1. **Key features of python (python basics read)**

* Interpreted language
* dynamically-typed
* object oriented
* High-level language
* free

1. **Difference between list and tuple (python basics read)**
2. **Ternary operator in python**

*[on true] if [expression] else [on false]*

e.g. print("Hi") if a<b else print("Bye")

1. **Slicing and negative indexing**

A "slice" in Python is powerful way of referring to sub-parts of a string. The syntax is s[i:j] meaning the substring starting at index i, running up to but not including index j.

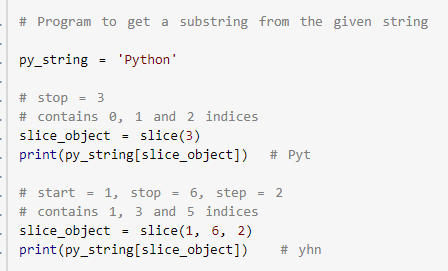
Python supports using negative numbers to index into a string: -1 means the last char, -2 is the next to last, and so on. In other words -1 is the same as the index len(s)-1, -2 is the same as len(s)-2. The negative numbers work analogously for the chars at the end of the string with -1, -2, etc. working from the right side.

s = 'Hello'

# -54321 ## negative index numbers

s[-2:] ## 'lo', begin slice with 2nd from the end

s[:-3] ## 'He', end slice 3rd from the end



1. **Is python case-sensitive? – Yes**
2. **istitle() will tell us if a string is in title case.**

'The Corpse Bride'.istitle()

other – lower(), upper(), islower(), isupper()

1. **Pass, break & continue statements**

the break statement breaks out of a loop.

for i in range(7):

if i==3: break

print(i)

1

2

the continue statement skips to the next iteration.

for i in range(7):

if i==3: continue

print(i)

1

2

4

5

6

1. **Explain help() and dir() functions in Python.**

The help() function displays the documentation string and help for its argument.

The dir() function displays all the members of an object(any kind).

1. **How do you get a list of all the keys in a dictionary?**

For this, we use the function keys().

mydict={'a':1,'b':2,'c':3,'e':5}

mydict.keys()

dict\_keys([‘a’, ‘b’, ‘c’, ‘e’])

1. **How will you check if all characters in a string are alphanumeric?**

For this, we use the method isalnum().

1. **How will you capitalize the first letter of a string?**

'ayushi'.capitalize()

1. **Limitations of python**

* Python’s interpreted nature imposes a speed penalty on it.
* While Python is great for a lot of things, it is weak in mobile computing, and in browsers.
* Being dynamically-typed, Python uses duck-typing (If it looks like a duck, it must be a duck). This can raise runtime errors.
* Python has underdeveloped database access layers. This renders it a less-than-perfect choice for huge database applications.

1. **Process large file in python without memory error**

|  |  |
| --- | --- |
|  | # The below program will read large csv file and breaks it into chunks for processing |
|  |  |
|  | import pandas as pd |
|  | batch=1 |
|  | for file in pd.read\_csv('file.csv',chunksize=50000,engine='python'): |
|  | filename='file\_'+str(batch)+'.csv' |
|  | file.to\_csv(filename,index=False) |
|  | batch+=1 |
|  | print('File divided into {0} chunks!'.format(batch)) |
|  |  |

1. **Sample code to get the next bus details**

* Usage of xml.etree.ElementTree
* make file executable using first line of code
* urllib to get details from webpage
* sys.argv to get any number of arguments. 0 being file name.



1. **Debug ways**

Run Python with the -i option. It will run the program and then drop you into Python's interactive prompt afterwords. This can be a very useful way to kind of poke around and look at what is happening.

Python debugger- if running with –i option then using

**import pdb**

**pdb.pm()** (post-mortem)

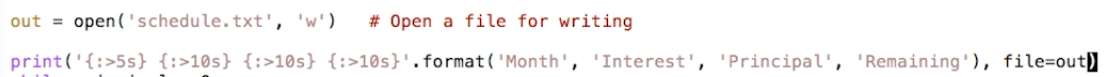
will give you exact line on which error occurred.

