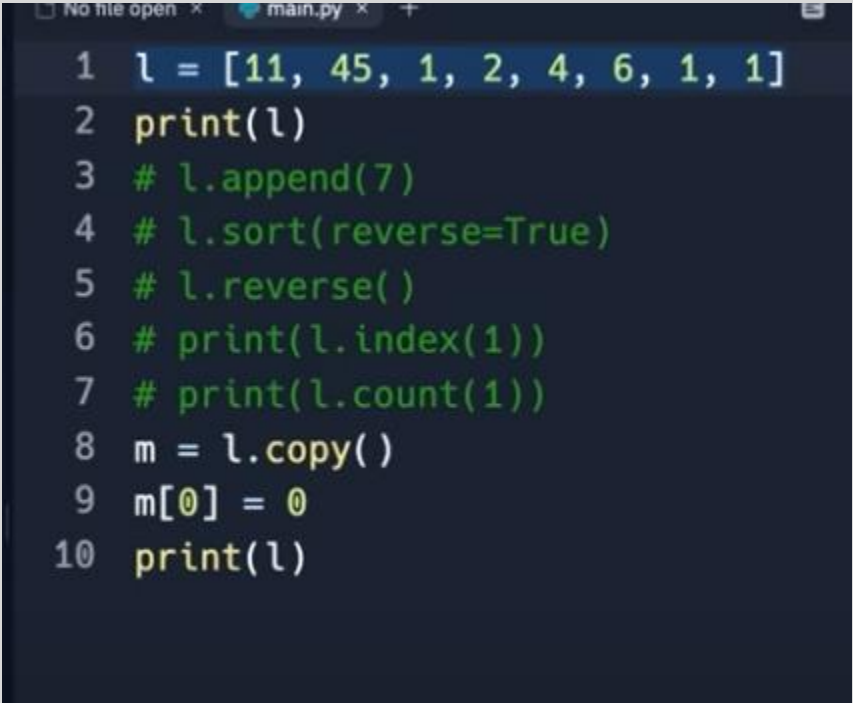
12

13 3.

14 UNSOLVED — 02/09/2023 7:40 PM

15 if m = l if we do m becomes a reference of l i.e. its just like a name if we change elements of m , it gets changed in l too if

16 we do m = l.copy() then it won't effect l and only the changes happen in m



17

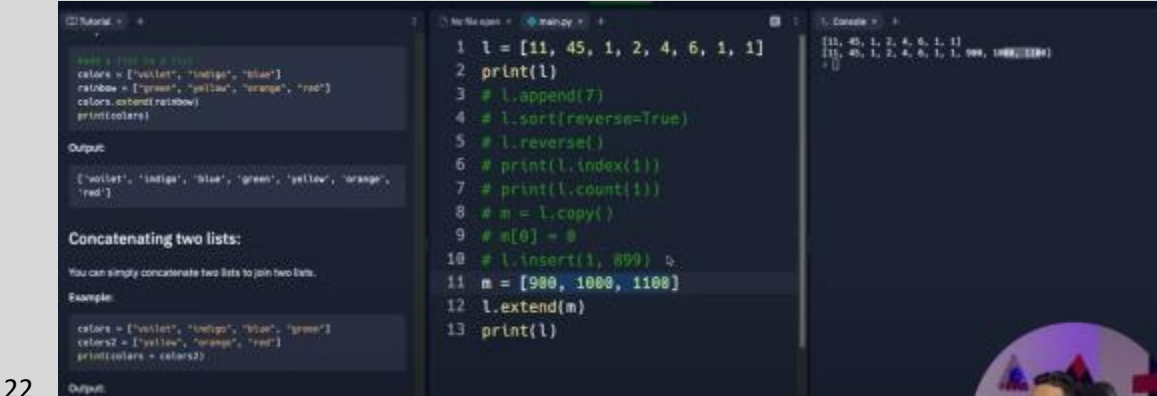
18 4. [7:41 PM]



19

20 5. [7:42 PM]

21 extends means add the list to list of the specified in the end



6. [7:44 PM]

if we want to create a list with l and m we can do : l.extend(m) this changes the list l and extends i.e. l gets effected
whereas if we do k = l + m it does the same thing but doesn't effect the list l

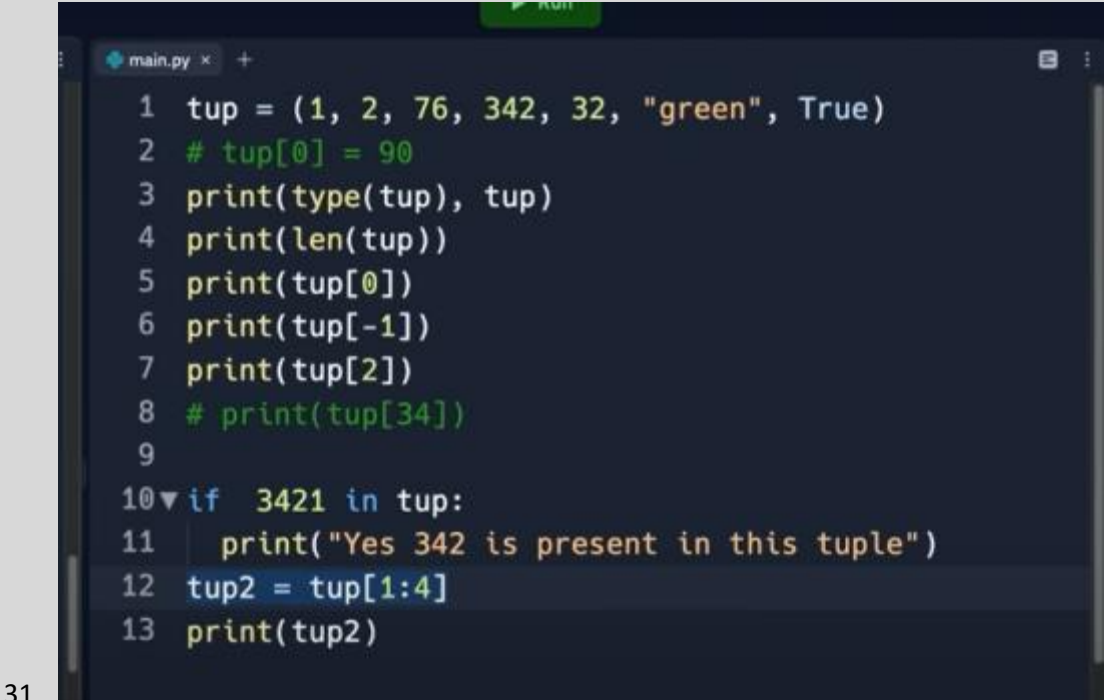
7.

UNSOLVED — 02/09/2023 7:54 PM

tuple with one number as the element it says its class is integer to say it is a tuple write (1,)

8. [7:59 PM]

if we do slicing and assign it to something it gets created as a new tuple



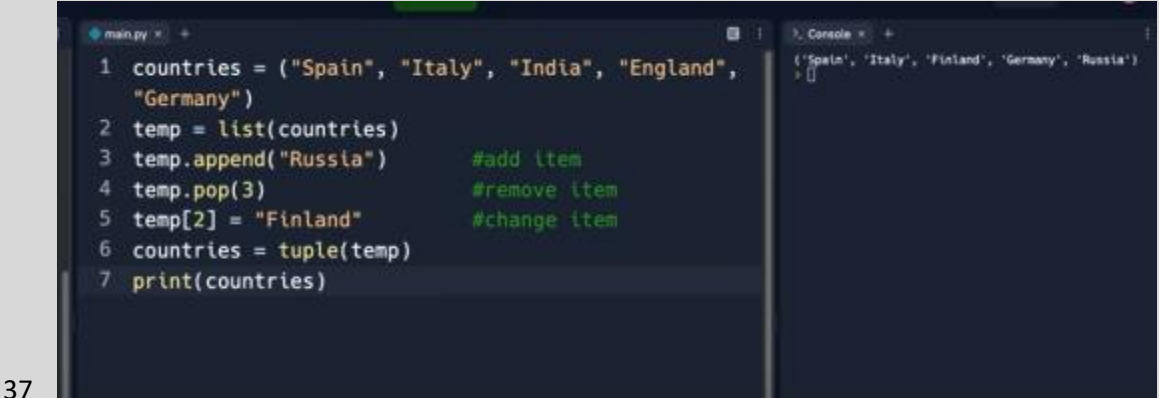
9.

UNSOLVED — 02/09/2023 8:03 PM

if we want to do anything to tuple we have to make a list make changes and make it tuple again

10. [8:04 PM]

like this



11. [8:09 PM]

count() in tuple in index() we can also specify the area to search (whattosearch, from , till) but it searches till till-1 only
(like always) (edited)

12.

UNSOLVED — 02/09/2023 8:18 PM

if sathvik = {} and if I print type(sathvik) it says dictionary even if you thought it is going to be a set

13. [8:18 PM]

to create a set do sathvik = set() or sathvik = {some-elements}

14. [8:23 PM]

s1.union(s2) just prints the union set s1.update(s2) updates the set s1 with the union of s1,s2 (i.e. it gets changed)

15. [8:24 PM]

if we create a set = set1.intersection(set2) set becomes a set with only those intersection elements

16. [8:24 PM]

if we want to update also do set1.intersection_update(set2)

17.

UNSOLVED — 02/09/2023 8:27 PM

set1.isdisjoint(set2) disjoint meaning is the sets have no element in common the function prints True if there are no elements in common the function prints False if there are is one or more elements in common

18. [8:29 PM]

if set2 is a subset of set1 then set1 is called the superset of set2 set1.issuperset(set2) meaning is is set1 superset of set2
prints true or false

19. [8:29 PM]

the same way there is a function called issubset()

20.

62 UNSOLVED — 02/09/2023 9:11 PM

63 there is add() function if we want to add more than one item update() remove() / discard() pop() = usually removes the
64 last element of the set since set is unordered we don't know what element it removes

65 21. [9:11 PM]

66 del set deletes the set and undefines it

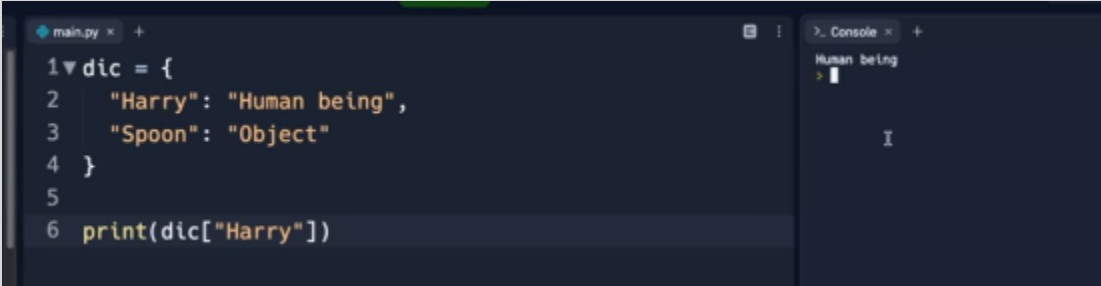
67 22. [9:12 PM]

68 if we don't want to delete the entire set but only want to remove all the elements we use clear()

69 23. [9:12 PM]

70 eg. set.clear()

71 24. [9:14 PM]



72

73 25. [9:16 PM]

74 in a way they are the same thing but if we do info["something"] since something is not there in the set it throws an error
75 but if we do info.get("something") we get None as the output (edited)



76

77 26. [9:18 PM]

78 to access keys do print (info.key()) to access values do print(info.values()) or run a for loop like for key in info.keys() print
79 (info[key]) (edited)

80 27.

