Python Programming Interview Questions [2022] – Crack your Coding Interviews

***Free Python course with 35 real-time projects***[***Start Now!!***](https://data-flair.training/python-course/)

Python Programming Interview Questions and Answers – Prepare with DataFlair’s Python Interview Series

* [**Python Coding Interview Questions for Freshers**](https://data-flair.training/blogs/top-python-interview-questions-answer/)
* [**Python Coding Interview Questions for Intermediates**](https://data-flair.training/blogs/python-interview-questions/)
* **Python Coding Interview Questions for Experts**

This is the second part of our Python Programming Interview Questions and Answers Series, soon we will publish more. Like our other parts of python programming interview questions, this part is also divided into further subcategories. The reason we are doing is so you can practice all type of Python Interview Questions along with their answers. Here is how I divided these Python coding interview questions for experienced –

* Python Project Related Interview Questions
* Top Python Programming Interview Questions
* Technical Python Programming Interview Questions
* Python Library Interview Questions
* Python Developer Interview Questions
* Python Data Science Interview Questions
* Advanced Python Programming Interview Questions
* Python OOPS Interview Questions
* Python Programming Interview Questions for Testers
* Python Framework Interview Questions
* Frequently asked Python coding Interview Questions

So, without wasting any time let’s start with our first category that is Python project-related Interview Questions and Answers.

Python Project Related-Interview Questions and Answers

If you are experienced then surely you have worked on projects and this is the only tool which can drive the interview on your side. Let’s explore the most important python interview questions for experienced.

**Q.1. What kinds of classifiers did you use for your projects?**

An MLPClassifier is a Multi-Layer Perceptron Classifier. It optimizes the log-loss function using the LBFGS or stochastic gradient descent. It is a feedforward ANN model. This is different to the SVM or Naive Bayes algorithms in that it has an internal neural network for classification. Along with the librosa and soundfile libraries, I used the MLPClassifier from the sklearn library to perform ***speech emotion recognition with Python***.

I used a softmax classifier to detect the presence of the IDC (Invasive Ductal Carcinoma) breast cancer. This classifier gets the probabilities for each class label. This is very commonly used as is SVM. The softmax classifier is a generalization of the binary logistic regression to multiple classes.

The XGBClassifier is the XGBoost model for classification. I used it to classify between healthy patients and those with Parkinson’s Disease- it uses eXtreme Gradient Boosting and falls under the category of Ensemble Learning in ML. Training and predicting using many models, it produces one superior output.

To ***detect fake news with Python***, I used the PassiveAggressiveClassifier. Such online learning algorithms that stays passive for a correct classification outcome, but turns aggressive if there is a miscalculation. This lets it update and adjust. Such an algorithm does not converge.

**Q.2. Why did you choose OpenCV for your project on gender and age detection? What helped you make the decision?**

OpenCV is a very popular library for computer vision, and is powerful too. I chose it for my ***project on gender and age detection*** because it had the tools required for my project. It had the following benefits:

* Free
* Fast and written in C/C++
* Low RAM usage (MATLAB needs a lot of resources)
* Portable

When I had to choose between OpenCV and MATLAB, however, the latter had some advantages over the former:

* Since OpenCV has a flann library of its own, it can create conflict when using OpenCV with PCL
* There can be memory leaks]
* OpenCV does not have an IDE like MATLAB does

In the end, I found OpenCV to be better suited for my project, and chose to go ahead with it.

**Q.3. What ratio did you choose to divide your dataset into training and testing sets?**

A classifier needs us to divide a dataset into training and testing sets. The ratio is very important because it can affect the accuracy of the model. If we have few training data, the parameter estimates will have greater variance. And if we have a few test data, the performance statistic will have greater variance. Depending on the purpose and the size of the dataset, we usually split the dataset into 60% to 80% for training, and 40% to 20% for testing.

For most of the projects, I used a ratio of 80:20. However, for the ***project on breast cancer classification***, I used 10% of that 80% of the training set for validation. A model’s performance can depend on the values of its parameters. I used the training set to train the model and the validation set to evaluate the performance of the model for different combinations of hyperparameter values so I could figure out which one was best.