It can also be put in code.

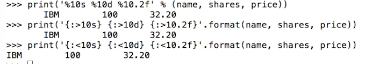
import pdb

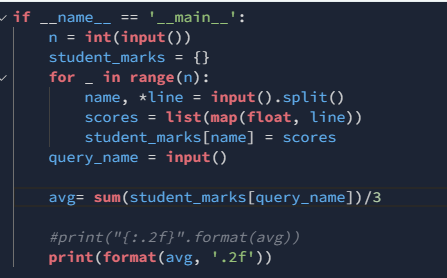
pdb.set\_trace()

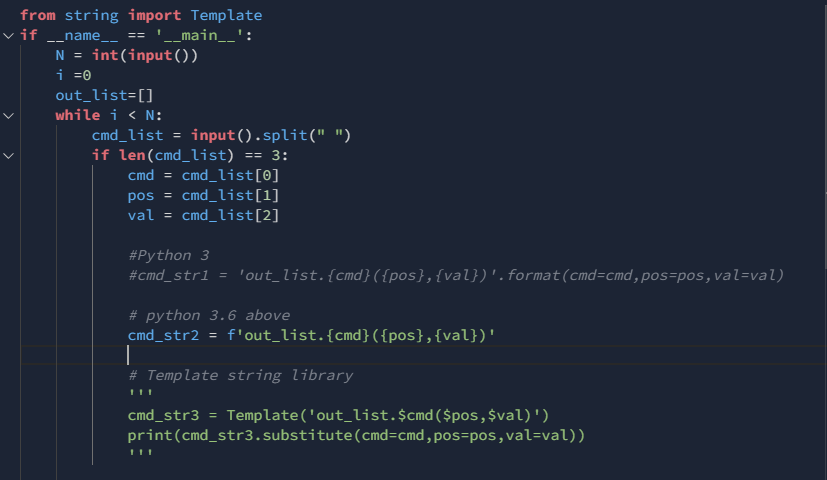
1. **write data into file**



1. **Format data**







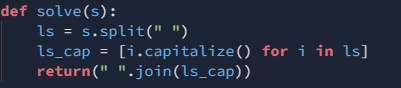
1. **eval**

to evaluate commands available as string

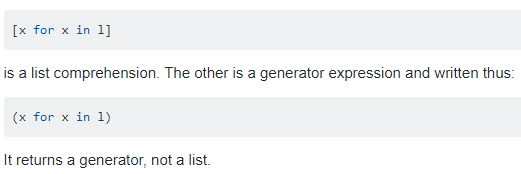
1. **capitalize()**

for capitalizing first character in string

**return**(' '.join(i.capitalize() **for** i **in** s.split(' ')))



1. **Generators and list comprehension**



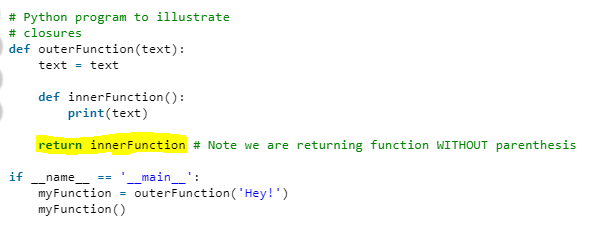
