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:

- Interpreted
- Dynamically-typed
- Object-oriented
- Concise and simple
- Free
- Has a large community

Q.2. Differentiate between lists and tuples.
The major difference is that a list is mutable, but a tuple is immutable. Examples:

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

Traceback (most recent call last):
File "<pyshell#97>", line 1, in <module>
mytuple[1]=2
TypeError: 'tuple' object does not support item assignment
Python Tuples vs Lists* – *A Detailed Comparison
Q.3. 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:

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

2

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

Hi
Q.4. What are negative indices?
Let's take a list for this.

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

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

```
>>> mylist[-3]
```

6

This also helps with slicing from the back:

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

[3, 4, 5, 6, 7]

Q.5. 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.

```
>>> myname='Ayushi'
>>> 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.

Q.6. How long can an identifier be in Python?

According to the official Python documentation, an identifier can be of any length. However, PEP 8 suggests that you should limit all lines to a maximum of 79 characters. Also, PEP 20 says 'readability counts'. So, a very long identifier will violate PEP-8 and PEP-20.

Apart from that, there are certain rules we must follow to name one:

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

and	def	False	import	not	True
as	del	finally	in	or	try
assert	elif	for	is	pass	while
break	else	from	lambda	print	with
class	except	global	None	raise	yield
continue	exec	if	nonlocal	return	

Learn everything about Python Identifiers

Q.7. How would you convert a string into lowercase?

We use the lower() method for this.

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

‘ayushi’

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

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

‘AYUSHI’

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

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

False

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

True

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

True

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

True

```
>>> '@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.

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

True

Q.8. What is the pass statement in Python?

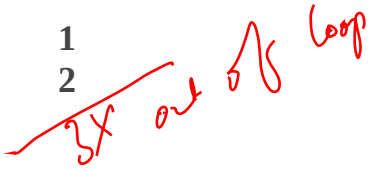
There may be times in our code when we haven’t decided what to do yet, but we must type something for it to be syntactically correct. In such a case, we use the pass statement.

```
>>> def func(*args):
pass
>>>
```

Similarly, the break statement breaks out of a loop.

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

1
2



Finally, the continue statement skips to the next iteration.

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

1
2
4
5
6

Hope you have read all the basic Python Interview Questions and Answers. Now, let's move towards the second part of the blog – Most asked Python Interview Questions and Answers for freshers

Frequently Asked Python Interview Questions and Answers for Freshers

While solving or answering these questions, if you feel any difficulty, comment us. DataFlair is always ready to help you.

Q.9. Explain help() and dir() functions in Python.
The help() function displays the documentation string and help for its argument.

```
>>> import copy
>>> 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).

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

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

Q.10. 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().

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

dict_keys([‘a’, ‘b’, ‘c’, ‘e’])

Q.11. What is slicing?
Slicing is a technique that allows us to retrieve only a part of a list, tuple, or string. For this, we use the slicing operator [].

```
>>> (1,2,3,4,5)[2:4]
```

(3, 4)

```
>>> [7,6,8,5,9][2:]
```

[8, 5, 9]

```
>>> 'Hello'[:-1]
```

‘Hell’

Q.12. How would you declare a comment in Python?
Unlike languages like C++, Python does not have multiline comments. All it has is octothorpe (#). Anything following a hash is considered a comment, and the interpreter ignores it.

```
>>>#line 1 of comment
>>> #line 2 of comment
```

In fact, you can place a comment anywhere in your code. You can use it to explain your code.

Q.13. How will you check if all characters in a string are alphanumeric?
For this, we use the method isalnum().

Q.14. How will you capitalize the first letter of a string?
Simply using the method capitalize().

```
>>>'ayushi'.capitalize()
```

‘Ayushi’

```
>>> type(str.capitalize)
```

<class ‘method_descriptor’>

However, it will let other characters be.

```
>>> '@yushi'.capitalize()
```

‘@yushi’

```
>>> 'Ayushi123'.isalnum()
```

True

```
>>> 'Ayushi123!'.isalnum()
```

False

Other methods that we have include:

```
>>> '123.3'.isdigit()
```

False

```
>>> '123'.isnumeric()
```

True

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

True

```
>>> 'Ayushi'.isupper()
```

False

```
>>> 'Ayushi'.istitle()
```

True

```
>>> ' '.isspace()
```

True

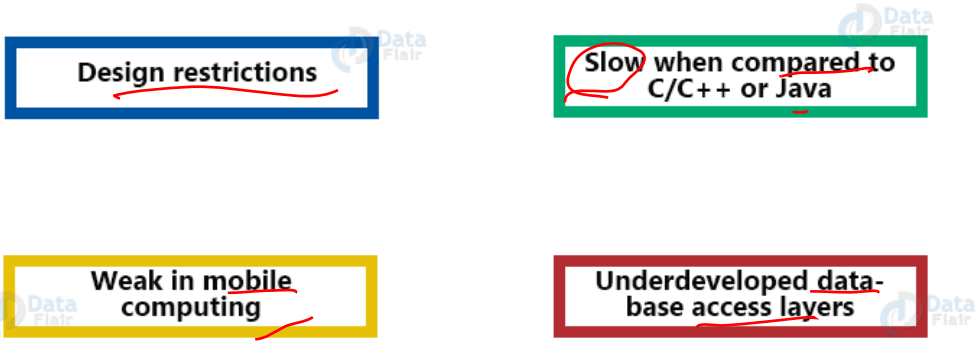
```
>>> '123F'.isdecimal()
```

False

Q.15. We know Python is all the rage these days. But to be truly accepting of a great technology, you must know its pitfalls as well. Would you like to talk about this?

Of course. To be truly yourself, you must be accepting of your flaws. Only then can you move forward to work on them. Python has its flaws too:

Limitations of Python



- Python's interpreted nature imposes a speed penalty on it.
- While Python is great for a lot of things, it is weak in mobile computing, and in browsers.
- Being dynamically-typed, Python uses duck-typing (If it looks like a duck, it must be a duck). This can raise runtime errors.
- Python has underdeveloped database access layers. This renders it a less-than-perfect choice for huge database applications.
- And then, well, of course. Being easy makes it addictive. Once a Python-coder, always a Python coder.

Q.16. With Python, how do you find out which directory you are currently in?
To find this, we use the function/method getcwd(). We import it from the module os.

```
>>> import os
>>> os.getcwd()
```

'C:\\Users\\lifei\\AppData\\Local\\Programs\\Python\\Python36-32'

```
>>> type(os.getcwd)
```

<class 'builtin_function_or_method'>
We can also change the current working directory with chdir().

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

'C:\\Users\\lifei\\Desktop'
Q.17. How do you insert an object at a given index in Python?
Let's build a list first.

```
>>> a=[1,2,4]
```

Now, we use the method insert. The first argument is the index at which to insert, the second is the value to insert.

```
>>> a.insert(2,3)
>>> a
```

[1, 2, 3, 4]

Q.18. And how do you reverse a list?

Using the reverse() method.

```
>>> a.reverse()
>>> a
```

[4, 3, 2, 1]

You can also do it via slicing from right to left:

```
>>> a[::-1]
>>> a
```

[1, 2, 3, 4]

This gives us the original list because we already reversed it once. However, this does not modify the original list to reverse it.

Q.19. What is the Python interpreter prompt?

This is the following sign for ***Python Interpreter***:

```
>>>
```

If you have worked with the IDLE, you will see this prompt.

Q.20. How does a function return values?

A function uses the 'return' keyword to return a value. Take a look:

```
>>> def add(a,b):
return a+b
```

Q.21. How would you define a block in Python?

For any kind of statements, we possibly need to define a block of code under them. However, Python does not support curly braces. This means we must end such statements with colons and then indent the blocks under those with the same amount.

```
>>> if 3>1:
print("Hello")
print("Goodbye")
```

Hello
Goodbye

Q.22. Why do we need break and continue in Python?

Both break and continue are statements that control flow in ***Python loops***. break stops the current loop from executing further and transfers the

control to the next block. continue jumps to the next iteration of the loop without exhausting it.

Q.23. Will the do-while loop work if you don't end it with a semicolon?

Trick question! Python does not support an intrinsic do-while loop. Secondly, to terminate do-while loops is a necessity for languages like C++.

Q.24. In one line, show us how you'll get the max alphabetical character from a string.

For this, we'll simply use the max function.

```
>>> max('flyiNg')
```

'y'

The following are the ASCII values for all the letters of this string-

f- 102

l- 108

y- 121

i- 105

N- 78

g- 103

By this logic, try to explain the following line of code-

```
>>> max('fly{}iNg')
```

'}'

(Bonus: } – 125)

Q.25. What is Python good for?

Python is a jack of many trades, check out Applications of Python to find out more.

Meanwhile, we'll say we can use it for:

- Web and Internet Development
- Desktop GUI
- Scientific and Numeric Applications
- Software Development Applications
- Applications in Education
- Applications in Business
- Database Access
- Network Programming
- Games, 3D Graphics

- Other Python Applications

Q.26. Can you name ten built-in functions in Python and explain each in brief?

Ten Built-in Functions, you say? Okay, here you go.

complex()- Creates a complex number.

```
>>> complex(3.5,4)
```

(3.5+4j)

eval()- Parses a string as an expression.

```
>>> eval('print(max(22,22.0)-min(2,3))')
```

20

filter()- Filters in items for which the condition is true.

```
>>> list(filter(lambda x:x%2==0,[1,2,0,False]))
```

[2, 0, False]

format()- Lets us format a string.

```
>>> print("a={0} but b={1}".format(a,b))
```

a=2 but b=3

hash()- Returns the hash value of an object.

```
>>> hash(3.7)
```

644245917

hex()- Converts an integer to a hexadecimal.

```
>>> hex(14)
```

'0xe'

input()- Reads and returns a line of string.

```
>>> input('Enter a number')
```

Enter a number7

'7'

len()- Returns the length of an object.

```
>>> len('Ayushi')
```

6

locals()- Returns a dictionary of the current local symbol table.

```
>>> locals()
```

{'__name__': '__main__', '__doc__': None, '__package__': None, '__loader__': <class '_frozen_importlib.BuiltinImporter'>, '__spec__': None, '__annotations__': {}, '__builtins__': <module 'builtins' (built-in)>, 'a': 2, 'b': 3}

open()- Opens a file.

```
>>> file=open('tabs.txt')
```

Q.27. 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.28. How will you convert a list into a string?

We will use the join() method for this.

```
>>> nums=['one','two','three','four','five','six','seven']
>>> s=' '.join(nums)
>>> s
```

'one two three four five six seven'

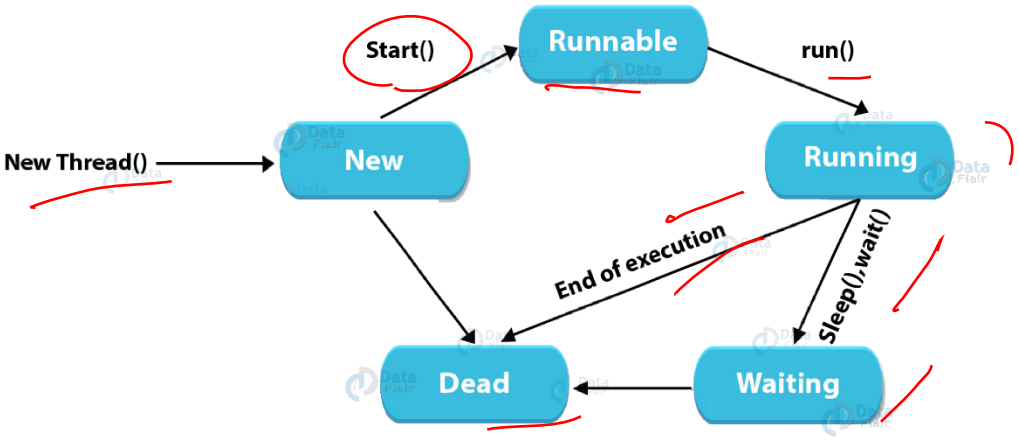
Q.29. How will you remove a duplicate element from a list?

We can turn it into a set to do that.

