Python Interview Questions

Complete & Continue

Enable fullscreen

Question 1: What Is Python?

Python is an open-source interpreted language (like PHP and Ruby) with automatic memory management, exceptions, modules, objects, and threads.The benefits of Python include its simplicity, portability, extensibility, and built-in data structures. As it's open-source, there's also a massive community backing it. Python is best suited for object-oriented programming. It's dynamically typed, so you won't have to state the types of variables when you declare them. Unlike C++, it doesn't have access to public or private specifiers.Python's functions are first-class objects that make difficult tasks simple. While you can write code quickly, running it will be comparatively slower than other compiled programming languages.

Question 2: What is Python really?

You can (and are encouraged) make comparisons to other technologies in your answer.Here are a few key points: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. 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 errorPython 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 objectsWriting 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 PythonPython finds use in many spheres - web applications, automation, scientific modeling, big data applications, and many more. It's also often used as a "glue" code to get other languages and components to play nicely.Python makes difficult things easy so programmers can focus on overriding algorithms and structures rather than nitty-gritty low-level details.

Question 3: What Native Data Structures Can You Name in Python?

Common native data structures in Python are as follows:DictionariesListsSetsStringsTuples

Question 4: Out of all datasets which Are Mutable, and Which Are Immutable?

ists, dictionaries, and sets are mutable. This means that you can change their content without changing their identity. Strings and tuples are immutable, as their contents can't be altered once they're created.

Question 5: What's the Difference Between a List and a Dictionary?

A list and a dictionary in Python are essentially different types of data structures. Lists are the most common data types that boast significant flexibility. Lists can be used to store a sequence of objects that are mutable (so they can be modified after they are created).However, they have

to be stored in a particular order that can be indexed into the list or iterated over it (and it can also take some time to apply iterations on list objects).For example:>>> a = [1,2,3]>>> a[2]=4>>> a[1, 2, 4]In Python, you can't use a list as a "key" for the dictionary (technically you can hash the list first via your own custom hash functions and use that as a key). A Python dictionary is fundamentally an unordered collection of key-value pairs. It's a perfect tool to work with an enormous amount of data since dictionaries are optimized for retrieving data (but you have to know the key to retrieve its value).It can also be described as the implementation of a hashtable and as a key-value store. In this scenario, you can quickly look up anything by its key, but since it's unordered, it will demand that keys are hashes. When you work with Python, dictionaries are defined within curly braces {} where each item will be a pair in the form key:value.

Question 6: In a List and in a Dictionary, What Are the Typical Characteristics of Elements?

Elements in lists maintain their ordering unless they are explicitly commanded to be re-ordered. They can be of any data type, they can all be the same, or they can be mixed. Elements in lists are always accessed through numeric, zero-based indices.In a dictionary, each entry will have a key and a value, but the order will not be guaranteed. Elements in the dictionary can be accessed by using their key.Lists can be used whenever you have a collection of items in an order. A dictionary can be used whenever you have a set of unique keys that map to values.

Question 7: Is There a Way to Get a List of All the Keys in a Dictionary?

If So, How Would You Do It?If the interviewer follows up with this question, try to make your answer as specific as possible.To obtain a list of all the keys in a dictionary, we have to use function keys():mydict={'a':1,'b':2,'c':3,'e':5}mydict.keys()dict_keys(['a', 'b', 'c', 'e'])

Question 8: Can You Explain What a List or Dict Comprehension Is?

When you need to create a new list from other iterables, you have to use list comprehensions. As lists comprehensions return list results, they will be made up of brackets that contain the expressions that need to be executed for each element. Along with the loop, these can be iterated over each element. Example of the basic syntax:new_list = [expression for_loop_one_or_more conditions]When you need to write for loops in Python, list comprehensions can make life a lot easier, as you can achieve this in a single line of code. However, comprehensions are not unique to lists. Dictionaries, which are common data structures used in data science, can also do comprehension. If you have to do a practical test to demonstrate your knowledge and experience, it will be critical to remember that a Python list is defined with square brackets []. On the other hand, a dictionary will be represented by curly braces {}. Determining dict comprehension follows the same principle and is defined with a similar syntax, but it has to have a key:value pair in the expression.

Question 9: When Would You Use a List vs. a Tuple vs. a Set in Python?