**Q.4. Why did you use config.py for this project on breast cancer classification?**

I created a config file for this project to keep every setting at one place. This made it easy to change the settings once and for all. This held the path to the input dataset, and the base path. It also held the relative paths to the training, validation, and testing sets. So if the base path was datasets/idc, the training set path was datasets/idc/training. It also mentioned the relative sizes of the training and validation sets, which were 0.8 and 0.1 respectively.

**Q.5. How was your experience with the XGBClassifier?**

My favorite thing about XGBoost is that it is scalable.  I used it for the project on ***detecting Parkinson’s Disease.*** It is also easy and efficient, and delivers high performance and accuracy compared to some other algorithms. In the multiple projects that I’ve tried it for, I found it delivered some of the best results. It is fast and has a Python API that makes its installation easier. It lets me access sparse datasets and is versatile. One thing I didn’t like about it is its documentation. It wasn’t very clear with its APIs and examples.

**Q.6 What challenges did you face in your best Python projects?**

Which algorithm to use was a challenge to figure out. Since the algorithm you use can affect your model’s accuracy, it is always a task to decide what algorithms and concepts to use together. Also, managing the changes to the code was tough, and that taught me the importance of version control. It was also important to make my code readable and easily modifiable.

Top Python Programming Interview Questions

**Q.7. What do you know about palindromes? Can you implement one in Python?**

A palindrome is a phrase, a word, or a sequence that reads the same forward and backward. One such example will be pip! An example of such a phrase will be ‘nurses run’. Let’s implement it, shall we?

>>> **def** isPalindrome(string):

left,right=0,len(string)-1

**while** right>=left:

**if** not string[left]==string[right]:

**return** **False**

left+=1;right-=1

**return** **True**

>>> isPalindrome('redrum murder')

**True**

>>> isPalindrome('CC.')

**False**

Well, there are other ways to do this too. Let’s try using an **iterator**.

>>> **def** isPalindrome(string):

left,right=iter(string),iter(string[::-1])

i=0

**while** i<len(string)/2:

**if** next(left)!=next(right):

**return** **False**

i+=1

**return** **True**

>>> isPalindrome('redrum murder')

**True**

>>> isPalindrome('CC.')

**False**

>>> isPalindrome('CCC.')

**False**

>>> isPalindrome('CCC')

**True**

**Q.8. What do you mean by \*args and \*\*kwargs?**

In cases when we don’t know how many arguments will be passed to a function, like when we want to pass a list or a tuple of values, we use \*args.

>>> **def** func(\*args):

**for** i **in** args:

print(i)

>>> func(3,2,1,4,7)

**3**  
**2**  
**1**  
**4**  
**7**

\*\*kwargs takes keyword arguments when we don’t know how many there will be.

>>> **def** func(\*\*kwargs):

**for** i **in** kwargs:

print(i,kwargs[i])

>>> func(a=1,b=2,c=7)

a.1  
b.2  
c.7  
The words args and kwargs are a convention, and we can use anything in their place.

**Q.9. What is a closure in Python?**

A ***closure in Python*** is said to occur when a nested function references a value in its enclosing scope. The whole point here is that it remembers the value.

>>> **def** A(x):

**def** B():

print(x)

**return** B

>>> A(7)()

**7**

**Q.10. Are these statements optimal? If not, optimize them.**

word=’word’

print(word.\_\_len\_\_())

No, these are not optimal. Let’s check the manual for this.

>>> help(str.\_\_len\_\_)

Help on wrapper\_descriptor:

\_\_len\_\_(self, /)

Return len(self).

\_\_len\_\_ is a wrapper descriptor which in turn makes a call to len(). So why not skip the work and do just that instead?

The optimal solution:

>>> word='word'

>>> len(word)

**4**

**Q.11. What is the iterator protocol?**

The iterator protocol for Python declares that we must make use of two functions to build an iterator- iter() and next().

iter()- To create an iterator

next()- To iterate to the next element

>>> a=iter([2,4,6,8,10])

>>> next(a)

**2**

>>> next(a)

**4**

>>> next(a)

**6**

>>> next(a)

**8**

>>> next(a)

**10**

>>> next(a)

**Traceback (most recent call last):**  
**File “<pyshell#31>”, line 1, in <module>**  
**next(a)**  
**StopIteration**

**Q.12. What is tuple unpacking?**

Suppose we have a tuple nums=(1,2,3). We can unpack its values into the variables a, b, and c. Here’s how:

>>> nums=(1,2,3)

>>> a,b,c=nums

>>> a

**1**

>>> b

**2**

>>> c

**3**

**Q.13. What will the following code output?**

>>> a=1

>>> a,b=a+1,a+1

>>> a,b

The output is (2, 2). This code increments the value of a by 1 and assigns it to both a and b. This is because this is a simultaneous declaration. The following code gives us the same:

>>> a=1

>>> b,a=a+1,a+1

>>> a,b

**(2, 2)**

**Q.14. What is a frozen set in Python?**

First, let’s discuss ***what a set is***. A set is a collection of items, where there cannot be any duplicates. A set is also unordered.

>>> myset={1,3,2,2}

>>> myset

**{1, 2, 3}**

This means that we cannot index it.

>>> myset[0]

**Traceback (most recent call last):**  
**File “<pyshell#197>”, line 1, in <module>**  
**myset[0]**  
**TypeError: ‘set’ object does not support indexing**

However, a set is mutable. A frozen set is immutable. This means we cannot change its values. This also makes it eligible to be used as a key for a dictionary.

>>> myset=frozenset([1,3,2,2])

>>> myset

**frozenset({1, 2, 3})**

>>> type(myset)

<**class** ‘frozenset’>

Before you start exploring the next category, do let me know (through comments) how many questions you have answered by yourself from the above top python programming interview questions.

Python Technical Interview Questions

**Q.15. When you exit Python, is all memory deallocated?**

Exiting Python deallocates everything except:

* modules with circular references
* Objects referenced from global namespaces
* Parts of memory reserved by the C library

**Q.16. What is the Dogpile effect?**

In case the cache expires, what happens when a client hits a website with multiple requests is what we call the dogpile effect. To avoid this, we can use a semaphore lock. When the value expires, the first process acquires the lock and then starts to generate the new value.

**Q.17. Explain garbage collection with Python.**

The following points are worth nothing for the garbage collector with CPython-

* Python maintains a count of how many references there are to each object in memory
* When a reference count drops to zero, it means the object is dead and Python can free the memory it allocated to that object
* The garbage collector looks for reference cycles and cleans them up
* Python uses heuristics to speed up garbage collection
* Recently created objects might as well be dead
* The garbage collector assigns generations to each object as it is created
* It deals with the younger generations first.

**Q.18. How will you use Python to read a random line from a file?**

We can borrow the choice() method from the random module for this.

>>> import random

>>> lines=open('tabs.txt').read().splitlines()

>>> random.choice(lines)

**‘https://data-flair.training/blogs/category/python/’**

Let’s restart the IDLE and do this again.

>>> import random

>>> lines=open('tabs.txt').read().splitlines()

>>> random.choice(lines)

**‘https://data-flair.training/blogs/’**

>>> random.choice(lines)

**‘https://data-flair.training/blogs/category/python/’**

>>> random.choice(lines)

**‘https://data-flair.training/blogs/category/python/’**

>>> random.choice(lines)

**‘https://data-flair.training/blogs/category/python/’**

**Q.19. What is JSON? Describe in brief how you’d convert JSON data into Python data?**

JSON stands for JavaScript Object Notation. It is a highly popular data format, and it stores data into NoSQL databases. JSON is generally built on the following two structures:

* A collection of <name,value> pairs
* An ordered list of values.

Python supports JSON parsers. In fact, JSON-based data is internally represented as a dictionary in Python. To convert JSON data into Python data, we use the load() function from the JSON module.

**Q.20. Differentiate between split(), sub(), and subn() methods of the remodule.**

The re module is what we have for processing regular expressions with Python. Let’s talk about the three methods we mentioned-

* split()- This makes use of a regex pattern to split a string into a list
* sub()- This looks for all substrings where the regex pattern matches, and replaces them with a different string
* subn()- Like sub(), this returns the new string and the number of replacements made

**Q.21. How would you display a file’s contents in reversed order?**

Let’s first get to the Desktop. We use the chdir() function/method form the os module for this.

