**1. How are global variables shared across modules in Python?**

The canonical way to share information

across modules within a single program is

to create a special configuration module

(often called config or cfg). Just import the

configuration module in all modules of your application;

the module then becomes available as a global name.

Because there is only one instance of each module,

any changes made to the module object get reflected everywhere. For example:

File: config.py

x = 0 # Default value of the 'x' configuration setting

File: mod.py

import config

config.x = 1

File: main.py

import config

import mod

print config.x

Module variables are also often used to implement the Singleton design pattern, for the same reason.

**2. Use input to prompt the user for a number, store the number entered as a float in a variable named num, and then print the contents of num.**

ànum = input(‘Enter float value’)

àprint(‘Float number is’, num)

1. **Variable kingdoms refers to the list ['Bacteria', 'Protozoa', 'Chromista', 'Plantae', 'Fungi', 'Animalia']. Using kingdoms and either slicing or indexing with positive indices, write expressions that produce the following:**

**a. The first item of kingdoms**

**b. The last item of kingdoms**

**c. The list ['Bacteria', 'Protozoa', 'Chromista']**

**d. The list ['Chromista', 'Plantae', 'Fungi']**

**e. The list ['Fungi', 'Animalia']**

**f. The empty list**

kingdoms = ['Bacteria', 'Protozoa', 'Chromista', 'Plantae', 'Fungi', 'Animalia']

testA = kingdoms[:1]

testB = kingdoms[5:]

testC = kingdoms[:3]

testD = kingdoms[2:5]

testE = kingdoms[4:6]

testF = kingdoms[:0]

print(testA)

print(testB)

print(testC)

print(testD)

print(testE)

print(testF)

**6. What happens if you attempt to use a variable within a program, and that variable has not been assigned a value?**

* Var1
* Print(Var1)
* Error occur Var1 not defined ..

**8. Classify each of the following as either a legal or illegal Python identifier:**

1. fred=1
2. if=1 #invalid syntax
3. 2x=1 #invalid syntax
4. -4=1 #can't assign to operator
5. sum\_total=1
6. sumTotal=1
7. sum-total=1 #can't assign to operator
8. sum total=1 #invalid syntax
9. sumtotal=1
10. While=1
11. x2=1
12. Private=1
13. public=1
14. $16=1 #invalid syntax
15. xTwo=1
16. \_static=1
17. \_4=1
18. \_\_=1
19. 10%=1 #can't assign to operator
20. a27834=1
21. wilma's=1 #EOL while scanning string literal

**7. In Python can you assign more than one variable in a single statement?**

* a,b,c=1,2,3
* print(a)
* print(b)
* print(c)
* output :- 1 2 3

Docstring

* Docstring is short for documentation string. It is a string that occurs as the first statement in a module, function, class, or method definition. We must write what a function/class does in the docstring. Triple quotes are used while writing docstrings. For example:
* def double(num):
* """Function to double the value"""
* return 2\*num
* Docstring is available to us as the attribute \_\_doc\_\_ of the function.
* >>> print(double.\_\_doc\_\_)
* Function to double the value

**10. What is the difference between the following two strings? 'n' and '\n'?**

* ‘\n’ use for characters put a new line character into the string at that point.
* ‘n’ it is a simple string character.

12**Write a generator function named evens that enables the following code:**

**for n in evens\_less\_than(50):**

**print(n, end=' ')**

**print()**

**to print all positive even numbers less than 50.**

for evens\_less\_than in range(50):

if evens\_less\_than%2==0:

print(evens\_less\_than)

**13.Can a Python list hold a mixture of integers and strings?**

à Yes, Python list hold a mixture of integers and strings

Example for,

**HD=[80,49.949,'hiren']**

**print(HD) # HD stand for HIREN DANGAR**

**14. What happens if you attempt to access an element of a list using a negative index?**

**à** You attempt to access an element of a list using a negative index the list display in reverse order

Example for,

var = [1,2,3,4,5]

print(var[0])

print(var[-2])

print(var[-5])

output à 1 , 4 , 1

**15. What happens if you create a string and you print [3:] of that string?**

**à** It displays the full string after the three (3) characters.

Example for,

var = 'hirendangar'

print(var[3:])

output à endanger

**16. If “Python Programming” is a string, what is at index position ‘0’?**

**à** Yes in python programming string index position consider as **0.**

Example for,

var = [1,2,3,4,5,6,7,8,9,0]

print(var[10])

output à IndexError: list index out of range

**17. How do we modify a tuple?**

Example for,

var = [13,15,16,17]

print(var)

var[0] = 14

print(var)

output à[13,15,16,17] , [14,15,16,17]

**18. How do we remove an item from a list?**

Example for,

var = [13,15,16,17]