81 UNSOLVED — 02/09/2023 9:19 PM

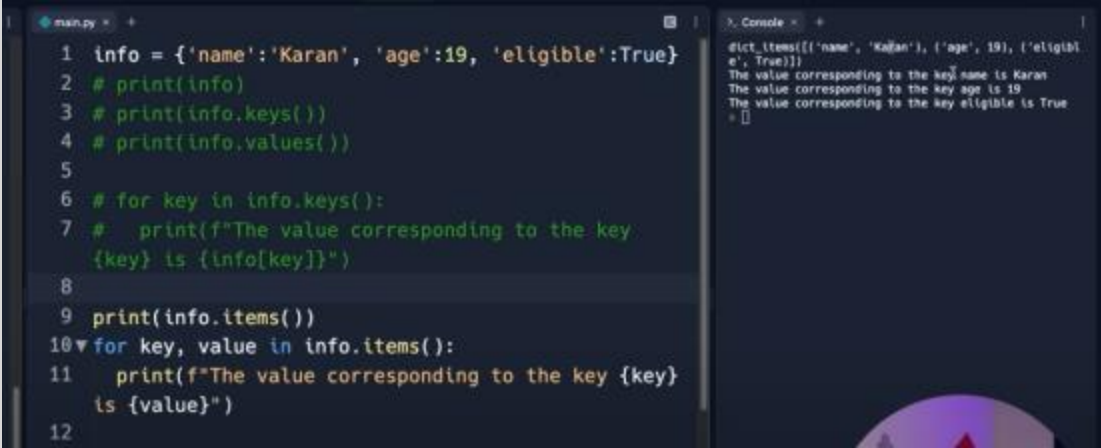
82 or the ultimate



83

84 28. [9:21 PM]

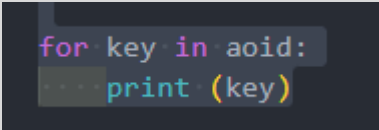
85 sets



86

87 29. [9:24 PM]

88 to get key names to get keys/values or keys and values do key,value in set.items() and run a f string



89

90 30. [9:26 PM]

91 ----- FUNCTIONS -----

92 31.

93 UNSOLVED — 02/09/2023 9:28 PM

94 FOR LIST :

- `sort()`: Sorts the list in ascending order.
- `type(list)`: It returns the class type of an object.
- `append()`: Adds one element to a list.
- `extend()`: Adds multiple elements to a list.
- `index()`: Returns the first appearance of a particular value.
- `max(list)`: It returns an item from the list with a max value.
- `min(list)`: It returns an item from the list with a min value.
- `len(list)`: It gives the overall length of the list.
- `clear()`: Removes all the elements from the list.
- `insert()`: Adds a component at the required position.
- `count()`: Returns the number of elements with the required value.
- `pop()`: Removes the element at the required position.
- `remove()`: Removes the primary item with the desired value.
- `reverse()`: Reverses the order of the list.
- `copy()`: Returns a duplicate of the list.

95

96 32. [9:29 PM]

97 FOR TUPLE

tuple. Therefore, we can call them tuple functions. Furthermore, these tuple functions make our work easy and efficient. Besides, there are a number of functions such as `cmp()`, `len()`, `max()`, `min()`, `tuple()`, `index()`, `count()`, `sum()`, `any()`, `all()`, `sorted()`, `reversed()`.

98

99 33. [9:29 PM]

100 FOR SET

tuple. Therefore, we can call them tuple functions. Furthermore, these tuple functions make our work easy and efficient. Besides, there are a number of functions such as `cmp()`, `len()`, `max()`, `min()`, `tuple()`, `index()`, `count()`, `sum()`, `any()`, `all()`, `sorted()`, `reversed()`.

101

102 34. [9:29 PM]

103 FOR DICTIONARY

Functions Name	Description
<code>clear()</code>	Removes all items from the dictionary
<code>copy()</code>	Returns a shallow copy of the dictionary
<code>fromkeys()</code>	Creates a dictionary from the given sequence
<code>get()</code>	Returns the value for the given key
<code>items()</code>	Return the list with all dictionary keys with values
<code>keys()</code>	Returns a view object that displays a list of all the keys in the dictionary in order of insertion
<code>pop()</code>	Returns and removes the element with the given key
<code>popitem()</code>	Returns and removes the key-value pair from the dictionary
<code>setdefault()</code>	Returns the value of a key if the key is in the dictionary else inserts the key with a value to the dictionary
<code>update()</code>	Updates the dictionary with the elements from another dictionary
<code>values()</code>	Returns a list of all the values available in a given dictionary

104

105 35.

106 UNSOLVED — 02/09/2023 10:19 PM

107 we can convert numbers, strings , tuples to list using `list()` function `list(123)` makes a list with 1, 2 and 3 as the elements

108 February 11, 2023

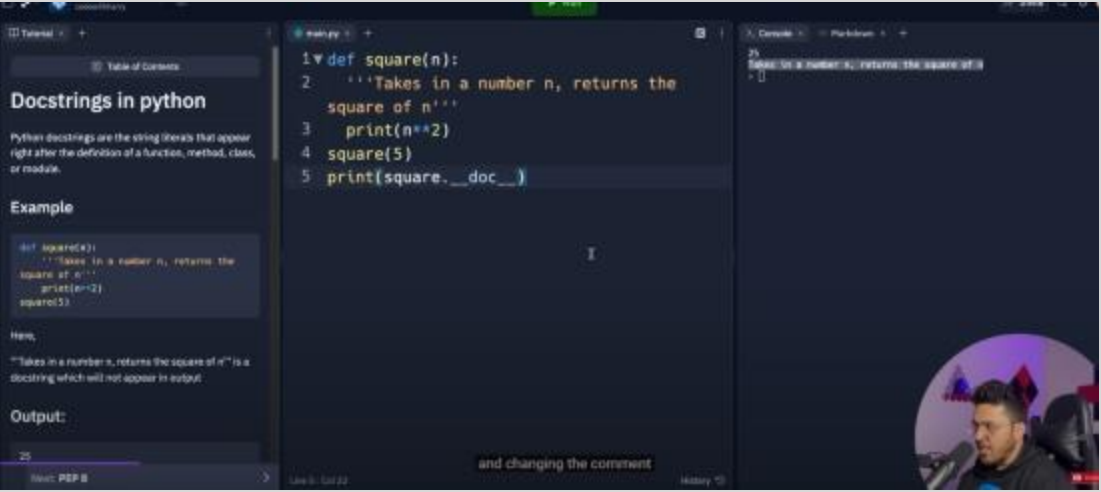
109 36.

110 UNSOLVED — 02/11/2023 6:47 PM

111 multi line comment `'''` comment `'''` or `"""` comment `"""`

112 37. [6:52 PM]

113 The comments which are written in a function which give us a brief explanation of how the function works `''' THE YH`
114 `AVETOB EWITTENABOVETHEFUNCTIONBODYANDRIGHTBELOWTHEFUNCTIONNAME`
115 `'''`

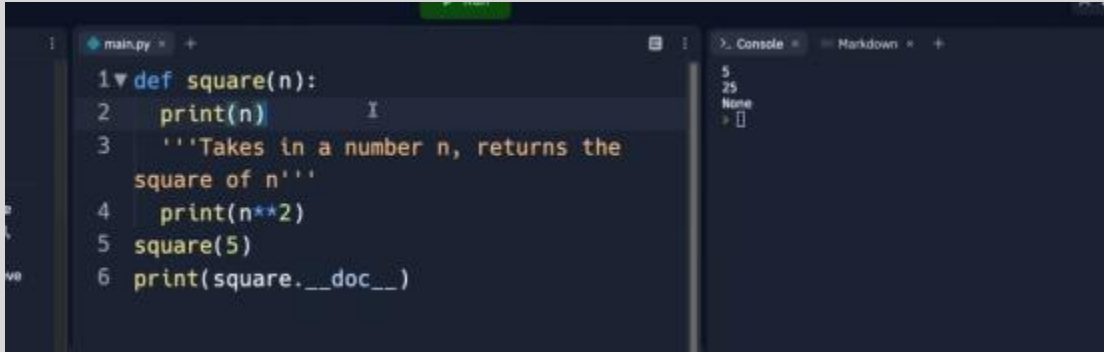


116

117 38. [6:53 PM]

118 and this doesn't print docstring of the function

119



```
1 def square(n):
2     print(n)
3     '''Takes in a number n, returns the
4     square of n'''
5     print(n**2)
6     square(5)
7     print(square.__doc__)
```

Console output:

```
5
25
None
```

120 39.

121 UNSOLVED — 02/11/2023 7:25 PM

122 dictionary if you del {setname} it removes the set from variable space

123



clear():
The clear() method removes all the items from the list.

Example:

```
info = {'name': 'Karan', 'age': 19, 'eligible': True}
info.clear()
print(info)
```

Output:

```
{}
```

pop():
The pop() method removes the key-value pair whose key is passed as a parameter.

Example:

```
info = {'name': 'Karan', 'age': 19, 'eligible': True}
info.pop('eligible')
print(info)
```

124



popitem():
The popitem() method removes the last key-value pair from the dictionary.

Example:

```
info = {'name': 'Karan', 'age': 19, 'eligible': True, 'DOB': 2003}
info.popitem()
print(info)
```

Output:

```
{'name': 'Karan', 'age': 19, 'eligible': True}
```

del:
we can also use the del keyword to remove a dictionary item.

Example:

```
info = {'name': 'Karan', 'age': 19, 'eligible': True, 'DOB': 2003}
del info['age']
print(info)
```

125 40. [7:26 PM]

126 WE CAN USE ELSE WITH FOR-LOOP for i in range(5): print ("SathviK") else: print ("If else executed that means the for
127 loop has run successfully till its last iteration and didn't break anywhere in between")

128 41.

129 UNSOLVED — 02/11/2023 7:34 PM

130 try : #here write that code which could give errors the name itself says it just tries the lines of the code written in try and
131 if it gets any error it goes to except part IF EXCEPT PART IS : 1) except Exception as e: print (e) This prints the Exception
132 which usually shows in the terminal in normal lines. (doesn't end the program and doesn't show it in the red lines) 2)
133 except : print ("Your random words which you want to tell the user") (edited)

134



```
1 a = input("Enter the number: ")
2 print(f"Multiplication table of {a} is: ")
3 try:
4     for i in range(1, 11):
5         print(f"{int(a)} X {i} = {int(a)*i}")
6 except Exception as e:
7     print(e)
8
9 print("Some imp lines of code")
10 print("End of program")
```

Console output:

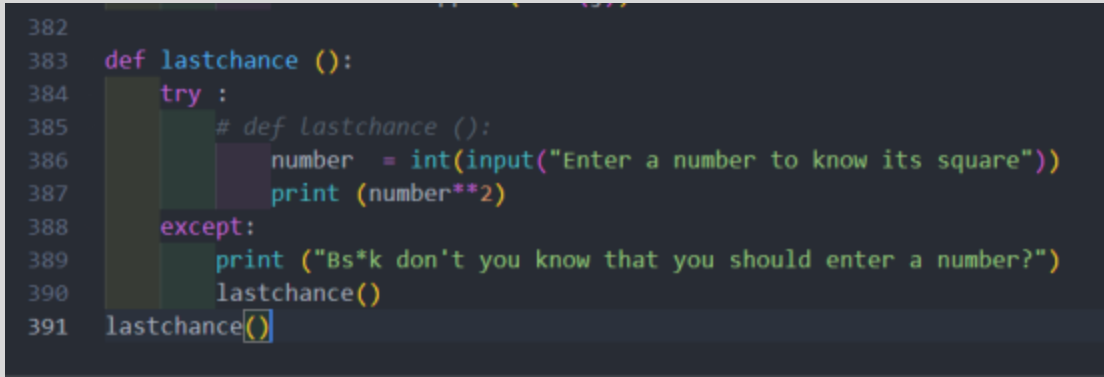
```
Enter the number: joker
Multiplication table of joker is:
Traceback (most recent call last):
  File "main.py", line 5, in <module>
    print(f"{int(a)} X {i} = {int(a)*i}")
ValueError: invalid literal for int() with base 10: 'joker'
```

135 42.

136 UNSOLVED — 02/11/2023 7:53 PM

137 if you want to give another chance so that they input the things without the error you can do this

138



```
382
383 def lastchance ():
384     try :
385         # def lastchance ():
386         number = int(input("Enter a number to know its square"))
387         print (number**2)
388     except:
389         print ("Bs*k don't you know that you should enter a number?")
390         lastchance()
391 lastchance()
```

139 43. [7:57 PM]

140 Types of errors: **Value error** arises when you type string as a input in place of integer or something and so on **Index error**
141 arises when you type an index as an input and it is out of range

142 44. [7:57 PM]

143

```
11
12▼try:
13     num = int(input("Enter an integer: "))
14     a = [6, 3]
15     print(a[num])
16▼except ValueError:
17     print("Number entered is not an integer.")
18
19▼except IndexError:
20     print("Index Error")
```

144 45.