>>> import os

>>> os.chdir('C:\\Users\\lifei\\Desktop')

The file we’ll use for this is Today.txt, and it has the following contents:

**OS, DBMS, DS, ADA**

**HTML, CSS, jQuery, JavaScript**

**Python, C++, Java**

**This sem’s subjects**

**Debugger**

**itertools**

**container**

Let’s read the contents into a list, and then call reversed() on it:

>>> **for** line **in** reversed(list(open('Today.txt'))):

print(line.rstrip())

**container**  
**itertools**  
**Debugger**

**This sem’s subjects**

**Python, C++, Java**  
**HTML, CSS, jQuery, JavaScript**  
**OS, DBMS, DS, ADA**

Without the rstrip(), we would get blank lines between the output.

**Q.22. Whenever you exit Python, is all memory de-allocated?**

The answer here is no. The modules with circular references to other objects, or to objects referenced from global namespaces, aren’t always freed on exiting Python.

Plus, it is impossible to de-allocate portions of memory reserved by the C library.

Python Programming Library Interview Questions

**Q.23. Can I dynamically load a module in Python?**

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

import importlib

module = importlib.import\_module('my\_package.my\_module')

**Q.24. What is speech\_recognition? Does this ship with Python by default?**

Speech\_recognition is a library for performing the task of recognizing speech with Python. This forms an integral part of AI. No, this does not ship with Python by default. We must download it from the PyPI and install it manually using pip.

**Score High in Interview – Don’t forget to practice**[**Speech Emotion Recognition Python Project with Source Code**](https://data-flair.training/blogs/python-mini-project-speech-emotion-recognition/)

**Q.25. How would you generate a random number in Python?**

To***generate a random number***, we import the function random() from the module random.

>>> from random import random

>>> random()

**0.7931961644126482**

Let’s call for help on this.

>>> help(random)

Help on built-in function random:

random(…) method of random.Random instance

random() -> x in the interval [0, 1).

This means that it will return a random number equal to or greater than 0, and less than 1.

We can also use the function randint(). It takes two arguments to indicate a range from which to return a random integer.

>>> from random import randint

>>> randint(2,7)

**6**

>>> randint(2,7)

**5**

>>> randint(2,7)

**7**

>>> randint(2,7)

**6**

>>> randint(2,7)

**2**

**Q.26. How will you locally save an image using its URL address?**

For this, we use the urllib module.

>>> import urllib.request

>>> urllib.request.urlretrieve('https://yt3.ggpht.com/a-/ACSszfE2YYTfvXCIVk4NjJdDfFSkSVrLBlalZwYsoA=s900-mo-c-c0xffffffff-rj-k-no','dataflair.jpg')

**(‘dataflair.jpg’, <http.client.HTTPMessage object at 0x02E90770>)**

You can then get to your Python’s location and confirm this.

Now, moving towards the next category that is Python Programming Interview Questions and Answers for Developers.

Python Programming Interview Questions for Developer

**Q.27.** Optionally, what statements can you put under a try-except block?

We have two of those:

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

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

>>> **try**:

print("Hello")

**except**:

print("Sorry")

**else**:

print("Oh then")

finally:

print("Bye")

**Hello**  
**Oh then**  
**Bye**

**Q.28.** Can you explain the filter(), map(), and reduce() functions?

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

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

**{5, 6}**

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

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

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

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

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

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

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

**3**

Let’s understand this:

**2-1=1**  
**3-1=2**  
**4-2=2**  
**5-2=3**

**Hence, 3.**

**Q.29.** How will you share global variables across modules?

To do this for modules within a single program, we create a special module, then import the config module in all modules of our application. This lets the module be global to all modules.

**Q.30. List some pdb commands.**

Some pdb commands include-

* <b> — Add breakpoint
* <c> — Resume execution
* <s> — Debug step by step
* <n> — Move to next line
* <l> — List source code
* <p> — Print an expression

**Q.31. What command do we use to debug a Python program?**

To start debugging, we first open the command prompt and get to the location the file is at.

Microsoft Windows [Version 10.0.16299.248]

(**c) 2017 Microsoft Corporation. All rights reserved.**

**C:\Users\lifei> cd Desktop**

**C:\Users\lifei\Desktop>**

**Then, we run the following command (for file try.py):**

**C:\Users\lifei\Desktop>python -m pdb try.py**

**> c:\users\lifei\desktop\try.py(1)<module>()**

**-> for i in range(5):**

**(Pdb)**

Then, we can start debugging.

