**Python**

**Q1. What is the difference between list and tuples?**

|  |  |
| --- | --- |
| **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) |

**Q2. What are the key features of Python?**

* Python is an **interpreted** language. 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), the justification for this point is given as “we are all adults here”
* In Python, **functions** are**first-class objects**. This means that they can be assigned to variables, returned from other functions and passed into functions. Classes are also first class objects
* **Writing Python code is quick** but running it is often slower than compiled languages. Fortunately Python allows the inclusion of C based extensions so bottlenecks can be optimized away and often are. The numpy package is a good example of this, it’s really quite quick because a lot of the number crunching it does isn’t actually done by Python
* Python finds **use in many spheres** – web applications, automation, scientific modelling, big data applications and many more. It’s also often used as “glue” code to get other languages and components to play nice.

**Q3. 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.

**Q4. How is Multithreading achieved in Python?**

Ans:

1. Python has a multi-threading package but if you want to multi-thread to speed your code up.
2. Python has a construct called the Global Interpreter Lock (GIL). The GIL makes sure that only one of your ‘threads’ can execute at any one time. A thread acquires the GIL, does a little work, then passes the GIL onto the next thread.
3. This happens very quickly so to the human eye it may seem like your threads are executing in parallel, but they are really just taking turns using the same CPU core.
4. All this GIL passing adds overhead to execution. This means that if you want to make your code run faster than using the threading package often isn’t a good idea.

**Q5. How can the ternary operators be used in python?**

**Ans:** The Ternary operator is the operator that is used to show the conditional statements. This consists of the true or false values with a statement that has to be evaluated for it.

Syntax: The Ternary operator will be given as:

**[on\_true] if [expression] else [on\_false]x, y = 25, 50big = x if x < y else y**

Example: The expression gets evaluated like if x<y else y, in this case if x<y is true then the value is returned as big=x and if it is incorrect then big=y will be sent as a result.

**Q6. How is memory managed in Python?**

**Ans:**

1. Python memory is managed by Python private heap space. All Python objects and data structures are located in a private heap. The programmer does not have an access to this private heap and interpreter takes care of this Python private heap.
2. The allocation of Python heap space for Python objects is done by Python memory manager. The core API gives access to some tools for the programmer to code.
3. Python also have an inbuilt garbage collector, which recycle all the unused memory and frees the memory and makes it available to the heap space.

**Q7. 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:

1. Single Inheritance – where a derived class acquires the members of a single super class.
2. Multi-level inheritance – a derived class d1 in inherited from base class base1, and d2 is inherited from base2.
3. Hierarchical inheritance – from one base class you can inherit any number of child classes
4. Multiple inheritance – a derived class is inherited from more than one base class.

**Q8. Explain what Flask is and its benefits?**

**Ans:** Flask is a web micro framework for Python based on “Werkzeug, Jinja2 and good intentions” BSD license. Werkzeug and Jinja2 are two of its dependencies. This means it will have little to no dependencies on external libraries.  It makes the framework light while there is little dependency to update and less security bugs.

A session basically allows you to remember information from one request to another. In a flask, a session uses a signed cookie so the user can look at the session contents and modify. The user can modify the session if only it has the secret key Flask.secret\_key.

**Q9. What is the usage of help() and dir() function in Python?**

Ans: Help() and dir() both functions are accessible from the Python interpreter and used for viewing a consolidated dump of built-in functions.

1. Help() function: The help() function is used to display the documentation string and also facilitates you to see the help related to modules, keywords, attributes, etc.
2. Dir() function: The dir() function is used to display the defined symbols.

**Q10. Whenever Python exits, why isn’t all the memory de-allocated?**

**Ans:**

1. Whenever Python exits, especially those Python modules which are having circular references to other objects or the objects that are referenced from the global namespaces are not always de-allocated or freed.
2. It is impossible to de-allocate those portions of memory that are reserved by the C library.
3. On exit, because of having its own efficient clean up mechanism, Python would try to de-allocate/destroy every other object.

**Q11. What is dictionary in Python?**

Ans: The built-in datatypes in Python is called dictionary. It defines one-to-one relationship between keys and values. Dictionaries contain pair of keys and their corresponding values. Dictionaries are indexed by keys.

Let’s take an example: The following example contains some keys. Country, Capital & PM. Their corresponding values are India, Delhi and Modi respectively.