145 UNSOLVED — 02/11/2023 8:05 PM

146 DIFFERENCE BETWEEN FINALLY AND NORMAL PRINT STATEMENT **If function got returned** it goes to exception if there is
147 a exception occurred **if function got returned without an exception** then it doesn't check remaining lines of statement in
148 the function (here print is not even touched) BUT **even if function got returned without an exception** then also it goes
149 to finally (after returning , since finally means finally)

150

```
():
Enter the index: 6
Some error occurred
0

I am always executed")
I am always executed")
()
```

151

152 46. [8:09 PM]

```
:
1
I am a
1
1
5, 6, 7]
(input("Enter the index: "))
[i])
1
Some error occurred")
0
I am always executed")
I am always executed")
```

153

154 February 12, 2023

155 47.

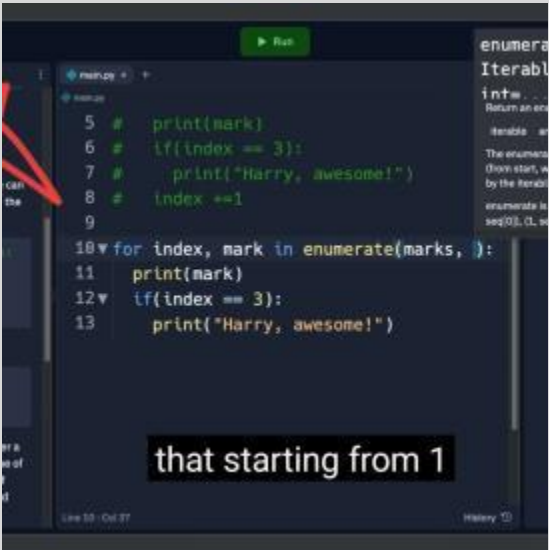
156 UNSOLVED — 02/12/2023 12:26 PM

```
1 a = 330000
2 b = 3303
3 print("A") if a > b else print("=") if
  else print("B")
4
5 c = 9 if a>b else 0
6 print(c)
```

157

to write simple if-else statements

158

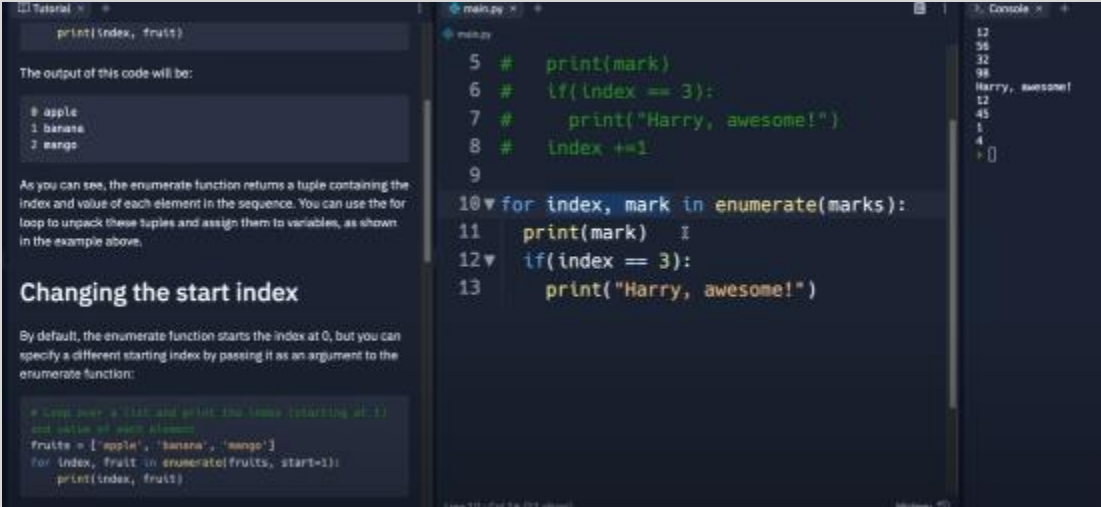


159

48.

160 UNSOLVED — 02/12/2023 5:34 PM

161



162

49. [5:37 PM]

163 actually enumerate(marks) has two values and they come as a pack of tuple (value1, value2) by writing index, mark will

164 unpack the values to the corresponding first and second

165

50.

166 UNSOLVED — 02/12/2023 5:45 PM

How importing in python works

Importing in Python is the process of loading code from a Python module into the current script. This allows you to use the functions and variables defined in the module in your current script, as well as any additional modules that the imported module may depend on.

To import a module in Python, you use the import statement followed by the name of the module. For example, to import the math module, which contains a variety of mathematical functions, you would use the following statement:

```
import math
```

Once a module is imported, you can use any of the functions and variables defined in the module by using the dot notation. For example, to use the sqrt function from the math module, you would write:

```
import math

result = math.sqrt(9)
print(result) # Output: 3.0
```

from keyword

You can also import specific functions or variables from a module using the from keyword. For example, to import only the sqrt function from the math module, you would write:

167

168

51.

169 UNSOLVED — 02/12/2023 5:58 PM

170 ALL THOSE ABOUT IMPORTS

```
# from math import sqrt, pi
from math import pi, sqrt as s

result = s(9) * pi
print(result) # Output: 3.0
```

171

```
rd

Output: 3.0

put: 3.141592653589793
```

172

173

```
from math import pi, sqrt
import math as m

result = m.sqrt(9) * m.pi
print(result) # Output: 3.0
```

174

```
import sqrt, pi
import pi, sqrt as s
import math as math_builtin_python

print(math_builtin_python.sqrt(9) *
python.pi) # Output: 3.0
```

175

52. [5:58 PM]

176

Practice of modules

177

53. [6:00 PM]

178

These should of same folder All these works from mod import * is not advised to use

```
mod.py
PythonLearnings > mod.py
1 def sathvik():
2     print ("This")
3     sath = "SATH is"
4
```

179

180

```
mod.py
PythonLearnings > mod.py
1 def sathvik():
2     print ("This")
3     sath = "SATH is"
4
```

181

```
mod.py
PythonLearnings > mod.py
1 def sathvik():
2     print ("This")
3     sath = "SATH is"
4
```

182

54.

183

UNSOLVED — 02/12/2023 6:06 PM

184

if there are some executable functions in a file and if there is a call of that function in the file on importing the file

185

they run the calling automatically so it runs once on importing the file (edited)

186

55. [6:11 PM]

187

just on importing it ran the file since there is a function call in the file itself

```
mod.py
PythonLearnings
1 def sa
2 pr
3 # sath
4 sathvi
```

188

189

190 56. [6:12 PM]

191 if I WRITE function call in the file where we are importing the module it ran the program twice in total

192

193 57.

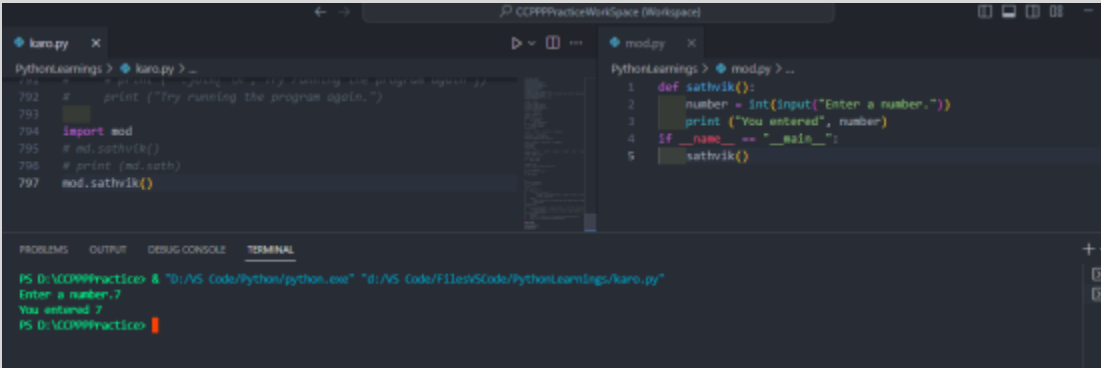
194 UNSOLVED — 02/12/2023 6:18 PM

195 here mod is the main function for its own code written in its file if mod.py is being executed then only the flow goes into

196 if condition otherwise it doesn't since we are running it from karo and since it is the main function / main file of that

197 code **only the calling of sathvik() in karo works and imports doesn't run the function** main simply means it is running /

198 executed from the same file (edited)

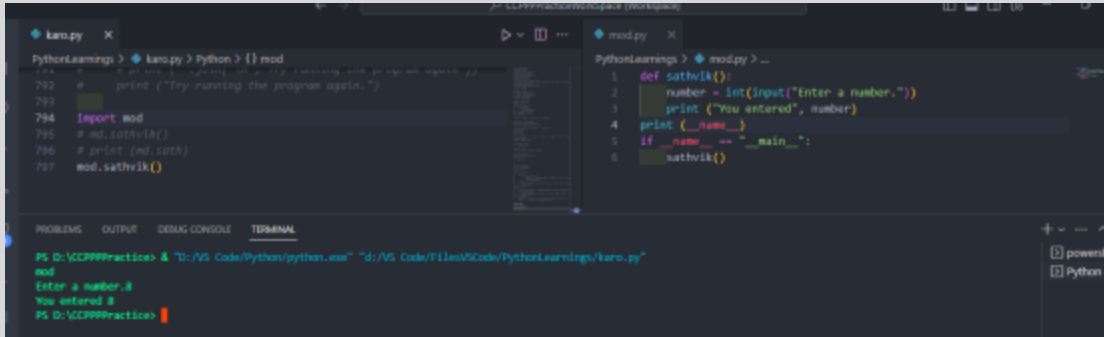


199

200 58. [6:23 PM]

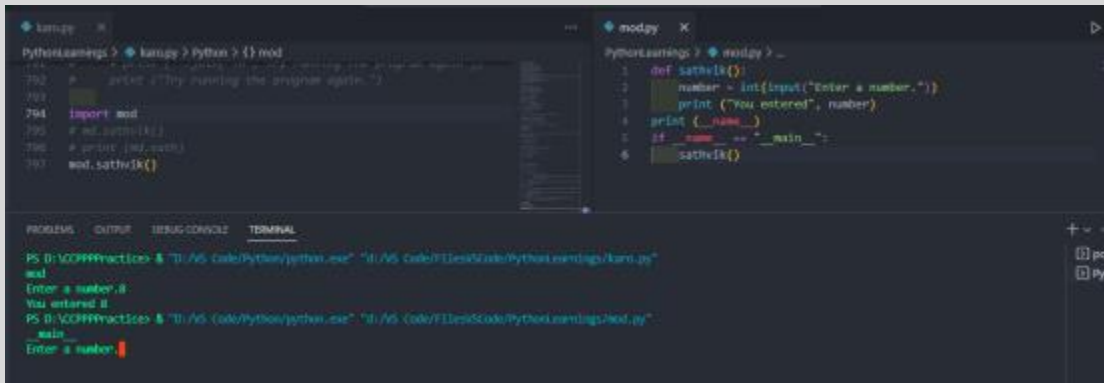
201 name tells how it is being executed/ran in the figure I ran the mod file from karo therefore it said it is being ran as **mod**

202 that is the file is ran somewhere else if it ran in the same place the name would be printed as main (edited)



203

204 59. [6:24 PM]



205

206 60.

207 UNSOLVED — 02/12/2023 6:27 PM

This can be useful if you have code that you want to reuse in multiple scripts, but you only want it to run when the script is run directly and not when it's imported as a module.

Is it a necessity?

It's important to note that the `if __name__ == "__main__":` idiom is not required to run a Python script. You can still run a script without it by simply calling the functions or running the code you want to execute directly. However, the `if __name__ == "__main__":` idiom can be a useful tool for organizing and separating code that should be run directly from code that should be imported and used as a module.

In summary, the `if __name__ == "__main__":` idiom is a common pattern used in Python scripts to determine whether the script is being run directly or being imported as a module into another script. It allows you to reuse code from a script by importing it as a module into another script, without running the code in the original script.

208

209 61.

