

[Blog Home](#)[Courses](#)[Data Science](#)[Big Data](#)[Categories](#)[Interview Questions](#)[Quizzes](#)[Job Portal](#)[Write For Us](#)[Python – Introduction](#)[Python – Features](#)[Python – Pros and Cons](#)[Python – Career Opportunities](#)[Python – Tools](#)[Python – Data Science Tutorial](#)[Python – For Data Science](#)[Python – Data Science Installation](#)[Python 3.6 Features](#)[Python – Install on Windows](#)[Python – Syntax](#)[Python – Comments, Indentations and Statements](#)[Python – Assert Statements](#)[Python – Number Types](#)[Python – Random Number](#)[Python – Variables and Data Types](#)[Python – Variable Scope](#)[Python – Identifiers](#)[Python – Strings](#)[Python – Interpreter](#)[Python – Operators](#)[Python – Bitwise Operators](#)[Python – Comparison Operators](#)[Python – Operator Overloading](#)[Python – Ternary Operator](#)[Python – Operator Precedence](#)[Python – Namespaces](#)[Python – Decision Making](#)[Python – Loops in Python](#)[Python – Implement Switch Case](#)[Python – Functions](#)

Python Interview Questions and Answers (Latest)

 4

Top 35 Python Interview Questions and Answers (Latest)

BY [DATAFLAIR TEAM](#) ·

PUBLISHED MARCH 7, 2018 ·

UPDATED DECEMBER 11, 2018

1. Python Interview Questions and Answers

To land a job with Python as a fresher, you must be acquainted with the basics. Here, we discuss some basic ***Python Interview Questions and answers*** and some advanced Python Questions and answers to help you ace your interview. There are Python developer interview questions, Python Coding interview questions, data structure interview

Python Tutorials

Python – Introduction
Python – Features
Python – Pros and Cons
Python – Applications
Python – Career Opportunities
Python – Tools
Python – Data Science Tutorial
Python – For Data Science
Python – Data Science Installation
Python 3.6 Features
Python – Install on Windows
Python – Syntax
Python – Comments, Indentations and Statements
Python – Assert Statements
Python – Number Types
Python – Random Number
Python – Variables and Data Types
Python – Variable Scope
Python – Identifiers
Python – Strings
Python – Interpreter
Python – Operators
Python – Bitwise Operators
Python – Comparison Operators
Python – Operator Overloading
Python – Ternary Operator
Python – Operator Precedence
Python – Namespaces
Python – Decision Making
Python – Loops in Python
Python – Implement Switch Case
Python – Functions

questions as well as Python Scripting Interview questions. Delve into Python Programming Interview questions one by one.

So, let's begin our journey towards acing your next Python interview.

“PREPARE like you have never won and PERFORM like you have never lost.”



Top 35 Python Interview Questions and Answers (Latest)

Q.1. What are the key features of Python?

If it makes for an introductory language to programming, Python must mean something. These are its qualities:

1. Interpreted
2. Dynamically-typed
3. Object-oriented
4. Concise and simple
5. Free
6. Has a large community

[Follow this link to explore more features of Python Programming](#)

Q.2. Differentiate between deep and shallow copy.

A deep copy copies an object into another. This means that if you make a change to a copy of an object, it won't affect the original object. In Python,

Python Tutorials

[Python – Introduction](#)

[Python – Features](#)

[Python – Pros and Cons](#)

[Python – Applications](#)

[Python – Career Opportunities](#)

[Python – Tools](#)

[Python – Data Science Tutorial](#)

[Python – For Data Science](#)

[Python – Data Science Installation](#)

[Python 3.6 Features](#)

[Python – Install on Windows](#)

[Python – Syntax](#)

[Python – Comments, Indentations and Statements](#)

[Python – Assert Statements](#)

[Python – Number Types](#)

[Python – Random Number](#)

[Python – Variables and Data Types](#)

[Python – Variable Scope](#)

[Python – Identifiers](#)

[Python – Strings](#)

[Python – Interpreter](#)

[Python – Operators](#)

[Python – Bitwise Operators](#)

[Python – Comparison Operators](#)

[Python – Operator Overloading](#)

[Python – Ternary Operator](#)

[Python – Operator Precedence](#)

[Python – Namespaces](#)

[Python – Decision Making](#)

[Python – Loops in Python](#)

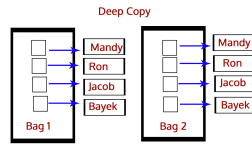
[Python – Implement Switch Case](#)

[Python – Functions](#)

we use the function `deepcopy()` for this, and we import the module `copy`.

We use it like:

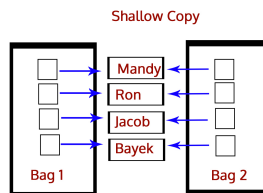
```
1. >>> import copy
2. >>> b=copy.deepcopy(a)
```



Deep Copy – Python Interview Questions and Answers

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

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



Shallow Copy – Python Interview Questions and Answers

Refer this link to know more about Deep & Shallow Copy in Python

Q.3. Differentiate between lists and tuples.