```
>>> list=[1,2,1,3,4,2]
>>> set(list)
```

{1, 2, 3, 4}

Q.30. Can you explain the life cycle of a thread?



- To create a thread, we create a class that we make override the run method of the thread class. Then, we instantiate it.

- A thread that we just created is in the new state. When we make a call to start() on it, it forwards the threads for scheduling. These are in the ready state.
- When execution begins, the thread is in the running state.
- Calls to methods like sleep() and join() make a thread wait. Such a thread is in the waiting/blocked state.
- When a thread is done waiting or executing, other waiting threads are sent for scheduling.
- A running thread that is done executing terminates and is in the dead state.

Basic Python Program Interview Questions and Answers

Q.31. 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.

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

<class 'dict'>

```
>>> roots[9]
```

3

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

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

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

Q.32. Explain the //, %, and ** operators in Python.

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

```
>>> 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.

```
>>> 2**10
```

1024

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

>>> 13%7

6

>>> 3.5%1.5

0.5

Q.33. What do you know about relational operators in Python.

Python Relational Operators

Operator	Description
<	Less than
>	Greater than
<=	Less than or equal to
>=	Greater than or equal to
=	Equal to
!=	Not equal to

Relational operators compare values.

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

>>> 'hi'<'Hi'

False

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

>>> 1.1+2.2>3.3

True

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

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

>>> 3.0<=3

True

Greater than or equal to (>=) If the value on the left is greater than or equal to, it returns True.

>>> True>=False

True

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

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

True

Not equal to (!=) If the two values are unequal, it returns True.

```
>>> True!=0.1
```

True

```
>>> False!=0.1
```

True

You will surely face a question from Python Operators. There are chances that question may be in an indirect way. Prepare yourself for it with the best guide – Python Operators

Q.34. What are assignment operators in Python?

Python Assignment Operators

Operator	Description
=	Assign
+=	Add and Assign
-=	Subtract and Assign
*=	Multiply and Assign
/=	Divide and Assign
%=	Modulus and Assign
**=	Exponent and Assign
//=	Floor-Divide and Assign

We can combine all arithmetic operators with the assignment symbol.

```
>>> a=7
>>> a+=1
>>> a
```

8

```
>>> a-=1
>>> a
```

7

```
>>> a*=2
>>> a
```

14

```
>>> a/=2
>>> a
```

7.0

```
>>> a**=2
>>> a
```

49.0

```
>>> a//=3
>>> a
```

16.0

```
>>> a%=4
>>> a
```

0.0

Q.35. Explain logical operators in Python.

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

```
>>> False and True
```

False

```
>>> 7<7 or True
```

True

```
>>> not 2==2
```

False

Q.36. What are membership operators?

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

```
>>> 'me' in 'disappointment'
```

True

```
>>> 'us' not in 'disappointment'
```

True

Q.37. Explain identity operators in Python.

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

```
>>> 10 is '10'
```

False

```
>>> True is not False
```

True

Q.38. Finally, tell us about bitwise operators in Python.

Python Bitwise Operators

Operator	Description
&	Binary AND
|	Binary OR
^	Binary XOR
~	Binary One's Complement
<<	Binary Left-Shift
>>	Binary Right-Shift

These operate on values bit by bit.

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

```
>>> 0b110 & 0b010
```

2

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

```
>>> 3|2
```

3

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

```
>>> 3^2
```

1

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

```
>>> ~2
```

-3

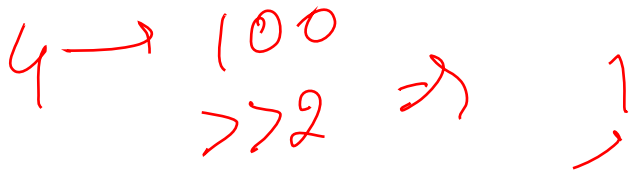
Binary Left-Shift (<<) This shifts the bits to the left by the specified amount.

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

4

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

Binary Right-Shift (>>)



```
>>> 4>>2
```

1

Q.39. What data types does Python support?

Python provides us with five kinds of data types:

Numbers – Numbers use to hold numerical values.

```
>>> a=7.0
>>>
```

Strings – A string is a sequence of characters. We declare it using single or double quotes.

```
>>> title="Ayushi's Book"
```

Lists – A list is an ordered collection of values, and we declare it using square brackets.

```
>>> colors=['red','green','blue']
>>> type(colors)
```

<class 'list'>

Tuples – A tuple, like a list, is an ordered collection of values. The difference. However, is that a tuple is immutable. This means that we cannot change a value in it.

```
>>> name=('Ayushi','Sharma')
>>> name[0]='Avery'
```

Traceback (most recent call last):
File "<pyshell#129>", line 1, in <module>
name[0]='Avery'
TypeError: 'tuple' object does not support item assignment

Dictionary – A dictionary is a data structure that holds key-value pairs. We declare it using curly braces.

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

<class 'dict'>

```
>>> type({})
```

<class 'dict'>

We can also use a dictionary comprehension:

```
>>> squares={x:x**2 for x in range(1,6)}
>>> squares
```

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

Don't miss the complete guide for Python Data Types and Variables

Q.40. What is a docstring?

A docstring is a documentation string that we use to explain what a construct does. We place it as the first thing under a function, class, or a method, to describe what it does. We declare a docstring using three sets of single or double-quotes.

```
>>> def sayhi():
"""
The function prints Hi
"""
print("Hi")
>>> sayhi()
```

Hi

To get a function's docstring, we use its __doc__ attribute.

```
>>> sayhi.__doc__
```

'\n\tThis function prints Hi\n\t'

A docstring, unlike a comment, is retained at runtime.

Q.41. How would you convert a string into an int in Python?

If a string contains only numerical characters, you can convert it into an integer using the int() function.

```
>>> int('227')
```

227

Let's check the types:

```
>>> type('227')
```

<class 'str'>

```
>>> type(int('227'))
```

<class 'int'>

Q.42. How do you take input in Python?

For taking input from the user, we have the function input(). In Python 2, we had another function raw_input().

The input() function takes, as an argument, the text to be displayed for the task:

```
>>> a=input('Enter a number')
```

Enter a number7

But if you have paid attention, you know that it takes input in the form of a string.

```
>>> type(a)
```

<class 'str'>

Multiplying this by 2 gives us this:

a=7

```
>>> a*=2
>>> a
```

'77'

So, what if we need to work on an integer instead?

We use the int() function for this.

```
>>> a=int(input('Enter a number'))
```

Enter a number7

Now when we multiply it by 2, we get this:

```
>>> a*=2
>>> a
```

14

Q.43. What is a function?

When we want to execute a sequence of statements, we can give it a name. Let's define a function to take two numbers and return the greater number.

```
>>> def greater(a,b):
return a if a>b else b
>>> greater(3,3.5)
```

3.5

Q.44. What is recursion?

When a function makes a call to itself, it is termed ***recursion***. But then, in order for it to avoid forming an infinite loop, we must have a base condition.

Let's take an example.

```
>>> def facto(n):
if n==1: return 1
return n*facto(n-1)
>>> facto(4)
```

24

Q.45. What does the function zip() do?

One of the less common functions with beginners, zip() returns an iterator of tuples.

```
>>> list(zip(['a','b','c'],[1,2,3]))
```

[('a', 1), ('b', 2), ('c', 3)]

Here, it pairs items from the two lists and creates tuples with those. But it doesn't have to be lists.

```
>>> list(zip(('a','b','c'),(1,2,3)))
```

[('a', 1), ('b', 2), ('c', 3)]

Q.46. How do you calculate the length of a string?

This is simple. We call the function len() on the string we want to calculate the length of.

```
>>> len('Ayushi Sharma')
```

13

Q.47. Explain Python List Comprehension.

The *list comprehension in python* is a way to declare a list in one line of code. Let's take a look at one such example.

```
>>> [i for i in range(1,11,2)]
```

[1, 3, 5, 7, 9]

```
>>> [i*2 for i in range(1,11,2)]
```

[2, 6, 10, 14, 18]

Q.48. How do you get all values from a Python dictionary?

We saw previously, to get all keys from a dictionary, we make a call to the keys() method. Similarly, for values, we use the method values().

```
>>> 'd' in {'a':1,'b':2,'c':3,'d':4}.values()
```

False

```
>>> 4 in {'a':1,'b':2,'c':3,'d':4}.values()
```

True

Q.49. What if you want to toggle case for a Python string?

We have the swapcase() method from the str class to do just that.

```
>>> 'AyuShi'.swapcase()
```

'aYUsHI'

Let's apply some concepts now, shall we? Questions 50 through 52 assume the string 'I love Python'. You need to do the needful.

Q.50. Write code to print only upto the letter t.

```
>>> i=0
>>> while s[i]!='t':
print(s[i],end='')
i+=1
```

I love Py

Q.51. Write code to print everything in the string except the spaces.

```
>>> for i in s:
if i==' ': continue
print(i,end='')
```

IlovePython

Q.52. Now, print this string five times in a row.

```
>>> for i in range(6):
print(s)
```

I love Python
I love Python
I love Python
I love Python
I love Python
I love Python

Okay, moving on to more domains to conquer.

Q.53. What is the purpose of bytes() in Python?

bytes() is a built-in function in Python that returns an immutable bytes object. Let's take an example.

```
>>> bytes([2,4,8])
```

b'\x02\x04\x08'

```
>>> bytes(5)
```

b'\x00\x00\x00\x00\x00'

```
>>> bytes('world','utf-8')
```

b'world'

Q.54. What is a control flow statement?

A Python program usually starts to execute from the first line. From there, it moves through each statement just once and as soon as it's done with the last statement, it transactions the program. However, sometimes, we may want to take a more twisted path through the code. Control flow statements let us disturb the normal execution flow of a program and bend it to our will.

Q.55. Create a new list to convert the following list of number strings to a list of numbers.

nums=['22','68','110','89','31','12']

We will use the int() function with a list comprehension to convert these strings into integers and put them in a list.

```
>>> [int(i) for i in nums]
```

[22, 68, 110, 89, 31, 12]

Q.56. Given the first and last names of all employees in your firm, what data type will you use to store it?
I can use a dictionary to store that. It would be something like this-

{'first_name':'Ayushi','second_name':'Sharma'

Top Python Interview Questions and Answers

Q.57. 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.

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

```
>>> int(0b1010)
```

10
To convert a number into its binary form, we use bin().

```
>>> bin(0xf)
```

'0b1111'
Octal numbers may have digits from 0 to 7. We use the prefix 0o or 0O.

```
>>> oct(8)
```

'0o10'
Hexadecimal numbers may have digits from 0 to 15. We use the prefix 0x or 0X.

```
>>> hex(16)
```

'0x10'

```
>>> hex(15)
```

'0xf'
DataFlair's latest article on Python Numbers with Examples
Q.58. What does the following code output?

```
>>> def extendList(val, list=[]):
list.append(val)
return list
>>> list1 = extendList(10)
>>> list2 = extendList(123,[])
>>> list3 = extendList('a')
```

```
>>> list1,list2,list3
```

([10, 'a'], [123], [10, 'a'])

You'd expect the output to be something like this:

([10],[123],['a'])

Well, this is because the list argument does not initialize to its default value ([]) every time we make a call to the function. Once we define the function, it creates a new list. Then, whenever we call it again without a list argument, it uses the same list. This is because it calculates the expressions in the default arguments when we define the function, not when we call it.

Q.59. How many arguments can the range() function take?

