1. **What is Python? What are the benefits of using Python?**

Ans:- 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.

**2) What is PEP 8?**

PEP 8 is a coding convention, a set of recommendation, about how to write your Python code more readable.

**3) What is pickling and unpickling?**

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.

**4) How Python is interpreted?**

Python language 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.

**5) How memory is managed in Python?**

* 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.
* 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.
* 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.

**6) What are the tools that help to find bugs or perform static analysis?**

PyChecker is a static analysis tool that detects the bugs in Python source code and warns about the style and complexity of the bug. Pylint is another tool that verifies whether the module meets the coding standard.

**7) What are Python decorators?**

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

A python decoder are the function that are the use of adds some extra information to another function or type.

It is also used for meta programming

**8) What is the difference between list and tuple?**

The difference between list and tuple is that list is mutable while tuple is not. Tuple can be hashed for e.g as a key for dictionaries.

**9) How are arguments passed by value or by reference?**

Everything in Python is an object and all variables hold references to the objects. The references values are according to the functions; as a result you cannot change the value of the references. However, you can change the objects if it is mutable.

**10) What is Dict and List comprehensions are?**

They are syntax constructions to ease the creation of a Dictionary or List based on existing iterable.

**11) What are the built-in type does python provides?**

There are mutable and Immutable types of Pythons built in types Mutable built-in types

* List
* Sets
* Dictionaries

Immutable built-in types

* Strings
* Tuples
* Numbers

**12) What is namespace in Python?**

In Python, every name introduced has a place where it lives and can be hooked for. This is known as namespace. It is like a box where a variable name is mapped to the object placed. Whenever the variable is searched out, this box will be searched, to get corresponding object.

**13) What is lambda in Python?**

It is a single expression anonymous function often used as inline function.

**14) Why lambda forms in python does not have statements?**

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

**15) What is pass in Python?**

Pass means, no-operation Python statement, or in other words it is a place holder in compound statement, where there should be a blank left and nothing has to be written there.

**16) In Python what are iterators?**

In Python, iterators are used to iterate a group of elements, containers like list.

**17) What is unittest in Python?**

A unit testing framework in Python is known as unittest. It supports sharing of setups, automation testing, shutdown code for tests, aggregation of tests into collections etc.

**18) In Python what is slicing?**

A mechanism to select a range of items from sequence types like list, tuple, strings etc. is known as slicing.

**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.

**20) What is docstring in Python?**

A Python documentation string is known as docstring, it is a way of documenting Python functions, modules and classes.

**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.

**22) What is negative index in Python?**

Python sequences can be index in positive and negative numbers. For positive index, 0 is the first index, 1 is the second index and so forth. For negative index, (-1) is the last index and (-2) is the second last index and so forth.

**23) How you can convert a number to a string?**

In order to convert a number into a string, use the inbuilt function str(). If you want a octal or hexadecimal representation, use the inbuilt function oct() or hex().

**24) What is the difference between Xrange and range?**

Xrange returns the xrange object while range returns the list, and uses the same memory and no matter what the range size is.

**25) What is module and package in Python?**

In Python, module is the way to structure program. Each Python program file is a module, which imports other modules like objects and attributes.

The folder of Python program is a package of modules. A package can have modules or subfolders.

**26) Mention what are the rules for local and global variables in Python?**

**Local variables**: If a variable is assigned a new value anywhere within the function's body, it's assumed to be local.

**Global variables**: Those variables that are only referenced inside a function are implicitly global.

**27) How can you share global variables across modules?**

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.

**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)

**29) Explain how to delete a file in Python?**

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

**30) Explain how can you generate random numbers in Python?**

To generate random numbers in Python, you need to import command as

import random

random.random()

This returns a random floating point number in the range [0,1)

**31) Explain how can you access a module written in Python from C?**

You can access a module written in Python from C by following method,

Module = =PyImport\_ImportModule("<modulename>");

**32) Mention the use of // operator in Python?**

It is a Floor Divisionoperator , which is used for dividing two operands with the result as quotient showing only digits before the decimal point. For instance, 10//5 = 2 and 10.0//5.0 = 2.0.

**33) Mention five benefits of using Python?**

* Python comprises of a huge standard library for most Internet platforms like Email, HTML, etc.
* Python does not require explicit memory management as the interpreter itself allocates the memory to new variables and free them automatically
* Provide easy readability due to use of square brackets
* Easy-to-learn for beginners
* Having the built-in data types saves programming time and effort from declaring variables

**34) Mention the use of the split function** **in Python**?

The use of the split function in Python is that it breaks a string into shorter strings using the defined separator. It gives a list of all words present in the string.

**35) Explain what is Flask & its benefits**?

Flask is a web micro framework for Python based on "Werkzeug, Jinja 2 and good intentions" BSD licensed. Werkzeug and jingja are two of its dependencies.

Flask is part of the micro-framework. Which 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.

**36) Mention what is the difference between Django, Pyramid, and Flask?**

Flask is a "microframework" primarily build for a small application with simpler requirements. In flask, you have to use external libraries. Flask is ready to use.

Pyramid are build for larger applications. It provides flexibility and lets the developer use the right tools for their project. The developer can choose the database, URL structure, templating style and more. Pyramid is heavy configurable.

Like Pyramid, Django can also used for larger applications. It includes an ORM.

**37) Mention what is Flask-WTF and what are their features?**

Flask-WTF offers simple integration with WTForms. Features include for Flask WTF are

* Integration with wtforms
* Secure form with csrf token
* Global csrf protection
* Internationalization integration
* Recaptcha supporting
* File upload that works with Flask Uploads

**38) Explain what is the common way for the Flask script to work?**

The common way for the flask script to work is

* Either it should be the import path for your application
* Or the path to a Python file

**39) Explain how you can access sessions in Flask?**

A session basically allows you to remember information from one request to another. In a flask, it 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.

**40) Is Flask an MVC model and if yes give an example showing MVC pattern for your application?**

Basically, Flask is a minimalistic framework which behaves same as MVC framework. So MVC is a perfect fit for Flask, and the pattern for MVC we will consider for the following example

|  |  |
| --- | --- |
| from flask import Flask  app = Flask(\_name\_)  @app.route("/")  Def hello():  return "Hello World"  app.run(debug = True) | In this code your,   * Configuration part will be   from flask import Flask  app = Flask(\_name\_)   * View part will be   @app.route("/")  Def hello():  return "Hello World"   * While you model or main part will be   app.run(debug = True) |

**41) Explain database connection in Python Flask?**

Flask supports database powered application (RDBS). Such system requires creating a schema, which requires piping the shema.sql file into a sqlite3 command. So you need to install sqlite3 command in order to create or initiate the database in Flask.

Flask allows to request database in three ways

* before\_request() : They are called before a request and pass no arguments
* after\_request() : They are called after a request and pass the response that will be sent to the client
* teardown\_request(): They are called in situation when exception is raised, and response are not guaranteed. They are called after the response been constructed. They are not allowed to modify the request, and their values are ignored.

**42) You are having multiple Memcache servers running Python, in which one of the memcacher server fails, and it has your data, will it ever try to get key data from that one failed server?**

The data in the failed server won't get removed, but there is a provision for auto-failure, which you can configure for multiple nodes. Fail-over can be triggered during any kind of socket or Memcached server level errors and not during normal client errors like adding an existing key, etc.

**43) Explain how you can minimize the Memcached server outages in your Python Development?**

* When one instance fails, several of them goes down, this will put larger load on the database server when lost data is reloaded as client make a request. To avoid this, if your code has been written to minimize cache stampedes then it will leave a minimal impact
* Another way is to bring up an instance of Memcached on a new machine using the lost machines IP address
* Code is another option to minimize server outages as it gives you the liberty to change the Memcached server list with minimal work
* Setting timeout value is another option that some Memcached clients implement for Memcached server outage. When your Memcached server goes down, the client will keep trying to send a request till the time-out limit is reached

**44) Explain what is Dogpile effect? How can you prevent this effect?**

Dogpile effect is referred to the event when cache expires, and websites are hit by the multiple requests made by the client at the same time. This effect can be prevented by using semaphore lock. In this system when value expires, first process acquires the lock and starts generating new value.

**45) Explain how Memcached should not be used in your Python project?**

* Memcached common misuse is to use it as a data store, and not as a cache
* Never use Memcached as the only source of the information you need to run your application. Data should always be available through another source as well
* Memcached is just a key or value store and cannot perform query over the data or iterate over the contents to extract information
* Memcached does not offer any form of security either in encryption or authentication

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

|  |  |
| --- | --- |
| **LIST vs 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, then it’s usually not a good idea to use it.
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 then 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. 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.
2. 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.
3. 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. 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 are 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 microframework 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 a little dependency to update and fewer 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.

|  |  |
| --- | --- |
| 1 | dict={'Country':'India','Capital':'Delhi','PM':'Modi'} |
| 1 | print dict[Country] |

India

|  |  |
| --- | --- |
| 1 | print dict[Capital] |

Delhi

|  |  |
| --- | --- |
| 1 | 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:

|  |  |
| --- | --- |
| 1  2  3  4 | # m.py  class MyClass:  def f(self):  print "f()" |

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

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | import m  def monkey\_f(self):  print "monkey\_f()"    m.MyClass.f = monkey\_f  obj = m.MyClass()  obj.f() |

The output will be as below:

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.

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | 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.

|  |  |
| --- | --- |
| 1 | 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.