The major difference is that a list is mutable, but a tuple is immutable. Examples:

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

Python Tutorials

Python – Introduction
Python – Features
Python – Pros and Cons
Python – Applications
Python – Career Opportunities
Python – Tools
Python – Data Science Tutorial
Python – For Data Science
Python – Data Science Installation
Python 3.6 Features
Python – Install on Windows
Python – Syntax
Python – Comments, Indentations and Statements
Python – Assert Statements
Python – Number Types
Python – Random Number
Python – Variables and Data Types
Python – Variable Scope
Python – Identifiers
Python – Strings
Python – Interpreter
Python – Operators
Python – Bitwise Operators
Python – Comparison Operators
Python – Operator Overloading
Python – Ternary Operator
Python – Operator Precedence
Python – Namespaces
Python – Decision Making
Python – Loops in Python
Python – Implement Switch Case
Python – Functions

Traceback (most recent call last):

File "<pyshell#97>", line 1, in <module>

mytuple[1]=2

TypeError: 'tuple' object does not support item assignment

For more insight, refer to **[Tuples vs Lists](#)**.

2. Basic Python Interview Questions for Freshers

Q.4 to Q.20 are some basic Python Interview question for freshers, however Experience can also refer these questions to revise basic concepts.

Q.4. Explain the ternary operator in Python.

Unlike C++, we don't have `?:` in Python, but we have this:

```
[on true] if [expression]
else [on false]
```

If the expression is True, the statement under [on true] is executed. Else, that under [on false] is executed.

Below is how you would use it:

```
1. >>> a,b=2,3
2. >>> min=a if a<b
   else b
3. >>> min
```

Python Tutorials

[Python – Introduction](#)

[Python – Features](#)

[Python – Pros and Cons](#)

[Python – Applications](#)

[Python – Career Opportunities](#)

[Python – Tools](#)

[Python – Data Science Tutorial](#)

[Python – For Data Science](#)

[Python – Data Science Installation](#)

[Python 3.6 Features](#)

[Python – Install on Windows](#)

[Python – Syntax](#)

[Python – Comments, Indentations and Statements](#)

[Python – Assert Statements](#)

[Python – Number Types](#)

[Python – Random Number](#)

[Python – Variables and Data Types](#)

[Python – Variable Scope](#)

[Python – Identifiers](#)

[Python – Strings](#)

[Python – Interpreter](#)

[Python – Operators](#)

[Python – Bitwise Operators](#)

[Python – Comparison Operators](#)

[Python – Operator Overloading](#)

[Python – Ternary Operator](#)

[Python – Operator Precedence](#)

[Python – Namespaces](#)

[Python – Decision Making](#)

[Python – Loops in Python](#)

[Python – Implement Switch Case](#)

[Python – Functions](#)

```
1. >>> print("Hi") if
    a<b else
    print("Bye")
```

Hi

Are you familiar with all kinds of Python Operators?

Q.5. 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.6. Explain inheritance in Python.

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

Python Tutorials

Python – Introduction
Python – Features
Python – Pros and Cons
Python – Applications
Python – Career Opportunities
Python – Tools
Python – Data Science Tutorial
Python – For Data Science
Python – Data Science Installation
Python 3.6 Features
Python – Install on Windows
Python – Syntax
Python – Comments, Indentations and Statements
Python – Assert Statements
Python – Number Types
Python – Random Number
Python – Variables and Data Types
Python – Variable Scope
Python – Identifiers
Python – Strings
Python – Interpreter
Python – Operators
Python – Bitwise Operators
Python – Comparison Operators
Python – Operator Overloading
Python – Ternary Operator
Python – Operator Precedence
Python – Namespaces
Python – Decision Making
Python – Loops in Python
Python – Implement Switch Case
Python – Functions

Python Interview Questions – inheritance in Python.

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

- 1. Single Inheritance-** A class inherits from a single base class.
- 2. Multiple Inheritance-** A class inherits from multiple base classes.
- 3. Multilevel Inheritance-** A class inherits from a base class, which, in turn, inherits from another base class.
- 4. Hierarchical Inheritance-** Multiple classes inherit from a single base class.
- 5. Hybrid Inheritance-** Hybrid inheritance is a combination of two or more types of inheritance.

For more on inheritance, refer to Python Inheritance.

Q.7. What is Flask?

Python Flask, as we've previously discussed, is a

Python Tutorials

Python – Introduction
Python – Features
Python – Pros and Cons
Python – Applications
Python – Career Opportunities
Python – Tools
Python – Data Science Tutorial
Python – For Data Science
Python – Data Science Installation
Python 3.6 Features
Python – Install on Windows
Python – Syntax
Python – Comments, Indentations and Statements
Python – Assert Statements
Python – Number Types
Python – Random Number
Python – Variables and Data Types
Python – Variable Scope
Python – Identifiers
Python – Strings
Python – Interpreter
Python – Operators
Python – Bitwise Operators
Python – Comparison Operators
Python – Operator Overloading
Python – Ternary Operator
Python – Operator Precedence
Python – Namespaces
Python – Decision Making
Python – Loops in Python
Python – Implement Switch Case
Python – Functions

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`.

Q.8. 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.9. Explain help() and dir() functions in Python.

The help() function displays the documentation

Python Tutorials

Python – Introduction

Python – Features

Python – Pros and Cons

Python – Applications

Python – Career Opportunities

Python – Tools

Python – Data Science Tutorial

Python – For Data Science

Python – Data Science Installation

Python 3.6 Features

Python – Install on Windows

Python – Syntax

Python – Comments, Indentations and Statements

Python – Assert Statements

Python – Number Types

Python – Random Number

Python – Variables and Data Types

Python – Variable Scope

Python – Identifiers

Python – Strings

Python – Interpreter

Python – Operators

Python – Bitwise Operators

Python – Comparison Operators

Python – Operator Overloading

Python – Ternary Operator

Python – Operator Precedence

Python – Namespaces

Python – Decision Making

Python – Loops in Python

Python – Implement Switch Case

Python – Functions

string and help for its argument.

```
1. >>> import copy
2. >>> help(copy.copy)
```

Help on function copy in module copy:

copy(x)

Shallow copy operation on arbitrary **Python objects**.