**Q.32. What is Tkinter?**

Tkinter is a famous***Python library*** with which you can craft a GUI. It provides support for different GUI tools and widgets like buttons, labels, text boxes, radio buttons, and more. These tools and widgets have attributes like dimensions, colors, fonts, colors, and more.

You can also import the tkinter module.

>>> import tkinter

>>> top=tkinter.Tk()

This will create a new window for you:

This creates a window with the title ‘My Game’. You can position your widgets on this.

Python Data Science Interview Questions

**Q.33. What is the process to calculate percentiles with NumPy?**

Refer to the code below.

>>> import numpy as np

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

>>> p=np.percentile(arr,50)

>>> p

**3.0**

**Q.34. How would you create an empty NumPy array?**

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

**a. Option 1**

>>> import numpy

>>> numpy.array([])

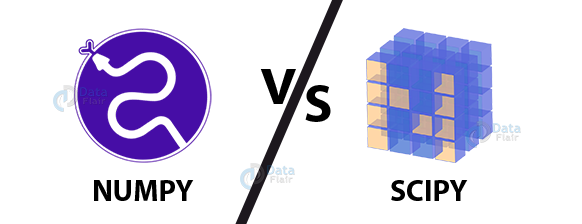
**array([], dtype=float64)**

**b. Option 2**

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

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

**Q.35. How is NumPy different from SciPy?**

[](https://data-flair.training/blogs/wp-content/uploads/sites/2/2019/09/numpy-vs-scipy-python-interview-questions.png)

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

* SciPy encompasses most new features
* NumPy does hold some linear algebra functions
* SciPy holds more fully-featured versions of the linear algebra modules and other numerical algorithms
* NumPy has compatibility as one of its essential goals; it attempts to retain all features supported by any of its predecessors
* NumPy holds the array data type and some basic operations: indexing, sorting, reshaping, and more

**Q.36. Explain different ways to create an empty NumPy array in Python.**

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

**First method-**

>>> import numpy

>>> numpy.array([])

**array([], dtype=float64)**

**Second method-**

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

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

**Q.37. What is monkey patching?**

Dynamically modifying a class or module at run-time.

>>> **class** A:

**def** func(self):

print("Hi")

>>> **def** monkey(self):

print "Hi, monkey"

>>> m.A.func = monkey

>>> a = m.A()

>>> a.func()

**Hi, monkey**

**Q.38. How will you find, in a string, the first word that rhymes with ‘cake’?**

For our purpose, we will use the function search(), and then use group() to get the output.

>>> import re

>>> rhyme=re.search('.ake','I would make a cake, but I hate to bake')

>>> rhyme.group()

**‘make’**

And as we know, the function search() stops at the first match. Hence, we have our first rhyme to ‘cake’.

**Q.39. Write a regular expression that will accept an email id. Use the re module.**

>>> import re

>>> e=re.search(r'[0-9a-zA-Z.]+@[a-zA-Z]+\.(com|co\.in)$','abc@gmail.com')

>>> e.group()

**‘abc@gmail.com’**

**Q.40. What is pickling and unpickling?**

To create portable serialized representations of Python objects, we have the module ‘pickle’. It accepts a Python object (remember, everything in Python is an object). It then converts it into a string representation and uses the dump() function to dump it into a file. We call this pickling. In contrast, retrieving objects from this stored string representation is termed ‘unpickling’.

**Q.41. What is the MRO in Python?**

MRO stands for Method Resolution Order. Talking of multiple inheritances, whenever we search for an attribute in a class, Python first searches in the current class. If found, its search is satiated. If not, it moves to the parent class. It follows an approach that is left-to-right and depth-first. It goes Child, Mother, Father, Object.

We can call this order a linearization of the class Child; the set of rules applied are the Method Resolution Order (MRO). We can borrow the \_\_mro\_\_ attribute or the mro() method to get this.

**Q.42. How do we make forms in Python?**

We use the cgi module for this; we borrow the FieldStorage class from it. It has the following attributes:

* form.name: Name of field.
* form.filename: Client-side filename for FTP transactions.
* form.value: Value of field as string.
* form.file: File object from which to read data.
* form.type: Content type.
* form.type\_options: Options of ‘content-type’ line of HTTP request, returned as dictionary.
* form.disposition: The field ‘content-disposition’.
* form.disposition\_options: Options for ‘content-disposition’.
* form.headers: All HTTP headers returned as dictionary.