A list is a common data type that is highly flexible. It can store a sequence of objects that are mutable, so it's ideal for projects that demand the storage of objects that can be changed later .A tuple is similar to a list in Python, but the key difference between them is that tuples are immutable. They also use less space than lists and can only be used as a key in a dictionary. Tuples are a perfect choice when you want a list of constants. Sets are a collection of unique elements that are used in Python. Sets are a good option when you want to avoid duplicate

elements in your list. This means that whenever you have two lists with common elements between them, you can leverage sets to eliminate them.

Question 10: What's the Difference between a For Loop and a While Loop?

n Python, a loop iterates over popular data types (like dictionaries, lists, or strings) while the condition is true. This means that the program control will pass to the line immediately following the loop whenever the condition is false. In this scenario, it's not a question of preference, but a question of what your data structures are.For LoopIn Python (and in almost any other programming language), For Loop is the most common type of loop. For Loop is often leveraged to iterate through the elements of an array.For example:For i=0, N_Elements (array) do...For Loop can also be used to perform a fixed number of iterations and iterate by a given (positive or even negative) increment. It's important to note that by default, the increment will always be one.While LoopWhile Loop can be used in Python to perform an indefinite number of iterations as long as the condition remains true.For example:While (condition) do...When using the While Loop, you have to explicitly specify a counter to keep track of how many times the loop was executed. However, While Loop can't define its own variable. Instead, it has to be previously defined and will continue to exist even after you exit the loop.When compared to For Loop, While Loop is inefficient because it's much slower. This can be attributed to the fact that it checks the condition after each iteration. However, if you need to perform one or more conditional checks in a For Loop, you will want to consider using While Loop instead (as these checks won't be required).

Question 11: Write Output of the following given code What does this code output:def f(x,l=[]):

for i in range(x):

l.append(i*i) print(l)

f(2)f(3,[3,2,1])f(3)

Question 12: What Packages in the Standard Library, Useful for Data Science Work, Do You Know?

When Guido van Rossum created Python in the 1990s, it wasn't built for data science. Yet, today, Python is the leading language for machine learning, predictive analytics, statistics, and simple data analytics.

This is because Python is a free and open-source language that data professionals could easily use to develop tools that would help them complete data tasks more efficiently.

The following packages in the Python Standard Library are very handy for data science projects:

NumPy

NumPy (or Numerical Python) is one of the principal packages for data science applications. It's often used to process large multidimensional arrays, extensive collections of high-level mathematical functions, and matrices. Implementation methods also make it easy to conduct multiple operations with these objects .

T here have been many improvements made over the last year that have resolved several bugs and compatibility issues. NumPy is popular because it can be used as a highly efficient multi-dimensional container of generic data. It's also an excellent library as it makes data analysis simple by processing data faster while using a lot less code than lists.

Pandas

Pandas is a Python library that provides highly flexible and powerful tools and high-level data structures for analysis. Pandas is an excellent tool for data analytics because it can translate highly complex operations with data into just one or two commands.

Pandas come with a variety of built-in methods for combining, filtering, and grouping data. It also boasts time-series functionality that is closely followed by remarkable speed indicators.

SciPy

SciPy is another outstanding library for scientific computing. It's based on NumPy and was created to extend its capabilities. Like NumPy, SciPy's data structure is also a multidimensional array that's implemented by NumPy.

The SciPy package contains powerful tools that help solve tasks related to integral calculus, linear algebra, probability theory, and much more .

Recently, this Python library went through some major build improvements in the form of continuous integration into multiple operating systems, methods, and new functions. Optimizers were also updated, and several new BLAS and LAPACK functions were wrapped.

Question 13: What does this stuff mean: *args, **kwargs? And why would we use it?

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. **kwargs is 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.

