Top 100 Basic Python Interview Questions

Q1. **What is Python? What are the benefits of using Python**? **What is the difference between list and tuples in Python?**

Python is a programming language with objects, modules, threads, exceptions and automatic memory management.

The benefits of pythons are that it is simple and easy, portable, extensible, build-in data structure and it is an open source.

LIST	**TUPLES**
Lists are mutable i.e they can be edited.	Tuples are immutable (tuples are lists which can't be edited).
Lists are slower than tuples.	Tuples are faster than list.
Syntax: list_1 = [10, 'Chelsea', 20]	Syntax: tup_1 = (10, 'Chelsea' , 20)

Examples:

```
>>> mylist=[1,3,3]
>>> mylist[1]=2
>>> mytuple=(1,3,3)
>>> mytuple[1]=2



Traceback (most recent call last):
File "<pyshell#97>", line 1, in <module>
mytuple[1]=2
TypeError: 'tuple' object does not support item assignment
```

Q2. What are the key features of Python?

Python is an **interpreted** language. Python program runs directly from the source code. It converts the source code that is written by the programmer into an intermediate language, which is again translated into machine language that has to be executed.That means that, unlike languages like *C* and its variants, Python does not need to be compiled before it is run. Other interpreted languages include *PHP* and *Ruby*.

- Python is **dynamically typed**, this means that you don’t need to state the types of variables when you declare them or anything like that. You can do things like x=111 and then x="I'm a string" without error
- Python is well suited to **object orientated programming** in that it allows the definition of classes along with composition and inheritance.
- Python does not have access specifiers (like C++’s public, private).
- Concise and simple
- Free
- Has a large community

Q3. What type of language is python? Programming or scripting?

Ans: Python is capable of scripting, but in general sense, it is considered as a general-purpose programming language. To know more about Scripting, you can refer to the Python Scripting Tutorial.

Q4. What are Python decorators?

A Python decorator is a specific change that we make in Python syntax to alter functions easily.

A decorator is a function that adds functionality to another function without modifying it. It wraps another function to add functionality to it. Take an example.

```
>>> def decor(func):
def wrap():
print("$$$$$$$$$$$$$$$$$")
func()
print("$$$$$$$$$$$$$$$$$")
return wrap
>>> @decor
def sayhi():
print("Hi")
>>> sayhi()
$$$$$$$$$$$$$$$$$
Hi
$$$$$$$$$$$$$$$$$
```

Decorators are an example of metaprogramming, where one part of the code tries to change another.

Q5.What is pep 8?

Ans: PEP stands for **Python Enhancement Proposal** .PEP 8 is a coding convention, a set of recommendation, about how to write your Python code more readable.

Q6. How is memory managed in Python?

Ans: Memory management in python is managed by *Python private heap space*. All Python objects and data structures are located in a private heap. The programmer does not have access to this private heap. The python interpreter takes care of this instead.

The allocation of heap space for Python objects is done by Python's memory manager. The core API gives access to some tools for the programmer to code.

Python also has an inbuilt garbage collector, which recycles all the unused memory and so that it can be made available to the heap space.

Q7. What is namespace in Python?

Ans: A namespace is a naming system used to make sure that names are unique to avoid naming conflicts.

Q8. What is PYTHONPATH?

Ans: It is an environment variable which is used when a module is imported. Whenever a module is imported, PYTHONPATH is also looked up to check for the presence of the imported modules in various directories. The interpreter uses it to determine which module to load.

Q9. What are python modules? Name some commonly used built-in modules in Python?

Ans: Python modules are files containing Python code. This code can either be functions classes or variables. A Python module is a .py file containing executable code.

Some of the commonly used built-in modules are:

- os
- sys
- math
- random
- data time
- JSON

Q10.What are local variables and global variables in Python? How can you share global variables across modules?

Global Variables:

Variables declared outside a function or in global space are called global variables. These variables can be accessed by any function in the program.

Local Variables:

Any variable declared inside a function is known as a local variable. This variable is present in the local space and not in the global space.

Example:

```
a=2
def add():
b=3
c=a+b
print(c)
add()
```

```
Output: 5
```

When you try to access the local variable outside the function add(), it will throw an error.

To share global variables across modules within a single program, create a special module. Import the config module in all modules of your application. The module will be available as a global variable across modules.

Q11. Is python case sensitive?

A language is case-sensitive if it distinguishes between identifiers like myname and Myname. In other words, it cares about case- lowercase or uppercase. Let’s try this with Python.

```
>>> myname='Ayushi'
>>> Myname


Traceback (most recent call last):
File “<pyshell#3>”, line 1, in <module>
Myname
NameError: name ‘Myname’ is not defined
```

As you can see, this raised a NameError. This means that Python is indeed case-sensitive.

Q12.What is type conversion in Python?

Ans: Type conversion refers to the conversion of one data type iinto another.

int() – converts any data type into integer type

float() – converts any data type into float type

ord() – converts characters into integer

**hex(**) – converts integers to hexadecimal

oct() – converts integer to octal

tuple() – This function is used to convert to a tuple.

set() – This function returns the type after converting to set.

list() – This function is used to convert any data type to a list type.

dict() – This function is used to convert a tuple of order (key,value) into a dictionary.

str() – Used to convert integer into a string.

complex(real,imag) – This functionconverts real numbers to complex(real,imag) number.

Q.57. How would you work with numbers other than those in the decimal number system?
With Python, it is possible to type numbers in binary, octal, and hexadecimal.

Binary numbers are made of 0 and 1. To type in binary, we use the prefix 0b or 0B.

```
>>> int(0b1010)
10
```

To convert a number into its binary form, we use bin().

```
>>> bin(0xf)
'0b1111'
```

Octal numbers may have digits from 0 to 7. We use the prefix 0o or 0O.

```
>>> oct(8)
'0o10'
```

Hexadecimal numbers may have digits from 0 to 15. We use the prefix 0x or 0X.

```
>>> hex(16)
'0x10'
>>> hex(15)
'0xf'
```

Q13. How to install Python on Windows and set path variable?

Ans: To install Python on Windows, follow the below steps:

Install python from this link: https://www.python.org/downloads/

After this, install it on your PC. Look for the location where PYTHON has been installed on your PC using the following command on your command prompt: cmd python.

Then go to advanced system settings and add a new variable and name it as PYTHON_NAME and paste the copied path.

Look for the path variable, select its value and select 'edit'.

Add a semicolon towards the end of the value if it's not present and then type %PYTHON_HOME%

Q14. Is indentation required in python?

Ans: Indentation is necessary for Python. It specifies a block of code. All code within loops, classes, functions, etc is specified within an indented block. It is usually done using four space characters. If your code is not indented necessarily, it will not execute accurately and will throw errors as well.

Q15. What is the difference between Python Arrays and lists?

Ans: Arrays and lists, in Python, have the same way of storing data. But, arrays can hold only a single data type elements whereas lists can hold any data type elements.

Example:

```
import array as arr

My_Array=arr.array('i',[1,2,3,4])

My_list=[1,'abc',1.20]

print(My_Array)

print(My_list)
```

Output:

```
array('i', [1, 2, 3, 4]) [1, 'abc', 1.2]
```

Q16. What are functions in Python?