[[](https://www.edureka.co/python-programming-certification-training)](https://www.edureka.co/python-programming-certification-training" \t "_blank)

**[Python Programming Certification Course](https://www.edureka.co/python-programming-certification-training" \t "_blank)**

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:

|  |  |
| --- | --- |
| 1  2  3  4 | 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:

|  |  |
| --- | --- |
| 1  2  3  4 | 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.**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | 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**

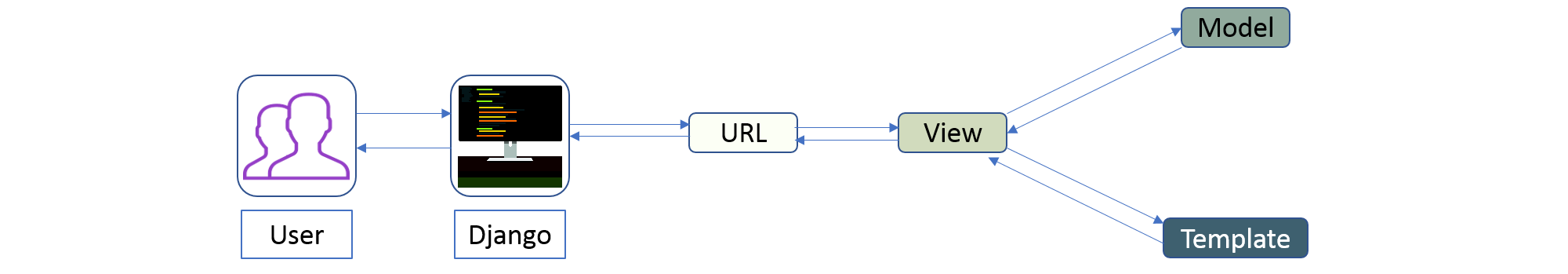
**Q24. Mention the differences between Django, Pyramid and Flask.**

**Ans:**

* Flask is a “microframework” primarily build for a small application with simpler requirements. In flask, you have to use external libraries. Flask is ready to use.
* Pyramid is built for larger applications. It provides flexibility and lets the developer use the right tools for their project. The developer can choose the database, URL structure, templating style and more. Pyramid is heavy configurable.
* Django can also used for larger applications just like Pyramid. It includes an ORM.

**Q25. Discuss the Django architecture.**

**Ans:** Django MVT Pattern:

**Figure:** *Python Interview Questions – Django Architecture*

The developer provides the Model, the view and the template then just maps it to a URL and Django does the magic to serve it to the user.

**Q26. Explain how you can set up the Database in Django.**

**Ans:** You can use the command edit mysite/setting.py , it is a normal python module with module level representing Django settings.

Django uses SQLite by default; it is easy for Django users as such it won’t require any other type of installation. In the case your database choice is different that you have to the following keys in the DATABASE ‘default’ item to match your database connection settings.

* **Engines**: you can change database by using ‘django.db.backends.sqlite3’ , ‘django.db.backeneds.mysql’, ‘django.db.backends.postgresql\_psycopg2’, ‘django.db.backends.oracle’ and so on
* **Name**: The name of your database. In the case if you are using SQLite as your database, in that case database will be a file on your computer, Name should be a full absolute path, including file name of that file.
* If you are not choosing SQLite as your database then settings like Password, Host, User, etc. must be added.

Django uses SQLite as default database, it stores data as a single file in the filesystem. If you do have a database server—PostgreSQL, MySQL, Oracle, MSSQL—and want to use it rather than SQLite, then use your database’s administration tools to create a new database for your Django project. Either way, with your (empty) database in place, all that remains is to tell Django how to use it. This is where your project’s settings.py file comes in.

We will add the following lines of code to the *setting.py* file:

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | DATABASES = {       'default': {            'ENGINE' : 'django.db.backends.sqlite3',            'NAME' : os.path.join(BASE\_DIR, 'db.sqlite3'),       }  } |

**Q27. Give an example how you can write a VIEW in Django?**

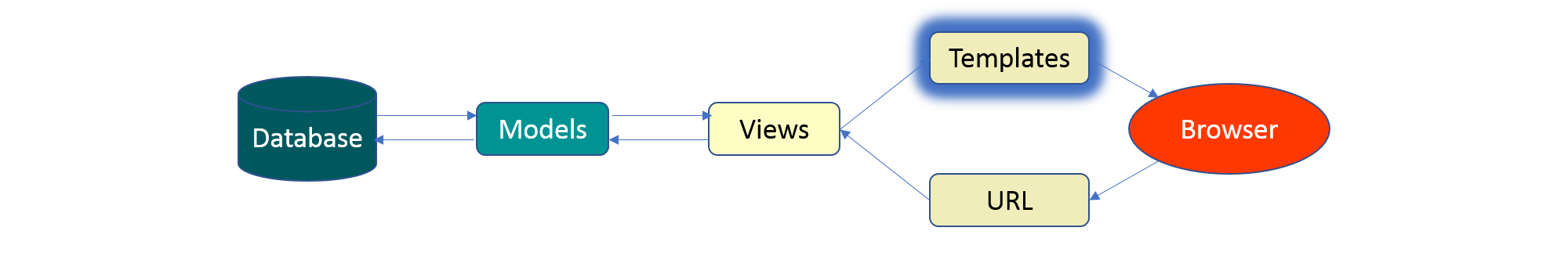
**Ans:** This is how we can use write a view in Django:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | from django.http import HttpResponse  import datetime    def Current\_datetime(request):       now = datetime.datetime.now()       html = "<html><body>It is now %s</body></html>" % now       return HttpResponse(html) |

*Returns the current date and time, as an HTML document*

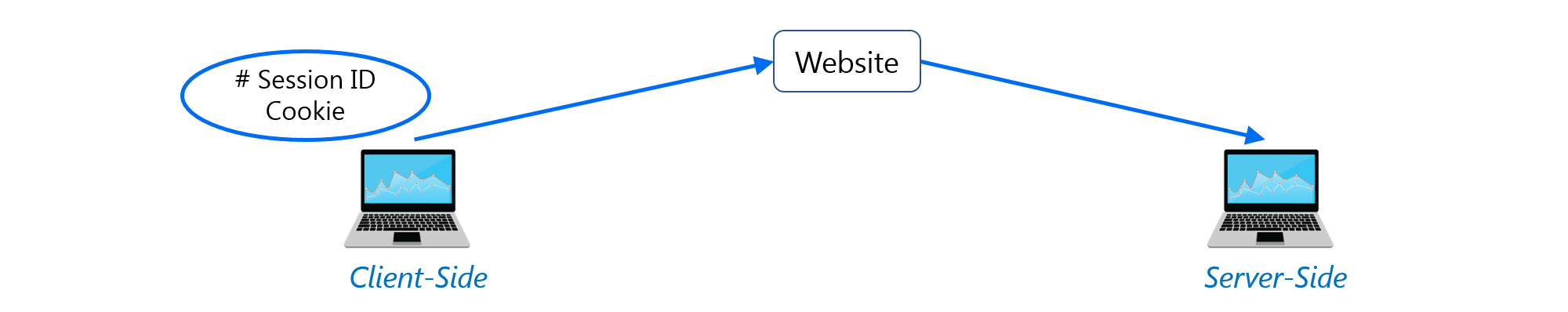
**Q28. Mention what the Django templates consists of.**

**Ans:** The template is a simple text file.  It can create any text-based format like XML, CSV, HTML, etc.  A template contains variables that get replaced with values when the template is evaluated and tags (% tag %) that controls the logic of the template.

**Figure:***Python Interview Questions – Django Template*

**Q29. Explain the use of session in Django framework?**

**Ans:** Django provides session that lets you store and retrieve data on a per-site-visitor basis. Django abstracts the process of sending and receiving cookies, by placing a session ID cookie on the client side, and storing all the related data on the server side.

**Figure:** *Python Interview Questions – Django Framework*

So the data itself is not stored client side. This is nice from a security perspective.

[[](https://www.edureka.co/python-programming-certification-training)](https://www.edureka.co/python-programming-certification-training" \t "_blank)

**[Python Programming Certification Course](https://www.edureka.co/python-programming-certification-training" \t "_blank)**

**Q30. List out the inheritance styles in Django.**

**Ans:** In Django, there is three possible inheritance styles:

1. Abstract Base Classes: This style is used when you only wants parent’s class to hold information that you don’t want to type out for each child model.
2. Multi-table Inheritance: This style is used If you are sub-classing an existing model and need each model to have its own database table.
3. Proxy models: You can use this model, If you only want to modify the Python level behavior of the model, without changing the model’s fields.

**Web Scraping – Python Interview Questions**

**Q31. 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

|  |  |
| --- | --- |
| 1  2 | import urllib.request  urllib.request.urlretrieve("URL", "local-filename.jpg") |

**Q32. 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

**Q33. 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:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19 | 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

**Data Analysis – Python Interview Questions**

**Q34. What is map function in Python?**

**Ans:** *map* function executes the function given as the first argument on all the elements of the iterable given as the second argument. If the function given takes in more than 1 arguments, then many iterables are given. #Follow the link to know more similar functions.