The range() function in Python can take up to 3 arguments. Let's see this one by one.

a. One argument

When we pass only one argument, it takes it as the stop value. Here, the start value is 0, and the step value is +1.

```
>>> list(range(5))
```

[0, 1, 2, 3, 4]

```
>>> list(range(-5))
```

[]

```
>>> list(range(0))
```

[]

b. Two arguments

When we pass two arguments, the first one is the start value, and the second is the stop value.

```
>>> list(range(2,7))
```

[2, 3, 4, 5, 6]

```
>>> list(range(7,2))
```

[]

```
>>> list(range(-3,4))
```

[-3, -2, -1, 0, 1, 2, 3]

c. Three arguments

Here, the first argument is the start value, the second is the stop value, and the third is the step value.

```
>>> list(range(2,9,2))
```

[2, 4, 6, 8]

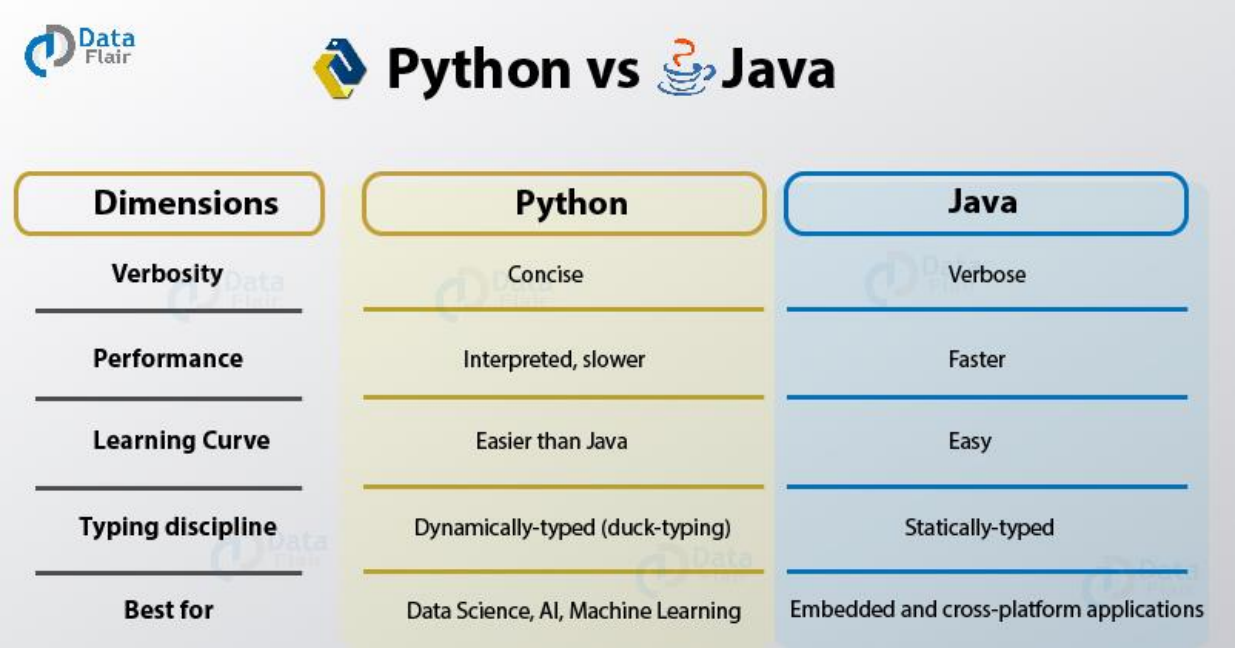
```
>>> list(range(9,2,-1))
```

[9, 8, 7, 6, 5, 4, 3]

Q.60. What is PEP 8?

PEP 8 is a coding convention that lets us write more readable code. In other words, it is a set of recommendations.

Q.61. How is Python different from Java?



Following is the comparison of Python vs Java –

- Java is faster than Python
- Python mandates indentation. Java needs braces.
- Python is dynamically-typed; Java is statically typed.
- Python is simple and concise; Java is verbose
- Python is interpreted
- Java is platform-independent
- Java has stronger database-access with JDBC

Python vs Java, the most commonly asked python interview question for freshers.

Learn it in detail – Python vs Java for Interview

Q.62. What is the best code you can write to swap two numbers?

I can perform the swapping in one statement.

```
>>> a,b=b,a
```

Here's the entire code, though-

```
>>> a,b=2,3
>>> a,b=b,a
>>> a,b
```

(3, 2)

Q.63. How can you declare multiple assignments in one statement?

This is one of the most asked interview questions for Python freshers –

There are two ways to do this:

First –

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

Second –

```
>>> a=b=c=3 #This assigns 3 to a, b, and c
```

Q.64. If you are ever stuck in an infinite loop, how will you break out of it?

For this, we press Ctrl+C. This interrupts the execution. Let's create an infinite loop to demonstrate this.

```
>>> def counterfunc(n):
while(n==7):print(n)
>>> counterfunc(7)
```

7
7
7
7
7
7
7
7
7
7
7
7
7
7
7
7

Traceback (most recent call last):
File "<pyshell#332>", line 1, in <module>
counterfunc(7)
File "<pyshell#331>", line 2, in counterfunc
while(n==7):print(n)
KeyboardInterrupt

Technical Python Interview Questions and Answers

Q.65. How do we execute Python?
Python files first compile to bytecode. Then, the host executes them.

Revise the concept of Python Compiler
Q.66. Explain Python's parameter-passing mechanism.
To pass its parameters to a function, Python uses pass-by-reference. If you change a parameter within a function, the change reflects in the calling function. This is its default behavior. However, when we pass literal arguments like strings, numbers, or tuples, they pass by value. This is because they are immutable.

Q.67. What is the with statement in Python?
The with statement in Python ensures that cleanup code is executed when working with unmanaged resources by encapsulating common preparation and cleanup tasks. It may be used to open a file, do something, and then automatically close the file at the end. It may be used to open a database connection, do some processing, then automatically close the connection to ensure resources are closed and available for others. with will cleanup the resources even if an exception is thrown. This statement is like the using statement in C#.
Consider you put some code in a try block, then in the finally block, you close any resources used. The with statement is like syntactic sugar for that.

The syntax of this control-flow structure is:

with expression [as variable]:
….with-block

```
>>> with open('data.txt') as data:
#processing statements
```

Q.68. How is a .pyc file different from a .py file?
While both files hold bytecode, .pyc is the compiled version of a Python file. It has platform-independent bytecode. Hence, we can execute it on any platform that supports the .pyc format. Python automatically generates it to improve performance(in terms of load time, not speed).

Python OOPS Interview Questions and Answers

Q.69. What makes Python object-oriented?
Again the frequently asked Python Interview Question

Python is object-oriented because it follows the Object-Oriented programming paradigm. This is a paradigm that revolves around classes and their instances (objects). With this kind of programming, we have the following features:

- Encapsulation

- Abstraction
- Inheritance
- Polymorphism
- Data hiding

Q.70. How many types of objects does Python support?

Objects in Python are mutable and immutable. Let's talk about these.

Immutable objects- Those which do not let us modify their contents. Examples of these will be tuples, booleans, strings, integers, floats, and complexes. Iterations on such objects are faster.

```
>>> tuple=(1,2,4)
>>> tuple
```

(1, 2, 4)

```
>>> 2+4j
```

(2+4j)

Mutable objects – Those that let you modify their contents. Examples of these are lists, sets, and dicts. Iterations on such objects are slower.

```
>>> [2,4,9]
```

[2, 4, 9]

```
>>> dict1={1:1,2:2}
>>> dict1
```

{1: 1, 2: 2}

While two equal immutable objects' reference variables share the same address, it is possible to create two mutable objects with the same content.

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

Ans: Python is a programming language with objects, modules, threads, exceptions and automatic memory management. The benefits of pythons are that it is simple and easy, portable, extensible, build-in data structure and it is an open source.

2. What is PEP 8?

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

3. What is pickling and unpickling?

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

4. How Python is interpreted?

Ans: Python language is an interpreted language. Python program runs directly from the source code. It converts the source code that is written by the programmer into an intermediate language, which is again translated into machine language that has to be executed.

5. How memory is managed in Python?

Ans: Python memory is managed by Python private heap space. All Python objects and data structures are located in a private heap. The programmer does not have an access to this private

heap and interpreter takes care of this Python private heap.
The allocation of Python heap space for Python objects is done by Python memory manager.
The core API gives access to some tools for the programmer to code.
Python also have an inbuilt garbage collector, which recycle all the unused memory and frees the memory and makes it available to the heap space.

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

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

7. What are Python decorators?

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

8. What is the difference between list and tuple?

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

9. How are arguments passed by value or by reference?

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

10. What is Dict and List comprehensions are?

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

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

Ans: There are mutable and Immutable types of Pythons built in types Mutable built-in types
List Sets
Dictionaries Immutable built-in types
Strings Tuples Numbers

12. What is namespace in Python?

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

13. What is lambda in Python?

Ans: It is a single expression anonymous function often used as In-line function.

14. Why lambda forms in python does not have statements?

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

15. What is pass in Python?

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

16. In Python what are iterators?

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

17. What is unit test in Python?

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

18. In Python what is slicing?

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

19. What are generators in Python?

Ans: The way of implementing iterators are known as generators. It is a normal function except that it yields expression in the function.

20. What is docstring in Python?

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

21. How can you copy an object in Python?

Ans: To copy an object in Python, you can try copy.copy () or copy.deepcopy() for the general case. You cannot copy all objects but most of them.

22. What is negative index in Python?

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

23. How you can convert a number to a string?

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

24. What is the difference between Xrange and range?

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

25. What is module and package in Python?

Ans: In Python, module is the way to structure program. Each Python program file is a module, which imports other modules like objects and attributes.
The folder of Python program is a package of modules. A package can have modules or subfolders.

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

Ans: Local variables: If a variable is assigned a new value anywhere within the function's body, it's assumed to be local.
Global variables: Those variables that are only referenced inside a function are implicitly global.

27. How can you share global variables across modules?

Ans: To share global variables across modules within a single program, create a special module. Import the config module in all modules of your application. The module will be available as a global variable across modules.

28. Explain how can you make a Python Script executable on Unix?To make a Python Script executable on Unix, you need to do two things,

Ans: Script file's mode must be executable and
the first line must begin with # (#!/usr/local/bin/python)

29. Explain how to delete a file in Python?

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

30. Explain how can you generate random numbers in Python?

Ans: To generate random numbers in Python, you need to import command as import random
random.random()
This returns a random floating point number in the range [0,1)

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

Ans: You can access a module written in Python from C by following method, Module = =PyImport_ImportModule("");

32. Mention the use of // operator in Python?

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

33. Mention five benefits of using Python?

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

34. Mention the use of the split function in Python?

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

35. Explain what is Flask & its benefits?

Ans: Flask is a web micro framework for Python based on "Werkzeug, Jinja 2 and good intentions" BSD licensed. Werkzeug and jingja are two of its dependencies.
Flask is part of the micro-framework. Which means it will have little to no dependencies on external libraries. It makes the framework light while there is little dependency to update and less security bugs.

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

Ans: Flask is a "micro framework" primarily build for a small application with simpler requirements. In flask, you have to use external libraries. Flask is ready to use.
Pyramid are build for larger applications. It provides flexibility and lets the developer use the right tools for their project. The developer can choose the database, URL structure, templating style and more. Pyramid is heavy configurable.
Like Pyramid, Django can also used for larger applications. It includes an ORM.

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

Ans: Flask-WTF offers simple integration with WTForms. Features include for Flask WTF are Integration with wtforms Secure form with csrf token Global csrf protection Internationalization integration Recaptcha supporting
File upload that works with Flask Uploads

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

Ans: The common way for the flask script to work is…
Either it should be the import path for your application Or the path to a Python file

39. Explain how you can access sessions in Flask?

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

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

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

```
from flask import Flaskapp =
Flask(_name_)

@app.route("/")
Def hello():
return "Hello World"
app.run(debug = True)
```

In this code your,

- Configuration part will be
  from flask import Flask
  app = Flask(_name_)
- View part will be
  @app.route("/")
  Def hello():
  return "Hello World"
- While you model or main part will be
  app.run(debug = True)

41. What type of a language is python? Interpreted or Compiled?

Ans: Beginner's Answer:
Python is an interpreted, interactive, objectoriented programming language.
Expert Answer:
Python is an interpreted language, as opposed to a compiled one, though the distinction can be blurry because of the presence of the bytecode compiler. This means that source files can be run directly without explicitly creating an executable which is then run.