Here is a little illustration:

```
def f(*args,**kwargs): print(args, kwargs)

l = [1,2,3]

t = (4,5,6)

d = {'a':7,'b':8,'c':9}

f()

f(1,2,3)                # (1, 2, 3) {}

f(1,2,3,"groovy")       # (1, 2, 3, 'groovy') {}

f(a=1,b=2,c=3)          # () {'a': 1, 'c': 3, 'b': 2}

f(a=1,b=2,c=3,zzz="hi")    # () {'a': 1, 'c': 3, 'b': 2, 'zzz': 'hi'}

f(1,2,3,a=1,b=2,c=3)      # (1, 2, 3) {'a': 1, 'c': 3, 'b': 2}
```

```
f(*l,**d)              # (1, 2, 3) {'a': 7, 'c': 9, 'b': 8}
f(*t,**d)              # (4, 5, 6) {'a': 7, 'c': 9, 'b': 8}
f(1,2,*t)              # (1, 2, 4, 5, 6) {}f(q="winning",**d)
 # () {'a': 7, 'q': 'winning', 'c': 9, 'b': 8}f(1,2,*t,q="winning",**d)
# (1, 2, 4, 5, 6) {'a': 7, 'q': 'winning', 'c': 9, 'b': 8}
def f2(arg1,arg2,*args,**kwargs):
print(arg1,arg2, args, kwargs)f2(1,2,3)
          # 1 2 (3,) {}f2(1,2,3,"groovy")          # 1 2 (3, 'groovy') {}f2(arg1=1,arg2=2,c=3)
   # 1 2 () {'c': 3}f2(arg1=1,arg2=2,c=3,zzz="hi")  # 1 2 () {'c': 3, 'zzz': 'hi'}f2(1,2,3,a=1,b=2,c=3)
      # 1 2 (3,) {'a': 1, 'c': 3, 'b': 2}
f2(*l,**d)             # 1 2 (3,) {'a': 7, 'c': 9, 'b': 8}f2(*t,**d)
    # 4 5 (6,) {'a': 7, 'c': 9, 'b': 8}f2(1,2,*t)            # 1 2 (4, 5, 6) {}f2(1,1,q="winning",**d)
# 1 1 () {'a': 7, 'q': 'winning', 'c': 9, 'b': 8}f2(1,2,*t,q="winning",**d)
# 1 2 (4, 5, 6) {'a': 7, 'q': 'winning', 'c': 9, 'b': 8}
```

Question 14: Consider the following code, what will it output?

```
class A(object):
   def go(self):
    print("go A go!")
   def stop(self):
    print("stop A stop!")
def pause(self):
raise Exception("Not Implemented")


class B(A):
 def go(self):
super(B, self).go()
  print("go B go!")
class C(A):
  def go(self):      super(C, self).go()      print("go C go!")   def stop(self):      super(C,
self).stop()      print("stop C stop!")
```

class D(B,C): def go(self): super(D, self).go() print("go D go!") def stop(self): super(D, self).stop() print("stop D stop!") def pause(self): print("wait D wait!")

class E(B,C): pass

a = A()b = B()c = C()d = D()e = E()

specify output from here onwards

a.go()b.go()c.go()d.go()e.go()

a.stop()b.stop()c.stop()d.stop()e.stop()

a.pause()b.pause()c.pause()d.pause()e.pause()

AnswerThe output is specified in the comments in the segment below:a.go()# go A go!

b.go()# go A go!# go B go!

c.go()# go A go!# go C go! d.go()# go A go!# go C go!# go B go!# go D go!

e.go()# go A go!# go C go!# go B go!

a.stop()# stop A stop!

b.stop()# stop A stop!

c.stop()# stop A stop!# stop C stop!

d.stop()# stop A stop!# stop C stop!# stop D stop!

e.stop()# stop A stop! a.pause()# ... Exception: Not Implemented

b.pause()# ... Exception: Not Implemented

c.pause()# ... Exception: Not Implemented

d.pause()# wait D wait!