Ans: A function is a block of code which is executed only when it is called. To define a Python function, the **def** keyword is used.

Example:

Let's define a function to take two numbers and return the greater number.

```
>>> def greater(a,b):
return a is a>b else b
```

```
>>> greater(3,3.5)
3.5
```

```
def Newfunc():
print("Hi, Welcome to Edureka")
Newfunc(); #calling the function
```

Output: Hi, Welcome to Edureka

Q17.What is __init__?

Ans: __init__ is a method or constructor in Python. This method is automatically called to allocate memory when a new object/ instance of a class is created. All classes have the __init__ method.

Here is an example of how to use it.

```
class Employee:
def __init__(self, name, age,salary):
self.name = name
self.age = age
self.salary = 20000
E1 = Employee("XYZ", 23, 20000)
# E1 is the instance of class Employee.
#__init__ allocates memory for E1.
print(E1.name)
print(E1.age)
print(E1.salary)
```

Output:

```
XYZ

23

20000
```

Q18.What is a lambda function? Why lambda forms in python does not have statements?

Ans: An anonymous function is known as a lambda function. This function can have any number of parameters but, can have just one statement.

A lambda form in python does not have statements as it is used to make new function object and then return them at runtime.

Example1:

```
a = lambda x,y : x+y
print(a(5, 6))
```

Output: 11:

Example2:

```
>>> (lambda a,b:a if a>b else b)(3,3.5)
```

3.5

Here, a and b are the inputs. a if a>b else b is the expression to return. The arguments are 3 and 3.5.

Example3:

It is possible to not have any inputs here.

```
>>> (lambda :print("Hi"))()
```

Hi

Q19.What is self in Python?

Ans: Self is an instance or an object of a class. In Python, this is explicitly included as the first parameter. However, this is not the case in Java where it's optional. It helps to differentiate between the methods and attributes of a class with local variables.

The self variable in the init method refers to the newly created object while in other methods, it refers to the object whose method was called.

Q20. How does break, continue and pass work?

Break	Allows loop termination when some condition is met and the control is transferred to the next statement.

Continue	Allows skipping some part of a loop when some specific condition is met and the control is transferred to the beginning of the loop
Pass	Used when you need some block of code syntactically, but you want to skip its execution. This is basically a null operation. Nothing happens when this is executed.

There may be times in our code when we haven't decided what to do yet, but we must type something for it to be syntactically correct. In such a case, we use the pass statement.

```
>>> def func(*args):
pass
>>>
```

Similarly, the break statement breaks out of a loop.

```
>>> for i in range(7):
if i==3: break
print(i)
```

```
1
2
```

Finally, the continue statement skips to the next iteration.

```
>>> for i in range(7):
if i==3: continue
print(i)
```

```
1
2
4
5
6
```

Q21. And how do you reverse a list?
Using the reverse() method.

```
>>> a.reverse()
>>> a
```

```
[4, 3, 2, 1]
```

You can also do it via slicing from right to left:

```
>>> a[::-1]
>>> a
```
[1, 2, 3, 4]
This gives us the original list because we already reversed it once. However, this does not modify the original list to reverse it.

Q22. How can you randomize the items of a list in place in Python?
Ans: Consider the example shown below:

```
from random import shuffle

x = ['Keep', 'The', 'Blue', 'Flag', 'Flying', 'High']

shuffle(x)

print(x)
```

The output of the following code is as below.

```
['Flying', 'Keep', 'Blue', 'High', 'The', 'Flag']
```

Q23. What are python iterators? In Python what are iterators?
Ans: Iterators are objects which can be iterated upon. In Python, iterators are used to iterate a group of elements, containers like list.

Q.24. In one line, show us how you'll get the max alphabetical character from a string.
For this, we'll simply use the max function.

```
>>> max('flyiNg')
'y'
```

The following are the ASCII values for all the letters of this string-

f- 102

l- 108

y- 121

105

N- 78

g- 103

By this logic, try to explain the following line of code-

```
>>> max('fly{}iNg')
```
'}'

(Bonus: } – 125)

Q.25. What is Python good for?
Python is a jack of many trades, check out Applications of Python to find out more.

Meanwhile, we'll say we can use it for:

- Web and Internet Development
- Desktop GUI
- Scientific and Numeric Applications
- Software Development Applications
- Applications in Education
- Applications in Business
- Database Access
- Network Programming
- Games, 3D Graphics
- Other Python Applications

Q.26. What does [::-1} do?
Ans: [::-1] is used to reverse the order of an array or a sequence.

For example:

```
import array as arr

My_Array=arr.array('i',[1,2,3,4,5])

My_Array[::-1]
```

blockOutput: array('i', [5, 4, 3, 2, 1])

[::-1] reprints a reversed copy of ordered data structures such as an array or a list. the original array or list remains unchanged.

Q.27. What will the following code output?

```
>>> word='abcdefghij'
>>> word[:3]+word[3:]
```

The output is 'abcdefghij'. The first slice gives us 'abc', the next gives us 'defghij'.

Q.28. How will you convert a list into a string?
We will use the join() method for this.

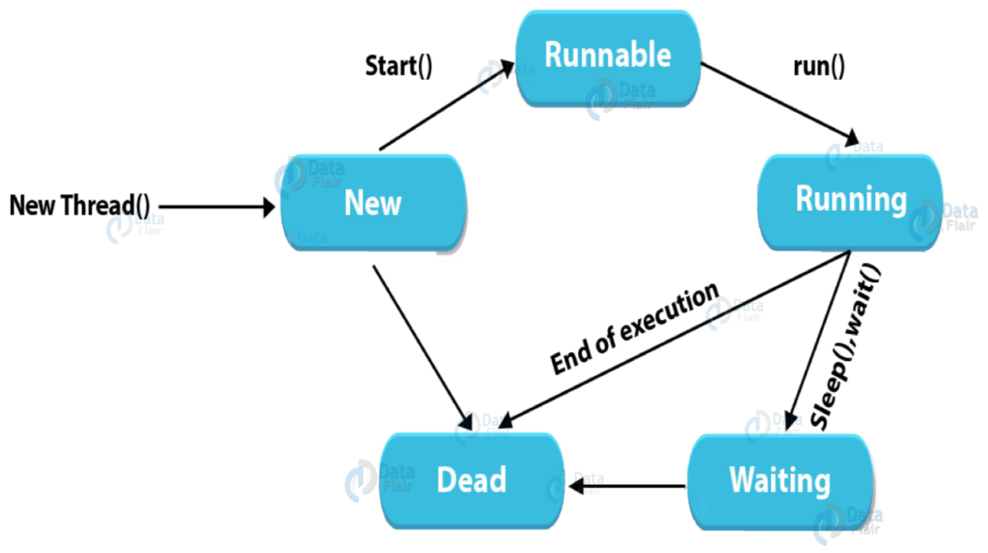
```
>>> nums=['one','two','three','four','five','six','seven']
>>> s=' '.join(nums)
>>> s
```

'one two three four five six seven'

Q.29. How will you remove a duplicate element from a list?
We can turn it into a set to do that.

```
>>> list=[1,2,1,3,4,2]
>>> set(list)
{1, 2, 3, 4}
```

Q.30. Can you explain the life cycle of a thread?