42. What do you mean by python being an "interpreted language"? (Continues from previous question)

Ans: An interpreted languageis a programming languagefor which most of its implementations execute instructions directly, without previously compiling a program into machinelanguageinstructions. In context of Python, it means that Python program runs directly from the source code.

43. What is python's standard way of identifying a block of code?

Ans: Indentation.

44. Please provide an example implementation of a function called "my_func" that returns the square of a given variable "x". (Continues from previous question)

Ans:

```
defmy_func(x):
returnx**2
```

45. Is python statically typed or dynamically typed?

Ans: Dynamic.

In a statically typed language, the type of variables must be known (and usually declared) at the point at which it is used. Attempting to use it will be an error. In a dynamically typed language, objects still have a type, but it is determined at runtime. You are free to bind names (variables) to different objects with a different type. So long as you only perform operations valid for the type the interpreter doesn't care what type they actually are.

46. Is python strongly typed or weakly typed language?

Ans: Strong.

In a weakly typed language a compiler / interpreter will sometimes change the type of a variable. For example, in some languages (like JavaScript) you can add strings to numbers 'x' + 3 becomes 'x3'. This can be a problem because if you have made a mistake in your program, instead of raising an exception execution will continue but your variables now have wrong and unexpected values. In a strongly typed language (like Python) you can't perform operations inappropriate to the type of the object attempting to add numbers to strings will fail. Problems like these are easier to diagnose because the exception is raised at the point where the error occurs rather than at some other, potentially far removed, place.

47. Create a unicode string in python with the string "This is a test string"?

Ans: some_variable=u'Thisisateststring'

Or

some_variable=u"Thisisateststring"

48. What is the python syntax for switch case statements?

Ans: Python doesn't support switchcase statements. You can use ifelse statements for this purpose.

49. What is a lambda statement? Provide an example.

Ans: A lambda statement is used to create new function objects and then return them at runtime. Example:

my_func=lambdax:x**2

creates a function called my_func that returns the square of the argument passed.

50.What are the rules for local and global variables in Python?

Ans: If a variable is defined outside function then it is implicitly global. If variable is assigned new value inside the function means it is local. If we want to make it global we need to explicitly define it as global. Variable referenced inside the function are implicit global

51.What is the output of the following program?

Ans:

```
#!/usr/bin/python
deffun1(a):
print'a:',a
a=33;
print'locala:',a
a=100
fun1(a)
print'aoutsidefun1:',a
```

Ans. Output:

```
a:100
locala:33
aoutsidefun1:100
```

52.What is the output of the following program?

Ans:

```
#!/usr/bin/python
deffun2():
globalb
print'b:',b
b=33
print'globalb:',b
b=100
fun2()
print'boutsidefun2',b
```

Ans. Output:

```
b:100
globalb:33
boutsidefun2:33
```

53. What is the output of the following program?

Ans:

```
#!/usr/bin/python
deffoo(x,y):
globala
a=42
x,y=y,x
b=33
b=17
```

```
c=100
print(a,b,x,y)
a,b,x,y=1,15,3,4
foo(17,4)
print(a,b,x,y)
```

Ans.Output:

```
42 17 4 17
42 15 3 4
```

54.What is the output of the following program?

Ans:

```
#!/usr/bin/python
deffoo(x=[]):
x.append(1)
returnx
foo()
foo()
```

Output:

```
[1]
[1,1]
```

55. What is the purpose of #!/usr/bin/pythonon the first line in the above code? Is there any advantage?

Ans: By specifying #!/usr/bin/pythonyou specify exactly which interpreter will be used to run the script on a particular system. This is the hardcoded path to the python interpreter for that particular system. The advantage of this line is that you can use a specific python version to run your code.

56.What is the output of the following program?

Ans:

```
list=['a','b','c','d','e']
printlist[10]
```

Ans. Output:

IndexError.Or Error.

57.What is the output of the following program?

Ans:

```
list=['a','b','c','d','e']
printlist[10:]
```

Ans. Output:

```
[]
```

Theabovecodewilloutput[],andwillnotresultinanIndexError.
As one would expect, attempting to access a member of a list using an index that exceeds the number of members results in an IndexError.

58. What does this list comprehension do:

Ans:

```
[x**2forxinrange(10)ifx%2==0]
```

Ans. Creates the following list:

```
[0,4,16,36,64]
```

59. Do sets, dictionaries and tuples also support comprehensions?

Ans: Sets and dictionaries support it. However tuples are immutable and have generators but not comprehensions.

Set Comprehension:
r={xforxinrange(2,101)
ifnotany(x%y==0foryinrange(2,x))}
Dictionary Comprehension:
{i:jfori,jin{1:'a',2:'b'}.items()}
since
{1:'a',2:'b'}.items()returnsalistof2-Tuple.iisthefirstelement
oftuplejisthesecond.

60.What are some mutable and immutable data-types/datastructures in python?

Ans:
Mutable Types Immutable Types
Dictionary number
List boolean
string
tuple

61.What are generators in Python?

Ans: Generators are functions that return an iterable collection of items, one at a time, in a set manner. Generators, in general, are used to create iterators with a different approach. They employ the use of `yield` keyword rather than `return` to return a **generator object**.
Let's try and build a generator for fibonacci numbers –

```
## generate fibonacci numbers upto n
def fib(n):
    p, q = 0, 1
    while(p < n):
        yield p
        p, q = q, p + q

x = fib(10)     # create generator object


## iterating using __next__(), for Python2, use next()
x.__next__()    # output => 0
x.__next__()    # output => 1
x.__next__()    # output => 1
x.__next__()    # output => 2
x.__next__()    # output => 3
x.__next__()    # output => 5
x.__next__()    # output => 8
x.__next__()    # error


## iterating using loop
```

```
for i in fib(10):
    print(i)     # output => 0 1 1 2 3 5 8
```

62.What can you use Python generator functions for?

Ans: One of the reasons to use generator is to make the solution clearer for some kind of solutions.
The other is to treat results one at a time, avoiding building huge lists of results that you would process separated anyway.

63.When is not a good time to use python generators?

Ans: Use list instead of generator when:
1 You need to access the data multiple times (i.e. cache the results instead of recomputing them)
2 You need random access (or any access other than forward sequential order):
3 You need to join strings (which requires two passes over the data)
4 You are using PyPy which sometimes can't optimize generator code as much as it can with normal function calls and list manipulations.

64.What's your preferred text editor?

Ans: Emacs. (Any alternate answer leads to instant disqualification of the applicant)

65.When should you use generator expressions vs. list comprehensions in Python and vice-versa?

Ans: Iterating over the generator expression or the list comprehension will do the same thing. However, the list comp will create the entire list in memory first while the generator expression will create the items on the fly, so you are able to use it for very large (and also infinite!) sequences.

66. What is a negative index in Python?

Ans: Python arrays and list items can be accessed with positive or negative numbers. A negative Index accesses the elements from the end of the list counting backwards.
Example:
a=[123]
printa[-3]
printa[-2]
Outputs:
1
2

67. What is the difference between range and xrange functions?

Ans: Range returns a list while xrange returns an xrange object which take the same memory no matter of the range size. In the first case you have all items already generated (this can take a lot of time and memory). In Python 3 however, range is implemented with xrange and you have to explicitly call the list function if you want to convert it to a list.

68. How can I find methods or attributes of an object in Python?

Ans: Builtin dir() function of Python ,on an instance shows the instance variables as well as the methods and class attributes defined by the instance's class and all its base classes alphabetically. So by any object as argument to dir() we can find all the methods & attributes of the object's class

69. What is the statement that can be used in Python if a statement is required syntactically but the program requires no action?

Ans:

pass

70. Do you know what is the difference between lists and tuples? Can you give me an example for their usage?

Ans:

First list are mutable while tuples are not, and second tuples can be hashed e.g. to be used as keys for dictionaries. As an example of their usage, tuples are used when the order of the elements in the sequence matters e.g. a geographic coordinates, "list" of points in a path or route, or set of actions that should be executed in specific order. Don't forget that you can use them a dictionary keys. For everything else use lists

71. What is the function of "self"?

Ans:

"Self" is a variable that represents the instance of the object to itself. In most of the object oriented programming languages, this is passed to the methods as a hidden parameter that is defined by an object. But, in python it is passed explicitly. It refers to separate instance of the variable for individual objects. The variables are referred as "self.xxx".

72. How is memory managed in Python?

Ans:

Memory management in Python involves a private heap containing all Python objects and data structures. Interpreter takes care of Python heap and the programmer has no access to it. The allocation of heap space for Python objects is done by Python memory manager. The core API of Python provides some tools for the programmer to code reliable and more robust program. Python also has a builtin garbage collector which recycles all the unused memory. The gc module defines functions to enable /disable garbage collector:

gc.enable() Enables automatic garbage collection.

gc.disable()-Disables automatic garbage collection

73. What is __init__.py?

Ans:

It is used to import a module in a directory, which is called package import.

74. Print contents of a file ensuring proper error handling?

Ans:

try:

withopen('filename','r')asf:

printf.read()
exceptIOError:
print"Nosuchfileexists"

75 How do we share global variables across modules in Python?

Ans:
We can create a config file and store the entire global variable to be shared across modules in it. By simply importing config, the entire global variable defined will be available for use in other modules.
For example I want a, b & c to share between modules.
config.py :

```
a=0
b=0
c=0
```

module1.py:

```
importconfig
config.a=1
config.b=2
config.c=3
print"a,b&resp.are:",config.a,config.b,config.c
```

output of module1.py will be

```
123
```

76. Does Python support Multithreading?

Ans: Yes
Medium

77. How do I get a list of all files (and directories) in a given directory in Python?

Ans: Following is one possible solution there can be other similar ones:

```
import os
for dirname,dirnames,filenames in os.walk('.'):
#printpathtoallsubdirectoriesfirst.
forsubdirnameindirnames:
printos.path.join(dirname,subdirname)
#printpathtoallfilenames.
forfilenameinfilenames:
printos.path.join(dirname,filename)
#Advancedusage:
#editingthe'dirnames'listwillstopos.walk()fromrecursing
intothere.
if'.git'indirnames:
#don'tgointoany.gitdirectories.
dirnames.remove('.git')
```

78. How to append to a string in Python?

Ans: The easiest way is to use the += operator. If the string is a list of character, join() function can also be used.

79. How to convert a string to lowercase in Python?

Ans: use lower() function.
Example:

```
s='MYSTRING'
prints.lower()
```

80. How to convert a string to lowercase in Python?

Ans: Similar to the above question. use upper() function instead.

81. How to check if string A is substring of string B?

Ans: The easiest way is to use the in operator.

```
>>> 'abc' in 'abcdefg'
True
```

82. Find all occurrences of a substring in Python

Ans: There is no simple builtin string function that does what you're looking for, but you could use the more powerful regular expressions:

```
>>>[m.start()forminre.finditer('test','testtesttesttest')]
[0,5,10,15]//thesearestartingindicesforthestring
```

83. What is GIL? What does it do?Talk to me about the GIL. How does it impact concurrency in Python? What kinds of applications does it impact more than others?

Ans: Python's GIL is intended to serialize access to interpreter internals from different threads. On multicore systems, it means that multiple threads can't effectively make use of multiple cores. (If the GIL didn't lead to this problem, most people wouldn't care about the GIL it's only being raised as an issue because of the increasing prevalence of multicore systems.)

Note that Python's GIL is only really an issue for CPython, the reference implementation. Jython and IronPython don't have a GIL. As a Python developer, you don't generally come across the GIL unless you're writing a C extension. C extension writers need to release the GIL when their extensions do blocking I/O, so that other threads in the Python process get a chance to run.