**Q35. 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:

|  |  |
| --- | --- |
| 1  2  3 | import numpy as np  arr = np.array([1, 3, 2, 4, 5])  print(arr.argsort()[-3:][::-1]) |

Output

[ 4 3 1 ]

**Q36. How do you calculate percentiles with Python/ NumPy?**

**Ans:** We can calculate percentiles with the following code

|  |  |
| --- | --- |
| 1  2  3  4 | 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

**Q37. What advantages do NumPy arrays offer over (nested) Python lists?**

**Ans:**

1. Python’s lists are efficient general-purpose containers. They support (fairly) efficient insertion, deletion, appending, and concatenation, and Python’s list comprehensions make them easy to construct and manipulate.
2. They have certain limitations: they don’t support “vectorized” operations like elementwise addition and multiplication, and the fact that they can contain objects of differing types mean that Python must store type information for every element, and must execute type dispatching code when operating on each element.
3. NumPy is not just more efficient; it is also more convenient. You get a lot of vector and matrix operations for free, which sometimes allow one to avoid unnecessary work. And they are also efficiently implemented.
4. NumPy array is faster and You get a lot built in with NumPy, FFTs, convolutions, fast searching, basic statistics, linear algebra, histograms, etc.

**Q38. Explain the use of decorators.**

**Ans:** Decorators in Python are used to modify or inject code in functions or classes. Using decorators, you can wrap a class or function method call so that a piece of code can be executed before or after the execution of the original code. Decorators can be used to check for permissions, modify or track the arguments passed to a method, logging the calls to a specific method, etc.

**Q39. What is the difference between NumPy and SciPy?**

**Ans:**

1. In an ideal world, NumPy would contain nothing but the array data type and the most basic operations: indexing, sorting, reshaping, basic elementwise functions, et cetera.
2. 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.
3. 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.
4. 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.

**Q40. 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.

**Multiple Choice Questions**

**Q41. Which of the following statements create a dictionary? (Multiple Correct Answers Possible)**

**a) d = {}  
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.

**Q42. Which one of these is floor division?**

**a) /  
b) //  
c) %  
d) None of the mentioned**

**Answer:** b) //

When both of the operands are integer then python chops out the fraction part and gives you the round off value, to get the accurate answer use floor division. For ex, 5/2 = 2.5 but both of the operands are integer so answer of this expression in python is 2. To get the 2.5 as the answer, use floor division using //. So, 5//2 = 2.5

**Q43. What is the maximum possible length of an identifier?**

**a) 31 characters  
b) 63 characters  
c) 79 characters  
d) None of the above**

**Answer:** d) None of the above

Identifiers can be of any length.

**Q44. Why are local variable names beginning with an underscore discouraged?**

**a) they are used to indicate a private variables of a class  
b) they confuse the interpreter  
c) they are used to indicate global variables  
d) they slow down execution**

**Answer:** a) they are used to indicate a private variables 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.

**Q45. 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.

**Q46. What is the output of the following?**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | try:      if '1' != 1:          raise "someError"      else:          print("someError has not occured")  except "someError":      print ("someError has occured") |

**a) someError has occured  
b) someError has not occured  
c) invalid code  
d) none of the above**

**Answer:** c) invalid code

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

**Q47. Suppose list1 is [2, 33, 222, 14, 25], What is list1[-1] ?**

**a) Error  
b) None  
c) 25  
d) 2**

**Answer:** c) 25

[[](https://www.edureka.co/python-programming-certification-training)](https://www.edureka.co/python-programming-certification-training" \t "_blank)

**[Python Programming Certification Course](https://www.edureka.co/python-programming-certification-training" \t "_blank)**

[Weekday / Weekend Batches](https://www.edureka.co/python-programming-certification-training" \t "_blank)

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

**Q48. To open a file c:\scores.txt for writing, we use**

**a) outfile = open(“c:\scores.txt”, “r”)  
b) outfile = open(“c:\\scores.txt”, “w”)  
c) outfile = open(file = “c:\scores.txt”, “r”)  
d) outfile = open(file = “c:\\scores.txt”, “o”)**

**Answer:** b) The location contains double slashes ( \\ ) and w is used to indicate that file is being written to.

**Q49. What is the output of the following?**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | f = None    for i in range (5):      with open("data.txt", "w") as f:          if i > 2:              break    print f.closed |

**a) True  
b) False  
c) None  
d) Error**

**Answer:** a) True

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

**Q50. When will the else part of try-except-else be executed?**

**a) always  
b) when an exception occurs  
c) when no exception occurs  
d) when an exception occurs in *to* except block**

**Answer:** c) when no exception occurs

The else part is executed when no exception occurs.

2. What is Python?

Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses English keywords frequently where as other languages use punctuation, and it has fewer syntactical constructions than other languages.

3. What is the purpose of PYTHONPATH environment variable?

PYTHONPATH − It has a role similar to PATH. This variable tells the Python interpreter where to locate the module files imported into a program. It should include the Python source library directory and the directories containing Python source code. PYTHONPATH is sometimes preset by the Python installer

4. What is the purpose of PYTHONSTARTUP,PYTHONCASEOK,PYTHONHOME,PYTHONSTARTUP environment variables?

PYTHONSTARTUP − It contains the path of an initialization file containing Python source code. It is executed every time you start the interpreter. It is named as .pythonrc.py in Unix and it contains commands that load utilities or modify PYTHONPATH

PYTHONCASEOK − It is used in Windows to instruct Python to find the first case-insensitive match in an import statement. Set this variable to any value to activate it.

PYTHONHOME − It is an alternative module search path. It is usually embedded in the PYTHONSTARTUP or PYTHONPATH directories to make switching module libraries easy.

5. What are the supported data types in Python?

Python has five standard data types −

* Numbers
* String
* List
* Tuple
* Dictionary

6. 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) |

7. How is memory managed in Python?

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.

8. Explain Inheritance in Python with an example.

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.

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

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.

10. What is dictionary in Python?

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]

11. 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.

Let us first write a multiple line solution and then convert it to one liner code.

1 with open(SOME\_LARGE\_FILE) as fh:

2 count = 0  
3 text = fh.read()  
4 for character in text:  
5 if character.isupper():  
6 count += 1

12. Write a sorting algorithm for a numerical dataset in Python.

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)

13. How will you reverse a list?

list.reverse() − Reverses objects of list in place.

14. How will you remove last object from a list?

list.pop(obj=list[-1]) − Removes and returns last object or obj from list.

15. What are negative indexes and why are they used?

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.

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

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.

17. What is the difference between range & xrange?

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.

18. What is pickling and unpickling?

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.

19. What is map function in Python?

*map* function executes the function given as the first argument on all the elements of the iterable given as the second argument. If the function given takes in more than 1 arguments, then many iterables are given. #Follow the link to know more similar functions

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

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])

21. What is a Python module?

A module is a Python script that generally contains import statements, functions, classes and variable definitions, and Python runnable code and it “lives” file with a ‘.py’ extension. zip files and DLL files can also be modules.Inside the module, you can refer to the module name as a string that is stored in the global variable name .

22. Name the File-related modules in Python?

Python provides libraries / modules with functions that enable you to manipulate text files and binary files on file system. Using them you can create files, update their contents, copy, and delete files. The libraries are : os, os.path, and shutil.

Here, os and os.path – modules include functions for accessing the filesystem

shutil – module enables you to copy and delete the files.

23. Explain the use of with statement?

In python generally “with” statement is used to open a file, process the data present in the file, and also to close the file without calling a close() method. “with” statement makes the exception handling simpler by providing cleanup activities.

General form of with:

with open(“filename”, “mode”) as file-var:

processing statements

note: no need to close the file by calling close() upon file-var.close()

24. Explain all the file processing modes supported by Python?

Python allows you to open files in one of the three modes. They are:

read-only mode, write-only mode, read-write mode, and append mode by specifying the flags “r”, “w”, “rw”, “a” respectively.

A text file can be opened in any one of the above said modes by specifying the option “t” along with

“r”, “w”, “rw”, and “a”, so that the preceding modes become “rt”, “wt”, “rwt”, and “at”.A binary file can be opened in any one of the above said modes by specifying the option “b” along with “r”, “w”, “rw”, and “a” so that the preceding modes become “rb”, “wb”, “rwb”, “ab”.

25. How many kinds of sequences are supported by Python? What are they?

Python supports 7 sequence types. They are str, list, tuple, unicode, byte array, xrange, and buffer. where xrange is deprecated in python 3.5.X.

26. How do you perform pattern matching in Python? Explain

Regular Expressions/REs/ regexes enable us to specify expressions that can match specific “parts” of a given string. For instance, we can define a regular expression to match a single character or a digit, a telephone number, or an email address, etc.The Python’s “re” module provides regular expression patterns and was introduce from later versions of Python 2.5. “re” module is providing methods for search text strings, or replacing text strings along with methods for splitting text strings based on the pattern defined.

27. How to display the contents of text file in reverse order?

1. convert the given file into a list.
2. reverse the list by using reversed()
3. Eg: for line in reversed(list(open(“file-name”,”r”))):
4. print(line)

28. What is the difference between NumPy and SciPy?

1. 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.
2. 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.
3. 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.
4. 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.

29. 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

30. What is the output of the following? try: if '1' != 1: raise

**a) some Error has  occured**

**b) some Error has not occured**

**c) invalid code**

**d) none of the above**

Answer: C

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

25

32. To open a file c:\scores.txt for writing?

fileWriter = open(“c:\\scores.txt”, “w”)

33. Name few Python modules for Statistical, Numerical and scientific computations ?

numPy – this module provides an array/matrix type, and it is useful for doing computations on arrays. scipy – this module provides methods for doing numeric integrals, solving differential equations, etc pylab – is a module for generating and saving plots

34. What is TkInter?

TkInter is Python library. It is a toolkit for GUI development. It provides support for various GUI tools or widgets (such as buttons, labels, text boxes, radio buttons, etc) that are used in GUI applications. The common attributes of them include Dimensions, Colors, Fonts, Cursors, etc.