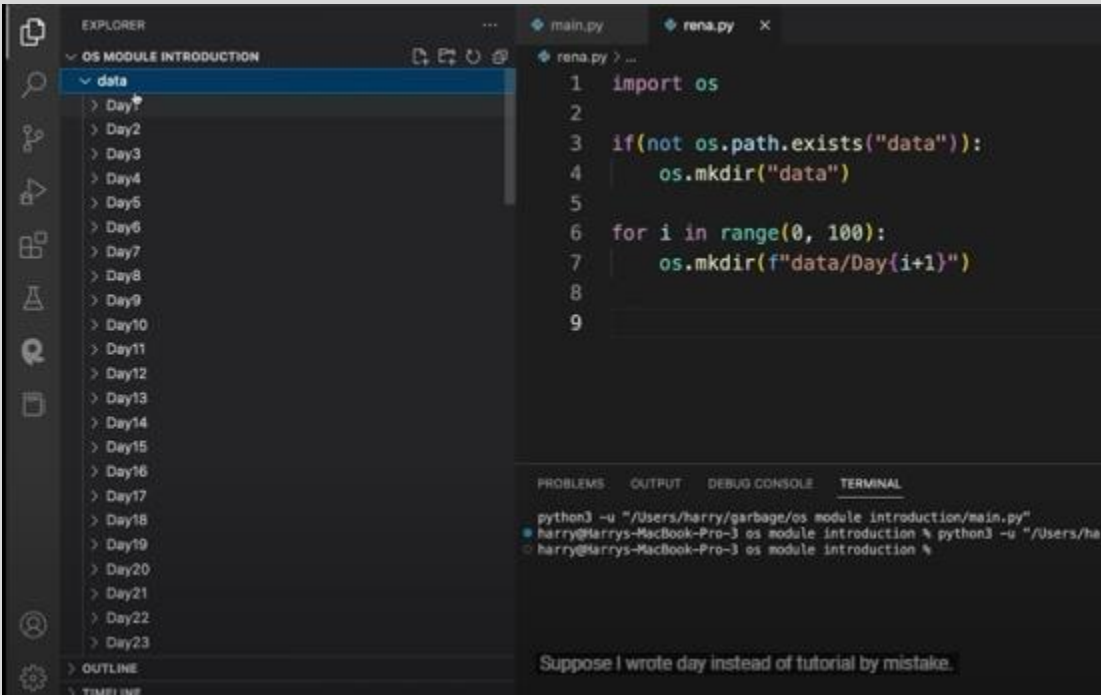
210 UNSOLVED — 02/12/2023 7:40 PM

211



212 62.

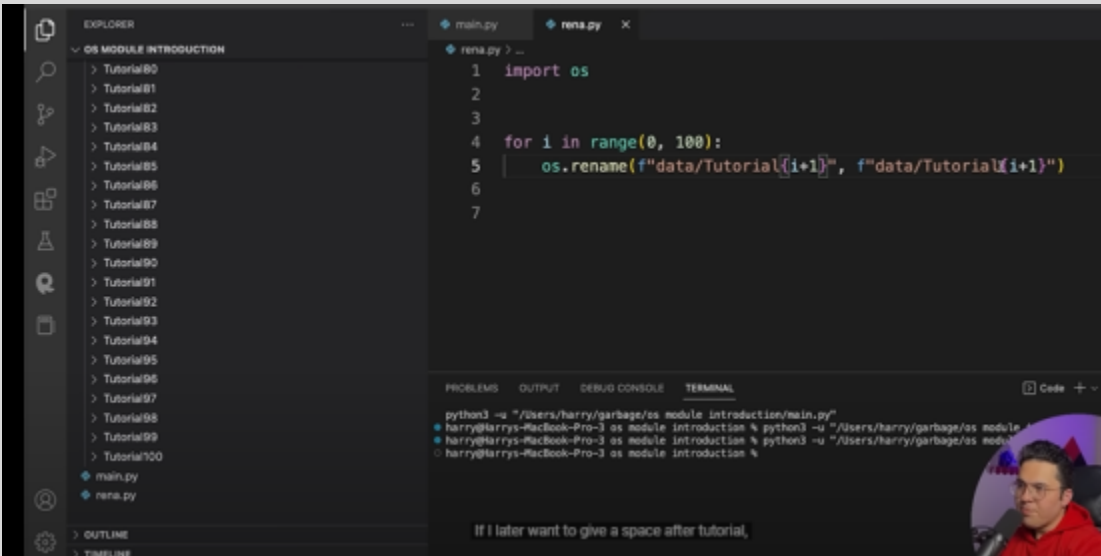
213 UNSOLVED — 02/12/2023 9:33 PM



214

215 63. [9:33 PM]

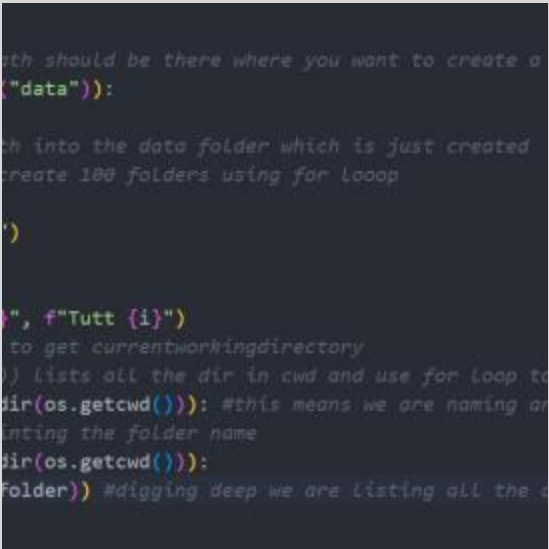
216



217 64.

218 UNSOLVED — 02/12/2023 10:24 PM

219 comment out the lines as necessary after creating the files comment it and run folders after seeing them if you don't
220 want to see them again comment again and also check the remaining functions remove and all



221



222

223 65. [10:28 PM]

224 if you want all at once.

```
836
837 print(os.listdir(os.getcwd()))
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
, 'Tut 20', 'Tut 21', 'Tut 22', 'Tut 23', 'Tut 24', 'Tut 25', 'Tut 26', 'Tut 27', 'Tut 28', 'Tut 29', 'Tut 3', 'Tut 30', 'Tut 31', 'Tut 32', 'Tut 33', 'Tut 34', 'Tut 35', 'Tut 36', 'Tut 37', 'Tut 38', 'Tut 39', 'Tut 4', 'Tut 40', 'Tut 41', 'Tut 42', 'Tut 43', 'Tut 44', 'Tut 45', 'Tut 46', 'Tut 47', 'Tut 48', 'Tut 49', 'Tut 5', 'Tut 50', 'Tut 51', 'Tut 52', 'Tut 53', 'Tut 54', 'Tut 55', 'Tut 56', 'Tut 57', 'Tut 58', 'Tut 59', 'Tut 6', 'Tut 60', 'Tut 61', 'Tut 62', 'Tut 63', 'Tut 64', 'Tut 65', 'Tut 66', 'Tut 67', 'Tut 68', 'Tut 69', 'Tut 7', 'Tut 70', 'Tut 71', 'Tut 72', 'Tut 73', 'Tut 74', 'Tut 75', 'Tut 76', 'Tut 77', 'Tut 78', 'Tut 79', 'Tut 8', 'Tut 80', 'Tut 81', 'Tut 82', 'Tut 83', 'Tut 84', 'Tut 85', 'Tut 86', 'Tut 87', 'Tut 88', 'Tut 89', 'Tut 9', 'Tut 90', 'Tut 91', 'Tut 92', 'Tut 93', 'Tut 94', 'Tut 95', 'Tut 96', 'Tut 97', 'Tut 98', 'Tut 99']
```

226 66.

227 UNSOLVED — 02/12/2023 10:54 PM

```
main.py x +
main.py
1 x = 10 # global variable
2
3 def my_function():
4     global x
5     x = 4
6     y = 5 # local variable
7     print(y)
8
9 my_function()
10 print(x)
11 # print(y) # this will cause an error
    because y is a local variable and is
```

229 67. [10:56 PM]

I will write two lines here
1) global x
2) x = 4
what is the meaning of each line in python?

1) The first line declares a global variable called "x" which can be used in any part of the program.
2) The second line assigns the value 4 to the global variable "x".

231 68.

232 UNSOLVED — 02/12/2023 11:21 PM

Modes in file

There are various modes in which we can open files.

1. read (r): This mode opens the file for reading only and gives an error if the file does not exist. This is the default mode if no mode is passed as a parameter.
2. write (w): This mode opens the file for writing only and creates a new file if the file does not exist.
3. append (a): This mode opens the file for appending only and creates a new file if the file does not exist.
4. create (x): This mode creates a file and gives an error if the file already exists.
5. text (t): Apart from these modes we also need to specify how the file must be handled. t mode is used to handle text files. t refers to the text mode. There is no difference between r and rt or w and wt since text mode is the default. The default mode is 'r' (open for reading text, synonym of 'rt').

233

234 69. [11:23 PM]

```
Run
No file open x main.py x +
main.py
1 f = open('myfile.txt', 'r')
2 # print(f)
3 text = f.read()
4 print(text)
5 f.close()
```

Console x +
Hey harry awesome man!
>

235

236 70.

237 UNSOLVED — 02/12/2023 11:29 PM

238 or else we can do this this with statement automatically closes it after the work

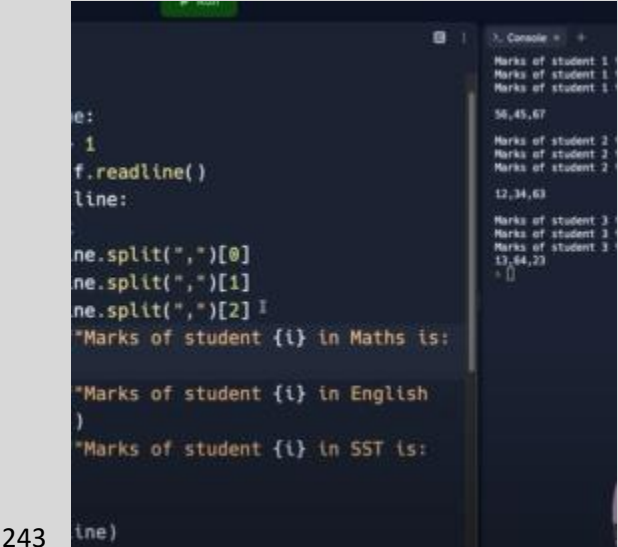
```
839 # f = open('newfile.txt', 'w')
840 # f.write("Hello Sathvik\nI am Python... How are you?")
841 # f.close()
842 # f=open ('newfile.txt', 'r')
843 # txtread = f.read()
844 # print (txtread)
845 with open ("newfile.txt", "r") as f:
846     print (f.read())
847
```

239

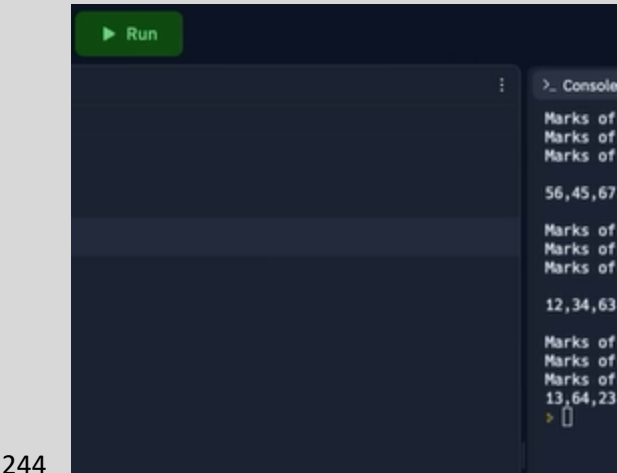
240 February 13, 2023

241 71.