84. Print the index of a specific item in a list?

Ans: use the index() function

```
>>>["foo","bar","baz"].index('bar')
1
```

.

85.How do you iterate over a list and pull element indices at the same time?

Ans: You are looking for the enumerate function. It takes each element in a sequence (like a list) and sticks it's location right before it. For example:

```
>>>my_list=['a','b','c']
>>>list(enumerate(my_list))
[(0,'a'),(1,'b'),(2,'c')]
```

Note that enumerate() returns an object to be iterated over, so wrapping it in list() just helps us see what enumerate() produces.

An example that directly answers the question is given below

```
my_list=['a','b','c']
fori,charinenumerate(my_list):
printi,char
```

The output is:

```
0a
1b
2c
```

86. How does Python's list.sort work at a high level? Is it stable? What's the runtime?

Ans: In early pythonversions, the sort function implemented a modified version of quicksort. However, it was deemed unstable and as of 2.3 they switched to using an adaptive mergesort algorithm.

87. What does the list comprehension do:

Ans:

my_list=[(x,y,z)forxinrange(1,30)foryinrange(x,30)forzin range(y,30)ifx**2+y**2==z**2]

It creates a list of tuples called my_list, where the first 2 elements are the perpendicular sides of right angle triangle and the third value 'z' is the hypotenuse.

[(3,4,5),(5,12,13),(6,8,10),(7,24,25),(8,15,17),(9,12,15), (10,24,26),(12,16,20),(15,20,25),(20,21,29)]

88. How can we pass optional or keyword parameters from one function to another in Python?

Ans:

Gather the arguments using the * and ** specifiers in the function's parameter list. This gives us positional arguments as a tuple and the keyword arguments as a dictionary. Then we can pass these arguments while calling another function by using * and **:

```
deffun1(a,*tup,**keywordArg):
…
keywordArg['width']='23.3c'
…
Fun2(a,*tup,**keywordArg)
```

89. Python How do you make a higher order function in Python?

Ans:

A higherorder function accepts one or more functions as input and returns a new function. Sometimes it is required to use function as data To make high order function , we need to import functools module The functools.partial() function is used often for high order function.

90. What is map?

Ans:

The syntax of map is:

map(aFunction,aSequence)

The first argument is a function to be executed for all the elements of the iterable given as the second argument. If the function given takes in more than 1 arguments, then many iterables are given.

91. Tell me a very simple solution to print every other element of this list?

Ans:

```
L=[0,10,20,30,40,50,60,70,80,90]
L[::2]
```

92. Are Tuples immutable?

Ans: Yes.

93. Why is not all memory freed when python exits?

Ans: Objects referenced from the global namespaces of Python modules are not always deallocated when Python exits. This may happen if there are circular references. There are also certain bits of memory that are allocated by the C library that are impossible to free (e.g. a tool like the one Purify will complain about these). Python is, however, aggressive about cleaning up memory on exit and does try to destroy every single object. If you want to force Python to delete certain things on deallocation, you can use the at exit module to register one or more exit functions to handle those deletions.

94. What is Java implementation of Python popularly know?

Ans: Jython.

95. What is used to create unicode strings in Python?

Ans:
Add u before the string.
u 'mystring'

96. What is a docstring?

Ans:
docstring is the documentation string for a function. It can be accessed by function_name.__doc__

97. Given the list below remove the repetition of an element.

Ans:
words=['one','one','two','three','three','two']
A bad solution would be to iterate over the list and checking for copies somehow and then remove them!
A very good solution would be to use the set type. In a Python set, duplicates are not allowed.
So, list(set(words)) would remove the duplicates.

98. Print the length of each line in the file 'file.txt' not including any whitespaces at the end of the lines?

Ans:
withopen("filename.txt","r")asf1:
printlen(f1.readline().rstrip())
rstrip() is an inbuilt function which strips the string from the right end of spaces or tabs (whitespace characters).

99. What is wrong with the code?

Ans:
func([1,2,3])#explicitlypassinginalist
func() #usingadefaultemptylist
deffunc(n=[]):
#dosomethingwithn
printn
This would result in a NameError. The variable n is local to function func and can't be accessesd outside. So, printing it won't be possible.

100. What does the below mean?

Ans:
s = a + '[' + b + ':' + c + ']'
seems like a string is being concatenated. Nothing much can be said without knowing types of variables a, b, c. Also, if all of the a, b, c are not of type string, TypeError would be raised. This is because of the string constants ('[' , ']') used in the statement.

101. What are Python decorators?

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

102. What is namespace in Python?

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

103. Explain the role of repr function.

Ans:
Python can convert any value to a string by making use of two functions repr() or str(). The str() function returns representations of values which are humanreadable, while repr() generates representations which can be read by the interpreter. repr() returns a machinereadable representation of values, suitable for an exec command.
Following code sniipets shows working of repr() & str() :

```
deffun():
y=2333.3
x=str(y)
z=repr(y)
print"y:",y
print"str(y):",x
print"repr(y):",z
fun()
————-
output
y:2333.3
str(y):2333.3
repr(y):2333.3000000000002
```

104. What is LIST comprehensions features of Python used for?

Ans:
LIST comprehensions features were introduced in Python version 2.0, it creates a new list based on existing list. It maps a list into another list by applying a function to each of the elements of the existing list. List comprehensions creates lists without using map() , filter() or lambda form.

105. Explain how to copy an object in Python.?

Ans:
There are two ways in which objects can be copied in python. Shallow copy & Deep copy. Shallow copies duplicate as minute as possible whereas Deep copies duplicate everything. If a is object to be copied then …
copy.copy(a) returns a shallow copy of a.
copy.deepcopy(a) returns a deep copy of a.

106. Describe how to send mail from a Python script?

Ans:
The smtplib module defines an SMTP client session object that can be used to send mail to any Internet machine.
A sample email is demonstrated below.

```
import smtplib
SERVER = smtplib.SMTP('smtp.server.domain')
FROM = sender@mail.com
TO = ["user@mail.com"] # must be a list
SUBJECT = "Hello!"
TEXT = "This message was sent with Python's smtplib."
# Main message
message = """
From: Lincoln < sender@mail.com >
To: CarreerRide user@mail.com
Subject: SMTP email msg
This is a test email. Acknowledge the email by responding.
""" % (FROM, ", ".join(TO), SUBJECT, TEXT)
server = smtplib.SMTP(SERVER)
server.sendmail(FROM, TO, message)
server.quit()
```

107. Which of the languages does Python resemble in its class syntax?

Ans: c++.

108. Python How to create a multidimensional list?

Ans: There are two ways in which Multidimensional list can be created:
By direct initializing the list as shown below to create myList below.

```
>>>myList=[[227,122,223],[222,321,192],[21,122,444]]
>>>printmyList[0]
>>>printmyList[1][2]
```

Output
[227, 122, 223]
192
The second approach is to create a list of the desired length first and then fill in each element with a newly created lists demonstrated below :

```
>>>list=[0]*3
>>>foriinrange(3):
>>>list[i]=[0]*2
>>>foriinrange(3):
>>>forjinrange(2):
>>>list[i][j]=i+j
>>>printlist
```

Output
[[0,1],[1,2],[2,3]]

109. Explain the disadvantages of python

Ans: Disadvantages of Python are: Python isn't the best for memory intensive tasks.
Python is interpreted language & is slow compared to C/C++ or Java.

110. Explain how to make Forms in python.

Ans. As python is scripting language forms processing is done by Python. We need to import cgi module to access form fields using FieldStorage class.
Every instance of class FieldStorage (for 'form') has the following attributes:
form.name: The name of the field, if specified.
form.filename: If an FTP transaction, the clientside filename.
form.value: The value of the field as a string.
form.file: file object from which data can be read.
form.type: The content type, if applicable.
form.type_options: The options of the 'contenttype' line of the HTTP request, returned as a dictionary.
form.disposition: The field 'contentdisposition'; None if unspecified.
form.disposition_options: The options for 'contentdisposition'.
form.headers: All of the HTTP headers returned as a dictionary.
A code snippet of form handling in python:

```
importcgi
form=cgi.FieldStorage()
ifnot(form.has_key("name")andform.has_key("age")):
print"<H1>Name&AgenotEntered</H1>"
print"FilltheName&Ageaccurately."
return
print"<p>name:",form["name"].value
print"<p>Age:",form["age"].value
```

111. Explain how python is interpreted.

Ans: Python program runs directly from the source code. Each type Python programs are executed code is required. Python converts source code written by the programmer into intermediate language which is again translated it into the native language machine language that is executed. So Python is an Interpreted language.

112. Explain how to overload constructors (or methods) in Python.?

Ans. _init__ () is a first

113.What is the difference between List & Tuple in Python.?

LIST vs TUPLES	
LIST	**TUPLES**
Lists are mutable i.e they can be edited.	Tuples are immutable (tuples are lists which can't be edited).
Lists are slower than tuples.	Tuples are faster than list.
Syntax: list_1 = [10, 'Chelsea', 20]	Syntax: tup_1 = (10, 'Chelsea' , 20)

114.What are the key features of Python?

Ans:

- Python is an **interpreted** language. That means that, unlike languages like *C* and its variants, Python does not need to be compiled before it is run. Other interpreted languages include *PHP* and *Ruby*.

- Python is **dynamically typed**, this means that you don't need to state the types of variables when you declare them or anything like that. You can do things like `x=111` and then `x="I'm a string"` without error
- Python is well suited to **object orientated programming** in that it allows the definition of classes along with composition and inheritance. Python does not have access specifiers (like C++'s `public`, `private`).
- In Python, **functions** are **first-class objects**. This means that they can be assigned to variables, returned from other functions and passed into functions. Classes are also first class objects
- **Writing Python code is quick** but running it is often slower than compiled languages. Fortunately, Python allows the inclusion of C based extensions so bottlenecks can be optimized away and often are. The numpy package is a good example of this, it's really quite quick because a lot of the number crunching it does isn't actually done by Python
- Python finds **use in many spheres** – web applications, automation, scientific modeling, big data applications and many more. It's also often used as "glue" code to get other languages and components to play nice.

115.How is Python an interpreted language?

Ans: An interpreted language is any programming language which is not in machine level code before runtime. Therefore, Python is an interpreted language.

116.How is memory managed in Python?

Ans:

1. Memory management in python is managed by ***Python private heap space***. All Python objects and data structures are located in a private heap. The programmer does not have access to this private heap. The python interpreter takes care of this instead.
2. The allocation of heap space for Python objects is done by Python's memory manager. The core API gives access to some tools for the programmer to code.
3. Python also has an inbuilt garbage collector, which recycles all the unused memory and so that it can be made available to the heap space.

117.What is PYTHONPATH?

Ans: It is an environment variable which is used when a module is imported. Whenever a module is imported, PYTHONPATH is also looked up to check for the presence of the imported modules in various directories. The interpreter uses it to determine which module to load.

118. What are python modules? Name some commonly used built-in modules in Python?

Ans: Python modules are files containing Python code. This code can either be functions classes or variables. A Python module is a .py file containing executable code.
Some of the commonly used built-in modules are:

- os
- sys

- math
- random
- data time
- JSON

119.What are local variables and global variables in Python?

Ans:
Global Variables:
Variables declared outside a function or in global space are called global variables. These variables can be accessed by any function in the program.
Local Variables:
Any variable declared inside a function is known as a local variable. This variable is present in the local space and not in the global space.
Example:

```
a=2
def add():
b=3
c=a+b
print(c)
add()
```

Output: 5
When you try to access the local variable outside the function add(), it will throw an error.

120. Is python case sensitive?

Ans: Yes. Python is a case sensitive language.

121.What is type conversion in Python?