35. Is Python object oriented? what is object oriented programming?

Yes. Python is Object Oriented Programming language. OOP is the programming paradigm based on classes and instances of those classes called objects. The features of OOP are:

Encapsulation, Data Abstraction, Inheritance, Polymorphism.

36. What is multithreading? Give an example.

It means running several different programs at the same time concurrently by invoking multiple threads. Multiple threads within a process refer the data space with main thread and they can communicate with each other to share information more easily.Threads are light-weight processes and have less memory overhead. Threads can be used just for quick task like calculating results and also running other processes in the background while the main program is running.

37. Does Python supports interfaces like in Java? Discuss.

Python does not provide interfaces like in Java. Abstract Base Class (ABC) and its feature are provided by the Python’s “abc” module. Abstract Base Class is a mechanism for specifying what methods must be implemented by its implementation subclasses. The use of ABC’c provides a sort of “understanding” about methods and their expected behaviour. This module was made available from Python 2.7 version onwards.

38. What are Accessors, mutators, @property?

Accessors and mutators are often called getters and setters in languages like “Java”. For example, if x is a property of a user-defined class, then the class would have methods called setX() and getX(). Python has an @property “decorator” that allows you to ad getters and setters in order to access the attribute of the class.

39. Differentiate between append() and extend() methods.?

Both append() and extend() methods are the methods of list. These methods a re used to add the elements at the end of the list.

append(element) – adds the given element at the end of the list which has called this method.

extend(another-list) – adds the elements of another-list at the end of the list which is called the extend method.

40. Name few methods that are used to implement Functionally Oriented Programming in Python?

Python supports methods (called iterators in Python3), such as filter(), map(), and reduce(), that are very useful when you need to iterate over the items in a list, create a dictionary, or extract a subset of a list.

filter() – enables you to extract a subset of values based on conditional logic.

map() – it is a built-in function that applies the function to each item in an iterable.

reduce() – repeatedly performs a pair-wise reduction on a sequence until a single value is computed.

41. What is the output of the following?

**x = [‘ab’, ‘cd’]  
print(len(map(list, x)))**

A TypeError occurs as map has no len().

42. What is the output of the following?

**x = [‘ab’, ‘cd’]  
print(len(list(map(list, x))))**

Explanation: The length of each string is 2.

43. Which of the following is not the correct syntax for creating a set?

1. **a) set([[1,2],[3,4]])**
2. **b) set([1,2,2,3,4])**
3. **c) set((1,2,3,4))**
4. **d) {1,2,3,4}**

Answer : a

Explanation : The argument given for the set must be an iterable.

44. Explain a few methods to implement Functionally Oriented Programming in Python.

Sometimes, when we want to iterate over a list, a few methods come in handy.

1. filter()

Filter lets us filter in some values based on conditional logic.

>>> list(filter(lambda x:x>5,range(8)))

[6, 7]

1. map()

Map applies a function to every element in an iterable.

>>> list(map(lambda x:x\*\*2,range(8)))

[0, 1, 4, 9, 16, 25, 36, 49]

1. reduce()

Reduce repeatedly reduces a sequence pair-wise until we reach a single value

>>> from functools import reduce

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

-13

45. Explain database connection in Python Flask?

Flask supports database powered application (RDBS). Such system requires creating a          schema, which requires piping the shema.sql file into a sqlite3 command.  So you need to install   sqlite3 command in order to create or initiate the database in Flask.

Flask allows to request database in three ways

* before\_request() : They are called before a request and pass no arguments
* after\_request() : They are called after a request and pass the response that will be sent to the client
* teardown\_request(): They are called in situation when exception is raised, and response are not guaranteed. They are called after the response been constructed. They are not allowed to modify the request, and their values are ignored.

46. Write a Python function that checks whether a passed string is palindrome Or not? Note: A palindrome is a word, phrase, or sequence that reads the same backward as forward, e.g., madam or nurses run.

def isPalindrome(string):  
left\_pos = 0  
right\_pos = len(string) – 1

while right\_pos >= left\_pos:  
if not string[left\_pos] == string[right\_pos]:  
return False  
left\_pos += 1  
right\_pos -= 1  
return True  
print(isPalindrome(‘aza’))

47. Write a Python program to calculate the sum of a list of numbers.

def list\_sum(num\_List):  
if len(num\_List) == 1:  
return num\_List[0]  
else:  
return num\_List[0] + list\_sum(num\_List[1:])

print(list\_sum([2, 4, 5, 6, 7]))

Sample Output:

24

48. How to retrieve data from a table in MySQL database through Python code? Explain.

1. import MySQLdb module as : import MySQLdb
2. establish a connection to the database.
3. db = MySQLdb.connect(“host”=”local host”, “database-user”=”user-name”, “password”=”password”, “database-name”=”database”)
4. initialize the cursor variable upon the established connection: c1 = db.cursor()
5. retrieve the information by defining a required query string. s = “Select \* from dept”
6. fetch the data using fetch() methods and print it. data = c1.fetch(s)
7. close the database connection. db.close()

49. Write a Python program to read a random line from a file.

import random  
def random\_line(fname):  
lines = open(fname).read().splitlines()  
return random.choice(lines)  
print(random\_line(‘test.txt’))

50. Write a Python program to count the number of lines in a text file.

def file\_lengthy(fname):  
with open(fname) as f:  
for i, l in enumerate(f):  
pass  
return i + 1  
print(“Number of lines in the file: “,file\_lengthy(“test.txt”))

**Q-1: What is Python, what are the benefits of using it, and what do you understand of PEP 8?**

Python is one of the most successful interpreted languages. When you write a Python script, it doesn’t need to get compiled before execution. Few other interpreted languages are PHP and Javascript.

**Benefits of Python Programming**

* Python is a dynamic-typed language. It means that you don’t need to mention the data type of variables during their declaration. It allows to set variables like var1=101 and var2 =” You are an engineer.” without any error.
* Python supports object orientated programming as you can define classes along with the composition and inheritance. It doesn’t use access specifiers like public or private).
* Functions in Python are like first-class objects. It suggests you can assign them to variables, return from other methods and pass as arguments.
* Developing using Python is quick but running it is often slower than compiled languages. Luckily, Python enables to include the “C” language extensions so you can optimize your scripts.
* Python has several usages like web-based applications, test automation, data modeling, big data analytics and much more. Alternatively, you can utilize it as a “glue” layer to work with other languages.

## Python Interview Questions

### Q-1: What is Python, what are the benefits of using it, and what do you understand of PEP 8?

Python is one of the most successful interpreted languages. When you write a Python script, it doesn’t need to get compiled before execution. Few other interpreted languages are PHP and Javascript.

#### Benefits of Python Programming

* Python is a dynamic-typed language. It means that you don’t need to mention the data type of variables during their declaration. It allows to set variables like var1=101 and var2 =” You are an engineer.” without any error.
* Python supports object orientated programming as you can define classes along with the composition and inheritance. It doesn’t use access specifiers like public or private).
* Functions in Python are like first-class objects. It suggests you can assign them to variables, return from other methods and pass as arguments.
* Developing using Python is quick but running it is often slower than compiled languages. Luckily, Python enables to include the “C” language extensions so you can optimize your scripts.
* Python has several usages like web-based applications, test automation, data modeling, big data analytics and much more. Alternatively, you can utilize it as a “glue” layer to work with other languages.

#### PEP 8.

PEP 8 is the latest Python coding standard, a set of coding recommendations. It guides to deliver more readable Python code.

### Q-2: What is the output of the following Python code fragment? Justify your answer.

def extendList(val, list=[]):

list.append(val)

return list

list1 = extendList(10)

list2 = extendList(123,[])

list3 = extendList('a')

print "list1 = %s" % list1

print "list2 = %s" % list2

print "list3 = %s" % list3

The result of the above Python code snippet is:

list1 = [10, 'a']

list2 = [123]

list3 = [10, 'a']

You may erroneously expect list1 to be equal to [10] and list3 to match with [‘a’], thinking that the list argument will initialize to its default value of [] every time there is a call to the extendList.

However, the flow is like that a new list gets created once after the function is defined. And the same get used whenever someone calls the extendList method without a list argument. It works like this because the calculation of expressions (in default arguments) occurs at the time of function definition, not during its invocation.

The list1 and list3 are hence operating on the same default list, whereas list2 is running on a separate object that it has created on its own (by passing an empty list as the value of the list parameter).

The definition of the extendList function can get changed in the following manner.

def extendList(val, list=None):

if list is None:

list = []

list.append(val)

return list

With this revised implementation, the output would be:

list1 = [10]

list2 = [123]

list3 = ['a']

### Q-3: What is the statement that can be used in Python if the program requires no action but requires it syntactically?

The pass statement is a null operation. Nothing happens when it executes. You should use “pass” keyword in lowercase. If you write “Pass,” you’ll face an error like “NameError: name Pass is not defined.” Python statements are case sensitive.

letter = "hai sethuraman"

for i in letter:

if i == "a":

pass

print("pass statement is execute ..............")

else:

print(i)

### Q-4: What’s the process to get the home directory using ‘~’ in Python?

You need to import the os module, and then just a single line would do the rest.

import os

print (os.path.expanduser('~'))

**Output:**

/home/runner

### Q-5: What are the built-in types available in Python?

Here is the list of most commonly used built-in types that Python supports:

* **Immutable built-in datatypes of Python**
  + Numbers
  + Strings
  + Tuples
* **Mutable built-in datatypes of Python**
  + List
  + Dictionaries
  + Sets

### Q-6: How to find bugs or perform static analysis in a Python application?

* You can use PyChecker, which is a static analyzer. It identifies the bugs in Python project and also reveals the style and complexity related bugs.
* Another tool is Pylint, which checks whether the Python module satisfies the coding standard.