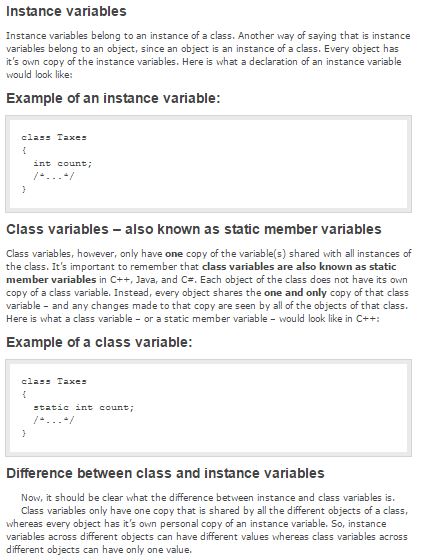
print(var)

var.remove(var[0])

print(var)

output à[13,15,16,17] , [15,16,17]

**19. Explain the similarities and differences between instance variables and “regular” function variables**

****

**20.** [**Explain how python is interpreted.**](http://www.careerride.com/python-how-it-is-interpreted.aspx)

àPython program runs directly from the source code. Each type Python programs are executed code is required. Python converts source code written by the programmer into intermediate language which is again translated it into the native language / machine language that are executed. So Python is an Interpreted language.

**21.** [**What are the rules for local and global variables in Python?**](http://www.careerride.com/python-rules-for-local-and-global-variables.aspx)

If a variable is defined outside function then it is implicitly **global**. If variable is assigned new value inside the function means it is **local**. If we want to make it global we need to explicitly define it as global. Variable referenced inside the function are implicit **global**.

def fun1(a):  
            print 'a:', a  
            a= 33;  
            print 'local a: ', a  
a = 100  
fun1(a)  
print 'a outside fun1:', a  
def fun2():  
           global b  
           print 'b: ', b  
           b = 33  
           print 'global b:', b  
b =100  
fun2()  
print 'b outside fun2', b

**Output** à   
$ python variable\_localglobal.py  
a: 100  
local a: 33  
a outside fun1: 100  
b :100  
global b: 33  
b outside fun2: 33

**22.** [**Explain indexing and slicing operation in sequences**](http://www.careerride.com/python-indexing-and-slicing-operation.aspx)

* Different types of sequences in python are strings, Unicode strings, lists, tuples, buffers, and xrange objects
* **Slicing & indexing operations** are salient features of sequence.
* indexing operation allows to access a particular item in the sequence directly ( similar to the array/list indexing) and the slicing operation allows to retrieve a part of the sequence.
* The slicing operation is used by specifying the name of the sequence followed by an optional pair of numbers separated by a colon within square brackets say **S[startno.:stopno]**.
* The startno in the slicing operation indicates the position from where the slice starts and the stopno indicates where the slice will stop at.
* If the startno is ommited, Python will start at the beginning of the sequence. If the stopno is ommited, Python will stop at the end of the sequence..
* Following code will further explain **indexing & slicing operation**:

>>> cosmeticList =[‘lipsstick’,’facepowder’,eyeliner’,’blusher’,kajal’]   
>>> print “Slicing operation :”,cosmeticList[2:]  
Slicing operation :[‘eyeliner’,’blusher’,kajal’]  
>>>print “Indexing operation :”,cosmeticList[0]  
“Indexing operation :lipsstick

**23. What are uses of lambda?**

* Using lambda keyword tiny anonymous functions can be created.
* It is a very powerful feature of Python which declares a one-line unknown small function on the fly. The lambda is used to create new function objects and then return them at runtime.
* The general format for lambda form is:   
  lambda parameter(s): expression using the parameter(s)
* For instance k is lambda function-  
  >>> k= lambda y: y + y  
  >>> k(30)  
  60  
  >>> k(40)  
  80

**24. What is the use of yield statement?**

Yield is a keyword that is used like return, except the function will return a generator.

>>> def createGenerator():

... mylist = range(3)

... for i in mylist:

... yield i\*i

...

>>> mygenerator = createGenerator() # create a generator

>>> print(mygenerator) # mygenerator is an object!

<generator object createGenerator at 0xb7555c34>

>>> for i in mygenerator:

... print(i)

0

1

4

Here it’s a useless example, but it’s handy when you know your function will return a huge set of values that you will only need to read once.

To master yield, you must understand that **when you call the function, the code you have written in the function body does not run.** The function only returns the generator object, this is a bit tricky.

**25. What are the steps to create own exceptions? Give example**

Python also allows you to create your own exceptions by deriving classes from the standard built-in exceptions.

Here is an example related to *RuntimeError*. Here, a class is created that is subclassed from *RuntimeError*. This is useful when you need to display more specific information when an exception is caught.

In the try block, the user-defined exception is raised and caught in the except block. The variable e is used to create an instance of the class *Networkerror*.

class Networkerror(RuntimeError):

def \_\_init\_\_(self, arg):

self.args = arg

So once you defined above class, you can raise the exception as follows −

try:

raise Networkerror("Bad hostname")

except Networkerror,e:

print e.args

**26. What is self in python? And Is it mandatory to use self keyword as a first argument of class function or we can change self to any other name?**

self represents the instance of the class. By using the “self” keyword we can access the attributes and methods of the class in python. It binds the attributes with the given arguments.

Self is not a mandatory keyword. But yes, the first argument of class function will always represent the instance of class.