e.pause()# ...Exception: Not Implemented

Question 15: Consider the following code, what will it output?

```
class Node(object):
  def __init__(self,sName):
   self._lChildren = []
    self.sName = sName
def __repr__(self):

  return "<Node '{}'>".format(self.sName)
```

```
 def append(self,*args,**kwargs)
  self._lChildren.append(*args,**kwargs)   def print_all_1(self):
  print(self)      for oChild in self._lChildren:          oChild.print_all_1()
 def print_all_2(self):
  def gen(o):
   lAll = [o,]         while lAll:
      oNext = lAll.pop(0)    lAll.extend(oNext._lChildren)
yield oNext      for oNode in gen(self):          print(oNode)
oRoot = Node("root")oChild1 = Node("child1")oChild2 = Node("child2")oChild3 =
Node("child3")oChild4 = Node("child4")oChild5 = Node("child5")oChild6 =
Node("child6")oChild7 = Node("child7")oChild8 = Node("child8")oChild9 =
Node("child9")oChild10 = Node("child10")
oRoot.append
(oChild1)oRoot.append(oChild2)oRoot.append(oChild3)oChild1.append(oChild4)oChild1.app
end(oChild5)oChild2.append(oChild6)oChild4.append(oChild7)oChild3.append(oChild8)oChil
d3.append(oChild9)oChild6.append(oChild10)
# specify output from here onwards
oRoot.print_all_1()oRoot.print_all_2
()AnsweroRoot.print_all_1() prints:<Node 'root'>
<Node 'child1'>
<Node 'child4'>
<Node 'child7'>
<Node 'child5'>
<Node 'child2'>
<Node 'child6'>
<Node 'child10'>
<Node 'child3'>
<Node 'child8'>
<Node 'child9'>
oRoot.print_all_2()
prints:<Node 'root'>
<Node 'child1'><Node 'child2'>
```

<Node 'child3'><Node 'child4'>

<Node 'child5'><Node 'child6'>

<Node 'child8'><Node 'child9'><Node 'child7'>

<Node 'child10'>

Question 16: Describe Python's garbage collection mechanism in brief.

AnswerA lot can be said here. There are a few main points that you should mention:Python maintains a count of the number of references to each object in memory. If a reference count goes to zero then the associated object is no longer live and the memory allocated to that object can be freed up for something elseoccasionally things called "reference cycles" happen. The garbage collector periodically looks for these and cleans them up. An example would be if you have two objects o1 and o2 such that o1.x == o2 and o2.x == o1. If o1 and o2 are not referenced by anything else then they shouldn't be live. But each of them has a reference count of 1.Certain heuristics are used to speed up garbage collection. For example, recently created objects are more likely to be dead. As objects are created, the garbage collector assigns them to generations. Each object gets one generation, and younger generations are dealt with first.This explanation is CPython specific.

Question 16: Place the following functions below in order of their efficiency. They all take in a list of numbers between 0 and 1. The list can be quite long. An example input list would be [random.random() for i in range(100000)]. How would you prove that your answer is correct?

```
def f1(lIn):    l1 = sorted(lIn)

l2 = [i for i in l1 if i<0.5]
 return [i*i for i in l2]

def f2(lIn):    l1 = [i for i in lIn if i<0.5]

l2 = sorted(l1)

return [i*i for i in l2]

def f3(lIn):    l1 = [i*i for i in lIn]    l2 = sorted(l1)

return [i for i in l1 if i<(0.5*0.5)]
```

Most to least efficient: f2, f1, f3. To prove that this is the case, you would want to profile your code. Python has a lovely profiling package that should do the trick.import cProfilelIn = [random.random() for i in range(100000)]cProfile.run('f1(lIn)')

cProfile.run('f2(lIn)')

cProfile.run('f3(lIn)')

For completion's sake, here is what the above profile outputs:>>> cProfile.run('f1(lIn)')

```
   4 function calls in 0.045 seconds

  Ordered by: standard name  ncalls  tottime  percall  cumtime  percall
filename:lineno(function)
```

1 0.009 0.009 0.044 0.044 <stdin>:1(f1) 1 0.001 0.001 0.045 0.045 <string>:1(<module>) 1 0.000 0.000 0.000 0.000 {method 'disable' of '_lsprof.Profiler' objects} 1 0.035 0.035 0.035 0.035 {sorted}

>>> cProfile.run('f2(lln)') 4 function calls in 0.024 seconds

Ordered by: standard name ncalls tottime percall cumtime percall filename:lineno(function) 1 0.008 0.008 0.023 0.023 <stdin>:1(f2) 1 0.001 0.001 0.024 0.024 <string>:1(<module>) 1 0.000 0.000 0.000 0.000 {method 'disable' of '_lsprof.Profiler' objects} 1 0.016 0.016 0.016 0.016 {sorted}

>>> cProfile.run('f3(lln)') 4 function calls in 0.055 seconds

Ordered by: standard name ncalls tottime percall cumtime percall filename:lineno(function) 1 0.016 0.016 0.054 0.054 <stdin>:1(f3) 1 0.001 0.001 0.055 0.055 <string>:1(<module>) 1 0.000 0.000 0.000 0.000 {method 'disable' of '_lsprof.Profiler' objects} 1 0.038 0.038 0.038 0.038 {sorted}

Question 17: In Python, How is Memory Managed?