To create a thread, we create a class that we make override the run method of the thread class. Then, we instantiate it.
A thread that we just created is in the new state. When we make a call to start() on it, it forwards the threads for scheduling. These are in the ready state.
When execution begins, the thread is in the running state.
Calls to methods like sleep() and join() make a thread wait. Such a thread is in the waiting/blocked state.
When a thread is done waiting or executing, other waiting threads are sent for scheduling.
A running thread that is done executing terminates and is in the dead state.

Q.31. What is a dictionary in Python?
A ***python dictionary*** is something I have never seen in other languages like C++ or Java programming. It holds key-value pairs.

```
>>> roots={25:5,16:4,9:3,4:2,1:1}
>>> type(roots)
<class 'dict'>
>>> roots[9]
3
```

A dictionary is mutable, and we can also use a comprehension to create it.

```
>>> roots={x**2        for x in range(5,0,-1)}
>>> roots
```

{25: 5, 16: 4, 9: 3, 4: 2, 1: 1}

Q.32. Take a look at this piece of code:

```
>>> A0= dict(zip(('a','b','c','d','e'),(1,2,3,4,5)))
>>> A1= range(10)
>>> A2= sorted([i for i in A1 if i in A0])
>>> A3= sorted([A0[s] for s in A0])
>>> A4= [i for i in A1 if i in A3]
>>> A5= {i:i*i for i in A1}
>>> A6= [[i,i*i] for i in A1]
>>> A0,A1,A2,A3,A4,A5,A6
```

Here you go:

A0= {‘a’: 1, ‘b’: 2, ‘c’: 3, ‘d’: 4, ‘e’: 5}

A1= range(0, 10)

A2= []

A3= [1, 2, 3, 4, 5]

A4= [1, 2, 3, 4, 5]

A5= {0: 0, 1: 1, 2: 4, 3: 9, 4: 16, 5: 25, 6: 36, 7: 49, 8: 64, 9: 81}

A6= [[0, 0], [1, 1], [2, 4], [3, 9], [4, 16], [5, 25], [6, 36], [7, 49], [8, 64], [9, 81]]

Now to find out what happened. A0 zips ‘a’ with 1, ‘b’ with 2, and so on. This results in tuples, which the call to dict() then turns into a dictionary by using these as keys and values.

A1 gives us a range object with start=0 and stop=10.
A2 checks each item in A1- does it exist in A0 as well? If it does, it adds it to a list. Finally, it sorts this list. Since no items exist in both A0 and A1, this gives us an empty list.
A3 takes each key in A0 and returns its value. This gives us the list [1,2,3,4,5].
A4 checks each item in A1- does it exist in A3 too? If it does, it adds it to a list and returns this list.
A5 takes each item in A1, squares it, and returns a dictionary with the items in A1 as keys and their squares as the corresponding values.
A6 takes each item in A1, then returns sublists containing those items and their squares- one at a time.

Q.33. Does Python have a switch-case statement?

In languages like C++, we have something like this:

```
switch(name)
{
case ‘Ayushi’:
cout<<”Monday”;
break;
case ‘Megha’:
cout<<”Tuesday”;
break;
default:
cout<<”Hi, user”;
}
```

But in Python, we do not have a ***switch-case statement***. Here, you may write a switch function to use. Else, you may use a set of if-elif-else statements. To implement a function for this, we may use a dictionary.

```
>>> def switch(choice):
switcher={
'Ayushi':'Monday',
'Megha':'Tuesday',
print(switcher.get(choice,'Hi, user'))
```

return

```
>>> switch('Megha')
```

Tuesday

```
>>> switch('Ayushi')
```

Monday

```
>>> switch('Ruchi')
```

Hi, user

Here, the get() method returns the value of the key. When no key matches, the default value (the second argument) is returned.

Q.34. Differentiate between deep and shallow copy.

A deep copy copies an object into another. This means that if you make a change to a copy of an object, it won't affect the original object. In Python, we use the function deepcopy() for this, and we import the module copy. We use it like:

```
>>> import copy
>>> b=copy.deepcopy(a)
```

A shallow copy, however, copies one object's reference to another. So, if we make a change in the copy, it will affect the original object. For this, we have the function copy(). We use it like:

```
>>> b=copy.copy(a)
```

21) How can you copy an object in Python?

To copy an object in Python, you can try copy.copy () or copy.deepcopy() for the general case. You cannot copy all objects but most of them.

Q49. What is the difference between deep and shallow copy?

Ans: *Shallow copy* is used when a new instance type gets created and it keeps the values that are copied in the new instance. Shallow copy is used to copy the reference pointers just like it copies the values. These references point to the original objects and the changes made in any member of the class will also affect the original copy of it. Shallow copy allows faster execution of the program and it depends on the size of the data that is used.

Deep copy is used to store the values that are already copied. Deep copy doesn't copy the reference pointers to the objects. It makes the reference to an object and the new object that is pointed by some other object gets stored. The changes made in the original copy won't affect any other copy that uses the object. Deep copy makes execution of the program slower due to making certain copies for each object that is been called.

Q.35. Can you make a local variable's name begin with an underscore? (developer)

You can, but you should not. This is because:

Local variables indicate private variables of a class, and so, they confuse the interpreter.

Q.36. Is a NumPy array better than a list?

NumPy arrays have 3 benefits over lists:

They are faster

They require less memory

They are more convenient to work with

Q.37. If you installed a module with pip but it doesn't import in your IDLE, what could it possibly be?

Well, for one, it could be that I installed two versions of Python on my system- possibly, both 32-bit and 64-bit.

The Path variable in my system's environment variables is probably set to both, but one of them prior to the other- say, the 32-bit.

This made the command prompt use the 32-bit version of pip to install the module I chose.

When I ran the IDLE, I ran the 64-bit version.

As this sequence of events unlapped, I couldn't import the module I just installed.

Q.38. Based on your previous answer, how will you solve this issue?

I could do two things.

The temporary solution- I will add the path to sys manually every time I work on a new session of the interpreter.

```
>>>
sys.path.append('C:\\Users\\Ayushi\\AppData\\Local\\Programs\\Python\\Python37\\Scripts')
```

The permanent solution- I will update the value of Path in my environment variables to hold the location of the Scripts folder for the 64-bit version first.

Q.39. If while installing a package with pip, you get the error No matching installation found, what can you do?

In such a situation, one thing I can do is to download the binaries for that package from the following location:

https://www.lfd.uci.edu/~gohlke/pythonlibs/

Then, I can install the wheel using pip.

Q.40. How can you keep track of different versions of code?

To make this happen, we implement version control. For this, one tool you can use is Git.

Q.41. How do you debug a program in Python? Answer in brief.

To debug a Python program, we use the pdb module. This is the ***Python debugger.*** If we start a program using pdb, it will let us step through the code.

Q.42. Can I dynamically load a module in Python?

Dynamic loading is where we do not load a module till we need it. This is slow, but lets us utilize the memory more efficiently. In Python, you can use the importlib module for this:

```
import importlib
```

```
module = importlib.import_module('my_package.my_module')
```

Q.43. Which methods/functions do we use to determine the type of instance and inheritance?