Python OOPS Interview Questions and Answers

**Q.43. Is Python call-by-value or call-by-reference?**

Python is neither call-by-value, nor call-by-reference. It is call-by-object-reference. Almost everything is an object in Python. Take this example:

>>> item='milk'

>>> groceries=[]

>>> groceries.append(item)

>>> groceries

**[‘milk’]**

>>> items=groceries

>>> item='cheese'

>>> items.append(item)

>>> item

**‘cheese’**

>>> groceries, items

**([‘milk’, ‘cheese’], [‘milk’, ‘cheese’])**

item is ‘milk’ and groceries is an empty list. We append item to the list of groceries; groceries is now the list [‘milk’]. Now, we assign groceries to the name items. item is now ‘cheese’; let’s append it to items. So now, the name item holds the value ‘cheese’, and items is the list [‘milk’, ‘cheese’]. But now, even groceries is the list [‘milk’, ‘cheese’]. The list groceries got updated too. With Python, you focus on objects, and not on names. It is one list object we modified for both names ‘groceries’ and ‘items’.

Functions will modify values of mutable objects, but not immutable ones:

>>> a=1

>>> **def** up(num):

num+=1

>>> up(a)

>>> a

1

------------------

------------------

>>> a=[1]

>>> **def** up(list):

list[0]+=1

>>> up(a)

>>> a

**[2]**

Variables are names bound to an object, not aliases for actual memory locations.

**Q.44. Why do we need to overload operators?**

To compare two objects, we can overload ***operators in Python***. We understand 3>2. But what is orange>apple? Let’s compare apples and oranges now.

>>> **class** fruit:

**def** \_\_init\_\_(self,type,size):

self.type='fruit'

self.type=type

self.size=size

**def** \_\_gt\_\_(self,other):

**if** self.size>other.size:

**return** **True**

**return** **False**

>>> orange=fruit('orange',7)

>>> apple=fruit('apple',8)

>>> apple>orange

**True**

>>> orange>apple

**False**

**Q.45. Why do we need the \_\_init\_\_() function in classes? What else helps?**

\_\_init\_\_() is what we need to initialize a class when we initiate it. Let’s take an example.

>>> **class** orange:

**def** \_\_init\_\_(self):

self.color='orange'

self.type='citrus'

**def** setsize(self,size):

self.size=size

**def** show(self):

print(f"color: {self.color}, type: {self.type}, size: {self.size}")

>>> o=orange()

>>> o.setsize(7)

>>> o.show()

**color: orange, type: citrus, size: 7**

In this code, we see that it is necessary to pass the parameter ‘self’ to tell Python it has to work with this object.

**Q.46. Does Python support interfaces like Java does?**

No. However, Abstract Base Classes (ABCs) are a feature from the abc module that let us declare what methods subclasses should implement. Python supports this module since version 2.7.

**Q.47. What are accessors, mutators, and @property?**

What we call getters and setters in languages like Java, we term accessors and mutators in Python. In Java, if we have a user-defined class with a property ‘x’, we have methods like getX() and setX(). In Python, we have @property, which is syntactic sugar for property(). This lets us get and set variables without compromising on the conventions. For a detailed explanation on property, refer to Python property.

**Q.48. Consider multiple inheritances here. Suppose class C inherits from classes A and B as class C(A,B). Classes A and B both have their own versions of method func(). If we call func() from an object of class C, which version gets invoked?**

In our article on [**Multiple Inheritance in Python**](https://data-flair.training/blogs/python-multiple-inheritance/), we discussed the Method Resolution Order (MRO). C does not contain its own version of func(). Since the interpreter searches in a left-to-right fashion, it finds the method in A, and does not go to look for it in B.

**Q.49. What do you mean by overriding methods?**

Suppose class B inherits from class A. Both have the method sayhello()- to each, their own version. B overrides the sayhello() of class A. So, when we create an object of class B, it calls the version that class B has.