See the module's `__doc__` string for more info.

The `dir()` function displays all the members of an object(any kind).

```
1. >>> dir(copy.copy)
```

```
[ '__annotations__',
  '__call__', '__class__',
  '__closure__', '__code__',
  '__defaults__',
  '__delattr__', '__dict__',
  '__dir__', '__doc__',
  '__eq__', '__format__',
  '__ge__', '__get__',
  '__getattr__',
  '__globals__', '__gt__',
  '__hash__', '__init__',
  '__init_subclass__',
  '__kwdefaults__', '__le__',
  '__lt__', '__module__',
  '__name__', '__ne__',
  '__new__',
  '__qualname__',
  '__reduce__',
  '__reduce_ex__',
  '__repr__', '__setattr__',
  '__sizeof__', '__str__',
  '__subclasshook__']
```

Let's explore more Functions in Python

Q.10. Whenever you exit Python, is all memory de-allocated?

Python Tutorials

Python – Introduction
Python – Features
Python – Pros and Cons
Python – Applications
Python – Career Opportunities
Python – Tools
Python – Data Science Tutorial
Python – For Data Science
Python – Data Science Installation
Python 3.6 Features
Python – Install on Windows
Python – Syntax
Python – Comments, Indentations and Statements
Python – Assert Statements
Python – Number Types
Python – Random Number
Python – Variables and Data Types
Python – Variable Scope
Python – Identifiers
Python – Strings
Python – Interpreter
Python – Operators
Python – Bitwise Operators
Python – Comparison Operators
Python – Operator Overloading
Python – Ternary Operator
Python – Operator Precedence
Python – Namespaces
Python – Decision Making
Python – Loops in Python
Python – Implement Switch Case
Python – Functions

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.

Q.11. What is monkey patching?

Dynamically modifying a class or module at run-time.

```

1. >>> class A:
2.     def func(self):
3.         print("Hi")
4. >>> def
5.     monkey(self):
6.         print "Hi,
7.         monkey"
8. >>> m.A.func =
9.     monkey
10. >>> a = m.A()
11. >>> a.func()
```

Hi, monkey

Q.12. What is a dictionary in Python?

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

```

1. >>> roots=
2.     {25:5,16:4,9:3,4:2,
3.     1:1}
4. >>> type(roots)
```

<class 'dict'>

```

1. >>> roots[9]
```

3

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

Python Tutorials

[Python – Introduction](#)

[Python – Features](#)

[Python – Pros and Cons](#)

[Python – Applications](#)

[Python – Career Opportunities](#)

[Python – Tools](#)

[Python – Data Science Tutorial](#)

[Python – For Data Science](#)

[Python – Data Science Installation](#)

[Python 3.6 Features](#)

[Python – Install on Windows](#)

[Python – Syntax](#)

[Python – Comments, Indentations and Statements](#)

[Python – Assert Statements](#)

[Python – Number Types](#)

[Python – Random Number](#)

[Python – Variables and Data Types](#)

[Python – Variable Scope](#)

[Python – Identifiers](#)

[Python – Strings](#)

[Python – Interpreter](#)

[Python – Operators](#)

[Python – Bitwise Operators](#)

[Python – Comparison Operators](#)

[Python – Operator Overloading](#)

[Python – Ternary Operator](#)

[Python – Operator Precedence](#)

[Python – Namespaces](#)

[Python – Decision Making](#)

[Python – Loops in Python](#)

[Python – Implement Switch Case](#)

[Python – Functions](#)

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

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

Q.13. 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.

```
1. >>> def
    func(*args):
2.         for i in args:
3.             print(i)
4. >>> 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.

```
1. >>> def
    func(**kwargs):
2.         for i in
    kwargs:
3.
    print(i, kwargs[i])
4. >>>
    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.

Any doubt yet in Basic Python Interview Questions

Python Tutorials

Python – Introduction
Python – Features
Python – Pros and Cons
Python – Applications
Python – Career Opportunities
Python – Tools
Python – Data Science Tutorial
Python – For Data Science
Python – Data Science Installation
Python 3.6 Features
Python – Install on Windows
Python – Syntax
Python – Comments, Indentations and Statements
Python – Assert Statements
Python – Number Types
Python – Random Number
Python – Variables and Data Types
Python – Variable Scope
Python – Identifiers
Python – Strings
Python – Interpreter
Python – Operators
Python – Bitwise Operators
Python – Comparison Operators
Python – Operator Overloading
Python – Ternary Operator
Python – Operator Precedence
Python – Namespaces
Python – Decision Making
Python – Loops in Python
Python – Implement Switch Case
Python – Functions

and answers for Freshers?
Please ask in Comments.

Q.14. Write Python logic to count the number of capital letters in a file.

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

26

Q.15. What are negative indices?

Let's take a list for this.

```
1. >>> mylist=[0,1,2,3,4,5,6,7,8]
```

A negative index, unlike a positive one, begins searching from the right.

```
1. >>> mylist[-3]
```

6

This also helps with slicing from the back:

```
1. >>> mylist[-6:-1]
```

[3, 4, 5, 6, 7]

Q.16. How would you randomize the contents of a list in-place?