Ans: Type conversion refers to the conversion of one data type iinto another.
int() – converts any data type into integer type
float() – converts any data type into float type
ord() – converts characters into integer
hex() – converts integers to hexadecimal
oct() – converts integer to octal
tuple() – This function is used to convert to a tuple.
set() – This function returns the type after converting to set.
list() – This function is used to convert any data type to a list type.
dict() – This function is used to convert a tuple of order (key,value) into a dictionary.
str() – Used to convert integer into a string.
complex(real,imag) – This functionconverts real numbers to complex(real,imag) number.

122. How to install Python on Windows and set path variable?

Ans: To install Python on Windows, follow the below steps:

- Install python from this link: https://www.python.org/downloads/

- After this, install it on your PC. Look for the location where PYTHON has been installed on your PC using the following command on your command prompt: cmd python.
- Then go to advanced system settings and add a new variable and name it as PYTHON_NAME and paste the copied path.
- Look for the path variable, select its value and select 'edit'.
- Add a semicolon towards the end of the value if it's not present and then type %PYTHON_HOME%

123. Is indentation required in python?

Ans: Indentation is necessary for Python. It specifies a block of code. All code within loops, classes, functions, etc is specified within an indented block. It is usually done using four space characters. If your code is not indented necessarily, it will not execute accurately and will throw errors as well.

124. What is the difference between Python Arrays and lists?

Ans: Arrays and lists, in Python, have the same way of storing data. But, arrays can hold only a single data type elements whereas lists can hold any data type elements.
Example:

```
import array as arr
My_Array=arr.array('i',[1,2,3,4])
My_list=[1,'abc',1.20]
print(My_Array)
print(My_list)
```

Output:
array('i', [1, 2, 3, 4]) [1, 'abc', 1.2]

125. What are functions in Python?

Ans:A function is a block of code which is executed only when it is called. To define a Python function, the **def** keyword is used.
Example:

```
def Newfunc():
print("Hi, Welcome to Edureka")
Newfunc(); #calling the function
```

Output: Hi, Welcome to Edureka

126.What is __init__?

Ans:__init__ is a method or constructor in Python. This method is automatically called to allocate memory when a new object/ instance of a class is created. All classes have the __init__ method. Here is an example of how to use it.

```
class Employee:
def __init__(self, name, age,salary):
```

```
        self.name = name
        self.age = age
        self.salary = 20000
E1 = Employee("XYZ", 23, 20000)
# E1 is the instance of class Employee.
# __init__ allocates memory for E1.
print(E1.name)
print(E1.age)
print(E1.salary)
```

Output:

XYZ
23
20000

127.What is a lambda function?

Ans: An anonymous function is known as a lambda function. This function can have any number of parameters but, can have just one statement.

Example:

```
a = lambda x,y : x+y
print(a(5, 6))
```

Output: 11

128. What is self in Python?

Ans: Self is an instance or an object of a class. In Python, this is explicitly included as the first parameter. However, this is not the case in Java where it's optional. It helps to differentiate between the methods and attributes of a class with local variables.

The self variable in the init method refers to the newly created object while in other methods, it refers to the object whose method was called.

129. How does break, continue and pass work?

Break	Allows loop termination when some condition is met and the control is transferred to the next statement.
Continue	Allows skipping some part of a loop when some specific condition is met and the control is transferred to the beginning of the loop

Pass	Used when you need some block of code syntactically, but you want to skip its execution. This is basically a null operation. Nothing happens when this is executed.

130. What does [::-1} do?

Ans: [::-1] is used to reverse the order of an array or a sequence.
For example:

```
import array as arr
My_Array=arr.array('i',[1,2,3,4,5])
My_Array[::-1]
```

Output: array('i', [5, 4, 3, 2, 1])
[::-1] reprints a reversed copy of ordered data structures such as an array or a list. the original array or list remains unchanged.

131. How can you randomize the items of a list in place in Python?

Ans: Consider the example shown below:

```
from random import shuffle
x = ['Keep', 'The', 'Blue', 'Flag', 'Flying', 'High']
shuffle(x)
print(x)
```

The output of the following code is as below.

```
['Flying', 'Keep', 'Blue', 'High', 'The', 'Flag']
```

132. What are python iterators?

Ans: Iterators are objects which can be traversed though or iterated upon.

133. How can you generate random numbers in Python?

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

```
import random
random.random
```

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

1. randrange(a, b): it chooses an integer and define the range in-between [a, b). It returns the elements by selecting it randomly from the range that is specified. It doesn't build a range object.

np.random.rand(3,2) [ ]

2. uniform(a, b): it chooses a floating point number that is defined in the range of [a,b).Iyt returns the floating point number
3. normalvariate(mean, sdev): it is used for the normal distribution where the mu is a mean and the sdev is a sigma that is used for standard deviation.
4. The Random class that is used and instantiated creates an independent multiple random number generators.

134. What is the difference between range & xrange?

Ans: For the most part, xrange and range are the exact same in terms of functionality. They both provide a way to generate a list of integers for you to use, however you please. The only difference is that range returns a Python list object and x range returns an xrange object.
This means that xrange doesn't actually generate a static list at run-time like range does. It creates the values as you need them with a special technique called yielding. This technique is used with a type of object known as generators. That means that if you have a really gigantic range you'd like to generate a list for, say one billion, xrange is the function to use.
This is especially true if you have a really memory sensitive system such as a cell phone that you are working with, as range will use as much memory as it can to create your array of integers, which can result in a Memory Error and crash your program. It's a memory hungry beast.

135. How do you write comments in python?

Ans: Comments in Python start with a # character. However, alternatively at times, commenting is done using docstrings(strings enclosed within triple quotes).
Example:

```
#Comments in Python start like this

print("Comments in Python start with a #")
```

Output: Comments in Python start with a #

136. What is pickling and unpickling?

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

137. What are the generators in python?

Ans: Functions that return an iterable set of items are called generators.

138. How will you capitalize the first letter of string?

Ans: In Python, the capitalize() method capitalizes the first letter of a string. If the string already consists of a capital letter at the beginning, then, it returns the original string.

139. How will you convert a string to all lowercase?

Ans: To convert a string to lowercase, lower() function can be used.
Example:

```
1
                    stg='ABCD'
2
                    print(stg.lower())
```

Output: abcd

140. How to comment multiple lines in python?

Ans: Multi-line comments appear in more than one line. All the lines to be commented are to be prefixed by a #. You can also a very good **shortcut method to comment multiple lines**. All you need to do is hold the ctrl key and **left click** in every place wherever you want to include a # character and type a # just once. This will comment all the lines where you introduced your cursor.

141.What are docstrings in Python?

Ans: Docstrings are not actually comments, but, they are ***documentation strings***. These docstrings are within triple quotes. They are not assigned to any variable and therefore, at times, serve the purpose of comments as well.

Example:

```
"""
Using docstring as a comment.
This code divides 2 numbers
"""
x=8
y=4
z=x/y
print(z)
```

Output: 2.0

141. What is the purpose of is, not and in operators?

Ans: Operators are special functions. They take one or more values and produce a corresponding result.

is: returns true when 2 operands are true (Example: “a” is ‘a’)

not: returns the inverse of the boolean value

in: checks if some element is present in some sequence

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

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

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

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

Ans:

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

143. What is a dictionary in Python?

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

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

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

```
1  print dict[Country]
```

```
India
```

```
1  print dict[Capital]
```

```
Delhi
```

```
1  print dict[PM]
```

```
Modi
```

144. How can the ternary operators be used in python?

Ans: The Ternary operator is the operator that is used to show the conditional statements. This consists of the true or false values with a statement that has to be evaluated for it.
Syntax:
The Ternary operator will be given as:
[on_true] if [expression] else [on_false]x, y = 25, 50big = x if x < y else y
Example:
The expression gets evaluated like if x<y else y, in this case if x<y is true then the value is returned as big=x and if it is incorrect then big=y will be sent as a result.

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

Ans: We use *args when we aren't sure how many arguments are going to be passed to a function, or if we want to pass a stored list or tuple of arguments to a function. **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.

147. What does len() do?

Ans: It is used to determine the length of a string, a list, an array, etc.
Example:

```
1
2  stg='ABCD'
   len(stg)
```

148. Explain split(), sub(), subn() methods of "re" module in Python.

Ans: To modify the strings, Python's "re" module is providing 3 methods. They are:

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

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

Ans: The sequences in Python are indexed and it consists of the positive as well as negative numbers. The numbers that are positive uses '0' that is uses as first index and '1' as the second index and the process goes on like that.
The index for the negative number starts from '-1' that represents the last index in the sequence and '-2' as the penultimate index and the sequence carries forward like the positive number.
The negative index is used to remove any new-line spaces from the string and allow the string to except the last character that is given as S[:-1]. The negative index is also used to show the index to represent the string in correct order.

150. What are Python packages?

Ans: Python packages are namespaces containing multiple modules.

151.How can files be deleted in Python?

Ans: To delete a file in Python, you need to import the OS Module. After that, you need to use the os.remove() function.
Example:

```
import os
os.remove("xyz.txt")
```

152. What are the built-in types of python?

Ans: Built-in types in Python are as follows –

- Integers
- Floating-point
- Complex numbers
- Strings
- Boolean
- Built-in functions

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

Ans:

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

4. NumPy array is faster and You get a lot built in with NumPy, FFTs, convolutions, fast searching, basic statistics, linear algebra, histograms, etc.

154. How to add values to a python array?

Ans: Elements can be added to an array using the **append()**, **extend()** and the **insert (i,x)** functions.

Example:

```
a=arr.array('d', [1.1 , 2.1 ,3.1] )
a.append(3.4)
print(a)
a.extend([4.5,6.3,6.8])
print(a)
a.insert(2,3.8)
print(a)
```

Output:

array('d', [1.1, 2.1, 3.1, 3.4])
array('d', [1.1, 2.1, 3.1, 3.4, 4.5, 6.3, 6.8])
array('d', [1.1, 2.1, 3.8, 3.1, 3.4, 4.5, 6.3, 6.8])

155. How to remove values to a python array?

Ans: Array elements can be removed using **pop()** or **remove()** method. The difference between these two functions is that the former returns the deleted value whereas the latter does not.

Example:

```
a=arr.array('d', [1.1, 2.2, 3.8, 3.1, 3.7, 1.2, 4.6])
print(a.pop())
print(a.pop(3))
a.remove(1.1)
print(a)
```

Output:

4.6
3.1
array('d', [2.2, 3.8, 3.7, 1.2])

156. Does Python have OOps concepts?

Ans: Python is an object-oriented programming language. This means that any program can be solved in python by creating an object model. However, Python can be treated as procedural as well as structural language.

157. What is the difference between deep and shallow copy?

Ans: *Shallow copy* is used when a new instance type gets created and it keeps the values that are copied in the new instance. Shallow copy is used to copy the reference pointers just like it copies the values. These references point to the original objects and the changes made in any member of the class will also affect the original copy of it. Shallow copy allows faster execution of the program and it depends on the size of the data that is used.

Deep copy is used to store the values that are already copied. Deep copy doesn't copy the reference pointers to the objects. It makes the reference to an object and the new object that is pointed by some other object gets stored. The changes made in the original copy won't affect any other copy that uses the object. Deep copy makes execution of the program slower due to making certain copies for each object that is been called.

158. How is Multithreading achieved in Python?

Ans:

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

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

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

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

160. What are Python libraries? Name a few of them.

Ans: Python libraries are a collection of Python packages. Some of the majorly used python libraries are – Numpy, Pandas, Matplotlib, Scikit-learn and many more.