242 UNSOLVED — 02/13/2023 1:28 PM

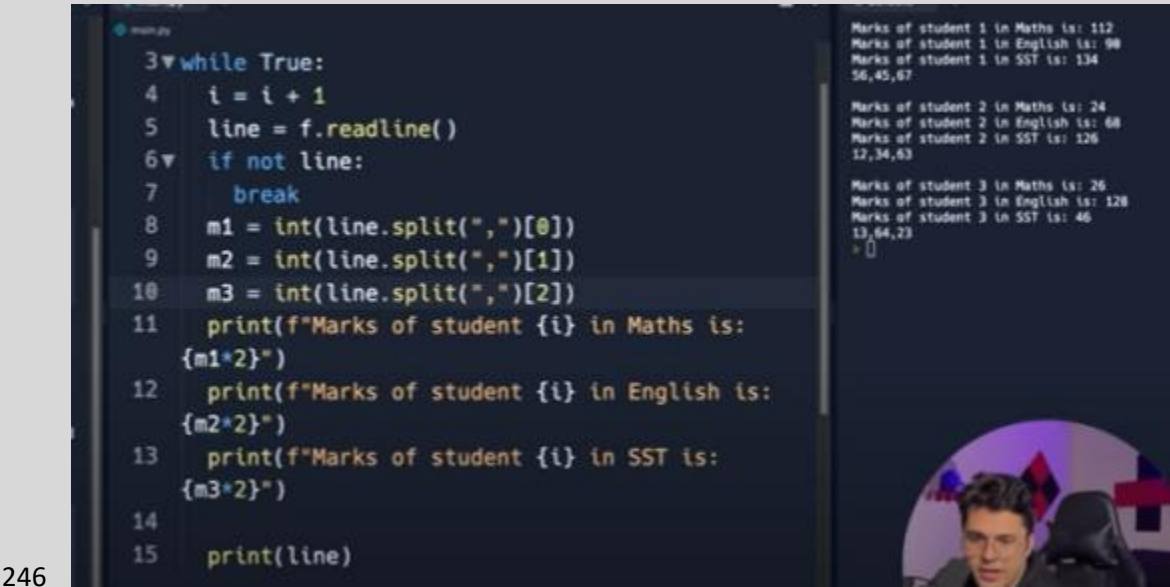


243



244

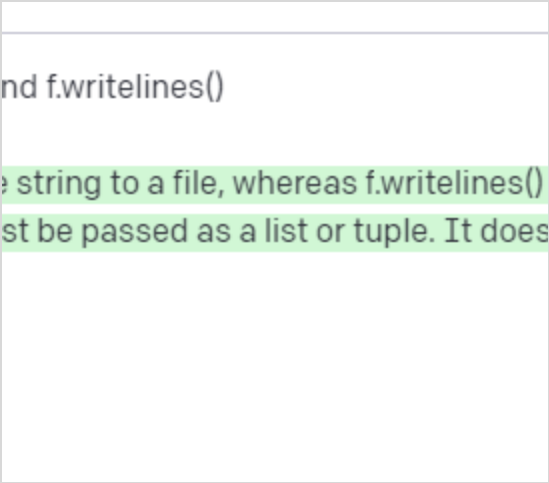
245 72. [1:29 PM]



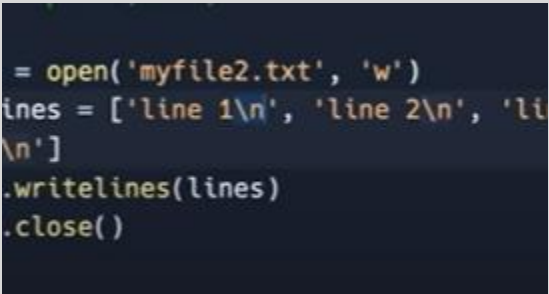
246

247 73.

248 UNSOLVED — 02/13/2023 1:36 PM



249



250

251 74.

252 UNSOLVED — 02/13/2023 2:28 PM

253 seek moves the pointer to the first letter to that byte



254

255 75.

256 UNSOLVED — 02/13/2023 2:38 PM

```
my_empty_set = set()

# or

my_empty_set = {}
```

257

258 76. [2:43 PM]

```
948
949 myset = set()
950 # print (type(myset))
951 for i in range(10):
952     myset.add(i*i)
953 myset.update([1,2,3,10000])
954 print (myset)
```

259

260 77. [2:44 PM]

261 you can use update([1,2,3.10000]) or ({1,2,3,100000})

262 78.

263 UNSOLVED — 02/13/2023 2:55 PM

```
1 with open('file.txt', 'r') as f:
2     print(type(f))
3     # Move to the 10th byte in the
  file
4     f.seek(10)
5
6     # Read the next 5 bytes
7     print(f.tell())
8     data = f.read(5)
9     print(data)
```

264

265 79. [3:01 PM]

```
956 with open ("newlines.txt", "r") as f:
957     f.seek(7)
958     print (f.tell())
959     print (f.read(40))
960 '''seek points the index given here 7
961 # tell tell the index it is pointing now
962 # read reads number of characters from the index
963 # if read(4) it reads the current position and next three'''
964
```

266

267 80.

268 UNSOLVED — 02/13/2023 3:35 PM

269 if we add truncate it cuts the file to that bytes it keeps only those number of first characters.

```
1 with open('sample.txt', 'w') as f:
2     f.write('Hello World!')
3     f.truncate(5)
4
5 with open('sample.txt', 'r') as f:
6     print(f.read())
```

270

271 81. [3:40 PM]

```
1 # def double(x):
2 #     return x*2
3
4 double = lambda x: x * 2
5 cube = lambda x: x * x * x
6 avg = lambda x, y, z: (x + y + z) / 3
7
8 print(double(5))
9 print(cube(5))
10 print(avg(3, 5, 10))
11
```

272

273 82.

274 UNSOLVED — 02/13/2023 3:44 PM

275 we can pass function as a argument also

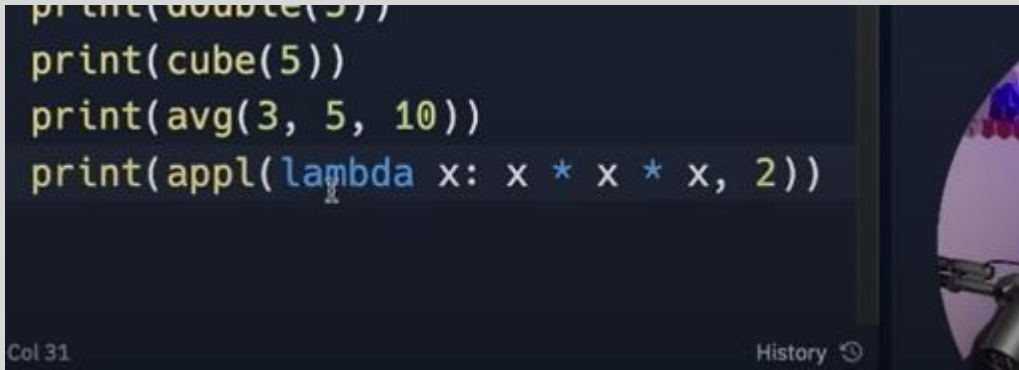


```
1 # def double(x):
2 #     return x*2
3
4 def appl(fx, value):
5     return 6 + fx(value)
6
7 double = lambda x: x * 2
8 cube = lambda x: x * x * x
9 avg = lambda x, y, z: (x + y + z) / 3
10
11 print(double(5))
12 print(cube(5))
13 print(avg(3, 5, 10))
14 print(appl(cube, 2))
```

276

277 83. [3:46 PM]

278 or also do this



```
print(double(5))
print(cube(5))
print(avg(3, 5, 10))
print(appl(lambda x: x * x * x, 2))
```

279

280 84. [3:50 PM]

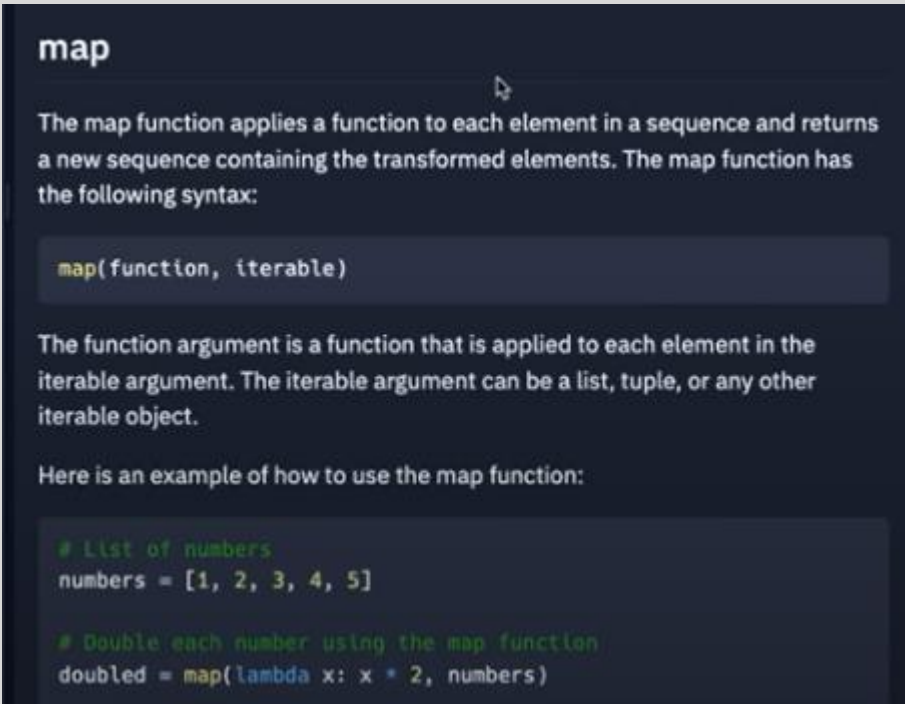
281 how to use map function



```
6
7 l = [1, 2, 4, 6, 4, 3]
8 # newl = []
9 # for item in l:
10 #     newl.append(cube(item))
11
12 newl = list(map(cube, l))
13 print(newl)
```

282

283 85.



map

The map function applies a function to each element in a sequence and returns a new sequence containing the transformed elements. The map function has the following syntax:

```
map(function, iterable)
```

The function argument is a function that is applied to each element in the iterable argument. The iterable argument can be a list, tuple, or any other iterable object.

Here is an example of how to use the map function:

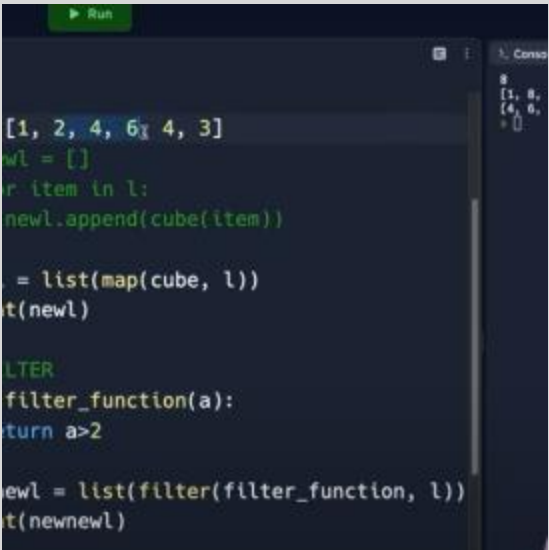
```
# List of numbers
numbers = [1, 2, 3, 4, 5]

# Double each number using the map function
doubled = map(lambda x: x * 2, numbers)
```

285

286 86. [4:02 PM]

287 how to use filter



```
[1, 2, 4, 6, 4, 3]
newl = []
for item in l:
    newl.append(cube(item))

newl = list(map(cube, l))
print(newl)

FILTER
filter_function(a):
    return a>2

newl = list(filter(filter_function, l))
print(newnewl)
```

288

h filters a sequence of elements based on a given predicate (a function that returns a boolean value) and returns a new sequence of elements that meet the predicate. The filter function has the following signature:

```
filter(predicate, iterable)
```

The predicate argument is a function that returns a boolean value. The iterable argument is an iterable object.

87.

UNSOLVED — 02/13/2023 4:20 PM

filter is used always with a function which returns a boolean value we have to turn map and filter to list

88.