For this, we'll import the function shuffle() from the module random.

```
1. >>> from random
2. >>> import shuffle
3. >>> shuffle(mylist)
```

Python Tutorials

Python – Introduction
Python – Features
Python – Pros and Cons
Python – Applications
Python – Career Opportunities
Python – Tools
Python – Data Science Tutorial
Python – For Data Science
Python – Data Science Installation
Python 3.6 Features
Python – Install on Windows
Python – Syntax
Python – Comments, Indentations and Statements
Python – Assert Statements
Python – Number Types
Python – Random Number
Python – Variables and Data Types
Python – Variable Scope
Python – Identifiers
Python – Strings
Python – Interpreter
Python – Operators
Python – Bitwise Operators
Python – Comparison Operators
Python – Operator Overloading
Python – Ternary Operator
Python – Operator Precedence
Python – Namespaces
Python – Decision Making
Python – Loops in Python
Python – Implement Switch Case
Python – Functions

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

Q.17. Explain join() and split() in Python.

join() lets us join characters from a string together by a character we specify.

```
1. >>>
    ','.join('12345')
```

‘1,2,3,4,5’

split() lets us split a string around the character we specify.

```
1. >>>
    '1,2,3,4,5'.split(',')
    ['1', '2', '3', '4', '5']
```

['1', '2', '3', '4', '5']

Q.18. Is Python case-sensitive?

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

```
1. >>> myname= 'Ayushi'
2. >>> Myname
```

Traceback (most recent call last):

File “<pyshell#3>”, line 1, in <module>

Myname

NameError: name

‘Myname’ is not defined

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

Let’s Discuss Errors and Exceptions in Python

Programming

Python Tutorials

Python – Introduction
Python – Features
Python – Pros and Cons
Python – Applications
Python – Career Opportunities
Python – Tools
Python – Data Science Tutorial
Python – For Data Science
Python – Data Science Installation
Python 3.6 Features
Python – Install on Windows
Python – Syntax
Python – Comments, Indentations and Statements
Python – Assert Statements
Python – Number Types
Python – Random Number
Python – Variables and Data Types
Python – Variable Scope
Python – Identifiers
Python – Strings
Python – Interpreter
Python – Operators
Python – Bitwise Operators
Python – Comparison Operators
Python – Operator Overloading
Python – Ternary Operator
Python – Operator Precedence
Python – Namespaces
Python – Decision Making
Python – Loops in Python
Python – Implement Switch Case
Python – Functions

Q.19. How long can an identifier be in Python?

In Python, an identifier can be of any length. Apart from that, there are certain rules we must follow to name one:

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

and	def	False	import
as	del	finally	in
assert	elif	for	is
break	else	from	lambda
class	except	global	None
continue	exec	if	nonloca

Q.20. How do you remove the leading whitespace in a string?

Leading whitespace in a string is the whitespace in a string before the first non-whitespace character. To remove it from a string, we use the method `lstrip()`.

Python Tutorials

[Python – Introduction](#)
[Python – Features](#)
[Python – Pros and Cons](#)
[Python – Applications](#)
[Python – Career Opportunities](#)
[Python – Tools](#)
[Python – Data Science Tutorial](#)
[Python – For Data Science](#)
[Python – Data Science Installation](#)
[Python 3.6 Features](#)
[Python – Install on Windows](#)
[Python – Syntax](#)
[Python – Comments, Indentations and Statements](#)
[Python – Assert Statements](#)
[Python – Number Types](#)
[Python – Random Number](#)
[Python – Variables and Data Types](#)
[Python – Variable Scope](#)
[Python – Identifiers](#)
[Python – Strings](#)
[Python – Interpreter](#)
[Python – Operators](#)
[Python – Bitwise Operators](#)
[Python – Comparison Operators](#)
[Python – Operator Overloading](#)
[Python – Ternary Operator](#)
[Python – Operator Precedence](#)
[Python – Namespaces](#)
[Python – Decision Making](#)
[Python – Loops in Python](#)
[Python – Implement Switch Case](#)
[Python – Functions](#)

```
1. >>> ' Ayushi
    >>> .lstrip()
```

‘Ayushi ‘

As you can see, this string had both leading and trailing whitespaces. `lstrip()` stripped the string of the leading whitespace. If we want to strip the trailing whitespace instead, we use `rstrip()`.

```
1. >>> ' Ayushi
    >>> .rstrip()
```

‘ Ayushi’

These were basic Python Interview Questions and answers for Freshers.

3. Advanced Python Interview Questions and Answers for Experienced

Q. 21 to Q. 35 are some Advanced Python Interview questions for Experience along with their answers and Examples.

Q.21. How would you convert a string into lowercase?

We use the `lower()` method for this.

```
1. >>> 'AyuShi'.lower()
```

‘ayushi’

Python Tutorials

Python – Introduction
Python – Features
Python – Pros and Cons
Python – Applications
Python – Career Opportunities
Python – Tools
Python – Data Science Tutorial
Python – For Data Science
Python – Data Science Installation
Python 3.6 Features
Python – Install on Windows
Python – Syntax
Python – Comments, Indentations and Statements
Python – Assert Statements
Python – Number Types
Python – Random Number
Python – Variables and Data Types
Python – Variable Scope
Python – Identifiers
Python – Strings
Python – Interpreter
Python – Operators
Python – Bitwise Operators
Python – Comparison Operators
Python – Operator Overloading
Python – Ternary Operator
Python – Operator Precedence
Python – Namespaces
Python – Decision Making
Python – Loops in Python
Python – Implement Switch Case
Python – Functions

To convert it into uppercase, then, we use upper().

```
1. >>> 'AyuShi'.upper()
```

‘AYUSHI’

Also, to check if a string is in all uppercase or all lowercase, we use the methods isupper() and islower().

```
1. >>> 'AyuShi'.isupper()
```

False

```
1. >>> 'AYUSHI'.isupper()
```

True

```
1. >>> 'ayushi'.islower()
```

True

```
1. >>> '@yu$hi'.islower()
```

True

```
1. >>> '@YU$HI'.isupper()
```

True

So, characters like @ and \$ will suffice for both cases.