### Q-7: When is the Python decorator used?

Python decorator is a relative change that you do in Python syntax to adjust the functions quickly.

### Q-8: What is the principal difference between a list and the tuple?

#### List vs. Tuple.

The principal difference between a list and the tuple is that the former is mutable while the tuple is not.

A tuple is allowed to be hashed, for example, using it as a key for dictionaries.

### Q-9: How does Python handle memory management?

* Python uses private heaps to maintain its memory. So the heap holds all the Python objects and the data structures. This area is only accessible to the Python interpreter; programmers can’t use it.
* And it’s the Python memory manager that handles the Private heap. It does the required allocation of the memory for Python objects.
* Python employs a built-in garbage collector, which salvages all the unused memory and offloads it to the heap space.

### Q-10: What are the principal differences between the lambda and def?

#### Lambda vs. def.

* Def can hold multiple expressions while lambda is a uni-expression function.
* Def generates a function and designates a name to call it later. Lambda forms a function object and returns it.
* Def can have a return statement. Lambda can’t have return statements.
* Lambda supports to get used inside a list and dictionary.

💡 **Also Check.**

**[Python Programming Quiz for Beginners](https://www.techbeamers.com/python-programming-quiz-for-beginners-part-1/" \t "_blank)**

### Q-11: Write a reg expression that confirms an email id using the python reg expression module “re”?

Python has a regular expression module “re.”

Check out the **“re”** expression that can check the email id for .com and .co.in subdomain.

import re

print(re.search(r"[0-9a-zA-Z.]+@[a-zA-Z]+\.(com|co\.in)$","micheal.pages@mp.com"))

### Q-12: What do you think is the output of the following code fragment? Is there any error in the code?

list = ['a', 'b', 'c', 'd', 'e']

print (list[10:])

The result of the above lines of code is []. There won’t be any error like an IndexError.

You should know that trying to fetch a member from the list using an index that exceeds the member count (for example, attempting to access list[10] as given in the question) would yield an IndexError. By the way, retrieving only a slice at the starting index that surpasses the no. of items in the list won’t result in an IndexError. It will just return an empty list.

### Q-13: Is there a switch or case statement in Python? If not then what is the reason for the same?

No, Python does not have a Switch statement, but you can write a Switch function and then use it.

### Q-14: What is a built-in function that Python uses to iterate over a number sequence?

Range() generates a list of numbers, which is used to iterate over for loops.

for i in range(5):

print(i)

The range() function accompanies two sets of parameters.

* **range(stop)**
  + stop: It is the no. of integers to generate and starts from zero. eg. range(3) == [0, 1, 2].
* **range([start], stop[, step])**
  + Start: It is the starting no. of the sequence.
  + Stop: It specifies the upper limit of the sequence.
  + Step: It is the incrementing factor for generating the sequence.
* **Points to note:**
  + Only integer arguments are allowed.
  + Parameters can be positive or negative.
  + The **range()** function in Python starts from the zeroth index.

### Q-15: What are the optional statements possible inside a try-except block in Python?

There are two optional clauses you can use in the **try-except** block.

* The “**else”** clause
  + It is useful if you want to run a piece of code when the try block doesn’t create an exception.
* The **“finally”** clause
  + It is useful when you want to execute some steps which run, irrespective of whether there occurs an exception or not.

### Q-16: What is a string in Python?

A string in Python is a sequence of alpha-numeric characters. They are immutable objects. It means that they don’t allow modification once they get assigned a value. Python provides several methods, such as join(), replace(), or split() to alter strings. But none of these change the original object.

### Q-17: What is slicing in Python?

Slicing is a string operation for extracting a part of the string, or some part of a list. In Python, a string (say text) begins at index 0, and the nth character stores at position text[n-1]. Python can also perform reverse indexing, i.e., in the backward direction, with the help of negative numbers. In Python, the slice() is also a constructor function which generates a slice object. The result is a set of indices mentioned by range(start, stop, step). The slice() method allows three parameters. 1. start – starting number for the slicing to begin. 2. stop – the number which indicates the end of slicing. 3. step – the value to increment after each index (default = 1).

### Q-18: What is %s in Python?

Python has support for formatting any value into a string. It may contain quite complex expressions.

One of the common usages is to push values into a string with the %s format specifier. The formatting operation in Python has the comparable syntax as the C function printf() has.

### Q-19: Is a string immutable or mutable in Python?

Python strings are indeed immutable.

Let’s take an example. We have an “str” variable holding a string value. We can’t mutate the container, i.e., the string, but can modify what it contains that means the value of the variable.

### Q-20: What is the index in Python?

An index is an integer data type which denotes a position within an ordered list or a string.

In Python, strings are also lists of characters. We can access them using the index which begins from zero and goes to the length minus one.

For example, in the string “Program,” the indexing happens like this:

Program 0 1 2 3 4 5

### Q-21: What is Docstring in Python?

A docstring is a unique text that happens to be the first statement in the following Python constructs:

Module, Function, Class, or Method definition.

A docstring gets added to the \_\_doc\_\_ attribute of the string object.

Now, read some of the Python interview questions on functions.

### Q-22: What is a function in Python programming?

A function is an object which represents a block of code and is a reusable entity. It brings modularity to a program and a higher degree of code reusability.

Python has given us many built-in functions such as print() and provides the ability to create user-defined functions.

### Q-23: How many basic types of functions are available in Python?

Python gives us two basic types of functions.

1. Built-in, and

2. User-defined.

The built-in functions happen to be part of the Python language. Some of these are print(), dir(), len(), and abs() etc.

### Q-24: How do we write a function in Python?

We can create a Python function in the following manner.

Step-1: to begin the function, start writing with the keyword def and then mention the function name.

Step-2: We can now pass the arguments and enclose them using the parentheses. A colon, in the end, marks the end of the function header.

Step-3: After pressing an enter, we can add the desired Python statements for execution.

### Q-25: What is a function call or a callable object in Python?

A function in Python gets treated as a callable object. It can allow some arguments and also return a value or multiple values in the form of a tuple. Apart from the function, Python has other constructs, such as classes or the class instances which fits in the same category.

### Q-26: What is the return keyword used for in Python?

The purpose of a function is to receive the inputs and return some output.

The return is a Python statement which we can use in a function for sending a value back to its caller.

### Q-27: What is “Call by Value” in Python?

In call-by-value, the argument whether an expression or a value gets bound to the respective variable in the function.

Python will treat that variable as local in the function-level scope. Any changes made to that variable will remain local and will not reflect outside the function.

### Q-28: What is “Call by Reference” in Python?

We use both “call-by-reference” and “pass-by-reference” interchangeably. When we pass an argument by reference, then it is available as an implicit reference to the function, rather than a simple copy. In such a case, any modification to the argument will also be visible to the caller.

This scheme also has the advantage of bringing more time and space efficiency because it leaves the need for creating local copies.

On the contrary, the disadvantage could be that a variable can get changed accidentally during a function call. Hence, the programmers need to handle in the code to avoid such uncertainty.

### Q-29: What is the return value of the trunc() function?

The Python trunc() function performs a mathematical operation to remove the decimal values from a particular expression and provides an integer value as its output.

### Q-30: Is it mandatory for a Python function to return a value?

It is not at all necessary for a function to return any value. However, if needed, we can use None as a return value.

### Q-31: What does the continue do in Python?

The continue is a jump statement in Python which moves the control to execute the next iteration in a loop leaving all the remaining instructions in the block unexecuted.

The continue statement is applicable for both the “while” and “for” loops.

### Q-32: What is the purpose of id() function in Python?

The id() is one of the built-in functions in Python.

Signature: id(object)

It accepts one parameter and returns a unique identifier associated with the input object.

### Q-33: What does the \*args do in Python?

We use \*args as a parameter in the function header. It gives us the ability to pass N (variable) number of arguments.

Please note that this type of argument syntax doesn’t allow passing a named argument to the function.

Example of using the \*args:

# Python code to demonstrate

# \*args for dynamic arguments

def fn(\*argList):

for argx in argList:

print (argx)

fn('I', 'am', 'Learning', 'Python')

The output:

I

am

Learning

Python

### Q-34: What does the \*\*kwargs do in Python?

We can also use the \*\*kwargs syntax in a Python function declaration. It let us pass N (variable) number of arguments which can be named or keyworded.

Example of using the \*\*kwargs:

# Python code to demonstrate

# \*\*kwargs for dynamic + named arguments

def fn(\*\*kwargs):

for emp, age in kwargs.items():

print ("%s's age is %s." %(emp, age))

fn(John=25, Kalley=22, Tom=32)

The output:

John's age is 25.

Kalley's age is 22.

Tom's age is 32.

### Q-35: Does Python have a Main() method?

The main() is the entry point function which happens to be called first in most programming languages.

Since Python is interpreter-based, so it sequentially executes the lines of the code one-by-one.

Python also does have a Main() method. But it gets executed whenever we run our Python script either by directly clicking it or starts it from the command line.

We can also override the Python default main() function using the Python if statement. Please see the below code.

print("Welcome")

print("\_\_name\_\_ contains: ", \_\_name\_\_)

def main():

print("Testing the main function")

if \_\_name\_\_ == '\_\_main\_\_':

main()

The output:

Welcome

\_\_name\_\_ contains: \_\_main\_\_

Testing the main function

### Q-36: What does the \_\_ Name \_\_ do in Python?

The \_\_name\_\_ is a unique variable. Since Python doesn’t expose the main() function, so when its interpreter gets to run the script, it first executes the code which is at level 0 indentation.