Here, we talk about three methods/functions- type(), isinstance(), and issubclass().

type()

This tells us the type of object we’re working with.

```
>>> type(3)
```

<class ‘int’>

```
>>> type(False)
```

<class ‘bool’>

```
>>> type(lambda :print("Hi"))
```

<class ‘function’>

```
>>> type(type)
```

<class ‘type’>

isinstance()

This takes in two arguments- a value and a type. If the value is of the kind of the specified type, it returns True. Else, it returns False.

```
>>> isinstance(3,int)
```

True

```
>>> isinstance((1),tuple)
```

False

```
>>> isinstance((1,),tuple)
```

True

issubclass()

This takes two classes as ***arguments***. If the first one inherits from the second, it returns True. Else, it returns False.

```
>>> class A: pass
>>> class B(A): pass
>>> issubclass(B,A)
```

True

```
>>> issubclass(A,B)
```

False

Q.44. Are methods and constructors the same thing?

No, there are subtle but considerable differences-

We must name a constructor in the name of the class; a method name can be anything.

Whenever we create an object, it executes a constructor; whenever we call a method, it executes a method.

For one object, a constructor executes only once; a method can execute any number of times for one object.

We use constructors to define and initialize non-static variables; we use methods to represent business logic to perform operations.

Q.45. What is a Python module?

A module is a script in Python that defines import statements, functions, classes, and variables. It also holds runnable Python code. ZIP files and DLL files can be modules too. The module holds its name as a string that is in a global variable.

Q.46. What are the file-related modules we have in Python?

We have the following libraries and modules that let us manipulate text and binary files on our file systems-

os

os.path

shutil

Q.47. Explain, in brief, the uses of the modules sqlite3, ctypes, pickle, traceback, and itertools.

sqlite3- Helps with handling databases of type SQLite

ctypes- Lets create and manipulate C data types in Python

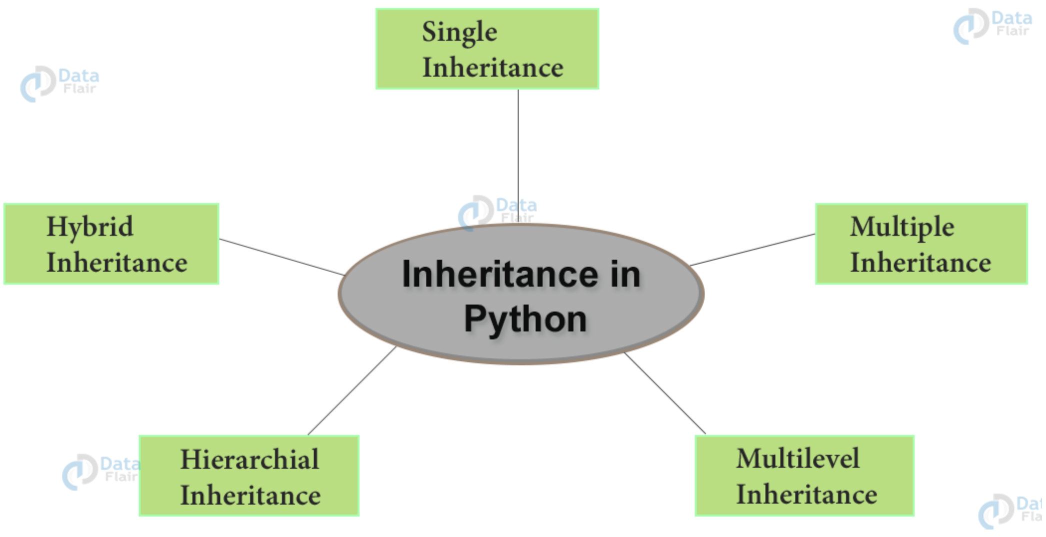
pickle- Lets put any data structure to external files

traceback- Allows extraction, formatting, and printing of stack traces

itertools– Supports working with permutations, combinations, and other useful iterables.

Q.48. Explain inheritance in Python.

When one class inherits from another, it is said to be the child/derived/sub class inheriting from the parent/base/super class. It inherits/gains all members (attributes and methods).



Inheritance lets us reuse our code, and also makes it easier to create and maintain applications. Python supports the following kinds of inheritance:

Single Inheritance- A class inherits from a single base class.

Multiple Inheritance- A class inherits from multiple base classes.

Multilevel Inheritance- A class inherits from a base class, which, in turn, inherits from another base class.

Hierarchical Inheritance- Multiple classes inherit from a single base class.

Hybrid Inheritance- Hybrid inheritance is a combination of two or more types of inheritance.

Q55. Explain Inheritance in Python with an example.

Ans: Inheritance allows One class to gain all the members(say attributes and methods) of another class. Inheritance provides code reusability, makes it easier to create and maintain an application. The class from which we are inheriting is called super-class and the class that is inherited is called a derived / child class.

They are different types of inheritance supported by Python:

Single Inheritance – where a derived class acquires the members of a single super class.

Multi-level inheritance – a derived class d1 in inherited from base class base1, and d2 are inherited from base2.

Hierarchical inheritance – from one base class you can inherit any number of child classes

Multiple inheritance – a derived class is inherited from more than one base class.

Q58. Does python support multiple inheritance?

Ans: Multiple inheritance means that a class can be derived from more than one parent classes. Python does support multiple inheritance, unlike Java.

Q.50. Write Python logic to count the number of capital letters in a file.

```
>>> import os
>>> os.chdir('C:\\Users\\lifei\\Desktop')
>>> with open('Today.txt') as today:
count=0
for i in today.read():
if i.isupper():
count+=1
print(count)
```

26

Q.51. How would you make a Python script executable on Unix?

For this to happen, two conditions must be met:

The script file’s mode must be executable

The first line must begin with a hash(#). An example of this will be: #!/usr/local/bin/python

28) Explain how can you make a Python Script executable on Unix?

To make a Python Script executable on Unix, you need to do two things,

Script file's mode must be executable and

the first line must begin with # (#!/usr/local/bin/python)

Q.52. What functions or methods will you use to delete a file in Python?

For this, we may use remove() or unlink().

```
>>> import os
>>> os.chdir('C:\\Users\\lifei\\Desktop')
>>> os.remove('try.py')
>>>
```

When we go and check our Desktop, the file is gone. Let’s go make it again so we can delete it again using unlink().

```
>>> os.unlink('try.py')
>>>
```

Both functions are the same, but unlink is the traditional Unix name for it.

Q43.How can files be deleted in Python?

Ans: To delete a file in Python, you need to import the OS Module. After that, you need to use the os.remove() function.

Example:

```
import os

os.remove("xyz.txt")
```

Q44. What are the built-in types of python?

Ans: Built-in types in Python are as follows –

Integers

Floating-point

Complex numbers

Strings

Boolean

Built-in functions

29) Explain how to delete a file in Python?

By using a command os.remove (filename) or os.unlink(filename)

Q47. How to remove values to a python array?

Ans: Array elements can be removed using **pop()** or **remove()** method. The difference between these two functions is that the former returns the deleted value whereas the latter does not.