In Python, memory is managed in private heap space. This means that all the objects and data structures will be located in a private heap. However, the programmer won't be allowed to access this heap. Instead, the Python interpreter will handle it. At the same time, the core API will enable access to some Python tools for the programmer to start coding.The memory manager will allocate the heap space for the Python objects while the inbuilt garbage collector will recycle all the memory that's not being used to boost available heap space.

Question 18: What is the purpose of the PYTHONSTARTUP environment variable?

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.

Question 19: Is python a case sensitive language?

Yes! Python is a case sensitive programming language.

Question 20: What are the supported data types in Python?

Python has five standard data types –NumbersStringListTupleDictionary

Question 21: What is the output of print str if str = 'Hello World!'?

It will print a complete string. The output would be Hello World!.

Question 22: What is the output of print str[0] if str = 'Hello World!'?

It will print the first character of the string. The output would be H.

Question 23: What is the output of print str[2:5] if str = 'Hello World!'?

It will print characters starting from 3rd to 5th. The output would be llo.

Question 24: What is the output of print str[2:] if str = 'Hello World!'?

It will print characters starting from 3rd character. The output would be llo World!.

Question 25: What is the output of print str * 2 if str = 'Hello World!'?

It will print the string two times. The output would be Hello World! Hello World!.

Question 26: What is the output of print str + "TEST" if str = 'Hello World!'?

It will print a concatenated string. The output would be Hello World!TEST.

Question 27: What is the output of print list if list = ['abcd', 786 , 2.23, 'john', 70.2]?

It will print complete list. Output would be ['abcd', 786, 2.23, 'john', 70.200000000000003].

Question 28: What is the output of print list[0] if list = ['abcd', 786 , 2.23, 'john', 70.2]?

It will print first element of the list. Output would be abcd.

Question 29: What is the output of print list[1:3] if list = ['abcd', 786 , 2.23, 'john', 70.2]?

It will print elements starting from 2nd till 3rd. Output would be [786, 2.23].

It will print elements starting from 3rd element. Output would be [2.23, 'john', 70.200000000000003].

Question 31: What is the output of print tinylist * 2 if tinylist = [123, 'john']?

It will print list two times. Output would be [123, 'john', 123, 'john'].

Question 32: What is the output of print list1 + list2, if list1 = ['abcd', 786 , 2.23, 'john', 70.2] and ist2 = [123, 'john']?I

t will print concatenated lists. Output would be ['abcd', 786, 2.23, 'john', 70.2, 123, 'john']

Question 33: What are tuples in Python?

A tuple is another sequence data type that is similar to the list. A tuple consists of a number of values separated by commas. Unlike lists, however, tuples are enclosed within parentheses.

Question 34: What is the difference between tuples and lists in Python?

The main differences between lists and tuples are – Lists are enclosed in brackets ([ ]) and their elements and size can be changed, while tuples are enclosed in parentheses (( )) and cannot be updated. Tuples can be thought of as read-only lists.

Question 35: What is the output of print tuple if tuple = ('abcd', 786 , 2.23, 'john', 70.2)?

It will print complete tuple. Output would be ('abcd', 786, 2.23, 'john', 70.200000000000003).

Question 36: What is the output of print tuple[0] if tuple = ('abcd', 786 , 2.23, 'john', 70.2)?

It will print the first element of the tuple. The output would be abcd.

Question 37: What is the output of print tuple[1:3] if tuple = ('abcd', 786 , 2.23, 'john', 70.2)?

It will print elements starting from 2nd till 3rd. The output would be (786, 2.23).

Question 38: What is the output of print tuple[2:] if tuple = ('abcd', 786 , 2.23, 'john', 70.2)?

It will print elements starting from the 3rd element. The output would be (2.23, 'john', 70.200000000000003).

Question 39: What is the output of print tinytuple * 2 if tinytuple = (123, 'john')?

It will print tuple two times. The output would be (123, 'john', 123, 'john').

Question 40: What is the output of print tuple + tinytuple if tuple = ('abcd', 786 , 2.23, 'john', 70.2) and tinytuple = (123, 'john')?

It will print concatenated tuples. Output would be ('abcd', 786, 2.23, 'john', 70.200000000000003, 123, 'john').