>>> **class** A:

**def** sayhello(self):

print("Hello, I'm A")

>>> **class** B(A):

**def** sayhello(self):

print("Hello, I'm B")

>>> a=A()

>>> b=B()

>>> a.sayhello()

**Hello, I’m A**

>>> b.sayhello()

**Hello, I’m B**

Python Interview Questions and Answers for Testers

**Q.50. How would you perform unit-testing on your Python code?**

For this purpose, we have the module unittest in Python. It has the following members:

* FunctionTestCase
* SkipTest
* TestCase
* TestLoader
* TestResult
* TestSuite
* TextTestResult
* TextTestRunner
* defaultTestLoader
* expectedFailure
* findTestCases
* getTestCaseNames
* installHandler
* main
* makeSuite
* registerResult
* removeHandler
* removeResult
* skip
* skipIf
* skipUnless

**Q.51. What is unit testing? How will you do it in Python?**

This is the first level of software testing, and it focuses on testing individual units of source code. This is to make sure everything works as expected. A unit is the smallest testable part of a software and usually comprises of a few inputs and a single output.

For this, we have the unittest framework. This was inspired by the JUnit framework and offers support for test automation, sharing of setup and shutdown code for tests, aggregation of tests into collections, and independence of tests from the framework. unittest is about text fixtures, text cases, test suites, and test runners.

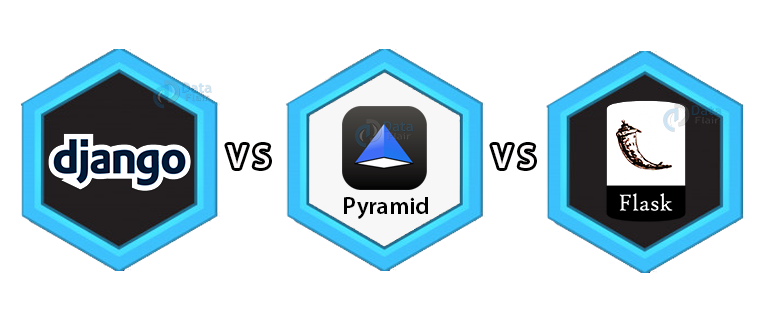
**Q.52. Can you use Selenium with Python?**

Selenium is an open-source, widely-used testing tool that is capable of automating tests on web browsers. Being open-source means there are no licensing costs. You can write test scripts with Python or other languages like Java, PHP, C#, Ruby, and Perl.

Practice some Python Programming Interview Questions based on its frameworks.

Python Framework Interview Questions

**Q.53. Differentiate between Django, Pyramid, and Flask.**

[](https://data-flair.training/blogs/wp-content/uploads/sites/2/2019/09/python-coding-interview-questions-django-vs-pyramid-vs-flask.png)

These are three major frameworks in Python. Here are the differences:

* We can also use Django for larger applications. It includes an ORM.
* Flask is a microframework for a small platform with simpler requirements. It is ready to use and you must use external libraries.
* The pyramid is for larger applications. It is flexible and you can choose the database, the URL structure, the templating style, and much more. It is also heavily configurable.

**Q.54. Explain the Inheritance Styles in Django.**

Talking on inheritance styles, we have three possible-

* **Abstract Base Classes-** For the parent class to hold information so we don’t have to type it out for each child model
* **Multi-table Inheritance-** For subclassing an existing model and letting each model have its own database
* **Proxy Models-** For modifying the model’s Python-level behavior without having to change its fields

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

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

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

**Q.56. What is Flask?**

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

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

Most Asked Python Coding Interview Questions and Answers

Here comes the most amazing part of Python programming interview questions part 2. Below are frequently asked Python Coding Interview Questions which you can’t miss. Check it out now –

**Q.57. How is multithreading achieved in Python?**

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

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

**Q.58. How is memory managed in Python?**

Python has a private heap space to hold all objects and data structures. Being programmers, we cannot access it; it is the interpreter that manages it. But with the core API, we can access some tools. The Python memory manager controls the allocation.

Additionally, an inbuilt garbage collector recycles all unused memory so it can make it available to the heap space.

**Q.59. What is tuple unpacking?**

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

>>> mytuple=3,4,5

>>> mytuple

**(3, 4, 5)**

This packs 3, 4, and 5 into mytuple.