To see whether the main() gets called, we can use the \_\_name\_\_ variable in an if clause compares with the value “\_\_main\_\_.”

### Q-37: What is the purpose of “end” in Python?

Python’s print() function always prints a newline in the end. The print() function accepts an optional parameter known as the ‘end.’ Its value is ‘\n’ by default. We can change the end character in a print statement with the value of our choice using this parameter.

# Example: Print a instead of the new line in the end.

print("Let's learn" , end = ' ')

print("Python")

# Printing a dot in the end.

print("Learn to code from techbeamers" , end = '.')

print("com", end = ' ')

The output is:

Let's learn Python

Learn to code from techbeamers.com

### Q-38: When should you use the “break” in Python?

Python provides a break statement to exit from a loop. Whenever the break hits in the code, the control of the program immediately exits from the body of the loop.

The break statement in a nested loop causes the control to exit from the inner iterative block.

### Q-39: What is the difference between pass and continue in Python?

The continue statement makes the loop to resume from the next iteration.

On the contrary, the pass statement instructs to do nothing, and the remainder of the code executes as usual.

### Q-40: What does the len() function do in Python?

In Python, the len() is a primary string function. It determines the length of an input string.

>>> some\_string = 'techbeamers'

>>> len(some\_string)

11

### Q-41: What does the chr() function do in Python?

The chr() function got re-added in Python 3.2. In version 3.0, it got removed.

It returns the string denoting a character whose Unicode code point is an integer.

For example, the chr(122) returns the string ‘z’ whereas the chr(1212) returns the string ‘Ҽ’.

### Q-42: What does the ord() function do in Python?

The ord(char) in Python takes a string of size one and returns an integer denoting the Unicode code format of the character in case of a Unicode type object, or the value of the byte if the argument is of 8-bit string type.

>>> ord("z")

122

### Q-43: What is Rstrip() in Python?

Python provides the rstrip() method which duplicates the string but leaves out the whitespace characters from the end.

The rstrip() escapes the characters from the right end based on the argument value, i.e., a string mentioning the group of characters to get excluded.

The signature of the rstrip() is:

str.rstrip([char sequence/pre>

#Example

test\_str = 'Programming '

# The trailing whitespaces are excluded

print(test\_str.rstrip())

### Q-44: What is whitespace in Python?

Whitespace represents the characters that we use for spacing and separation.

They possess an “empty” representation. In Python, it could be a tab or space.

### Q-45: What is isalpha() in Python?

Python provides this built-in isalpha() function for the string handling purpose.

It returns True if all characters in the string are of alphabet type, else it returns False.

### Q-46: How do you use the split() function in Python?

Python’s split() function works on strings to cut a large piece into smaller chunks, or sub-strings. We can specify a separator to start splitting, or it uses the space as one by default.

#Example

str = 'pdf csv json'

print(str.split(" "))

print(str.split())

The output:

['pdf', 'csv', 'json']

['pdf', 'csv', 'json']

### Q-47: What does the join method do in Python?

Python provides the join() method which works on strings, lists, and tuples. It combines them and returns a united value.

### Q-48: What does the Title() method do in Python?

Python provides the title() method to convert the first letter in each word to capital format while the rest turns to Lowercase.

#Example

str = 'lEaRn pYtHoN'

print(str.title())

The output:

Learn Python

Now, check out some general purpose Python interview questions.

### Q-49: What makes the CPython different from Python?

CPython has its core developed in C. The prefix ‘C’ represents this fact. It runs an interpreter loop used for translating the Python-ish code to C language.

### Q-50: Which package is the fastest form of Python?

PyPy provides maximum compatibility while utilizing CPython implementation for improving its performance.

The tests confirmed that PyPy is nearly five times faster than the CPython. It currently supports Python 2.7.

### Q-51: What is GIL in Python language?

Python supports GIL (the global interpreter lock) which is a mutex used to secure access to Python objects, synchronizing multiple threads from running the Python bytecodes at the same time.

### Q-52: How is Python thread safe?

Python ensures safe access to threads. It uses the GIL mutex to set synchronization. If a thread loses the GIL lock at any time, then you have to make the code thread-safe.

For example, many of the Python operations execute as atomic such as calling the sort() method on a list.

### Q-53: How does Python manage the memory?

Python implements a heap manager internally which holds all of its objects and data structures.

This heap manager does the allocation/de-allocation of heap space for objects.

### Q-54: What is a tuple in Python?

A tuple is a collection type data structure in Python which is immutable.

They are similar to sequences, just like the lists. However, There are some differences between a tuple and list; the former doesn’t allow modifications whereas the list does.

Also, the tuples use parentheses for enclosing, but the lists have square brackets in their syntax.

### Q-55: What is a dictionary in Python programming?

A dictionary is a data structure known as an associative array in Python which stores a collection of objects.

The collection is a set of keys having a single associated value. We can call it a hash, a map, or a hashmap as it gets called in other programming languages.

### Q-56: What is the set object in Python?

Sets are unordered collection objects in Python. They store unique and immutable objects. Python has its implementation derived from mathematics.

### Q-57: What is the use of the dictionary in Python?

A dictionary has a group of objects (the keys) map to another group of objects (the values). A Python dictionary represents a mapping of unique Keys to Values.

They are mutable and hence will not change. The values associated with the keys can be of any Python types.

### Q-58: Is Python list a linked list?

A Python list is a variable-length array which is different from C-style linked lists.

Internally, it has a contiguous array for referencing to other objects and stores a pointer to the array variable and its length in the list head structure.

Here are some Python interview questions on classes and objects.

### Q-59: What is Class in Python?

Python supports object-oriented programming and provides almost all OOP features to use in programs.

A Python class is a blueprint for creating the objects. It defines member variables and gets their behavior associated with them.

We can make it by using the keyword “class.” An object gets created from the constructor. This object represents the instance of the class.

In Python, we generate classes and instances in the following way.

>>>class Human: # Create the class

... pass

>>>man = Human() # Create the instance

>>>print(man)

<\_\_main\_\_.Human object at 0x0000000003559E10>

### Q-60: What are Attributes and Methods in a Python class?

A class is useless if it has not defined any functionality. We can do so by adding attributes. They work as containers for data and functions. We can add an attribute directly specifying inside the class body.

>>> class Human:

... profession = "programmer" # specify the attribute 'profession' of the class

>>> man = Human()

>>> print(man.profession)

programmer

After we added the attributes, we can go on to define the functions. Generally, we call them methods. In the method signature, we always have to provide the first argument with a self-keyword.

>>> class Human:

profession = "programmer"

def set\_profession(self, new\_profession):

self.profession = new\_profession

>>> man = Human()

>>> man.set\_profession("Manager")

>>> print(man.profession)

Manager

### Q-61: How to assign values for the Class attributes at runtime?

We can specify the values for the attributes at runtime. We need to add an init method and pass input to object constructor. See the following example demonstrating this.

>>> class Human:

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

self.profession = profession

def set\_profession(self, new\_profession):

self.profession = new\_profession

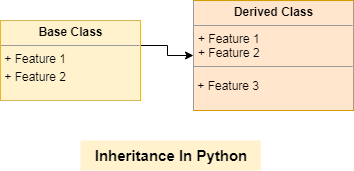
>>> man = Human("Manager")

>>> print(man.profession)

Manager

### Q-62: What is Inheritance in Python programming?

Inheritance is an OOP mechanism which allows an object to access its parent class features. It carries forward the base class functionality to the child.



We do it intentionally to abstract away the similar code in different classes.

The common code rests with the base class, and the child class objects can access it via inheritance. Check out the below example.

class PC: # Base class

processor = "Xeon" # Common attribute

def set\_processor(self, new\_processor):

processor = new\_processor

class Desktop(PC): # Derived class

os = "Mac OS High Sierra" # Personalized attribute

ram = "32 GB"

class Laptop(PC): # Derived class

os = "Windows 10 Pro 64" # Personalized attribute

ram = "16 GB"

desk = Desktop()

print(desk.processor, desk.os, desk.ram)

lap = Laptop()

print(lap.processor, lap.os, lap.ram)

The output:

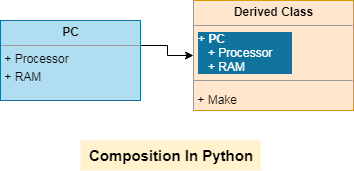
Xeon Mac OS High Sierra 32 GB

Xeon Windows 10 Pro 64 16 GB

### Q-63: What is Composition in Python?

The composition is also a type of inheritance in Python. It intends to inherit from the base class but a little differently, i.e., by using an instance variable of the base class acting as a member of the derived class.

See the below diagram.



To demonstrate composition, we need to instantiate other objects in the class and then make use of those instances.

class PC: # Base class

processor = "Xeon" # Common attribute

def \_\_init\_\_(self, processor, ram):

self.processor = processor

self.ram = ram

def set\_processor(self, new\_processor):

processor = new\_processor

def get\_PC(self):

return "%s cpu & %s ram" % (self.processor, self.ram)

class Tablet():

make = "Intel"

def \_\_init\_\_(self, processor, ram, make):

self.PC = PC(processor, ram) # Composition

self.make = make

def get\_Tablet(self):

return "Tablet with %s CPU & %s ram by %s" % (self.PC.processor, self.PC.ram, self.make)

if \_\_name\_\_ == "\_\_main\_\_":

tab = Tablet("i7", "16 GB", "Intel")

print(tab.get\_Tablet())

The output is:

Tablet with i7 CPU & 16 GB ram by Intel

### Q-64: What are Errors and Exceptions in Python programs?

Errors are coding issues in a program which may cause it to exit abnormally.