Q**uestion 41: What are Python's dictionaries?**

Python's dictionaries are kind of hash table type. They work like associative arrays or hashes found in Perl and consist of key-value pairs. A dictionary key can be almost any Python type, but are usually numbers or strings. Values, on the other hand, can be any arbitrary Python object.

Question 42: How will you create a dictionary in python?

Dictionaries are enclosed by curly braces ({ }) and values can be assigned and accessed using square braces ([]).dict = {}dict['one'] = "This is one"dict[2] = "This is two"tinydict = {'name': 'john','code':6734, 'dept': 'sales'}

Question 43: How will you get all the keys from the dictionary?

Using dictionary.keys() function, we can get all the keys from the dictionary object.print dict.keys() # Prints all the keys

Question 44: How will you get all the values from the dictionary?

Using dictionary.values() function, we can get all the values from the dictionary object.print dict.values() # Prints all the values

Q**uestion 45: How will you convert a string to an int in python?**

int(x [,base]) - Converts x to an integer. base specifies the base if x is a string.

Question 46: How will you convert a string to a long in python?

long(x [,base]) - Converts x to a long integer. base specifies the base if x is a string.

Question 47: How will you convert a string to a float in python?

float(x) – Converts x to a floating-point number.

Question 48: How will you convert an object to a string in python?

str(x) – Converts object x to a string representation.

Question 49: How will you convert an object to a regular expression in python?

repr(x) – Converts object x to an expression string.

Question 50: How will you convert a string to an object in python?

eval(str) – Evaluates a string and returns an object.

Question 51: How will you convert a string to a tuple in python?

tuple(s) – Converts s to a tuple.

Question 52: How will you convert a string to a list in python?

list(s) – Converts s to a list.

Question 53: How will you convert a string to a set in python?

set(s) – Converts s to a set.

Question 54: How will you create a dictionary using tuples in python?

dict(d) – Creates a dictionary. d must be a sequence of (key, value) tuples.

Question 55: How will you convert a string to a frozen set in python?

frozenset(s) – Converts s to a frozen set.

Question 56: How will you convert an integer to a character in python?

chr(x) – Converts an integer to a character.

Question 57: How will you convert an integer to an Unicode character in python?

unichr(x) – Converts an integer to a Unicode character.

Question 58: How will you convert a single character to its integer value in python?

ord(x) – Converts a single character to its integer value.

Python Interview Questions

hex(x) – Converts an integer to a hexadecimal string.

Question 60: How will you convert an integer to an octal string in python?

oct(x) – Converts an integer to an octal string.

Question 61: What is the purpose of ** operator?**

Exponent – Performs exponential (power) calculation on operators. a**b = 10 to the power 20 if a = 10 and b = 20.

Question 62: What is the purpose of // operator?//

Floor Division – The division of operands where the result is the quotient in which the digits after the decimal point are removed.

Question 63: What is the purpose of is the operator?

is – Evaluates to true if the variables on either side of the operator point to the same object and false otherwise. x is y, here is results in 1 if id(x) equals id(y).

Question 64: What is the purpose of not in the operator?

not in – Evaluates to true if it does not finds a variable in the specified sequence and false otherwise. x not in y, here not in results in a 1 if x is not a member of sequence y.

Question 65: What is the purpose break statement in python?

break statement – Terminates the loop statement and transfers execution to the statement immediately following the loop.

Question 66: What is the purpose of continue statement in python?

continue statement – Causes the loop to skip the remainder of its body and immediately retest its condition prior to reiterating.

Question 67: What is the purpose pass statement in python?

pass statement – The pass statement in Python is used when a statement is required syntactically but you do not want any command or code to execute.

Question 68: How can you pick a random item from a list or tuple?

choice(seq) – Returns a random item from a list, tuple, or string.

Question 69: How can you pick a random item from a range?

randrange ([start,] stop [,step]) – returns a randomly selected element from range(start, stop, step).

Question 70: How can you get a random number in python?

random() – returns a random float r, such that 0 is less than or equal to r and r is less than 1.

Question 71: How will you set the starting value in generating random numbers?

seed([x]) – Sets the integer starting value used in generating random numbers. Call this function before calling any other random module function. Returns None.