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

>>> x,y,z=mytuple

>>> x+y+z

**12**

**Q.60. What is a namedtuple?**

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

>>> from collections import namedtuple

>>> result=namedtuple('result','Physics Chemistry Maths') #format

>>> Ayushi=result(Physics=86,Chemistry=95,Maths=86) #declaring the tuple

>>> Ayushi.Chemistry

**95**

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

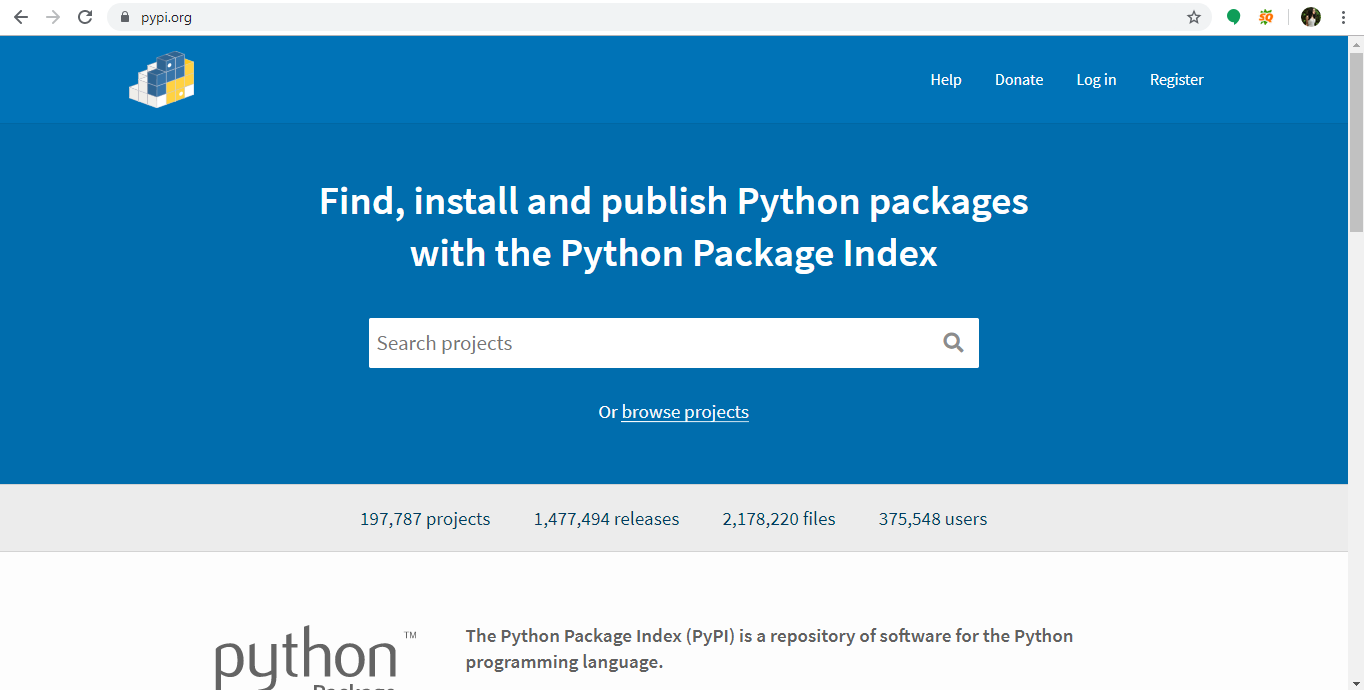
**Q.61. How do you create your own package in Python?**

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

To create a Python package of our own, we create a directory and create a file \_\_init\_\_.py in it. We leave it empty. Then, in that package, we create a module(s) with whatever code we want. For a detailed explanation with pictures, refer to [***Python Packages***](https://data-flair.training/blogs/python-packages/).

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

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

[](https://data-flair.training/blogs/wp-content/uploads/sites/2/2019/09/PyPI-python-coding-interview-question.png)

**Q.63. What will the following code output?**

>>> word=’abcdefghij’

>>> word[:3]+word[3:]

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

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

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

>>> **def** odds(n):

odd=[i **for** i **in** range(n+1) **if** i%2!=0]

**for** i **in** odd:

yield i

>>> **for** i **in** odds(8):

print(i)

**1**  
**3**  
**5**  
**7**

**Q.65. If a function does not have a return statement, is it valid?**

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

So, this was the last category of our Python programming interview questions and answers. Hope this helped you. If you have more python interview questions for experienced or freshers or any interview experience do share with us through comments.