161. What is split used for?

Ans: The split() method is used to separate a given string in Python.
Example:

```
a="KausalVikash python"
print(a.split())
```

Output: ['KausalVikash', 'python']

162. How to import modules in python?

Ans: Modules can be imported using the **import** keyword. You can import modules in three ways-
Example:

```
import array              #importing using the original module name
import array as arr       # importing using an alias name
from array import *  #imports everything present in the array module
```

163. Explain Inheritance in Python with an example.

Ans: Inheritance allows One class to gain all the members(say attributes and methods) of another class. Inheritance provides code reusability, makes it easier to create and maintain an application. The class from which we are inheriting is called super-class and the class that is inherited is called a derived / child class.
They are different types of inheritance supported by Python:

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

164. How are classes created in Python?

Ans: Class in Python is created using the **class** keyword.
Example:

```
class Employee:
def __init__(self, name):
self.name = name
E1=Employee("abc")
print(E1.name)
```

Output: abc

165. What is monkey patching in Python?

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

```
# m.py
class MyClass:
def f(self):
print "f()"
```

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

```
import m
def monkey_f(self):
```

```
print "monkey_f()"
m.MyClass.f = monkey_f
obj = m.MyClass()
obj.f()
```

The output will be as below:

```
monkey_f()
```

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

166. Does python support multiple inheritance?

Ans: Multiple inheritance means that a class can be derived from more than one parent classes. Python does support multiple inheritance, unlike Java.

167. What is Polymorphism in Python?

Ans: Polymorphism means the ability to take multiple forms. So, for instance, if the parent class has a method named ABC then the child class also can have a method with the same name ABC having its own parameters and variables. Python allows polymorphism.

168. Define encapsulation in Python?

Ans: Encapsulation means binding the code and the data together. A Python class in an example of encapsulation.

169. How do you do data abstraction in Python?

Ans: Data Abstraction is providing only the required details and hiding the implementation from the world. It can be achieved in Python by using interfaces and abstract classes.

170.Does python make use of access specifiers?

Ans: Python does not deprive access to an instance variable or function. Python lays down the concept of prefixing the name of the variable, function or method with a single or double underscore to imitate the behavior of protected and private access specifiers.

171. How to create an empty class in Python?

Ans: An empty class is a class that does not have any code defined within its block. It can be created using the *pass* keyword. However, you can create objects of this class outside the class itself. IN PYTHON THE PASS command does nothing when its executed. it's a null statement.

For example-

```
class a:
    pass
obj=a()
obj.name="xyz"
print("Name = ",obj.name)
```

Output:

```
Name =  xyz
```

172.What's The Process To Get The Home Directory Using '~' In Python?

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

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

Output:

```
/home/runner
```

173.How To Find Bugs Or Perform Static Analysis In A Python Application?

Ans:

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

174.When Is The Python Decorator Used?

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

175.Can Python be used for web client and web server side programming? And which one is best suited to Python?

Ans: Python is best suited for web server-side application development due to its vast set of features for creating business logic, database interactions, web server hosting etc. However, Python can be used as a web client-side application which needs some conversions for a browser to interpret the client side logic. Also, note that Python can be used to create desktop applications which can run as a standalone application such as utilities for test automation.

176. Mention at least 3-4 benefits of using Python over the other scripting languages such as Javascript.

Ans: Enlisted below are some of the benefits of using Python.

1. Application development is faster and easy.
2. Extensive support of modules for any kind of application development including data analytics/machine learning/math-intensive applications.
3. An excellent support community to get your answers.

177.What is the type () in Python?

Ans: The built-in method which decides the types of the variable at the program runtime is known as type() in Python. When a single argument is passed through it, then it returns given object type. When 3 arguments pass through this, then it returns a new object type.

178.What are the key points of Python?

Ans:

- Similar to PERL and PHP, Python is processed by the interpreter at runtime. Python supports Object-Oriented style of programming, which encapsulates code within objects.

- Derived from other languages, such as ABC, C, C++, Modula-3, SmallTalk, Algol-68, Unix shell, and other scripting languages.
- Python is copyrighted, and its source code is available under the GNU General Public License (GPL).
- Supports the development of many applications, from text processing to games.
- Works for scripting, embedded code and compiled the code.
- Detailed

179.How is memory managed in Python?

Ans: Memory is managed by the private heap space. All objects and data structures are located in a private heap, and the programmer has no access to it. Only the interpreter has access. Python memory manager allocates heap space for objects. The programmer is given access to some tools for coding by the core API. The inbuilt garbage collector recycles the unused memory and frees up the memory to make it available for the heap space.

180.What tools can help find bugs or perform the static analysis?

Ans: For performing Static Analysis, PyChecker is a tool that detects the bugs in source code and warns the programmer about the style and complexity. Pylint is another tool that authenticates whether the module meets the coding standard.

181.How Does Python Handle Memory Management?

Ans:

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

182.What Are The Principal Differences Between The Lambda And Def?

Ans:

Lambda Vs. Def.

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

183.Write A Reg Expression That Confirms An Email Id Using The Python Reg Expression Module "Re"?

Ans: Python has a regular expression module "re."
Check out the **"re"** expression that can check the email id for .com and .co.in subdomain.

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

184.What Do You Think Is The Output Of The Following Code Fragment? Is There Any Error In The Code?

Ans:

```
list = ['a', 'b', 'c', 'd', 'e']
print (list[10:])
```

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

185. Is There A Switch Or Case Statement In Python? If Not Then What Is The Reason For The Same?

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

186.What Is A Built-In Function That Python Uses To Iterate Over A Number Sequence?

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

```
for i in range(5):
    print(i)
```

The range() function accompanies two sets of parameters.

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

187.What Are The Optional Statements Possible Inside A Try-Except Block In Python?

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

- The "**else**" clause

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

188.What Is A String In Python?

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

189. What Is Slicing In Python?

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

190. What Is %S In Python?

Ans: Python has support for formatting any value into a string. It may contain quite complex expressions.
One of the common usages is to push values into a string with the %s format specifier. The formatting operation in Python has the comparable syntax as the C function printf() has.

191.What Is The Index In Python?

Ans: An index is an integer data type which denotes a position within an ordered list or a string. In Python, strings are also lists of characters. We can access them using the index which begins from zero and goes to the length minus one.
For example, in the string "Program," the indexing happens like this:

```
Program 0 1 2 3 4 5
```

192. What Is Docstring In Python?

Ans: A docstring is a unique text that happens to be the first statement in the following Python constructs:
Module, Function, Class, or Method definition.
A docstring gets added to the __doc__ attribute of the string object.

193.What Is A Function In Python Programming?

Ans: A function is an object which represents a block of code and is a reusable entity. It brings modularity to a program and a higher degree of code reusability.
Python has given us many built-in functions such as print() and provides the ability to create user-defined functions.

194. How Many Basic Types Of Functions Are Available In Python?

Ans: Python gives us two basic types of functions.
1. Built-in, and
2. User-defined.
The built-in functions happen to be part of the Python language. Some of these are print(), dir(), len(), and abs() etc.

195. How Do We Write A Function In Python?

Ans: We can create a Python function in the following manner.

Step-1: to begin the function, start writing with the keyword def and then mention the function name.
Step-2: We can now pass the arguments and enclose them using the parentheses. A colon, in the end, marks the end of the function header.
Step-3: After pressing an enter, we can add the desired Python statements for execution.

196. What Is A Function Call Or A Callable Object In Python?

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

197. What Is The Return Keyword Used For In Python?

Ans: The purpose of a function is to receive the inputs and return some output.
The return is a Python statement which we can use in a function for sending a value back to its caller.

198. What Is "Call By Value" In Python?

Ans: In call-by-value, the argument whether an expression or a value gets bound to the respective variable in the function.
Python will treat that variable as local in the function-level scope. Any changes made to that variable will remain local and will not reflect outside the function.

199. What Is "Call By Reference" In Python?

Ans: We use both "call-by-reference" and "pass-by-reference" interchangeably. When we pass an argument by reference, then it is available as an implicit reference to the function, rather than a simple copy. In such a case, any modification to the argument will also be visible to the caller.
This scheme also has the advantage of bringing more time and space efficiency because it leaves the need for creating local copies.
On the contrary, the disadvantage could be that a variable can get changed accidentally during a function call. Hence, the programmers need to handle in the code to avoid such uncertainty.

200. What Is The Return Value Of The Trunc() Function?

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

201. Is It Mandatory For A Python Function To Return A Value?

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

202. What Does The Continue Do In Python?

Ans: The continue is a jump statement in Python which moves the control to execute the next iteration in a loop leaving all the remaining instructions in the block unexecuted.
The continue statement is applicable for both the "while" and "for" loops.

203. What Is The Purpose Of Id() Function In Python?

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

```
Signature: id(object)
```

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

204. What Does The *Args Do In Python?

Ans: We use *args as a parameter in the function header. It gives us the ability to pass N (variable) number of arguments.
Please note that this type of argument syntax doesn't allow passing a named argument to the function.
Example of using the *args:

```
# Python code to demonstrate
# *args for dynamic arguments
```

```
def fn(*argList):
    for argx in argList:
        print (argx)

fn('I', 'am', 'Learning', 'Python')
```
The output:
```
I
am
Learning
Python
```

205.Does Python Have A Main() Method?

Ans: The main() is the entry point function which happens to be called first in most programming languages.
Since Python is interpreter-based, so it sequentially executes the lines of the code one-by-one. Python also does have a Main() method. But it gets executed whenever we run our Python script either by directly clicking it or starts it from the command line.
We can also override the Python default main() function using the Python if statement. Please see the below code.
```
print("Welcome")
print("__name__ contains: ", __name__)
def main():
    print("Testing the main function")
if __name__ == '__main__':
    main()
```
The output:
```
Welcome
__name__ contains:  __main__
Testing the main function
```

206. What Does The __ Name __ Do In Python?

Ans: The __name__ is a unique variable. Since Python doesn't expose the main() function, so when its interpreter gets to run the script, it first executes the code which is at level 0 indentation. To see whether the main() gets called, we can use the __name__ variable in an if clause compares with the value "__main__."

207. What Is The Purpose Of "End" In Python?

Ans: Python's print() function always prints a newline in the end. The print() function accepts an optional parameter known as the 'end.' Its value is '\n' by default. We can change the end character in a print statement with the value of our choice using this parameter.
```
# Example: Print a  instead of the new line in the end.
print("Let's learn" , end = ' ')
print("Python")

# Printing a dot in the end.
print("Learn to code from techbeamers" , end = '.')
print("com", end = ' ')
```
The output is:
```
Let's learn Python
Learn to code from techbeamers.com
```

208. When Should You Use The "Break" In Python?

Ans: Python provides a break statement to exit from a loop. Whenever the break hits in the code, the control of the program immediately exits from the body of the loop.
The break statement in a nested loop causes the control to exit from the inner iterative block.

209.What Is The Difference Between Pass And Continue In Python?

Ans: The continue statement makes the loop to resume from the next iteration.
On the contrary, the pass statement instructs to do nothing, and the remainder of the code executes as usual.

210. What Does The Len() Function Do In Python?

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

```
>>> some_string = 'techbeamers'
>>> len(some_string)
11
```

211. What Does The Chr() Function Do In Python?

Ans: The chr() function got re-added in Python 3.2. In version 3.0, it got removed.
It returns the string denoting a character whose Unicode code point is an integer.
For example, the chr(122) returns the string 'z' whereas the chr(1212) returns the string 'Ҽ'.

212. What Does The Ord() Function Do In Python?

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

```
>>> ord("z")
122
```

213. What Is Rstrip() In Python?

Ans: Python provides the rstrip() method which duplicates the string but leaves out the whitespace characters from the end.
The rstrip() escapes the characters from the right end based on the argument value, i.e., a string mentioning the group of characters to get excluded.
The signature of the rstrip() is:

```
str.rstrip([char sequence/pre>
#Example
test_str = 'Programming    '
# The trailing whitespaces are excluded
print(test_str.rstrip())
```

214.What Is Whitespace In Python?

Ans: Whitespace represents the characters that we use for spacing and separation.
They possess an "empty" representation. In Python, it could be a tab or space.

215. What Is Isalpha() In Python?

Ans: Python provides this built-in isalpha() function for the string handling purpose.
It returns True if all characters in the string are of alphabet type, else it returns False.

216. How Do You Use The Split() Function In Python?

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

```
#Example
str = 'pdf csv json'
print(str.split(" "))
print(str.split())
```

The output:

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

217. What Does The Join Method Do In Python?

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

218. What Does The Title() Method Do In Python?

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

```
#Example
str = 'lEaRn pYtHoN'
print(str.title())
```

The output:

```
Learn Python
```

Now, check out some general purpose Python interview questions.