Also, istitle() will tell us if a string is in title case.

```
1. >>> 'The Corpse Bride'.istitle()
```

True

[Refer this link to explore more Python Strings with functions](#)

Python Tutorials[Python – Introduction](#)[Python – Features](#)[Python – Pros and Cons](#)[Python – Applications](#)[Python – Career Opportunities](#)[Python – Tools](#)[Python – Data Science Tutorial](#)[Python – For Data Science](#)[Python – Data Science Installation](#)[Python 3.6 Features](#)[Python – Install on Windows](#)[Python – Syntax](#)[Python – Comments, Indentations and Statements](#)[Python – Assert Statements](#)[Python – Number Types](#)[Python – Random Number](#)[Python – Variables and Data Types](#)[Python – Variable Scope](#)[Python – Identifiers](#)[Python – Strings](#)[Python – Interpreter](#)[Python – Operators](#)[Python – Bitwise Operators](#)[Python – Comparison Operators](#)[Python – Operator Overloading](#)[Python – Ternary Operator](#)[Python – Operator Precedence](#)[Python – Namespaces](#)[Python – Decision Making](#)[Python – Loops in Python](#)[Python – Implement Switch Case](#)[Python – Functions](#)**Q.22. What is the pass statement in Python?**

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

```
1. >>> def
   func(*args):
2.         pass
3. >>>
```

Similarly, the [break statement](#) breaks out of a loop.

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

1

2

Finally, the continue statement skips to the next iteration.

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

1

2

4

5

6

Q.23. What is a closure in Python?

A closure 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.

Python Tutorials

Python – Introduction

Python – Features

Python – Pros and Cons

Python – Applications

Python – Career Opportunities

Python – Tools

Python – Data Science Tutorial

Python – For Data Science

Python – Data Science Installation

Python 3.6 Features

Python – Install on Windows

Python – Syntax

Python – Comments, Indentations and Statements

Python – Assert Statements

Python – Number Types

Python – Random Number

Python – Variables and Data Types

Python – Variable Scope

Python – Identifiers

Python – Strings

Python – Interpreter

Python – Operators

Python – Bitwise Operators

Python – Comparison Operators

Python – Operator Overloading

Python – Ternary Operator

Python – Operator Precedence

Python – Namespaces

Python – Decision Making

Python – Loops in Python

Python – Implement Switch Case

Python – Functions

```
1. >>> def A(x):
2.     def B():
3.         print(x)
4.     return B
5. >>> A(7)()
```

7

For more depth on closures, refer to [Closures in Python](#).

Q.24. Explain the //, %, and ** operators in Python.

The // operator performs floor division. It will return the integer part of the result on division.

```
1. >>> 7//2
```

3

Normal division would return 3.5 here.

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

```
1. >>> 2**10
```

1024

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

```
1. >>> 13%7
```

6

```
1. >>> 3.5%1.5
```

0.5

Any Doubt yet in Advanced Python Interview Questions and Answers for

Python Tutorials

Python – Introduction
Python – Features
Python – Pros and Cons
Python – Applications
Python – Career Opportunities
Python – Tools
Python – Data Science Tutorial
Python – For Data Science
Python – Data Science Installation
Python 3.6 Features
Python – Install on Windows
Python – Syntax
Python – Comments, Indentations and Statements
Python – Assert Statements
Python – Number Types
Python – Random Number
Python – Variables and Data Types
Python – Variable Scope
Python – Identifiers
Python – Strings
Python – Interpreter
Python – Operators
Python – Bitwise Operators
Python – Comparison Operators
Python – Operator Overloading
Python – Ternary Operator
Python – Operator Precedence
Python – Namespaces
Python – Decision Making
Python – Loops in Python
Python – Implement Switch Case
Python – Functions

Experienced? Please
Comment.

Q.24. How many kinds of operators do we have in Python? Explain arithmetic operators.

This type of Python Interview Questions and Answers can decide your knowledge in Python. Answer the Python Interview Questions with some good Examples.

Here in Python, we have 7 kinds of operators: arithmetic, relational, assignment, logical, membership, identity, and bitwise.