1. **Closures**

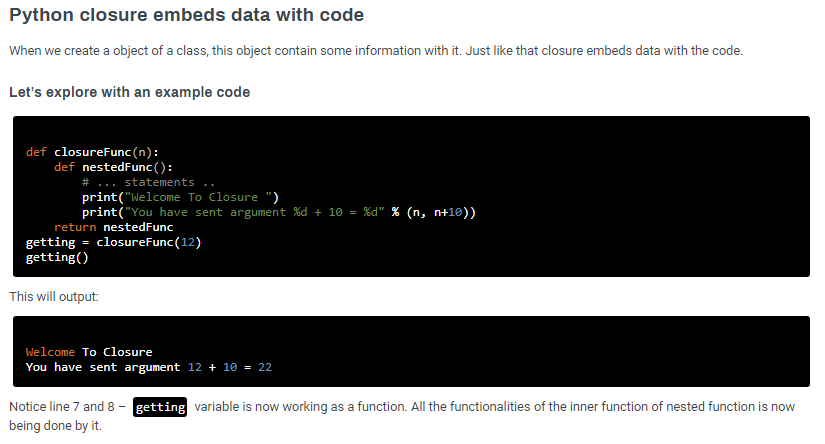
When a function returns another function defined in ( ie. nested function) it to get all its functionalities , it’s called a closure.

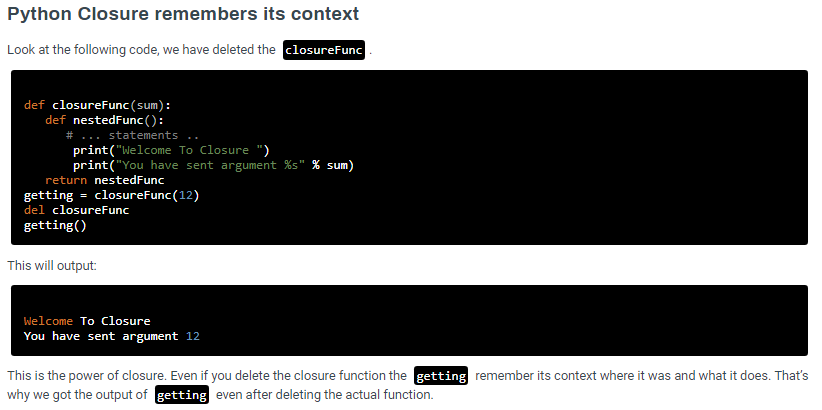
A Closure is a function object that remembers values in enclosing scopes even if they are not present in memory.Closures help to invoke function outside their scope.

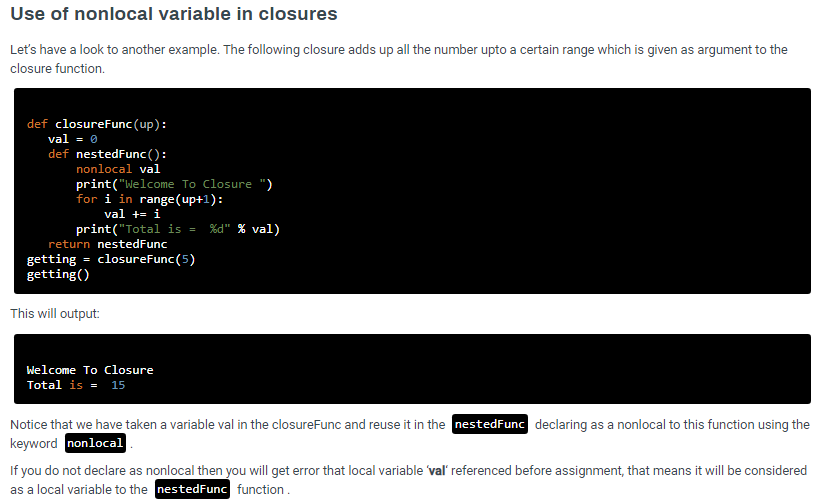
Closures can avoid the use of global values and provides some form of **data hiding**.

It can also provide an **object oriented solution** to the problem.