UNSOLVED — 02/13/2023 7:40 PM

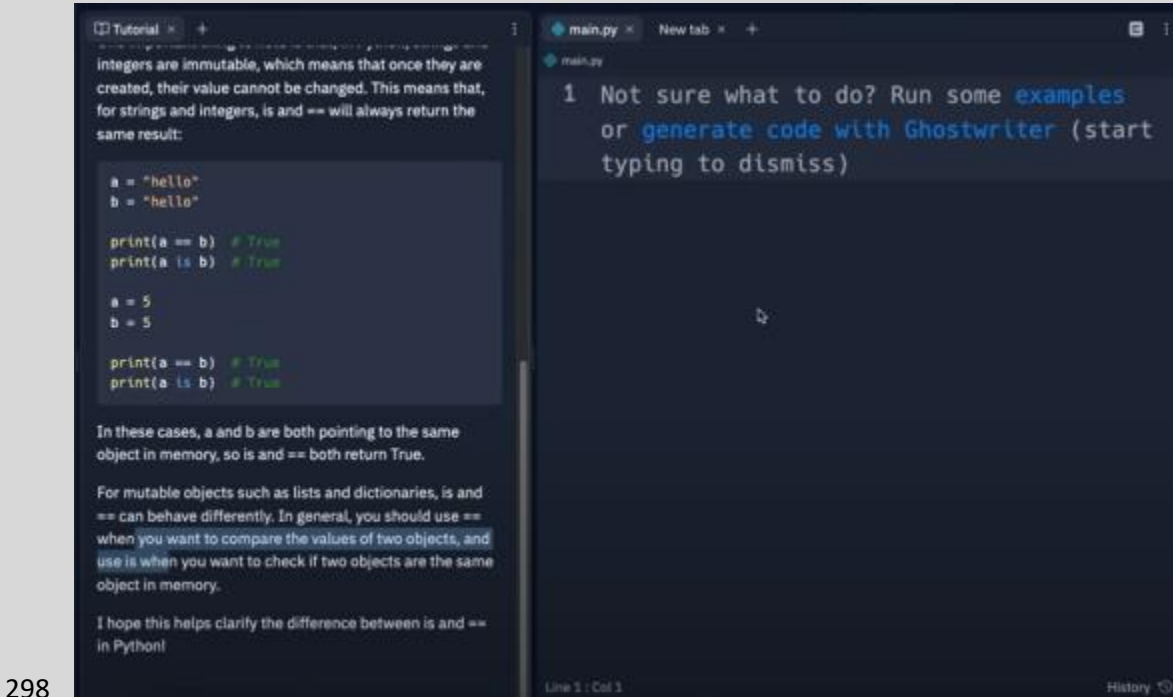
```
1033
1034 from functools import reduce
1035 number = [32,2,2,2,2]
1036 def helpredu (x,y):
1037     return x/y
1038 newnumber = reduce(helpredu,number)
1039 print (newnumber)
```

PROBLEMS OUTPUT DEBUG CONSOLE JUPYTER TERMINAL

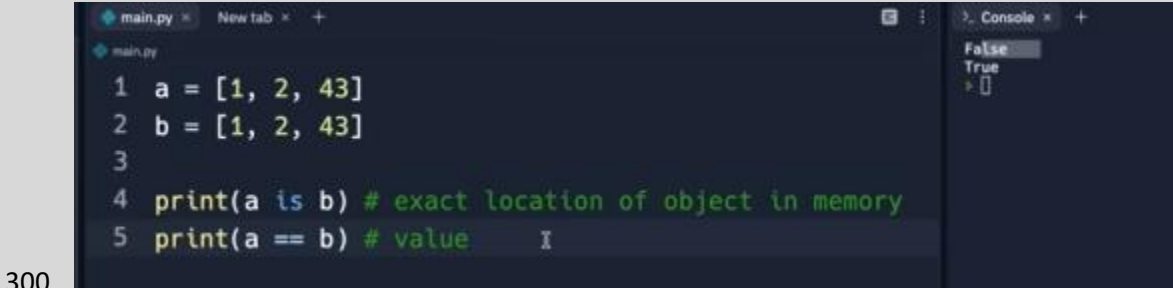
```
PS D:\VS Code\FilesVSCode\PythonLearnings> & "D:/VS Code/Python/python.exe" -i
2.0
PS D:\VS Code\FilesVSCode\PythonLearnings>
```

89.

UNSOLVED — 02/13/2023 9:06 PM

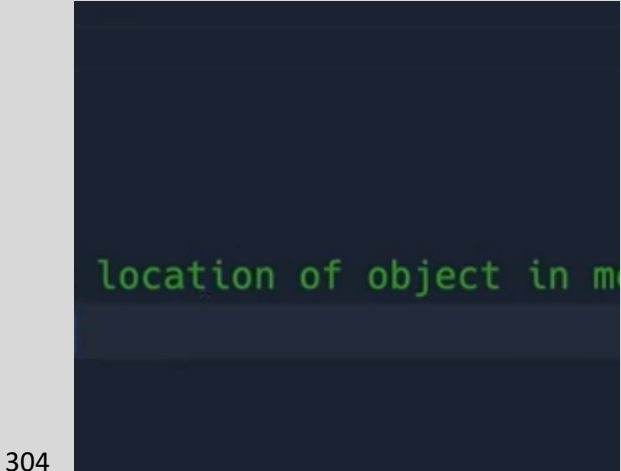


90. [9:08 PM]



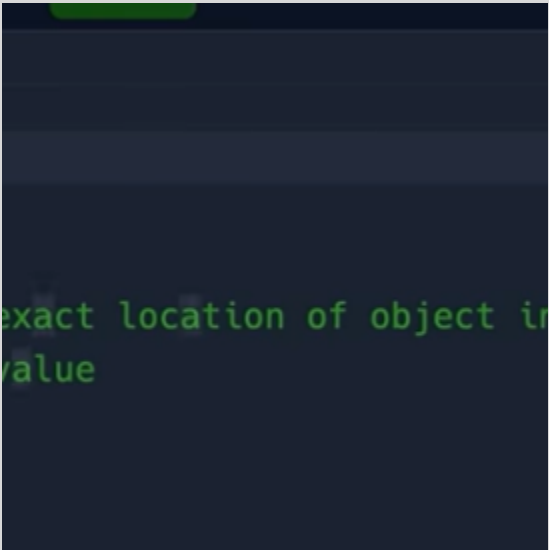
91. [9:10 PM]

here a and b are pointing at immutable things python don't allocate different memories for the same immutable things
so a and b points at the same location



304

305



306

92. [9:10 PM]

307



308

February 14, 2023

309

93.

310

UNSOLVED — 02/14/2023 4:08 PM

311

Introduction to Object-oriented programming

Introduction to Object-Oriented Programming in Python: In programming languages, mainly there are two approaches that are used to write program or code.

- 1). Procedural Programming
- 2). Object-Oriented Programming

The procedure we are following till now is the "Procedural Programming" approach. So, in this session, we will learn about Object Oriented Programming (OOP). The basic idea of object-oriented programming (OOP) in Python is to use classes and objects to represent real-world concepts and entities.

A class is a blueprint or template for creating objects. It defines the properties and methods that an object of that class will have. Properties are the data or state of an object, and methods are the actions or behaviors that an object can perform.

An object is an instance of a class, and it contains its own data and methods. For example, you could create a class called "Person" that has properties such as name and age, and methods such as speak() and walk(). Each instance of the Person class would be a unique object with its own name and age, but they would all have the same methods to speak and walk.

One of the key features of OOP in Python is encapsulation, which means that the internal state of an object is hidden and can only be accessed or modified through the object's methods. This helps to protect the object's data and prevent it from being modified in unexpected ways.

312

94. [4:13 PM]

313

A class is a blueprint or template for creating objects. It defines the properties and methods that an object of that class will have. Properties are the data or state of an object, and methods are the actions or behaviors that an object can perform.

An object is an instance of a class, and it contains its own data and methods. For example, you could create a class called "Person" that has properties such as name and age, and methods such as speak() and walk(). Each instance of the Person class would be a unique object with its own name and age, but they would all have the same methods to speak and walk.

One of the key features of OOP in Python is encapsulation, which means that the internal state of an object is hidden and can only be accessed or modified through the object's methods. This helps to protect the object's data and prevent it from being modified in unexpected ways.

Another key feature of OOP in Python is inheritance, which allows new classes to be created that inherit the properties and methods of an existing class. This allows for code reuse and makes it easy to create new classes that have similar functionality to existing classes.

Polymorphism is also supported in Python, which means that objects of different classes can be treated as if they were objects of a common class. This allows for greater flexibility in code and makes it easier to write code that can work with multiple types of objects.

In summary, OOP in Python allows developers to model real-world concepts and entities using classes and objects, encapsulate data, reuse code through inheritance, and write more flexible code through polymorphism.

314

95.

315

UNSOLVED — 02/14/2023 4:37 PM

316

```
main.py
4  networth = 10
5  def info(self):
6      print(f"{self.name} is a {self.occupation}")
7
8
9  a = Person()
10 b = Person()
11 a.name = "Shubham"
12 a.occupation = "Accountant"
13
14 b.name = "Nitika"
15 b.occupation = "HR"
16 # print(a.name, a.occupation)
17 a.info()
18 b.info()
19
```

Shubham is a Accountant
Nitika is a HR

317 96.

318 UNSOLVED — 02/14/2023 6:52 PM

```
435
436 class Person:
437     name = "Sathvik"
438     study = "IITDh"
439     networth = 99999
440     def info (self):
441         print (f"{self.name} is studying in {self.study} and is having a networth of {
442 per = Person()
443 # print (per.name,per.networth)
444 # per.info()
445 b = Person()
446 b.name = "Meripe"
447 # b.study = "LocalCollege"
448 b.networth = 9
449 b.info() #If you don't provide full details of b some default details will come which
```

320 97. [6:52 PM]

321 or if you don't want this also put None for everything (edited)

322 98. [6:53 PM]

```
435
436 class Person:
437     name = "Sathvik"
438     study = "IITDh"
439     networth = 99999
440     def info (self):
441         print (f"{self.name} is studying in {self.study} and is having a networth of {
442 per = Person()
443 # print (per.name,per.networth)
444 # per.info()
445 b = Person()
446 b.name = "Meripe"
447 # b.study = "LocalCollege"
448 b.networth = 9
449 b.info() #If you don't provide full details of b some default details will come which
```

324 99.

325 UNSOLVED — 02/14/2023 10:28 PM

```
person():
name = "Sathvik"
occ = "Developer"
def __init__(self,n,o):
    self.name = n
    self.occ = o
def info(self):
    print (f"{self.name} is a {self
person("Sathvik", "Developer")
person("Lavanya", "HR")
o()
o()
```

326

```
class Person:
    def __init__(self, name, occ):
        print("Hey I am a person")
        self.name = name
        self.occ = occ
    def info(self):
        print(f"{self.name} is a {self.occ}")

a = Person("Harry", "Developer")
b = Person("Divya", "HR")
a.info()
b.info()
# print(a.name)
```

Hey I am a person
Hey I am a person
Harry is a Developer
Divya is a HR

327

328 February 15, 2023

329 100.

330 UNSOLVED — 02/15/2023 10:11 AM

331

is source code.

A decorator is a function that takes another function as an argument and returns a new function that modifies the behavior of the original function. The new function is often referred to as a "decorated" function. The basic syntax for using a decorator is the following:

```
@decorator_function
def my_function():
    pass
```

The @decorator_function notation is just a shorthand for the following code:

```
def my_function():
    pass
my_function = decorator_function(my_function)
```

Decorators are often used to add functionality to functions and methods, such as logging, memoization, and access control.

332

```
def greet(fx):
    def mfx():
        print("Good Morning")
        fx()
        print("Thanks for using this function")
    return mfx

greet
def hello():
    print("Hello world")

def add(a, b):
    print(a+b)

hello()
```

333 101. [10:12 AM]

334 or it also works fine

335

```
1
2 def greet(fx):
3     def mfx():
4         print("Good Morning")
5         fx()
6         print("Thanks for using this function")
7     return mfx
8
9 # @greet
10 def hello():
11     print("Hello world")
12
13 def add(a, b):
14     print(a+b)
15
16 greet(hello)()
17
```

336

102. [10:14 AM]

337 NICE see this

338

methods, such as logging, memoization, and access control.

Practical use case

One common use of decorators is to add logging to a function. For example, you could use a decorator to log the arguments and return value of a function each time it is called:

```
import logging

def log_function_call(func):
    def decorated(*args, **kwargs):
        logging.info(f"Calling {func.__name__} with args={args}, kwargs={kwargs}")
        result = func(*args, **kwargs)
        logging.info(f"{func.__name__} returned {result}")
        return result
    return decorated

@log_function_call
def my_function(a, b):
    return a + b
```

In this example, the log_function_call decorator takes a function as an argument and returns a new function that logs the function call before and after the original function is called.