Example:

```
a=arr.array('d', [1.1, 2.2, 3.8, 3.1, 3.7, 1.2, 4.6])

print(a.pop())

print(a.pop(3))

a.remove(1.1)

print(a)
```

Output:

4.6

3.1

array(‘d’, [2.2, 3.8, 3.7, 1.2])

Q.8. Is del the same as remove()? What are they?

del and remove() are methods on lists/ ways to eliminate elements.

```
>>> list=[3,4,5,6,7]
>>> del list[3]
>>> list
```

[3, 4, 5, 7]

```
>>> list.remove(5)
>>> list
```

[3, 4, 7]

While del lets us delete an element at a certain index, remove() lets us remove an element by its value.

Q.53. Can you write a function to generate the following pyramid?

```
*
***
*****
*******
*********
```

```
def pyramid(n):
for row in range(n):
for space in range(n-row):
print(' ',end='')
for star in range(row):
print('*',end='')
for star in range(row+1):
print('*',end='')
print()

pyramid(5)
```

Q.54. How will you print the contents of a file?

```
>>> try:
with open('tabs.txt','r') as f:
print(f.read())
```

```
except IOError:
print("File not found")
```

Q.55. What is Flask? What is Flask- WTF? Explain its features.

Python Flask, as we've previously discussed, is a web microframework for Python. It is based on the 'Werkzeug, Jinja 2 and good intentions' BSD license. Two of its dependencies are Werkzeug and Jinja2. This means it has around no dependencies on external libraries. Due to this, we can call it a light framework.

A session uses a signed cookie to allow the user to look at and modify session contents. It will remember information from one request to another. However, to modify a session, the user must have the secret key Flask.secret_key.

Flask-WTF supports simple integration with WTForms. It has the following features-

- Integration with wtforms
- Global csrf protection
- Recaptcha supporting
- Internationalization integration
- Secure form with csrf token
- File upload compatible with Flask uploads

Q28. What are the generators in python?
Ans: Functions that return an iterable set of items are called generators.

19) What are generators in Python?

The way of implementing iterators are known as generators. It is a normal function except that it yields expression in the function.

Q.58. Okay, we asked you about generators and iterators, and you gave us the right answers. But don't they sound similar?

They do, but there are subtle differences:

For a generator, we create a function. For an iterator, we use in-built functions iter() and next().
For a generator, we use the keyword ‘yield’ to yield/return an object at a time.
A generator may have as many ‘yield’ statements as you want.
A generator will save the states of the local variables every time ‘yield’ will pause the loop. An iterator does not use local variables; it only needs an iterable to iterate on.
Using a class, you can implement your own iterator, but not a generator.
Generators are fast, compact, and simpler.
Iterators are more memory-efficient.

Q.56. What is a generator?

Python generator produces a sequence of values to iterate on. This way, it is kind of an iterable. We define a function that ‘yields’ values one by one, and then use a for loop to iterate on it.

```
>>> def squares(n):
i=1
while(i<=n):
yield i**2
i+=1
>>> for i in squares(7):
print(i)
```

```
1
4
9
16
25
36
49
```

Q.57. So, what is an iterator, then?

An iterator returns one object at a time to iterate on. To create an iterator, we use the iter() function.

odds=iter([1,3,5,7,9])

Then, we call the next() function on it every time we want an object.

```
>>> next(odds)
```

1

```
>>> next(odds)
```

3

```
>>> next(odds)
```

5

```
>>> next(odds)
```

7

```
>>> next(odds)
```

9

And now, when we call it again, it raises a StopIteration exception. This is because it has reached the end of the values to iterate on.

```
>>> next(odds)
```

```
Traceback (most recent call last):
File "<pyshell#295>", line 1, in <module>
next(odds)
Stop Iteration
```

Q.58. How is multithreading achieved in Python?

A thread is a lightweight process, and multithreading allows us to execute multiple threads at once. As you know, Python is a multithreaded language. It has a multi-threading package.

The GIL (Global Interpreter Lock) ensures that a single thread executes at a time. A thread holds the GIL and does a little work before passing it on to the next thread. This makes for an illusion of parallel execution. But in reality, it is just threaded taking turns at the CPU. Of course, all the passing around adds overhead to the execution.

Q.59. What is tuple unpacking?

First, let's discuss tuple packing. It is a way to pack a set of values into a tuple.

```
>>> mytuple=3,4,5
>>> mytuple
```

(3, 4, 5)

This packs 3, 4, and 5 into mytuple.

Now, we will unpack the values from the tuple into variables x, y, and z.

```
>>> x,y,z=mytuple
>>> x+y+z
```

12

Q.60. What is a namedtuple?

A namedtuple will let us access a tuple’s elements using a name/label. We use the function namedtuple() for this, and import it from collections.

```
>>> from collections import namedtuple
>>> result=namedtuple('result','Physics Chemistry Maths') #format
>>> Ayushi=result(Physics=86,Chemistry=95,Maths=86) #declaring the tuple
>>> Ayushi.Chemistry
```

95

As you can see, it let us access the marks in Chemistry using the Chemistry attribute of object Ayushi.

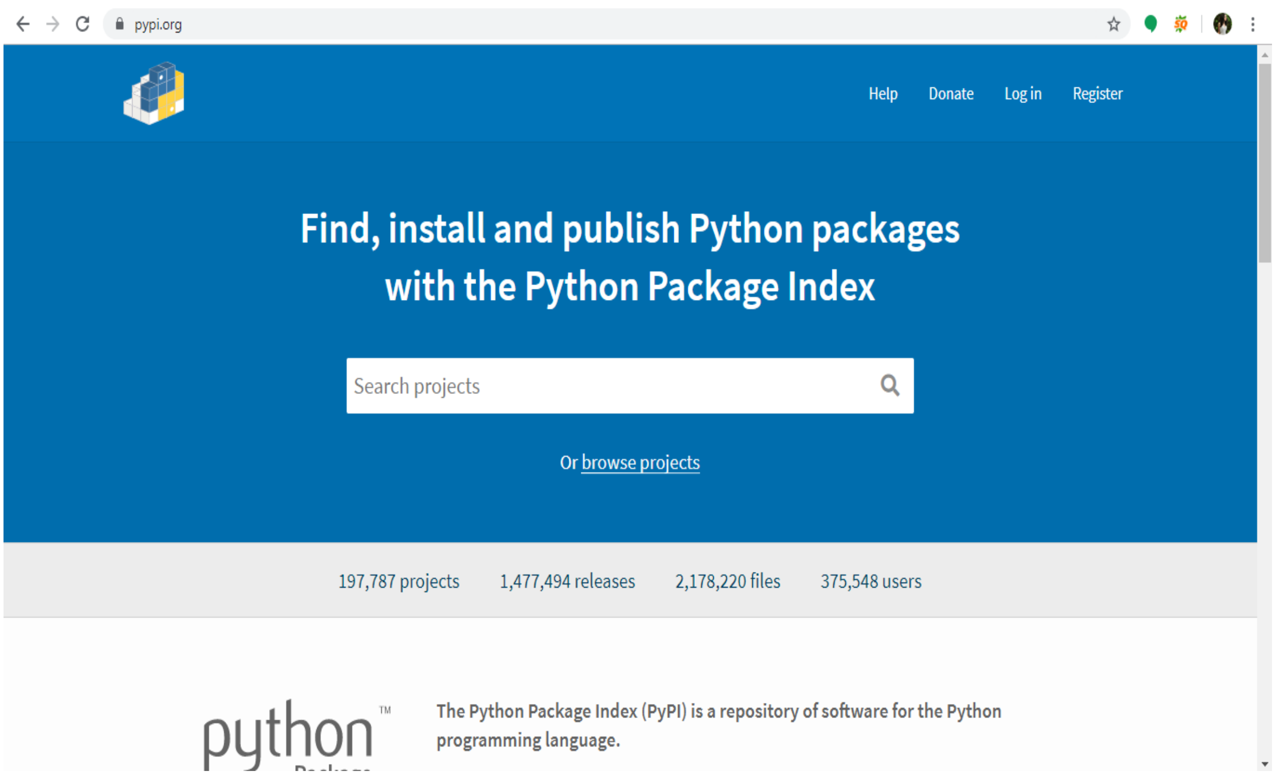
Q.61. How do you create your own package in Python?