On the contrary, exceptions happen due to the occurrence of an external event which interrupts the normal flow of the program.

### Q-65: How do you handle exceptions with Try/Except/Finally in Python?

Python lay down Try, Except, Finally constructs to handle errors as well as Exceptions. We enclose the unsafe code indented under the try block. And we can keep our fall-back code inside the except block. Any instructions intended for execution last should come under the finally block.

try:

print("Executing code in the try block")

print(exception)

except:

print("Entering in the except block")

finally:

print("Reached to the final block")

The output is:

Executing code in the try block

Entering in the except block

Reached to the final block

### Q-66: How do you raise exceptions for a predefined condition in Python?

We can raise an exception based on some condition.

For example, if we want the user to enter only odd numbers, else will raise an exception.

# Example - Raise an exception

while True:

try:

value = int(input("Enter an odd number- "))

if value%2 == 0:

raise ValueError("Exited due to invalid input!!!")

else:

print("Value entered is : %s" % value)

except ValueError as ex:

print(ex)

break

The output is:

Enter an odd number- 2

Exited due to invalid input!!!

Enter an odd number- 1

Value entered is : 1

Enter an odd number-

### Q-67: What are Python Iterators?

Iterators in Python are array-like objects which allow moving on the next element. We use them in traversing a loop, for example, in a “for” loop.

Python library has a no. of iterators. For example, a list is also an iterator and we can start a for loop over it.

### Q-68: What is the difference between an Iterator and Iterable?

The collection type like a list, tuple, dictionary, and set are all iterable objects whereas they are also iterable containers which return an iterator while traversing.

Here are some advanced-level Python interview questions.

### Q-69: What are Python Generators?

A Generator is a kind of function which lets us specify a function that acts like an iterator and hence can get used in a “for” loop.

In a generator function, the yield keyword substitutes the return statement.

# Simple Python function

def fn():

return "Simple Python function."

# Python Generator function

def generate():

yield "Python Generator function."

print(next(generate()))

The output is:

Python Generator function.

### Q-70: What are Closures in Python?

Python closures are function objects returned by another function. We use them to eliminate code redundancy.

In the example below, we’ve written a simple closure for multiplying numbers.

def multiply\_number(num):

def product(number):

'product() here is a closure'

return num \* number

return product

num\_2 = multiply\_number(2)

print(num\_2(11))

print(num\_2(24))

num\_6 = multiply\_number(6)

print(num\_6(1))

The output is:

22

48

6

**Q-71: What are Decorators in Python?**

Python decorator gives us the ability to add new behavior to the given objects dynamically. In the example below, we’ve written a simple example to display a message pre and post the execution of a function.

def decorator\_sample(func):

def decorator\_hook(\*args, \*\*kwargs):

print("Before the function call")

result = func(\*args, \*\*kwargs)

print("After the function call")

return result

return decorator\_hook

@decorator\_sample

def product(x, y):

"Function to multiply two numbers."

return x \* y

print(product(3, 3))

The output is:

Before the function call

After the function call

9

**Q-72: How do you create a dictionary in Python?**

Let’s take the example of building site statistics. For this, we first need to break up the key-value pairs using a colon(“:”). The keys should be of an immutable type, i.e., so we’ll use the data-types which don’t allow changes at runtime. We’ll choose from an int, string, or tuple.

However, we can take values of any kind. For distinguishing the data pairs, we can use a comma(“,”) and keep the whole stuff inside curly braces({…}).

>>> site\_stats = {'site': 'tecbeamers.com', 'traffic': 10000, "type": "organic"}

>>> type(site\_stats)

<class 'dict'>

>>> print(site\_stats)

{'type': 'organic', 'site': 'tecbeamers.com', 'traffic': 10000}

**Q-73: How do you read from a dictionary in Python?**

To fetch data from a dictionary, we can directly access using the keys. We can enclose a “key” using brackets […] after mentioning the variable name corresponding to the dictionary.

>>> site\_stats = {'site': 'tecbeamers.com', 'traffic': 10000, "type": "organic"}

>>> print(site\_stats["traffic"])

We can even call the get method to fetch the values from a dict. It also let us set a default value. If the key is missing, then the KeyError would occur.

>>> site\_stats = {'site': 'tecbeamers.com', 'traffic': 10000, "type": "organic"}

>>> print(site\_stats.get('site'))

tecbeamers.com

**Q-74: How do you traverse through a dictionary object in Python?**

We can use the “for” and “in” loop for traversing the dictionary object.

>>> site\_stats = {'site': 'tecbeamers.com', 'traffic': 10000, "type": "organic"}

>>> for k, v in site\_stats.items():

print("The key is: %s" % k)

print("The value is: %s" % v)

print("++++++++++++++++++++++++")

The output is:

The key is: type

The value is: organic

++++++++++++++++++++++++

The key is: site

The value is: tecbeamers.com

++++++++++++++++++++++++

The key is: traffic

The value is: 10000

++++++++++++++++++++++++

**Q-75: How do you add elements to a dictionary in Python?**

We can add elements by modifying the dictionary with a fresh key and then set the value to it.

>>> # Setup a blank dictionary

>>> site\_stats = {}

>>> site\_stats['site'] = 'google.com'

>>> site\_stats['traffic'] = 10000000000

>>> site\_stats['type'] = 'Referral'

>>> print(site\_stats)

{'type': 'Referral', 'site': 'google.com', 'traffic': 10000000000}

We can even join two dictionaries to get a bigger dictionary with the help of the update() method.

>>> site\_stats['site'] = 'google.co.in'

>>> print(site\_stats)

{'site': 'google.co.in'}

>>> site\_stats\_new = {'traffic': 1000000, "type": "social media"}

>>> site\_stats.update(site\_stats\_new)

>>> print(site\_stats)

{'type': 'social media', 'site': 'google.co.in', 'traffic': 1000000}

**Q-76: How do you delete elements of a dictionary in Python?**

We can delete a key in a dictionary by using the del() method.

>>> site\_stats = {'site': 'tecbeamers.com', 'traffic': 10000, "type": "organic"}

>>> del site\_stats["type"]

>>> print(site\_stats)

{'site': 'google.co.in', 'traffic': 1000000}

Another method, we can use is the pop() function. It accepts the key as the parameter. Also, a second parameter, we can pass a default value if the key doesn’t exist.

>>> site\_stats = {'site': 'tecbeamers.com', 'traffic': 10000, "type": "organic"}

>>> print(site\_stats.pop("type", None))

organic

>>> print(site\_stats)

{'site': 'tecbeamers.com', 'traffic': 10000}

**Q-77: How do you check the presence of a key in a dictionary?**

We can use Python’s “in” operator to test the presence of a key inside a dict object.

>>> site\_stats = {'site': 'tecbeamers.com', 'traffic': 10000, "type": "organic"}

>>> 'site' in site\_stats

True

>>> 'traffic' in site\_stats

True

>>> "type" in site\_stats

True

Earlier, Python also provided the has\_key() method which got deprecated.

**Q-78: What is the syntax for List comprehension in Python?**

The signature for the list comprehension is as follows:

[ expression(var) for var in iterable ]

For example, the below code will return all the numbers from 10 to 20 and store them in a list.

>>> alist = [var for var in range(10, 20)]

>>> print(alist)

**Q-79: What is the syntax for Dictionary comprehension in Python?**

A dictionary has the same syntax as was for the list comprehension but the difference is that it uses curly braces:

{ aKey, itsValue for aKey in iterable }

For example, the below code will return all the numbers 10 to 20 as the keys and will store the respective squares of those numbers as the values.

>>> adict = {var:var\*\*2 for var in range(10, 20)}

>>> print(adict)

**Q-80: What is the syntax for Generator expression in Python?**

The syntax for generator expression matches with the list comprehension, but the difference is that it uses parenthesis:

( expression(var) for var in iterable )

For example, the below code will create a generator object that generates the values from 10 to 20 upon using it.

>>> (var for var in range(10, 20))

at 0x0000000003668728>

>>> list((var for var in range(10, 20)))

Now, see more Python interview questions for practice.

**Q-81: How do you write a conditional expression in Python?**

We can utilize the following single statement as a conditional expression. default\_statment if Condition else another\_statement

>>> no\_of\_days = 366

>>> is\_leap\_year = "Yes" if no\_of\_days == 366 else "No"

>>> print(is\_leap\_year)

Yes

**Q-82: What do you know about the Python enumerate?**

While using the iterators, sometimes we might have a use case to store the count of iterations. Python gets this task quite easy for us by giving a built-in method known as the enumerate().

The enumerate() function attaches a counter variable to an iterable and returns it as the “enumerated” object.

We can use this object directly in the “for” loops or transform it into a list of tuples by calling the list() method. It has the following signature:

enumerate(iterable, to\_begin=0)

Arguments:

iterable: array type object which enables iteration

to\_begin: the base index for the counter is to get started, its default value is 0

# Example - enumerate function

alist = ["apple","mango", "orange"]

astr = "banana"

# Let's set the enumerate objects

list\_obj = enumerate(alist)

str\_obj = enumerate(astr)

print("list\_obj type:", type(list\_obj))

print("str\_obj type:", type(str\_obj))

print(list(enumerate(alist)) )

# Move the starting index to two from zero

print(list(enumerate(astr, 2)))

The output is:

list\_obj type: <class 'enumerate'>

str\_obj type: <class 'enumerate'>

[(0, 'apple'), (1, 'mango'), (2, 'orange')]

[(2, 'b'), (3, 'a'), (4, 'n'), (5, 'a'), (6, 'n'), (7, 'a')]

**Q-83: What is the use of globals() function in Python?**

The globals() function in Python returns the current global symbol table as a dictionary object.