Conclusion

Decorators are a powerful and flexible feature in Python that can be used to add functionality to functions and methods without modifying the original function code.

```
2 def greet(fx):
3     def mfx(*args, **kwargs):
4         print("Good Morning")
5         fx(*args, **kwargs)
6         print("Thanks for using this function")
7     return mfx
8
9 @greet
10 def hello():
11     print("Hello world")
12
13 # @greet
14 def add(a, b):
15     print(a+b)
16
17 # greet(hello)()
18 hello()
19 greet(add)(1, 2)
```

339

103. [10:16 AM]

340

Conclusion

Decorators are a powerful and flexible feature in Python that can be used to add functionality to functions and methods without modifying their source code. They are a great tool for separating concerns, reducing code duplication, and making your code more readable and maintainable.

In conclusion, python decorators are a way to extend the functionality of functions and methods, by modifying its behavior without modifying the source code. They are used for a variety of purposes, such as logging, memoization, access control, and more. They are a powerful tool that can be used to make your code more readable, maintainable, and extendable.


341

104.

342

UNSOLVED — 02/15/2023 10:27 AM

343



calo 4 years ago

The proper way to never forget how decorators work is just keeping in mind what that '@' syntax means:

```
@decorator
def function():
    ...
```

is equivalent to: function = decorator(function).

Show less

344

February 23, 2023

345

105.

346

UNSOLVED — 02/23/2023 1:59 AM

347

REGULAR METHODS AUTOMATICALLY TAKES OBJECT AS ARGUMENT (SELF) CLASS METHODS TAKE CLASS AS ARGUMENT

348

STATIC METHODS DON'T PASS ANYTHING AS ARGUMENT (edited)

349

106.

350

UNSOLVED — 02/23/2023 2:14 PM

351

If our subclass contains just one / two more information than its parent class we can use

352

super().init(arugementswhichshouldbecopiedfromparentclass) self.newargument or

353

Classname.__init(self,arugementswhichshouldbecopiedfromparentclass) self.newargument

354

107.

355

UNSOLVED — 02/23/2023 2:21 PM

356

We should not pass mutable datatypes as default argument

357

108.

358

UNSOLVED — 02/23/2023 9:51 PM

359

It will only work this way we can't a call a private function private method or function : leading it has two __ so we have

360

to create a public method which call it in the class itself and private function can't be accessed from the outside

361

```
pythonLearnings > queprac.py > ...
1 class tess:
2     def __init__(self):
3         print("I run once")
4     def __nicera():
5         print("I am nicera")
6     def iwillcallnicera(self):
7         tess.__nicera()
8 obj = tess()
9 obj.__init__()
10 obj.iwillcallnicera()
```

362

February 24, 2023

363

109.

364

UNSOLVED — 02/24/2023 11:55 AM

365

This is how we can give all the details necessary to init while giving what other methods wants sepcifically to them

366

```
python
Copy code

class Person:
    def __init__(self, name):
        self.name = name
        self.gender = None

    def set_gender(self, gender):
        self.gender = gender

    def display(self):
        print("The whole details are here\nName: {}\nGender: {}".format(self.name,

per = Person('Sathvik')
per.set_gender('Male')
per.display()
```


367 March 1, 2023

368 110.

369 UNSOLVED — 03/01/2023 11:40 PM

370 numpy

```
In [9]: a = np.array([1,2,3])
print(a)
[1 2 3]

In [10]: b = np.array([[9.0,8.0,7.0],[6.0,5.0,4.0]])
print(b)
[[9. 8. 7.]
 [6. 5. 4.]]

In [16]: # Get Dimension
a.ndim

Out[16]: 1

In [18]: # Get Shape
b.shape

Out[18]: (2, 3)

In [20]: # Get Type
a.dtype

Out[20]: dtype('int32')
```

371

372 111. [11:41 PM]

373 we can also specify the datatype we want

The Basics

```
In [21]: a = np.array([1,2,3], dtype='int16')
print(a)
[1 2 3]

In [10]: b = np.array([[9.0,8.0,7.0],[6.0,5.0,4.0]])
print(b)
[[9. 8. 7.]
 [6. 5. 4.]]

In [16]: # Get Dimension
a.ndim

Out[16]: 1

In [18]: # Get Shape
b.shape

Out[18]: (2, 3)

In [20]: # Get Type
a.dtype

Out[20]: dtype('int32')
```

374

375 112.

376 UNSOLVED — 03/01/2023 11:48 PM

```
Out[18]: (2, 3)

In [22]: # Get Type
a.dtype

Out[22]: dtype('int16')

In [25]: # Get Size
a.itemsize

Out[25]: 4

In [29]: # Get total size
a.nbytes

Out[29]: 12
```

377

378 113.

379 UNSOLVED — 03/01/2023 11:58 PM

Accessing/Changing specific elements, rows, columns, etc

```
In [9]: a = np.array([[1,2,3,4,5,6,7],[8,9,10,11,12,13,14]])
print(a)
[[ 1  2  3  4  5  6  7]
 [ 8  9 10 11 12 13 14]]

In [14]: # Get a specific element [r, c]
a[1, 5]

Out[14]: 13

In [15]: # Get a specific row
a[0, :]

Out[15]: array([1, 2, 3, 4, 5, 6, 7])

In [16]: # Get a specific column
a[:, 2]

Out[16]: array([ 3, 10])

In [20]: # Getting a little more fancy [startindex:endindex:stepsize]
a[0, 1:-1:2]

Out[20]: array([2, 4, 6])
```

380

381 114. [11:59 PM]

382

383 March 2, 2023

384 115.

385 UNSOLVED — 03/02/2023 12:16 AM

```
In [15]: # Get a specific row
a[0, :]

Out[15]: array([1, 2, 3, 4, 5, 6, 7])

In [16]: # Get a specific column
a[:, 2]

Out[16]: array([ 3, 10])

In [20]: # Getting a little more fancy (startindex:end)
a[0, 1:-1:2]

Out[20]: array([2, 4, 6])

In [23]: a[1,5] = 20
print(a)

a[:,2] = 5
print(a)

[[ 1  2  3  4  5  6  7]
 [ 8  9 10 11 12 20 14]]
[[ 1  2  5  4  5  6  7]
 [ 8  9  5 11 12 20 14]]
```

386

387 116. [12:17 AM]

```
[[ 1  2  5  4  5  6  7]
 [ 8  9  5 11 12 20 14]]
[[ 1  2  1  4  5  6  7]
 [ 8  9  2 11 12 20 14]]

*3-d example

In [25]: b = np.array([[[1,2],[3,4]],[[5,6],[7,8]]])
print(b)

[[[1 2]
  [3 4]]

 [[5 6]
  [7 8]]]

In [28]: # Get specific element (work outside in)
b[:,0,:]

Out[28]: array([[1, 2],
               [5, 6]])
```

388

389 117. [12:20 AM]

```
In [40]: # All 0s matrix
np.zeros((2,3))

Out[40]: array([[0., 0., 0.],
               [0., 0., 0.]])

In [42]: # All 1s matrix
np.ones((4,2,2), dtype='int32')

Out[42]: array([[[[1, 1],
                  [1, 1]],
                 [[1, 1],
                  [1, 1]],
                 [[1, 1],
                  [1, 1]],
                 [[1, 1],
                  [1, 1]]]])

In [43]: # Any other number
np.full((2,2), 99)

Out[43]: array([[99, 99],
               [99, 99]])
```

390

391 118.


392 UNSOLVED — 03/02/2023 12:34 AM

```
In [49]: # Any other number (full_like)
np.full_like(a, 4)

Out[49]: array([[4, 4, 4, 4, 4, 4, 4],
               [4, 4, 4, 4, 4, 4, 4]])

In [52]: # Random decimal numbers
np.random.rand(4,2)

Out[52]: array([[0.83329042, 0.4378868 ],
               [0.75339986, 0.12535178],
               [0.4241573 , 0.67521787],
               [0.44924988, 0.97534681]])
```

Bharath B N

9:19 AM

Clarification regarding arr > x:

```
import numpy as np

arr = np.array([1,3,-2,4,5,3,0,7])

print(arr > 2)
```

output: [False True False True True True False True]

418

419 128.

420 UNSOLVED — 03/02/2023 11:00 PM

Some basic syntax (Apart from slides)

```
1. arr.flatten() converts arr into a 1D array with rows in order.
2. arr > x boolean array same shape, and True iff arr[element] > x.
3. np.sum(a) gives sum of values in a
4. np.random.randint(start, end, size) random integers between start, end of given size.
5. np.linalg.det(A) determinant of  $N \times N$  matrix A.
6. np.cross(x, y) cross product between vector x and y.
7. np.sin(x) sin(x), np.cos(x) cos(x), np.tan(x) tan(x)
8. np.std(x, axis=n), np.median(x, axis=n), np.mean(x, axis=n)
9. np.max(a) gives maximum in a
10. np.random.randint(start, end, size) random integers between start, end (inclusive) of given size.
11. np.linalg.det(A) determinant of  $N \times N$  matrix A.
12. arr > x boolean array of size arr.shape, and True if arr[element] > x.
13. np.sum(a) gives sum of values in a.
```

421

422 March 9, 2023

423 129.

424 UNSOLVED — 03/09/2023 1:33 AM

Method Overloading

```
class Area:
    def find_area(self,a=None,b=None):
        if a!=None and b!=None:
            print("Area of Rectangle:",(a*b))
        elif a!=None:
            print("Area of square:",(a*a))
        else:
            print("Nothing to find")

obj1=Area()
obj1.find_area()
obj1.find_area(10)
obj1.find_area(10,20)
```

425

426 130. [1:34 AM]

427 If we use same method in parent class and sub class also then if we point the object to the subclass the method of
428 subclass will be executed this is called method overriding

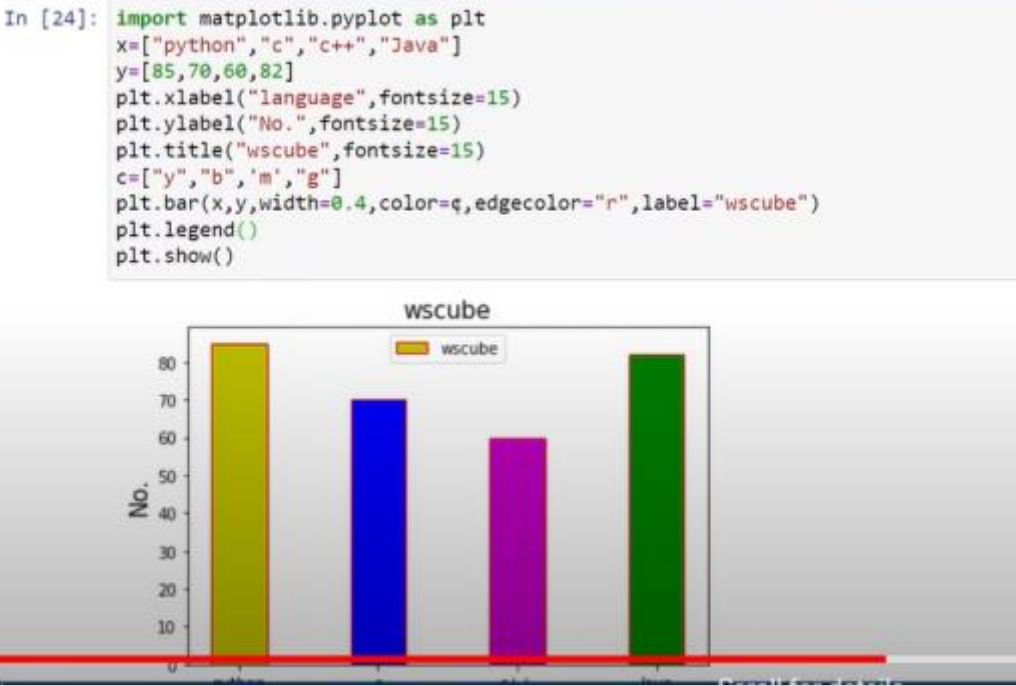
429 131.