We know that a package may contain sub-packages and modules. A module is nothing but Python code.

To create a Python package of our own, we create a directory and create a file __init__.py in it. We leave it empty. Then, in that package, we create a module(s) with whatever code we want.

Q.62. You mentioned PyPI in your previous answer. Can you elaborate?

Sure. PyPI is the Python Package Index. This is a repository of software for Python. It has a large collection of packages and their binaries for a wide range of uses. Here’s a hint of what it looks like-



Find, install and publish Python packages with the Python Package Index

197,787 projects 1,477,494 releases 2,178,220 files 375,548 users

The Python Package Index (PyPI) is a repository of software for the Python programming language.

Q.63. What will the following code output?

```
>>> word='abcdefghij'
>>> word[:3]+word[3:]
```

The output is 'abcdefghij'. The first slice gives us 'abc', the next gives us 'defghij'.

Q.64. Have you heard of the yield keyword in Python?

Yes, I have. This keyword bears the ability to turn any function into a generator. Much like the standard return keyword, but returns a generatorobject. It is also true that one function may observe multiple yields.

```
>>> def odds(n):
odd=[i for i in range(n+1) if i%2!=0]
for i in odd:
yield i
>>> for i in odds(8):
print(i)
```

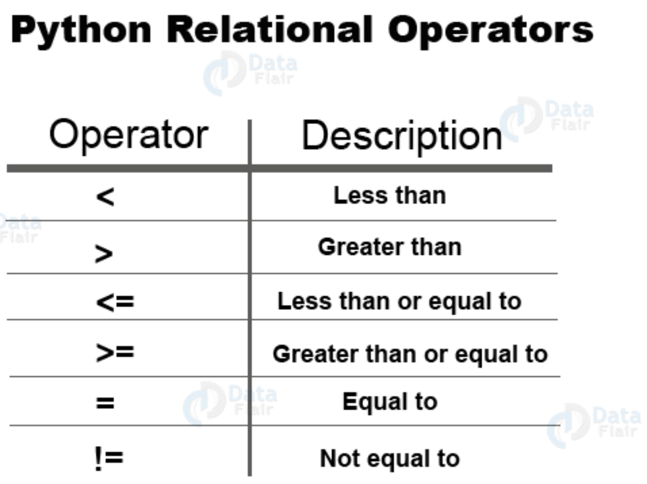
1

3

5

7

Q65. **What do you know about relational operators in Python.**


Python Relational Operators

Operator	Description
<	Less than
>	Greater than
<=	Less than or equal to
>=	Greater than or equal to
=	Equal to
!=	Not equal to

Relational operators compare values.
Less than (<) If the value on the left is lesser, it returns True.

```
>>> 'hi'<'Hi'
False
```

Greater than (>) If the value on the left is greater, it returns True.

```
>>> 1.1+2.2>3.3
True
```

This is because of the flawed floating-point arithmetic in Python, due to hardware dependencies.

Less than or equal to (<=) If the value on the left is lesser than or equal to, it returns True.

```
>>> 3.0<=3
True
```

Greater than or equal to (>=) If the value on the left is greater than or equal to, it returns True.

```
>>> True>=False
True
```

Equal to (==) If the two values are equal, it returns True.

```
>>> {1,3,2,2}=={1,2,3}
True
```

Not equal to (!=) If the two values are unequal, it returns True.

```
>>> True!=0.1
True
```

```
>>> False!=0.1
True
```

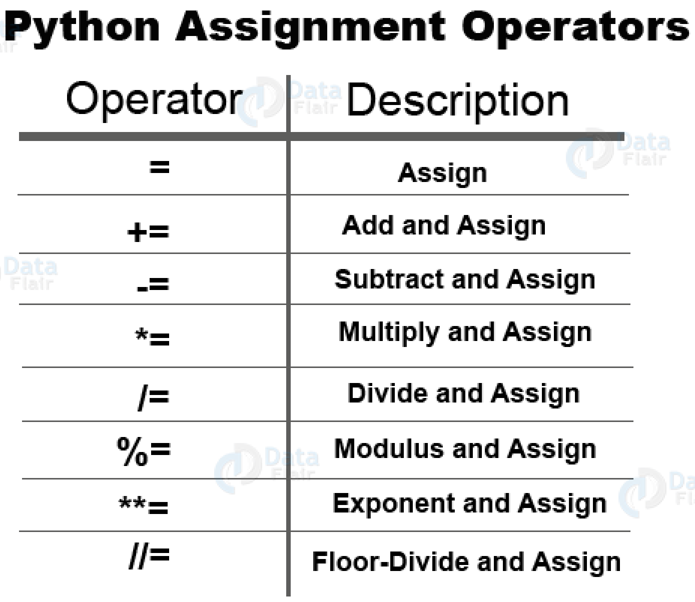
Q66. **Write a program in Python to produce Star triangle.**

```
def pyfunc(r):
for x in range(r):
print(' '*(r-x-1)+'*'*(2*x+1))
pyfunc(9)
```

Output:

```
*
***
*****
*******
*********
***********
*************
***************
*****************
```

Q67. **What are assignment operators in Python?**



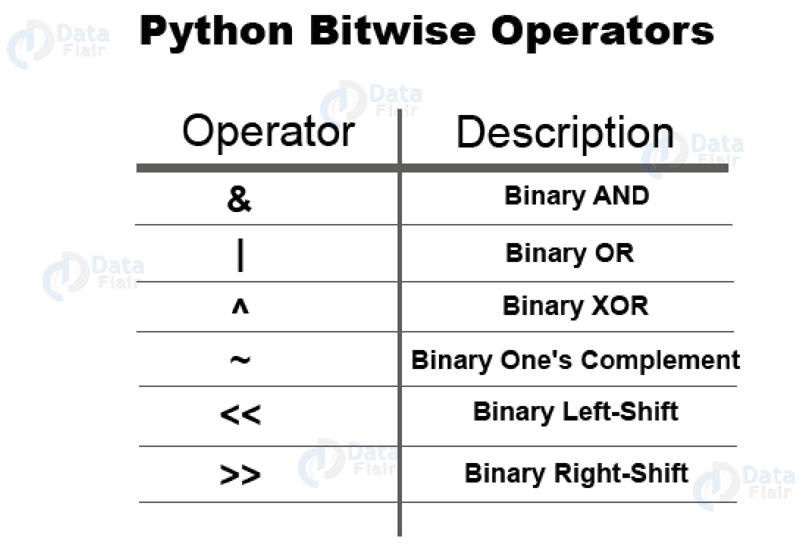
Python Assignment Operators

Operator	Description
=	Assign
+=	Add and Assign
-=	Subtract and Assign
*=	Multiply and Assign
/=	Divide and Assign
%=	Modulus and Assign
**=	Exponent and Assign
//=	Floor-Divide and Assign

We can combine all arithmetic operators with the assignment symbol.

```
>>> a=7
>>> a+=1
>>> a
8
>>> a-=1
>>> a
7
>>> a*=2
>>> a
14
>>> a/=2
>>> a
7.0
>>> a**=2
>>> a
49.0
>>> a//=3
>>> a
16.0
>>> a%=4
>>> a
0.0
```

Q68. Finally, tell us about bitwise operators in Python.