Python maintains a symbol table to keep all necessary information about a program. This info includes the names of variables, methods, and classes used by the program.

All the information in this table remains in the global scope of the program and Python allows us to retrieve it using the globals() method.

Signature: globals()

Arguments: None

# Example: globals() function

x = 9

def fn():

y = 3

z = y + x

# Calling the globals() method

z = globals()['x'] = z

return z

# Test Code

ret = fn()

print(ret)

The output is:

12

**Q-84: Why do you use the zip() method in Python?**

The zip method lets us map the corresponding index of multiple containers so that we can use them using as a single unit.

Signature:

zip(\*iterators)

Arguments:

Python iterables or collections (e.g., list, string, etc.)

Returns:

A single iterator object with combined mapped values

# Example: zip() function

emp = [ "tom", "john", "jerry", "jake" ]

age = [ 32, 28, 33, 44 ]

dept = [ 'HR', 'Accounts', 'R&D', 'IT' ]

# call zip() to map values

out = zip(emp, age, dept)

# convert all values for printing them as set

out = set(out)

# Displaying the final values

print ("The output of zip() is : ",end="")

print (out)

The output is:

The output of zip() is : {('jerry', 33, 'R&D'), ('jake', 44, 'IT'), ('john', 28, 'Accounts'), ('tom', 32, 'HR')}

**Q-85: What are Class or Static Variables in Python programming?**

In Python, all the objects share common class or static variables.

But the instance or non-static variables are altogether different for different objects.

The programming languages like C++ and Java need to use the static keyword to make a variable as the class variable. However, Python has a unique way to declare a static variable.

All names initialized with a value in the class declaration becomes the class variables. And those which get assigned values in the class methods becomes the instance variables.

# Example

class Test:

aclass = 'programming' # A class variable

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

self.ainst = ainst # An instance variable

# Objects of CSStudent class

test1 = Test(1)

test2 = Test(2)

print(test1.aclass)

print(test2.aclass)

print(test1.ainst)

print(test2.ainst)

# A class variable is also accessible using the class name

print(Test.aclass)

The output is:

programming

programming

1

2

programming

**Let’s now answer some advanced-level Python interview questions**.

**Q-86: How does the ternary operator work in Python?**

The ternary operator is an alternative for the conditional statements. It combines true or false values with a statement that you need to test.

The syntax would look like the one given below.

**[onTrue] if [Condition] else [onFalse]**

x, y = 35, 75

smaller = x if x < y else y

print(smaller)

**Q-87: What does the “self” keyword do?**

The **self** is a Python keyword which represents a variable that holds the instance of an object.

In almost, all the object-oriented languages, it is passed to the methods as a hidden parameter.

**Q-88: What are the different methods to copy an object in Python?**

There are two ways to copy objects in Python.

* **copy.copy() function**
  + It makes a copy of the file from source to destination.
  + It’ll return a shallow copy of the parameter.
* **copy.deepcopy() function**
  + It also produces the copy of an object from the source to destination.
  + It’ll return a deep copy of the parameter that you can pass to the function.

**Q-89: What is the purpose of docstrings in Python?**

In Python, the docstring is what we call as the docstrings. It sets a process of recording Python functions, modules, and classes.

**Q-90: Which Python function will you use to convert a number to a string?**

For converting a number into a string, you can use the built-in function **str()**.  If you want an octal or hexadecimal representation, use the inbuilt function **oct()** or **hex()**.

💡 **Also Check.**

[**Python Multithreading Quiz**](https://www.techbeamers.com/python-multithreading-quiz-to-test-your-skills/)

**Q-91: How do you debug a program in Python? Is it possible to step through the Python code?**

Yes, we can use the Python debugger (**pdb**) to debug any Python program. And if we start a program using **pdb**, then it let us even step through the code.

**Q-92: List down some of the PDB commands for debugging Python programs?**

Here are a few PDB commands to start debugging Python code.

* Add breakpoint **(b)**
* Resume execution **(c)**
* Step by step debugging **(s)**
* Move to the next line **(n)**
* List source code **(l)**
* Print an expression **(p)**

**Q-93: What is the command to debug a Python program?**

The following command helps run a Python program in debug mode.

$ python -m pdb python-script.py

**Q-94: How do you monitor the code flow of a program in Python?**

In Python, we can use **the sys** module’s **settrace()** method to setup trace hooks and monitor the functions inside a program.

You need to define a trace callback method and pass it to the **settrace()** function. The callback should specify three arguments as shown below.

import sys

def trace\_calls(frame, event, arg):

# The 'call' event occurs before a function gets executed.

if event != 'call':

return

# Next, inspect the frame data and print information.

print 'Function name=%s, line num=%s' % (frame.f\_code.co\_name, frame.f\_lineno)

return

def demo2():

print 'in demo2()'

def demo1():

print 'in demo1()'

demo2()

sys.settrace(trace\_calls)

demo1()

**Q-95: Why and when do you use generators in Python?**

A generator in Python is a function which returns an iterable object. We can iterate on the generator object using the **yield** keyword. But we can only do that once because their values don’t persist in memory, they get the values on the fly.

Generators give us the ability to hold the execution of a function or a step as long as we want to keep it. However, here are a few examples where it is beneficial to use generators.

* We can replace loops with generators for efficiently calculating results involving large data sets.
* Generators are useful when we don’t want all the results and wish to hold back for some time.
* Instead of using a callback function, we can replace it with a generator. We can write a loop inside the function doing the same thing as the callback and turns it into a generator.

**Q-96: What does the yield keyword do in Python?**

The **yield** keyword can turn any function into a generator. It works like a standard return keyword. But it’ll always return a generator object. Also, a method can have multiple calls to the **yield** keyword.

See the example below.

def testgen(index):

weekdays = ['sun','mon','tue','wed','thu','fri','sat']

yield weekdays[index]

yield weekdays[index+1]

day = testgen(0)

print next(day), next(day)

#output: sun mon

**Q-97: How to convert a list into other data types?**

Sometimes, we don’t use lists as is. Instead, we have to convert them to other types.

**Turn a list into a string.**

We can use the **”.join()** method which combines all elements into one and returns as a string.

weekdays = ['sun','mon','tue','wed','thu','fri','sat']

listAsString = ' '.join(weekdays)

print(listAsString)

#output: sun mon tue wed thu fri sat

**Turn a list into a tuple.**

Call Python’s **tuple()** function for converting a list into a tuple.

This function takes the list as its argument.

But remember, we can’t change the list after turning it into a tuple because it becomes immutable.

weekdays = ['sun','mon','tue','wed','thu','fri','sat']

listAsTuple = tuple(weekdays)

print(listAsTuple)

#output: ('sun', 'mon', 'tue', 'wed', 'thu', 'fri', 'sat')

**Turn a list into a set.**

Converting a list to a set poses two side-effects.

* Set doesn’t allow duplicate entries so that the conversion will remove any such item.
* A set is an ordered collection, so the order of list items would also change.

However, we can use the **set()** function to convert a list into a Set.

weekdays = ['sun','mon','tue','wed','thu','fri','sat','sun','tue']

listAsSet = set(weekdays)

print(listAsSet)

#output: set(['wed', 'sun', 'thu', 'tue', 'mon', 'fri', 'sat'])

**Turn a list into a dictionary.**

In a dictionary, each item represents a key-value pair. So converting a list isn’t as straightforward as it were for other data types.

However, we can achieve the conversion by breaking the list into a set of pairs and then call the **zip()** function to return them as tuples.

Passing the tuples into the **dict()** function would finally turn them into a dictionary.

weekdays = ['sun','mon','tue','wed','thu','fri']

listAsDict = dict(zip(weekdays[0::2], weekdays[1::2]))

print(listAsDict)

#output: {'sun': 'mon', 'thu': 'fri', 'tue': 'wed'}

**Q-98: How do you count the occurrences of each item present in the list without explicitly mentioning them?**

Unlike sets, lists can have items with the same values.

In Python, the list has a **count()** function which returns the occurrences of a particular item.

**Count the occurrences of an individual item.**

weekdays = ['sun','mon','tue','wed','thu','fri','sun','mon','mon']

print(weekdays.count('mon'))

#output: 3

**Count the occurrences of each item in the list.**

We’ll use the list comprehension along with the **count()** method. It’ll print the frequency of each of the items.

weekdays = ['sun','mon','tue','wed','thu','fri','sun','mon','mon']

print([[x,weekdays.count(x)] for x in set(weekdays)])

#output: [['wed', 1], ['sun', 2], ['thu', 1], ['tue', 1], ['mon', 3], ['fri', 1]]

**Q-99: What is NumPy and how is it better than a list in Python?**

NumPy is a Python package for scientific computing which can deal with large data sizes. It includes a powerful N-dimensional array object and a set of advanced functions.

Also, the NumPy arrays are superior to the built-in lists. There are a no. of reasons for this.

* NumPy arrays are more compact than lists.
* Reading and writing items is faster with NumPy.
* Using NumPy is more convenient than to the standard list.
* NumPy arrays are more efficient as they augment the functionality of lists in Python.

**Q-100: What are different ways to create an empty NumPy array in Python?**

There are two methods which we can apply to create empty NumPy arrays.

**The first method to create an empty array.**

import numpy

numpy.array([])

**The second method to create an empty array.**

# Make an empty NumPy array

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