430 UNSOLVED — 03/09/2023 3:13 PM

```
plt.xlabel("language",fontsize=15)
plt.ylabel("No.",fontsize=15)
plt.title("wscube",fontsize=15)
c=["y","b",'m',"g"]
plt.bar(x,y,width=0.4,color=c,edgecolor="r",linewidth=5,linestyle=":",alpha=0.5)
plt.show()
```

431

432 132. [3:14 PM]



- Want a **change color**?
- Example plot:
 - Plots $f(x) = x^2$ for $-5 \leq x \leq 5$

```
x = np.linspace(-5,5, 100)
plt.plot(x, x**2,'g-')
plt.show()
```

In [15]: `import matplotlib.pyplot as plt`
`day=[1,2,3,4,5,6,7]`
`no=[2,3,1,4,5,3,6]`
`colors=["r","y","g","b","r",'g',"r"]`
`sizes=[400,200,400,300,200,100,600]`
`plt.scatter(day,no,c=colors,s=sizes)`
`plt.title("Scatter Plot",fontsize=15)`
`plt.xlabel("Day",fontsize=15)`
`plt.ylabel("No.",fontsize=15)`
`plt.show()`

[3:40 PM]

In [24]: `import matplotlib.pyplot as plt`
`day=[1,2,3,4,5,6,7]`
`no=[2,3,1,4,5,3,6]`
`colors=["r","y","g","b","r",'g',"r"]`
`sizes=[400,200,400,300,200,100,600]`
`plt.scatter(day,no,c=colors,s=sizes,marker="*",edgecolor="g",linewidth=2)`
`plt.title("Scatter Plot",fontsize=15)`
`plt.xlabel("Day",fontsize=15)`
`plt.ylabel("No.",fontsize=15)`
`plt.show()`

433

434 133.

435 UNSOLVED — 03/09/2023 3:20 PM

In [35]: `import matplotlib.pyplot as plt`
`import numpy as np`

`x=["python","c","c++","Java"]`
`y=[85,70,60,82]`
`z=[20,30,40,50]`

`width=0.2`
`p=np.arange(len(x))`
`p1=[j+width for j in p]`

`plt.xlabel("language",fontsize=15)`
`plt.ylabel("No.",fontsize=15)`
`plt.title("wscube",fontsize=15)`

`plt.bar(p,y,width,color="r",label="popularity")`
`plt.bar(p1,z,width,color="y",label="popularity1")`

`plt.xticks(p+width/2,x,rotation=20)`
`plt.legend()`
`plt.show()`

436

437 134. [3:27 PM]

438

439 135.

440 UNSOLVED — 03/09/2023 3:38 PM

441

442 136.

443

444 137.

445 UNSOLVED — 03/09/2023 3:48 PM

```
In [36]: import matplotlib.pyplot as plt
day=[1,2,3,4,5,6,7]
no=[2,3,1,4,5,3,6]
no2=[3,2,4,5,1,6,2]
colors=[10,49,30,29,56,20,30]
sizes=[400,200,400,300,200,100,600]
plt.scatter(day,no,c=colors,s=sizes,cmap="viridis",alpha=0.5)
plt.scatter(day,no2,color="r",s=sizes,alpha=0.5)
t=plt.colorbar()
t.set_label("ColorBar",fontsize=15)
plt.title("Scatter Plot",fontsize=15)
plt.xlabel("Day",fontsize=15)
plt.ylabel("No.",fontsize=15)
plt.show()
```

446

447 138.

448 UNSOLVED — 03/09/2023 4:01 PM



449

450 139.

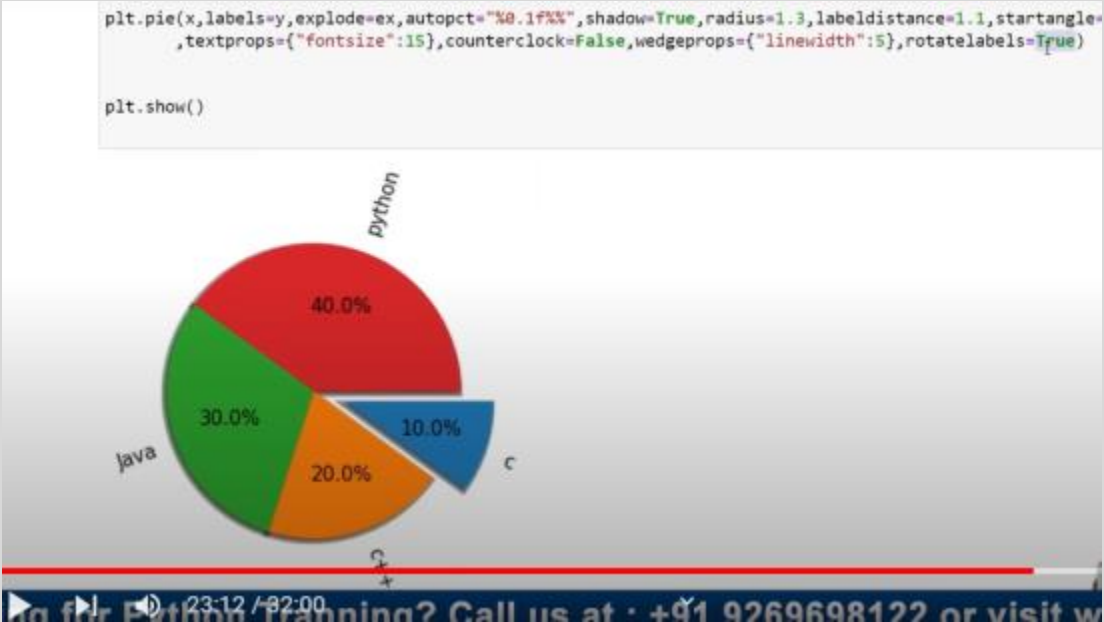
451 UNSOLVED — 03/09/2023 10:15 PM

```
plt.pie(x,labels=y,explode=ex,autopct="%0.1f%%",shadow=True,radius=1.3,labeldistance=1.1,startangle=0
,textprops={"fontsize":15},counterclock=False,wedgeprops={"linewidth":5,"edgecolor":"m"},
center=(3,6))

plt.show()
```

452

453 140. [10:15 PM]



454

455 141. [10:18 PM]

```
import matplotlib.pyplot as plt

x=[10,20,30,40]
y=["c","c++","Java","python"]
ex=[0.3,0.0,0.0,0.0]
c=["r","b","g","y"]

plt.pie(x,labels=y,explode=ex,autopct="%0.1f%%",shadow=True,radius=1,labeldistance=1.1,startangle=0
,textprops={"fontsize":15},counterclock=False,wedgeprops={"linewidth":5},rotatelabels=False)

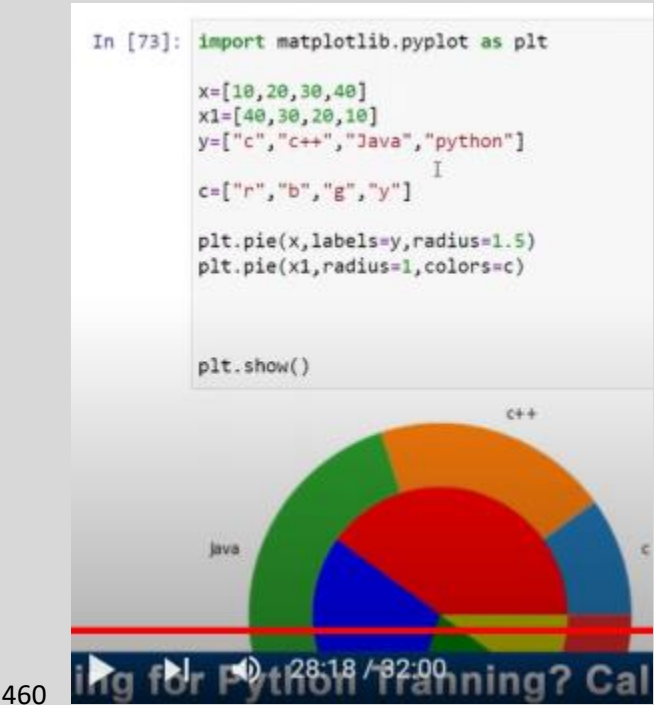
plt.title("Wscube Tech")
plt.legend(loc=2)
plt.show()
```

456

457 142. [10:18 PM]

458 dot pie chart

459 143. [10:19 PM]

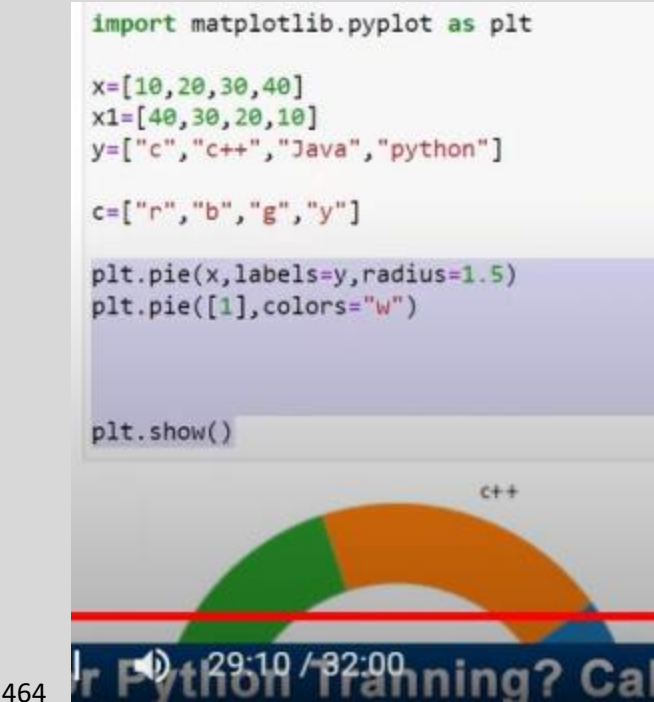


460

461 144. [10:20 PM]

462 To make a ring

463 145. [10:20 PM]



464

465 146. [10:20 PM]

466 or this



467

468 147.

469 UNSOLVED — 03/09/2023 11:51 PM

```
- `` : blue solid line
-- `` : green dashed line
^ `` : red triangles
D `` : cyan diamonds
> `` : magenta filled triangles pointing right
x `` : yellow X's
```

470

b	blue	T	T
g	green	s	square
r	red	d	diamond
c	cyan	v	triangle (down)
m	magenta	^	triangle (up)
y	yellow	<	triangle (left)

471

472

k	black	>	triangle (right)
w	white	p	pentagram
.	point	h	hexagram
o	circle	-	solid
x	x-mark	:	dotted
+	plus	-.	dashdot

473 March 14, 2023

474 148.

475 UNSOLVED — 03/14/2023 2:32 PM

476 that commented code is without using walrus operator and the uncommented code is by using the walrus operator

```
12 # foods = list()
13 # while True:
14 #     food = input("What food do you like?: ")
15 #     if food == "quit":
16 #         break
17 #     foods.append(food)
18 foods = list()
19 while (food := input("What food do you like?: ")) != "quit":
20     foods.append(food)
```

477

478 149.

479 UNSOLVED — 03/14/2023 3:50 PM

```
PythonLearnings > karo.py > ...
1 import re
2 pattern = "was"
3 text = '''Intense Tropical Cyclone Dumazile was a strong
4     tropical cyclone that brought flooding to Madagascar
5     and Réunion in early March 2018. Dumazile originated
6     from an area of low pressure that formed in the South-West
7     Indian Ocean near Agaléga on 27 February. The system
8     concentrated into a tropical disturbance on 2 March
9     and was named the next day, as it intensified into a
10    tropical storm. Amid conditions conducive for intensification,
11    Dumazile strengthened over the next two days and reached
12    peak intensity on 5 March as an intense tropical cyclone,
13    with 10-minute sustained winds of 165 km/h (105 mph),
14    1-minute sustained winds of 205 km/h (125 mph), and a
15    central pressure of 945 hPa (27.91 inHg). The system
16    weakened steadily over the next couple days because
17    of increasing wind shear as it tracked to the southeast.
18    Dumazile became post-tropical on 7 March'''
19
20 print (text.index(pattern))
21 # print (len('Intense Tropical Cyclone Dumazile'))
22 print (text.find(pattern))
23 print (re.search(pattern,text))
```

480

481 March 16, 2023

482 150.

483 UNSOLVED — 03/16/2023 5:20 PM

Sure, I will try to explain more clearly.

xlim is a function that controls the range of the x-axis on your plot. You can use it to zoom in or out on your data. For example, if you have data from 0 to 100 on the x-axis, but you only want to see the data from 20 to 80, you can use xlim(20, 80) to change the range of the x-axis.

xrange is a function that creates a sequence of numbers that you can use as data for plotting. For example, if you want to plot a line that goes from (0, 0) to (9, 9), you can use xrange(10) to create a sequence of numbers from 0 to 9 for both the x and y values.

Does this make sense?

484

485