Question 72: How will you randomize the items of a list in place?

shuffle(lst) – Randomizes the items of a list in place. Returns None.

Question 73: How will you capitalize first letter of string?

capitalize() – Capitalizes first letter of string.

Question 74: How will you check in a string that all characters are alphanumeric?

isalnum() – Returns true if the string has at least 1 character and all characters are alphanumeric and false otherwise.

Question 75: How will you check in a string that all characters are digits?

isdigit() – Returns true if the string contains only digits and false otherwise.

Question 76: How will you check in a string that all characters are in lowercase?

islower() – Returns true if the string has at least 1 cased character and all cased characters are in lowercase and false otherwise.

Question 77: How will you check in a string that all characters are numerics?

isnumeric() – Returns true if a Unicode string contains only numeric characters and false otherwise.

Question 78: How will you check in a string that all characters are whitespaces?

isspace() – Returns true if the string contains only whitespace characters and false otherwise.

Question 79: How will you check in a string that it is properly titlecased?

istitle() – Returns true if the string is properly "titlecased" and false otherwise.

Question 80: How will you check in a string that all characters are in uppercase?

isupper() – Returns true if the string has at least one cased character and all cased characters are in uppercase and false otherwise.

Question 81: How will you merge elements in a sequence?

join(seq) – Merges (concatenates) the string representations of elements in sequence seq into a string, with separator string.

Question 82: How will you get the length of the string?

len(string) – Returns the length of the string.

Question 83: How will you get a space-padded string with the original string left-justified to a total of width columns?

ljust(width[, fillchar]) – Returns a space-padded string with the original string left-justified to a total of width columns.

Question 84: How will you convert a string to all lowercase?

lower() – Converts all uppercase letters in a string to lowercase.

Question 85: How will you remove all leading whitespace in a string?

lstrip() – Removes all leading whitespace in string.

Question 86: How will you get the max alphabetical character from the string?

max(str) – Returns the max alphabetical character from the string str.

Question 87: How will you get the min alphabetical character from the string?

min(str) – Returns the min alphabetical character from the string str.

Question 88: How will you replace all occurrences of an old substring in a string with new string?

replace(old, new [, max]) – Replaces all occurrences of old in a string with new or at most max occurrences if max gave.

Question 89: How will you remove all leading and trailing whitespace in a string?

strip([chars]) – Performs both lstrip() and rstrip() on string.

Question 90: How will you change the case for all letters in a string?

swapcase() – Inverts case for all letters in the string.

Question 91: How will you get title cased version of string?

title() – Returns "titlecased" version of the string, that is, all words begin with uppercase and the rest are lowercase.

Question 92: How will you convert a string to all uppercase?

upper() – Converts all lowercase letters in a string to uppercase.

Question 93: How will you check in a string that all characters are decimal?

isdecimal() – Returns true if a Unicode string contains only decimal characters and false otherwise.

Question 94: What is the difference between del() and remove() methods of list?

To remove a list element, you can use either the del statement if you know exactly which element(s) you are deleting or the remove() method if you do not know.

Question 95: What is the output of len([1, 2, 3])?

3.

Question 96: What is the output of [1, 2, 3] + [4, 5, 6]?

[1, 2, 3, 4, 5, 6]

Question 97: What is the output of ['Hi!'] * 4?

['Hi!', 'Hi!', 'Hi!', 'Hi!']

Question 98: What is the output of 3 in [1, 2, 3]?

True

Question 99: What is the output of x in [1, 2, 3]: print x?

123

Question 100: What is the output of L[2] if L = [1,2,3]?

3, Offsets start at zero.

Question 101: What is the output of L[-2] if L = [1,2,3]?

1, Negative: count from the right.

Question 102: What is the output of L[1:] if L = [1,2,3]?

2, 3, Slicing fetches sections.

Question 103: How will you compare two lists?

cmp(list1, list2) – Compares elements of both lists.

Question 104: How will you get the length of a list?l

en(list) – Gives the total length of the list.

Question 105: How will you get the max valued item of a list?

max(list) – Returns item from the list with max value.

Question 106: How will you get the min valued item of a list?

min(list) – Returns item from the list with min value.

Question 107: How will you get the index of an object in a list?

list.index(obj) – Returns the lowest index in the list that obj appears.

Question 108: How will you insert an object at a given index in a list?

list.insert(index, obj) – Inserts object obj into list at offset index.