Python Bitwise Operators

Operator	Description
&	Binary AND
|	Binary OR
^	Binary XOR
~	Binary One's Complement
<<	Binary Left-Shift
>>	Binary Right-Shift

These operate on values bit by bit.

AND (&) This performs & on each bit pair.

```
>>> 0b110 & 0b010
```

```
2
```

OR (|) This performs | on each bit pair.

```
>>> 3|2
3
```

XOR (^) This performs an exclusive-OR operation on each bit pair.

```
>>> 3^2
1
```

Binary One’s **Complement (~)** This returns the one’s complement of a value.

```
>>> ~2
-3
```

Binary Left-Shift (<<) This shifts the bits to the left by the specified amount.

```
>>> 1<<2
4
```

Here, 001 was shifted to the left by two places to get 100, which is binary for 4.

Binary Right-Shift (>>)

```
>>> 4>>2
1
```

Q69. Write a program in Python to check if a sequence is a Palindrome.

```
a=input("enter sequence")
b=a[::-1]
if a==b:
&nbsp; print("palindrome")
else:
&nbsp; print("Not a Palindrome")
```

Output:

```
enter sequence 323 palindrome
```

Q70. Write a one-liner that will count the number of capital letters in a file. Your code should work even if the file is too big to fit in memory.
Ans: Let us first write a multiple line solution and then convert it to one-liner code.

```
with open(SOME_LARGE_FILE) as fh:
count = 0
text = fh.read()
for character in text:
if character.isupper():
count += 1
```

We will now try to transform this into a single line.

```
count sum(1 for line in fh for character in line if character.isupper())
```

Q71. Write a sorting algorithm for a numerical dataset in Python.
Ans: The following code can be used to sort a list in Python:

```
list = ["1", "4", "0", "6", "9"]
list = [int(i) for i in list]
list.sort()
print (list)
```

Q72. **What does the function zip() do?**
One of the less common functions with beginners, zip() returns an iterator of tuples.

>>> **list**(**zip**(['a','b','c'],[1,2,3]))
[('a', 1), ('b', 2), ('c', 3)]
Here, it pairs items from the two lists and creates tuples with those. But it doesn't have to be lists.

>>> **list**(**zip**(('a','b','c'),(1,2,3)))
[('a', 1), ('b', 2), ('c', 3)]

Looking at the below code, write down the final values of A0, A1, …An.

```
A0 = dict(zip(('a','b','c','d','e'),(1,2,3,4,5)))
A1 = range(10)A2 = sorted([i for i in A1 if i in A0])
```

```
A3 = sorted([A0[s] for s in A0])

A4 = [i for i in A1 if i in A3]

A5 = {i:i*i for i in A1}

A6 = [[i,i*i] for i in A1]

print(A0,A1,A2,A3,A4,A5,A6)
```

Ans: The following will be the final outputs of A0, A1, … A6

```
A0 = {'a': 1, 'c': 3, 'b': 2, 'e': 5, 'd': 4} # the order may vary
A1 = range(0, 10)
A2 = []
A3 = [1, 2, 3, 4, 5]
A4 = [1, 2, 3, 4, 5]
A5 = {0: 0, 1: 1, 2: 4, 3: 9, 4: 16, 5: 25, 6: 36, 7: 49, 8: 64, 9: 81}
A6 = [[0, 0], [1, 1], [2, 4], [3, 9], [4, 16], [5, 25], [6, 36], [7, 49], [8, 64], [9, 81]]
```

Q73. What is the maximum possible length of an identifier?

Identifiers can be of any length.

Q74. **Explain the //, %, and ** operators in Python.**
The // operator performs floor division. It will return the integer part of the result on division.

```
>>> 7//2
3
```

Normal division would return 3.5 here.

Similarly, ** performs exponentiation. a**b returns the value of a raised to the power b.

```
>>> 2**10
1024
```

Finally, % is for modulus. This gives us the value left after the highest achievable division.

```
>>> 13%7
6
>>> 3.5%1.5
0.5
```

Q75. **Explain different ways to create an empty NumPy array in Python.**

We'll talk about two methods to create NumPy array-

First method-

```
>>> import numpy
>>> numpy.array([])
```

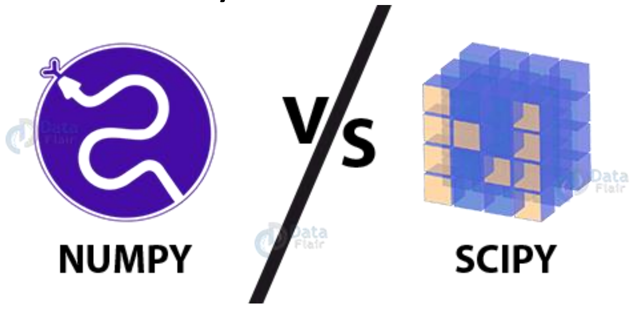
array([], dtype=float64)

Second method-

```
>>> numpy.empty(shape=(0,0))
```

array([], shape=(0, 0), dtype=float64)

Q76. **How is NumPy different from SciPy?**



We have so far seen them used together. But they have subtle differences:

SciPy encompasses most new features

NumPy does hold some linear algebra functions

SciPy holds more fully-featured versions of the linear algebra modules and other numerical algorithms

NumPy has compatibility as one of its essential goals; it attempts to retain all features supported by any of its predecessors

NumPy holds the array data type and some basic operations: indexing, sorting, reshaping, and more

Q77. **How would you create an empty NumPy array?**

To create an empty array with NumPy, we have two options:

Option 1

```
>>> import numpy
>>> numpy.array([])
```

array([], dtype=float64)

Option 2

```
>>> numpy.empty(shape=(0,0))
```

array([], shape=(0, 0), dtype=float64)

Q78. **If a function does not have a return statement, is it valid?**

Very conveniently. A function that doesn't return anything returns a None object. Not necessarily does the return keyword mark the end of a function; it merely ends it when present in the function. Normally, a block of code marks a function and where it ends, the function body ends.

Q79. What does an object() do?

Ans: It returns a featureless object that is a base for all classes. Also, it does not take any parameters.

Q80. How to create an empty class in Python?

Ans: An empty class is a class that does not have any code defined within its block. It can be created using the *pass* keyword. However, you can create objects of this class outside the class itself. IN PYTHON THE PASS command does nothing when its executed. it's a null statement.