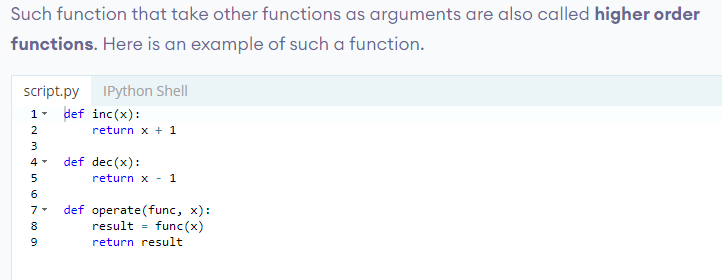








1. **Higher order function**



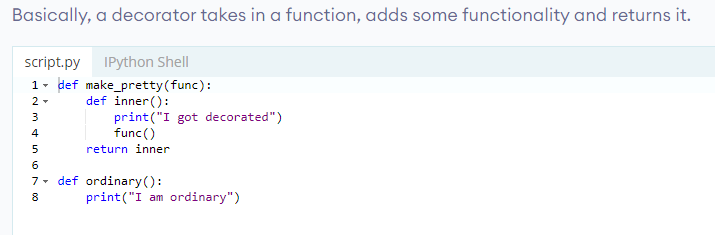
1. **Decorators**

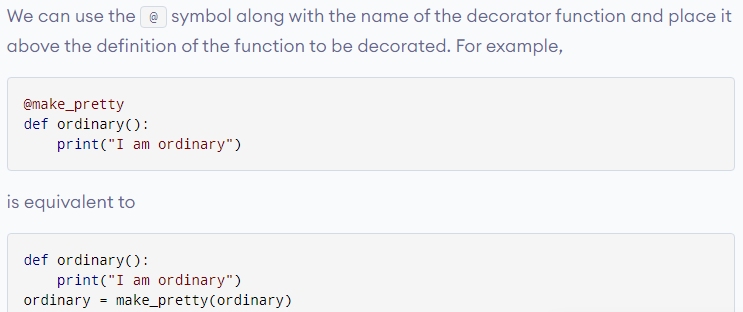
A decorator takes in a function, adds some functionality and returns it.

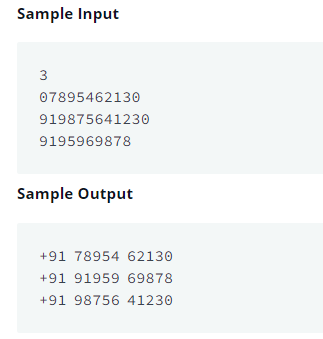
Python has an interesting feature called decorators to add functionality to an existing code.

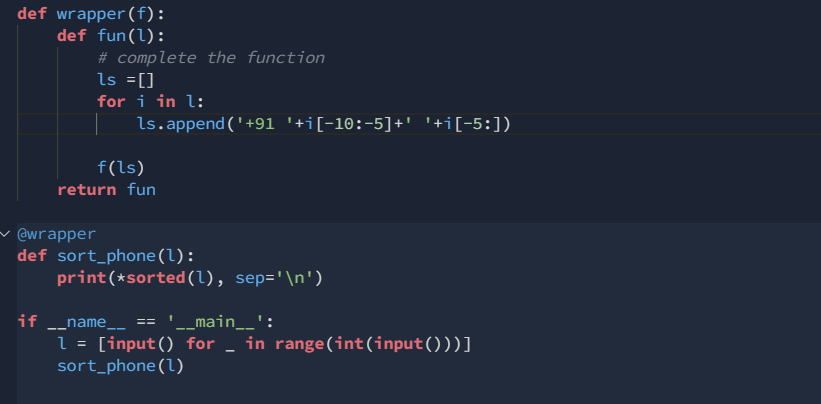
This is also called metaprogramming as a part of the program tries to modify another part of the program at compile time.

[Decorators](https://www.geeksforgeeks.org/function-decorators-in-python-set-1-introduction/) are very powerful and useful tool in Python since it allows programmers to modify the behavior of function or class. Decorators allow us to wrap another function in order to extend the behavior of wrapped function, without permanently modifying it. In Decorators, functions are taken as the argument into another function and then called inside the wrapper function.

****

****

****

****