Question 109: How will you remove the last object from a list?

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

Question 110: How will you remove an object from a list?

list.remove(obj) – Removes object obj from list.

Question 111: How will you reverse a list?

list.reverse() – Reverses objects of the list in place.

Question 112: How will you sort a list?

list.sort([func]) – Sorts objects of list, use compare func if given.

Question 113: What is lambda function in python?'

lambda' is a keyword in python which creates an anonymous function. Lambda does not contain block of statements. It does not contain return statements.

Question 114: What we call a function that is an incomplete version of a function?

Stub.

Question 115: When a function is defined then the system stores parameters and local variables in an area of memory. What this memory is known as?

Stack.

Python Interview Questions

(Yes/No)Yes.

Question 117: Is the Python platform independent?

NoThere are some modules and functions in python that can only run on certain platforms.

Question 118: Do you think Python has a compiler?

YesYes, it has a compiler that works automatically so we don't notice the compiler of python.

Q**uestion 119: What are the applications of Python?**

Django (web framework of Python).

2. Micro Framework such as Flask and Bottle.

3. Plone and Django CMS for advanced content Management.

Question 120: What is the basic difference between Python version 2 and Python version 3?

Table below explains the difference between Python version 2 and Python version 3.S.NoSectionPython Version2Python Version31.Print Function The print command can be used without parentheses. Python 3 needs parentheses to print any string. It will raise errors without parentheses.2.UnicodeASCII str() types and separate Unicode() but there is no byte type code in

Python 2.Unicode (utf-8) and it has two-byte classes –Bytebytearray S .3.ExceptionsPython 2 accepts both new and old notations of syntax. Python 3 raises a SyntaxError in turn when we don't enclose the exception argument in parentheses.4.Comparing UnorderableIt does not raise any error .It raises 'TypeError' as a warning if we try to compare unorderable types.

Question 121: Which programming language is an implementation of Python programming language designed to run on Java Platform?

Jython(Jython is the successor of Jpython.)

Question 122: Is there any double data type in Python?

No

Question 123: Is String in Python are immutable?

(Yes/No)Yes.

Question 124: Can True = False be possible in Python?

No.

Question 125: Which module of python is used to apply the methods related to OS.?

OS.

Question 126: When does a new block begin in python?

A block begins when the line is intended by 4 spaces.

Question 127: Write a function in python that detects whether the given two strings are anagrams or not.

```
def check(a,b):
if(len(a)!=len(b)):
  return False   else:
if(sorted(list(a)) =
= sorted(list(b))):
  return True   else:
return False
```

Question 128: Name the python Library used for Machine learning.

Scikit-learn python Library used for Machine learning

Question 129: What does pass operation do?

Pass indicates that nothing is to be done i.e. it signifies a no operation.

Question 130: Name the tools that python uses to find bugs (if any).

Pylint and pychecker.

Question 131: Write a function to give the sum of all the numbers in the list?

Sample list – (100, 200, 300, 400, 0, 500)

Expected output – 1500

Question 132: Program for the sum of all the numbers in the list is –

```
def sum(numbers):

total = 0   for num in numbers:

   total+=num   print(''Sum of the numbers: '

', total)sum((100, 200, 300, 400, 0, 500))
```

We define a function 'sum' with numbers as a parameter. The in for loop we store the sum of all the values of the list.

Question 133: Write a program in Python to reverse a string without using the inbuilt function reverse string?

Program to reverse a string is given below –

```
def string_reverse(str1):

              rev_str = ' '

      index = len(str1) #defining index as length of string.

            while(index>0):

         rev_str = rev_str + str1[index-1]

  index = index-1

      return(rev_str)


print(string_reverse('1tniop'))
```

First, we declare a variable to store the reversed string. Then using while loop and indexing of string (index is calculated by string length) we reverse the string. While loop starts when the index is greater than zero. The index is reduced to value 1 each time. When the index reaches zero we obtain the reverse of a string.

Question 134: Write a program to test whether the number is in the defined range or not?

Program is –

```
def test_range(num):   if num in range(0, 101):

   print(''%s is in range''%str(num))

 else:

 print(''%s is not in range''%str(num))
```

Output –

test_range(101)

101 is not in the rangeTo test any number in a particular range we make use of the method ‘if.. in’ and else condition.