We have seven arithmetic operators. These allow us to perform arithmetic operations on values:

1. Addition (+) This adds two values.

```
1. >>> 7+8
```

15

2. Subtraction (-) This subtracts the second value from the first.

```
1. >>> 7-8
```

-1

3. Multiplication (*) This multiplies two numbers.

```
1. >>> 7*8
```

56

Python Tutorials[Python – Introduction](#)[Python – Features](#)[Python – Pros and Cons](#)[Python – Applications](#)[Python – Career Opportunities](#)[Python – Tools](#)[Python – Data Science Tutorial](#)[Python – For Data Science](#)[Python – Data Science Installation](#)[Python 3.6 Features](#)[Python – Install on Windows](#)[Python – Syntax](#)[Python – Comments, Indentations and Statements](#)[Python – Assert Statements](#)[Python – Number Types](#)[Python – Random Number](#)[Python – Variables and Data Types](#)[Python – Variable Scope](#)[Python – Identifiers](#)[Python – Strings](#)[Python – Interpreter](#)[Python – Operators](#)[Python – Bitwise Operators](#)[Python – Comparison Operators](#)[Python – Operator Overloading](#)[Python – Ternary Operator](#)[Python – Operator Precedence](#)[Python – Namespaces](#)[Python – Decision Making](#)[Python – Loops in Python](#)[Python – Implement Switch Case](#)[Python – Functions](#)

4. Division (/) This

divides the first value
by the second.1. `>>> 7/8`

0.875

1. `>>> 1/1`

1.0

For floor division, modulus,
and exponentiation refer to
the previous question.**Q.25. Explain relational
operators in Python.**Relational operators
compare values.

1. Less than (<) If the
value on the left is
lesser, it returns True.

1. `>>> 'hi' < 'Hi'`

False

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

1. `>>> 1.1+2.2>3.3`

True

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

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

1. `>>> 3.0<=3`

True

Python Tutorials

[Python – Introduction](#)

[Python – Features](#)

[Python – Pros and Cons](#)

[Python – Applications](#)

[Python – Career Opportunities](#)

[Python – Tools](#)

[Python – Data Science Tutorial](#)

[Python – For Data Science](#)

[Python – Data Science Installation](#)

[Python 3.6 Features](#)

[Python – Install on Windows](#)

[Python – Syntax](#)

[Python – Comments, Indentations and Statements](#)

[Python – Assert Statements](#)

[Python – Number Types](#)

[Python – Random Number](#)

[Python – Variables and Data Types](#)

[Python – Variable Scope](#)

[Python – Identifiers](#)

[Python – Strings](#)

[Python – Interpreter](#)

[Python – Operators](#)

[Python – Bitwise Operators](#)

[Python – Comparison Operators](#)

[Python – Operator Overloading](#)

[Python – Ternary Operator](#)

[Python – Operator Precedence](#)

[Python – Namespaces](#)

[Python – Decision Making](#)

[Python – Loops in Python](#)

[Python – Implement Switch Case](#)

[Python – Functions](#)

4. Greater than or equal to (\geq) If the value on the left is greater than or equal to, it returns True.

```
1. >>> True>=False
```

True

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

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

True

6. Not equal to (\neq) If the two values are unequal, it returns True.

```
1. >>> True!=0.1
```

True

```
1. >>> False!=0.1
```

True

Q.26. What are assignment operators in Python?

This one is an Important Interview question in Python Interview.

We can combine all arithmetic operators with the assignment symbol.

```
1. >>> a=7
2. >>> a+=1
3. >>> a
```

8

```
1. >>> a-=1
2. >>> a
```

7

```
1. >>> a*=2
```

Python Tutorials

Python – Introduction

Python – Features

Python – Pros and Cons

Python – Applications

Python – Career Opportunities

Python – Tools

Python – Data Science Tutorial

Python – For Data Science

Python – Data Science Installation

Python 3.6 Features

Python – Install on Windows

Python – Syntax

Python – Comments, Indentations and Statements

Python – Assert Statements

Python – Number Types

Python – Random Number

Python – Variables and Data Types

Python – Variable Scope

Python – Identifiers

Python – Strings

Python – Interpreter

Python – Operators

Python – Bitwise Operators

Python – Comparison Operators

Python – Operator Overloading

Python – Ternary Operator

Python – Operator Precedence

Python – Namespaces

Python – Decision Making

Python – Loops in Python

Python – Implement Switch Case

Python – Functions

```
2. >>> a
```

14

```
1. >>> a/=2
2. >>> a
```

7.0

```
1. >>> a**=2
2. >>> a
```

49.0

```
1. >>> a//=3
2. >>> a
```

16.0

```
1. >>> a%=4
2. >>> a
```

0.0

Q.27. Explain logical operators in Python.

We have three logical operators- and, or, not.

```
1. >>> False and True
```

False

```
1. >>> 7<7 or True
```

True

```
1. >>> not 2==2
```

False

Q.28. What are membership operators?

With the operators 'in' and 'not in', we can confirm if a value is a member in another.

```
1. >>> 'me' in 'disappointment'
```

True

Python Tutorials

Python – Introduction

Python – Features

Python – Pros and Cons

Python – Applications

Python – Career Opportunities

Python – Tools

Python – Data Science Tutorial

Python – For Data Science

Python – Data Science Installation

Python 3.6 Features

Python – Install on Windows

Python – Syntax

Python – Comments, Indentations and Statements

Python – Assert Statements

Python – Number Types

Python – Random Number

Python – Variables and Data Types

Python – Variable Scope

Python – Identifiers

Python – Strings

Python – Interpreter

Python – Operators

Python – Bitwise Operators

Python – Comparison Operators

Python – Operator Overloading

Python – Ternary Operator

Python – Operator Precedence

Python – Namespaces

Python – Decision Making

Python – Loops in Python

Python – Implement Switch Case

Python – Functions

```
1. >>> 'us' not in
'disappointment'
```

True

Q.29. Explain identity operators in Python.

This is one of the very commonly asked Python Interview Questions and answers it with examples.

The operators ‘is’ and ‘is not’ tell us if two values have the same identity.

```
1. >>> 10 is '10'
```

False

```
1. >>> True is not
False
```

True

Q.30. Finally, tell us about bitwise operators in Python.