>>> dict={'Country':'India','Capital':'Delhi','PM':'Modi'}

>>> print dict[Country]

India

>>> print dict[Capital]

Delhi

>>> print dict[PM]

Modi

**Q12. What is monkey patching in Python?**

**Ans:** In Python, the term monkey patch only refers to dynamic modifications of a class or module at run-time.

Consider the below example:

# m.py

class MyClass:

def f(self):

print "f()"

We can then run the monkey-patch testing like this:

import m

def monkey\_f(self):

print "monkey\_f()"

m.MyClass.f = monkey\_f

obj = m.MyClass()

obj.f()

The output will be: monkey\_f()

As we can see, we did make some changes in the behavior of*f()* in *MyClass*using the function we defined, *monkey\_f()*, outside of the module *m*.

**Q13. What does this mean: \*args, \*\*kwargs? And why would we use it?**

Ans: We use \*args when we aren’t sure how many arguments are going to be passed to a function, or if we want to pass a stored list or tuple of arguments to a function. \*\*kwargsis used when we don’t know how many keyword arguments will be passed to a function, or it can be used to pass the values of a dictionary as keyword arguments. The identifiers args and kwargs are a convention, you could also use \*bob and \*\*billy but that would not be wise.

**Q14. 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())

**Q15. What are negative indexes and why are they used?**

Ans: The sequences in Python are indexed and it consists of the positive as well as negative numbers. The numbers that are positive uses ‘0’ that is uses as first index and ‘1’ as the second index and the process goes on like that.

The index for the negative number starts from ‘-1’ that represents the last index in the sequence and ‘-2’ as the penultimate index and the sequence carries forward like the positive number.

The negative index is used to remove any new-line spaces from the string and allow the string to except the last character that is given as S[:-1]. The negative index is also used to show the index to represent the string in correct order.

**Q16. 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']

**Q17. What is the process of compilation and linking in python?**

**Ans:** The compiling and linking allows the new extensions to be compiled properly without any error and the linking can be done only when it passes the compiled procedure. If the dynamic loading is used then it depends on the style that is being provided with the system. The python interpreter can be used to provide the dynamic loading of the configuration setup files and will rebuild the interpreter.

The steps that is required in this as:

1. Create a file with any name and in any language that is supported by the compiler of your system. For example file.c or file.cpp
2. Place this file in the Modules/ directory of the distribution which is getting used.
3. Add a line in the file Setup.local that is present in the Modules/ directory.
4. Run the file using spam file.o
5. After successful run of this rebuild the interpreter by using the make command on the top-level directory.
6. If the file is changed then run rebuildMakefile by using the command as ‘make Makefile’.

**Q18. 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)

**Q19. 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]]

**Q20. Explain split(), sub(), subn() methods of “re” module in Python.**

**Ans:** To modify the strings, Python’s “re” module is providing 3 methods. They are:

* split() – uses a regex pattern to “split” a given string into a list.
* sub() – finds all substrings where the regex pattern matches and then replace them with a different string
* subn() – it is similar to sub() and also returns the new string along with the no. of replacements.

**Q21. How can you generate random numbers in Python?**

**Ans:** Random module is the standard module that is used to generate the random number. The method is defined as:

|  |  |
| --- | --- |
| 1  2 | import random  random.random |

The statement random.random() method return the floating point number that is in the range of [0, 1). The function generates the random float numbers. The methods that are used with the random class are the bound methods of the hidden instances. The instances of the Random can be done to show the multi-threading programs that creates different instance of individual threads. The other random generators that are used in this are:

1. randrange(a, b): it chooses an integer and define the range in-between [a, b). It returns the elements by selecting it randomly from the range that is specified. It doesn’t build a range object.
2. uniform(a, b): it chooses a floating point number that is defined in the range of [a,b).Iyt returns the floating point number
3. normalvariate(mean, sdev): it is used for the normal distribution where the mu is a mean and the sdev is a sigma that is used for standard deviation.
4. The Random class that is used and instantiated creates an independent multiple random number generators.

**Q22. What is the difference between range & xrange?**

**Ans:** For the most part, xrange and range are the exact same in terms of functionality. They both provide a way to generate a list of integers for you to use, however you please. The only difference is that range returns a Python list object and x range returns an xrange object.

This means that xrange doesn’t actually generate a static list at run-time like range does. It creates the values as you need them with a special technique called yielding. This technique is used with a type of object known as generators. That means that if you have a really gigantic range you’d like to generate a list for, say one billion, xrange is the function to use.

This is especially true if you have a really memory sensitive system such as a cell phone that you are working with, as range will use as much memory as it can to create your array of integers, which can result in a Memory Error and crash your program. It’s a memory hungry beast.

**Q23. What is pickling and unpickling?**

Ans: Pickle module accepts any Python object and converts it into a string representation and dumps it into a file by using dump function, this process is called pickling. While the process of retrieving original Python objects from the stored string representation is called unpickling.

## ****Django – Python Interview Questions****