For example-

```
class a:
&nbsp; pass
obj=a()
obj.name="xyz"
print("Name = ",obj.name)
```

Output:

```
Name =  xyz
```

Q81. **Does python make use of access specifiers?**

Ans: Python does not deprive access to an instance variable or function. Python lays down the concept of prefixing the name of the variable, function or method with a single or double underscore to imitate the behavior of protected and private access specifiers.

Q82. How To Save An Image Locally Using Python Whose URL Address I Already Know?

Ans: We will use the following code to save an image locally from an URL address

```
import urllib.request
urllib.request.urlretrieve("URL", "local-filename.jpg")
```

Q83. How can you Get the Google cache age of any URL or web page?

Ans: Use the following URL format:

http://webcache.googleusercontent.com/search?q=cache:URLGOESHERE

Be sure to replace “URLGOESHERE” with the proper web address of the page or site whose cache you want to retrieve and see the time for. For example, to check the Google Webcache age of edureka.co you’d use the following URL:

http://webcache.googleusercontent.com/search?q=cache:edureka.co

Q84. You are required to scrap data from IMDb top 250 movies page. It should only have fields movie name, year, and rating.

Ans: We will use the following lines of code:

```
from bs4 import BeautifulSoup

import requests
import sys

url = 'http://www.imdb.com/chart/top'
response = requests.get(url)
soup = BeautifulSoup(response.text)
tr = soup.findChildren("tr")
tr = iter(tr)
next(tr)

for movie in tr:
title = movie.find('td', {'class': 'titleColumn'} ).find('a').contents[0]
year = movie.find('td', {'class': 'titleColumn'} ).find('span', {'class': 'secondaryInfo'}).contents[0]
rating = movie.find('td', {'class': 'ratingColumn imdbRating'} ).find('strong').contents[0]
row = title + ' - ' + year + ' ' + ' ' + rating

print(row)
```

The above code will help scrap data from IMDb’s top 250 list

Q85. **Can you explain the filter(), map(), and reduce() functions?**

Let's see these ***Python Functions***.

filter()- This function lets us keep the values that satisfy some conditional logic. Let's take an example.

```
>>> set(filter(lambda x:x>4, range(7)))
```

{5, 6}

This filters in the elements from 0 to 6 that are greater than the number 4.

map()- This function applies a function to each element in the iterable.

```
>>> set(map(lambda x:x**3, range(7)))
```

{0, 1, 64, 8, 216, 27, 125}

This calculates the cube for each element in the range 0 to 6 and stores them in a set.

reduce()- This function reduces a sequence pair-wise, repeatedly until we arrive at a single value.

```
>>> reduce(lambda x,y:y-x, [1,2,3,4,5])
```

3

Let's understand this:

2-1=1

3-1=2

4-2=2

5-2=3

Hence, 3.

Q86. Is python numpy better than lists?

Ans: We use python numpy array instead of a list because of the below three reasons:

- Less Memory
- Fast
- Convenient

Q87. How to get indices of N maximum values in a NumPy array?

Ans: We can get the indices of N maximum values in a NumPy array using the below code:

```
import numpy as np

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

print(arr.argsort()[-3:][::-1])
```

Output

```
[ 4 3 1 ]
```

Q88. How do you calculate percentiles with Python/ NumPy?

Ans: We can calculate percentiles with the following code

```
import numpy as np

a = np.array([1,2,3,4,5])

p = np.percentile(a, 50) #Returns 50th percentile, e.g. median

print(p)
```

Output

```
3
```

Q89. What is the difference between NumPy and SciPy?

Ans:

In an ideal world, NumPy would contain nothing but the array data type and the most basic operations: indexing, sorting, reshaping, basic element wise functions, et cetera.

All numerical code would reside in SciPy. However, one of NumPy’s important goals is compatibility, so NumPy tries to retain all features supported by either of its predecessors.

Thus NumPy contains some linear algebra functions, even though these more properly belong in SciPy. In any case, SciPy contains more fully-featured versions of the linear algebra modules, as well as many other numerical algorithms.

If you are doing scientific computing with python, you should probably install both NumPy and SciPy. Most new features belong in SciPy rather than NumPy.

Q90. How do you make 3D plots/visualizations using NumPy/SciPy?

Ans: Like 2D plotting, 3D graphics is beyond the scope of NumPy and SciPy, but just as in the 2D case, packages exist that integrate with NumPy. Matplotlib provides basic 3D plotting in the mplot3d subpackage, whereas Mayavi provides a wide range of high-quality 3D visualization features, utilizing the powerful VTK engine.

Q91. Which of the following statements create a dictionary? (Multiple Correct Answers Possible)

b) d = {“john”:40, “peter”:45}

c) d = {40:”john”, 45:”peter”}

d) d = (40:”john”, 45:”50”)

Answer: b, c & d.

Dictionaries are created by specifying keys and values.

Q92. **Optionally, what statements can you put under a try-except block?**

We have two of those:

else- To run a piece of code when the try-block doesn’t create an exception.

finally- To execute some piece of code regardless of whether there is an exception.

```
>>> try:
print("Hello")
except:
print("Sorry")
else:
print("Oh then")
finally:
print("Bye")
```

Hello

Ohthen

Bye

Q93. **How long can an identifier be in Python?**

According to the official Python documentation, an identifier can be of any length. However, PEP 8 suggests that you should limit all lines to a maximum of 79 characters. Also, PEP 20 says ‘readability counts’. So, a very long identifier will violate PEP-8 and PEP-20.

Apart from that, there are certain rules we must follow to name one:

It can only begin with an underscore or a character from A-Z or a-z.
The rest of it can contain anything from the following: A-Z/a-z/_/0-9.
Python is case-sensitive, as we discussed in the previous question.
Keywords cannot be used as identifiers. Python has the following keywords:

And	Def	False	Import	Not	True
As	Del	finally	In	Or	try
assert	Elif	for	Is	Pass	while
break	Else	from	Lambda	Print	with
class	Except	global	None	Raise	yield
continue	Exec	If	nonlocal	return	

Q94. Why are local variable names beginning with an underscore discouraged?

Answer: **they are used to indicate a private variable of a class**
As Python has no concept of private variables, leading underscores are used to indicate variables that must not be accessed from outside the class.

Q95. Which of the following is an invalid statement?

a)abc = 1,000,000
b) a b c = 1000 2000 3000
c) a,b,c = 1000, 2000, 3000
d) a_b_c = 1,000,000

Answer: b) a b c = 1000 2000 3000

Spaces are not allowed in variable names.

Q96. What is the output of the following?

```
try:
if '1' != 1:
raise "someError"
else:
print("someError has not occured")
except "someError":
print ("someError has occured")
```

Answer: invalid code

A new exception class must inherit from a BaseException. There is no such inheritance here.

Q97. Suppose list1 is [2, 33, 222, 14, 25], What is list1[-1] ?
Answer: 25

The index -1 corresponds to the last index in the list.

Q98. To open a file c:scores.txt for writing, we use

outfile = open(“c:scores.txt”, “w”)

The location contains double slashes ( ) and w is used to indicate that file is being written to.

Q99. What is the output of the following?

```
f = None

for i in range (5):
with open("data.txt", "w") as f:
if i > 2:
break

print f.closed
```

The WITH statement when used with open file guarantees that the file object is closed when the with block exits.

Q100. When will the else part of try-except-else be executed?

The else part is executed when no exception occurs.