These operate on values bit by bit.

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

```
1. >>> 0b110 & 0b010
```

2

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

```
1. >>> 3|2
```

3

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

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

Python Tutorials

Python – Introduction

Python – Features

Python – Pros and Cons

Python – Applications

Python – Career Opportunities

Python – Tools

Python – Data Science Tutorial

Python – For Data Science

Python – Data Science Installation

Python 3.6 Features

Python – Install on Windows

Python – Syntax

Python – Comments, Indentations and Statements

Python – Assert Statements

Python – Number Types

Python – Random Number

Python – Variables and Data Types

Python – Variable Scope

Python – Identifiers

Python – Strings

Python – Interpreter

Python – Operators

Python – Bitwise Operators

Python – Comparison Operators

Python – Operator Overloading

Python – Ternary Operator

Python – Operator Precedence

Python – Namespaces

Python – Decision Making

Python – Loops in Python

Python – Implement Switch Case

Python – Functions

1

4. Binary One's

Complement (~) This returns the one's complement of a value.

```
1. >>> ~2
```

-3

5. Binary Left-Shift (<<)

This shifts the bits to the left by the specified amount.

```
1. >>> 1<<2
```

4

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

6. Binary Right-Shift (>>)

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

1

For more insight on operators, refer to [Operators in Python](#).

Q.31. How would you work with numbers other than those in the decimal number system?

With Python, it is possible to type numbers in binary, octal, and hexadecimal.

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

```
1. >>> int(0b1010)
```

10

Python Tutorials

[Python – Introduction](#)

[Python – Features](#)

[Python – Pros and Cons](#)

[Python – Applications](#)

[Python – Career Opportunities](#)

[Python – Tools](#)

[Python – Data Science Tutorial](#)

[Python – For Data Science](#)

[Python – Data Science Installation](#)

[Python 3.6 Features](#)

[Python – Install on Windows](#)

[Python – Syntax](#)

[Python – Comments, Indentations and Statements](#)

[Python – Assert Statements](#)

[Python – Number Types](#)

[Python – Random Number](#)

[Python – Variables and Data Types](#)

[Python – Variable Scope](#)

[Python – Identifiers](#)

[Python – Strings](#)

[Python – Interpreter](#)

[Python – Operators](#)

[Python – Bitwise Operators](#)

[Python – Comparison Operators](#)

[Python – Operator Overloading](#)

[Python – Ternary Operator](#)

[Python – Operator Precedence](#)

[Python – Namespaces](#)

[Python – Decision Making](#)

[Python – Loops in Python](#)

[Python – Implement Switch Case](#)

[Python – Functions](#)

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

```
1. >>> bin(0xf)
```

`'0b1111'`

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

```
1. >>> oct(8)
```

`'0o10'`

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

```
1. >>> hex(16)
```

`'0x10'`

```
1. >>> hex(15)
```

`'0xf'`

Q.32. How do you get a list of all the keys in a dictionary?

Be specific in these type of Python Interview Questions and Answers.

For this, we use the function `keys()`.

```
1. >>> mydict={ 'a':1, 'b':2, 'c':3, 'e':5}
2. >>> mydict.keys()
```

`dict_keys(['a', 'b', 'c', 'e'])`

Q.33. Why are identifier names with a leading underscore disparaged?

Since Python does not have a concept of private

Python Tutorials

Python – Introduction
Python – Features
Python – Pros and Cons
Python – Applications
Python – Career Opportunities
Python – Tools
Python – Data Science Tutorial
Python – For Data Science
Python – Data Science Installation
Python 3.6 Features
Python – Install on Windows
Python – Syntax
Python – Comments, Indentations and Statements
Python – Assert Statements
Python – Number Types
Python – Random Number
Python – Variables and Data Types
Python – Variable Scope
Python – Identifiers
Python – Strings
Python – Interpreter
Python – Operators
Python – Bitwise Operators
Python – Comparison Operators
Python – Operator Overloading
Python – Ternary Operator
Python – Operator Precedence
Python – Namespaces
Python – Decision Making
Python – Loops in Python
Python – Implement Switch Case
Python – Functions

variables, it is a convention to use leading underscores to declare a variable private. This is why we mustn't do that to variables we do not want to make private.

Q.34. How can you declare multiple assignments in one statement?

There are two ways to do this:

```
1. >>> a,b,c=3,4,5
   #This assigns 3, 4,
   #and 5 to a, b, and
   #c respectively
2. >>> a=b=c=3 #This
   #assigns 3 to a, b,
   #and c
```

Q.35. What is tuple unpacking?

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

```
1. >>> mytuple=3,4,5
2. >>> 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.

```
1. >>> x,y,z=mytuple
2. >>> x+y+z
```

12

[Follow this link to know about Data Structures in Python – Lists, Tuples, Sets, Dictionaries](#)

Python Tutorials

[Python – Introduction](#)[Python – Features](#)[Python – Pros and Cons](#)[Python – Applications](#)[Python – Career Opportunities](#)[Python – Tools](#)[Python – Data Science Tutorial](#)[Python – For Data Science](#)[Python – Data Science Installation](#)[Python 3.6 Features](#)[Python – Install on Windows](#)[Python – Syntax](#)[Python – Comments, Indentations and Statements](#)[Python – Assert Statements](#)[Python – Number Types](#)[Python – Random Number](#)[Python – Variables and Data Types](#)[Python – Variable Scope](#)[Python – Identifiers](#)[Python – Strings](#)[Python – Interpreter](#)[Python – Operators](#)[Python – Bitwise Operators](#)[Python – Comparison Operators](#)[Python – Operator Overloading](#)[Python – Ternary Operator](#)[Python – Operator Precedence](#)[Python – Namespaces](#)[Python – Decision Making](#)[Python – Loops in Python](#)[Python – Implement Switch Case](#)[Python – Functions](#)

These were the Advanced Python Interview Questions and Answers for Experiences. Freshers may Also Refer the Python Interview Questions for advanced knowledge.

4. Conclusion

These are some of the important [Python](#) Interview questions and answers you should take a look at before you appear for an interview. We will be back with more. Till then, feel free to add a question.

This was all About Python Interview Questions and Answers.

Also, see:

- [Top 26 Python Programming Interview Questions](#)
- [The Tremendous Python Career Opportunities](#)

Tags: [python advanced interview questions](#)

[python coding interview questions](#)

[python developer interview questions](#)

[python interview questions](#)

[python interview questions and answers](#)

[python interview questions for experienced](#)

[python interview questions for freshers](#)

[python programming questions](#)

[python questions](#)



4 RESPONSES

Python Tutorials


[Python – Introduction](#)[Python – Features](#)[Python – Pros and Cons](#)[Python – Applications](#)[Python – Career Opportunities](#)[Python – Tools](#)[Python – Data Science Tutorial](#)[Python – For Data Science](#)[Python – Data Science Installation](#)[Python 3.6 Features](#)[Python – Install on Windows](#)[Python – Syntax](#)[Python – Comments, Indentations and Statements](#)[Python – Assert Statements](#)[Python – Number Types](#)[Python – Random Number](#)[Python – Variables and Data Types](#)[Python – Variable Scope](#)[Python – Identifiers](#)[Python – Strings](#)[Python – Interpreter](#)[Python – Operators](#)[Python – Bitwise Operators](#)[Python – Comparison Operators](#)[Python – Operator Overloading](#)[Python – Ternary Operator](#)[Python – Operator Precedence](#)[Python – Namespaces](#)[Python – Decision Making](#)[Python – Loops in Python](#)[Python – Implement Switch Case](#)[Python – Functions](#)

nidhi  July 31, 2018 at 5:46 am

I have faced a number of difficult interview questions like what is your career goal etc. Some of these questions require prior preparation. This I came to know after reading your blog. Thanks for the practical tips to answer such questions.

Reply

Data Flair

 August 24, 2018 at 6:04 am


Hi Nidhi,

We are glad we could make a difference to you. Luckily, we have prepared many more Python Interview questions for loyal readers like you looking to find all the important questions in one place. We have also curated questions for you that deal with issues you'd normally run into with Python-questions that could be part of an interview.

Why don't you check [Best Python Interview Questions](#)

Reply

Damian Oguche

 November 28, 2018 at 10:01 pm

Your website has has given me serious insight about what Python is all about. The content of your tutorials are great. I am encouraged to continue my journey into the wonderful world of Python programming.

Reply

DataFlair Team



🕒 December 1, 2018 at 4:43 pm

Hello Damian,
Very glad to see your review on Python Interview Questions. Our readers feedback always motivate us to publish more articles. If you are planning for Python Career refer our article [Python Career Opportunities](#).
Regards,
Data-Flair

Reply

Python Tutorials

- Python – Introduction
- Python – Features
- Python – Pros and Cons
- Python – Applications
- Python – Career Opportunities
- Python – Tools
- Python – Data Science Tutorial
- Python – For Data Science
- Python – Data Science Installation
- Python 3.6 Features
- Python – Install on Windows
- Python – Syntax
- Python – Comments, Indentations and Statements
- Python – Assert Statements
- Python – Number Types
- Python – Random Number
- Python – Variables and Data Types
- Python – Variable Scope
- Python – Identifiers
- Python – Strings
- Python – Interpreter
- Python – Operators
- Python – Bitwise Operators
- Python – Comparison Operators
- Python – Operator Overloading
- Python – Ternary Operator
- Python – Operator Precedence
- Python – Namespaces
- Python – Decision Making
- Python – Loops in Python
- Python – Implement Switch Case
- Python – Functions

LEAVE A REPLY

Comment

Name *

Email *

This site is protected by reCAPTCHA and the Google [Privacy Policy](#) and [Terms of Service](#) apply.

Post Comment

Popular Courses

Hadoop + Spark
Course

Popular Tutorials

Hadoop Tutorials
Spark Tutorials

Popular Tutorials

Data Science
Tutorials

Big Data Hadoop
Course
Spark Scala Course
Apache Flink Course

Flink Tutorials
Tableau Tutorials
Power BI Tutorials
QlikView Tutorials

Machine Learning
Tutorials
Python Tutorials
R Tutorials
SAS Tutorials
SQL Tutorials



DataFlair © 2019. All Rights Reserved.

Learn today, Lead tomorrow



Python – Install on Windows

Python – Syntax

Python – Comments, Indentations and Statements

Python – Assert Statements

Python – Number Types

Python – Random Number

Python – Variables and Data Types

Python – Variable Scope

Python – Identifiers

Python – Strings

Python – Interpreter

Python – Operators

Python – Bitwise Operators

Python – Comparison Operators

Python – Operator Overloading

Python – Ternary Operator

Python – Operator Precedence

Python – Namespaces

Python – Decision Making

Python – Loops in Python

Python – Implement Switch Case

Python – Functions