219. What Makes The CPython Different From Python?

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

220. Which Package Is The Fastest Form Of Python?

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

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

221. What Is GIL In Python Language?

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

222. How Is Python Thread Safe?

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

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

223. How Does Python Manage The Memory?

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

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

224.What Is The Set Object In Python?

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

225. What Is The Use Of The Dictionary In Python?

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

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

226. Is Python List A Linked List?

Ans: A Python list is a variable-length array which is different from C-style linked lists. Internally, it has a contiguous array for referencing to other objects and stores a pointer to the array variable and its length in the list head structure.

Here are some Python interview questions on classes and objects

227.What Is Class In Python?

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

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

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

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

```
>>>class Human:  # Create the class
...     pass
>>>man = Human()  # Create the instance
>>>print(man)
<__main__.Human object at 0x0000000003559E10>
```

228. What Are Attributes And Methods In A Python Class?

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

```
>>> class Human:
...     profession = "programmer" # specify the attribute 'profession'
of the class
>>> man = Human()
>>> print(man.profession)
programmer
```

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

```
>>> class Human:
    profession = "programmer"
    def set_profession(self, new_profession):
        self.profession = new_profession
>>> man = Human()
>>> man.set_profession("Manager")
>>> print(man.profession)
Manager
```

229. How To Assign Values For The Class Attributes At Runtime?

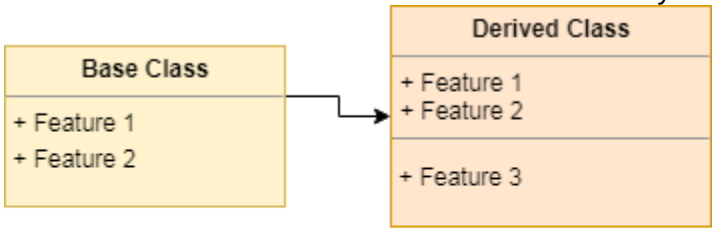
Ans: We can specify the values for the attributes at runtime. We need to add an init method and pass input to object constructor. See the following example demonstrating this.

```
>>> class Human:
    def __init__(self, profession):
        self.profession = profession
    def set_profession(self, new_profession):
        self.profession = new_profession

>>> man = Human("Manager")
>>> print(man.profession)
Manager
```

230.What Is Inheritance In Python Programming?

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



Inheritance In Python

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

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

```
class PC: # Base class
    processor = "Xeon" # Common attribute
    def set_processor(self, new_processor):
        processor = new_processor

class Desktop(PC): # Derived class
    os = "Mac OS High Sierra" # Personalized attribute
    ram = "32 GB"

class Laptop(PC): # Derived class
    os = "Windows 10 Pro 64" # Personalized attribute
    ram = "16 GB"

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

lap = Laptop()
print(lap.processor, lap.os, lap.ram)
```

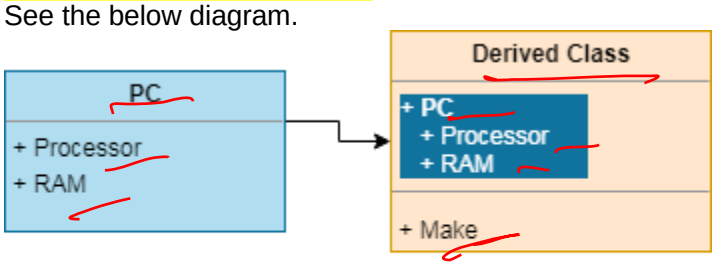
The output:

```
Xeon Mac OS High Sierra 32 GB
Xeon Windows 10 Pro 64 16 GB
```

231.What Is Composition In Python?

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

See the below diagram.



Composition In Python

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

```
class PC: # Base class
    processor = "Xeon" # Common attribute
    def __init__(self, processor, ram):
        self.processor = processor
        self.ram = ram

    def set_processor(self, new_processor):
        processor = new_processor

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

class Tablet():
    make = "Intel"
    def __init__(self, processor, ram, make):
        self.PC = PC(processor, ram) # Composition
```

```
        self.make = make

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

if __name__ == "__main__":
    tab = Tablet("i7", "16 GB", "Intel")
    print(tab.get_Tablet())
```

The output is:

```
Tablet with i7 CPU & 16 GB ram by Intel
```

232.What Are Errors And Exceptions In Python Programs?

Ans: Errors are coding issues in a program which may cause it to exit abnormally.
On the contrary, exceptions happen due to the occurrence of an external event which interrupts the normal flow of the program.

233. How Do You Handle Exceptions With Try/Except/Finally In Python?

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

```
try:
    print("Executing code in the try block")
    print(exception)
except:
    print("Entering in the except block")
finally:
    print("Reached to the final block")
```

The output is:

```
Executing code in the try block
Entering in the except block
Reached to the final block
```

234. How Do You Raise Exceptions For A Predefined Condition In Python?

Ans: We can raise an exception based on some condition.
For example, if we want the user to enter only odd numbers, else will raise an exception.

```
# Example - Raise an exception
while True:
    try:
        value = int(input("Enter an odd number- "))
        if value%2 == 0:
            raise ValueError("Exited due to invalid input!!!")
        else:
            print("Value entered is : %s" % value)
    except ValueError as ex:
        print(ex)
        break
```

The output is:

```
Enter an odd number- 2
Exited due to invalid input!!!
Enter an odd number- 1
Value entered is : 1
Enter an odd number-
```

235. What Are Python Iterators?

Ans: Iterators in Python are array-like objects which allow moving on the next element. We use them in traversing a loop, for example, in a “for” loop.
Python library has a no. of iterators. For example, a list is also an iterator and we can start a for loop over it.

236. What Is The Difference Between An Iterator And Iterable?

Ans: The collection type like a list, tuple, dictionary, and set are all iterable objects whereas they are also iterable containers which return an iterator while traversing.
Here are some advanced-level Python interview questions.

237. What Are Python Generators?

Ans: A Generator is a kind of function which lets us specify a function that acts like an iterator and hence can get used in a “for” loop.
In a generator function, the yield keyword substitutes the return statement.

```
# Simple Python function
def fn():
    return "Simple Python function."

# Python Generator function
def generate():
    yield "Python Generator function."

print(next(generate()))
```

The output is:

```
Python Generator function.
```

238.What Are Closures In Python?

Ans: Python closures are function objects returned by another function. We use them to eliminate code redundancy.
In the example below, we’ve written a simple closure for multiplying numbers.

```
def multiply_number(num):
    def product(number):
        'product() here is a closure'
        return num * number
    return product

num_2 = multiply_number(2)
print(num_2(11))
print(num_2(24))

num_6 = multiply_number(6)
print(num_6(1))
```

The output is:

```
22
48
6
```

239. What Are Decorators In Python?

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

```
def decorator_sample(func):
    def decorator_hook(*args, **kwargs):
        print("Before the function call")
        result = func(*args, **kwargs)
```

```
        print("After the function call")
        return result
    return decorator_hook

@decorator_sample
def product(x, y):
    "Function to multiply two numbers."
    return x * y

print(product(3, 3))
```

The output is:

```
Before the function call
After the function call
9
```

240. How Do You Create A Dictionary In Python?

Ans: Let's take the example of building site statistics. For this, we first need to break up the key-value pairs using a colon(":"). The keys should be of an immutable type, i.e., so we'll use the data-types which don't allow changes at runtime. We'll choose from an int, string, or tuple. However, we can take values of any kind. For distinguishing the data pairs, we can use a comma(",") and keep the whole stuff inside curly braces({…}).

```
>>> site_stats = {'site': 'tecbeamers.com', 'traffic': 10000, "type": "organic"}
>>> type(site_stats)
<class 'dict'>
>>> print(site_stats)
{'type': 'organic', 'site': 'tecbeamers.com', 'traffic': 10000}
```

241. How Do You Read From A Dictionary In Python?

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

```
>>> site_stats = {'site': 'tecbeamers.com', 'traffic': 10000, "type": "organic"}
>>> print(site_stats["traffic"])
```

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

```
>>> site_stats = {'site': 'tecbeamers.com', 'traffic': 10000, "type": "organic"}
>>> print(site_stats.get('site'))
tecbeamers.com
```

242. How Do You Traverse Through A Dictionary Object In Python?

Ans: We can use the "for" and "in" loop for traversing the dictionary object.

```
>>> site_stats = {'site': 'tecbeamers.com', 'traffic': 10000, "type": "organic"}
>>> for k, v in site_stats.items():
    print("The key is: %s" % k)
    print("The value is: %s" % v)
    print("++++++++++++++++++++++++")
```

The output is:

```
The key is: type
The value is: organic
++++++++++++++++++++++++
The key is: site
The value is: tecbeamers.com
++++++++++++++++++++++++
The key is: traffic
The value is: 10000
```

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

243. How Do You Add Elements To A Dictionary In Python?

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

```
>>> # Setup a blank dictionary
>>> site_stats = {}
>>> site_stats['site'] = 'google.com'
>>> site_stats['traffic'] = 10000000000
>>> site_stats['type'] = 'Referral'
>>> print(site_stats)
{'type': 'Referral', 'site': 'google.com', 'traffic': 10000000000}
```

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

```
>>> site_stats['site'] = 'google.co.in'
>>> print(site_stats)
{'site': 'google.co.in'}
>>> site_stats_new = {'traffic': 1000000, "type": "social media"}
>>> site_stats.update(site_stats_new)
>>> print(site_stats)
{'type': 'social media', 'site': 'google.co.in', 'traffic': 1000000}
```

244. How Do You Delete Elements Of A Dictionary In Python?

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

```
>>> site_stats = {'site': 'tecbeamers.com', 'traffic': 10000, "type":
"organic"}
>>> del site_stats["type"]
>>> print(site_stats)
{'site': 'google.co.in', 'traffic': 1000000}
```

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

```
>>> site_stats = {'site': 'tecbeamers.com', 'traffic': 10000, "type":
"organic"}
>>> print(site_stats.pop("type", None))
organic
>>> print(site_stats)
{'site': 'tecbeamers.com', 'traffic': 10000}
```

245. How Do You Check The Presence Of A Key In A Dictionary?

Ans: We can use Python's "in" operator to test the presence of a key inside a dict object.

```
>>> site_stats = {'site': 'tecbeamers.com', 'traffic': 10000, "type":
"organic"}
>>> 'site' in site_stats
True
>>> 'traffic' in site_stats
True
>>> "type" in site_stats
True
```

Earlier, Python also provided the has_key() method which got deprecated.

246. What Is The Syntax For List Comprehension In Python?

Ans: The signature for the list comprehension is as follows:

```
[ expression(var) for var in iterable ]
```

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

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

247. What Is The Syntax For Dictionary Comprehension In Python?

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

```
{ aKey, itsValue for aKey in iterable }
```

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

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

248. What Is The Syntax For Generator Expression In Python?

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

```
( expression(var) for var in iterable )
```

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

```
>>> (var for var in range(10, 20))
at 0x0000000003668728>
>>> list((var for var in range(10, 20)))
```

Now, see more Python interview questions for practice.

249. How Do You Write A Conditional Expression In Python?

Ans: We can utilize the following single statement as a conditional expression. default_statment if Condition else another_statement

```
>>> no_of_days = 366
>>> is_leap_year = "Yes" if no_of_days == 366 else "No"
>>> print(is_leap_year)
Yes
```

250. What Do You Know About The Python Enumerate?

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

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

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

```
enumerate(iterable, to_begin=0)
Arguments:
iterable: array type object which enables iteration
to_begin: the base index for the counter is to get started, its default
value is 0
# Example - enumerate function
alist = ["apple","mango", "orange"]
astr = "banana"

# Let's set the enumerate objects
list_obj = enumerate(alist)
str_obj = enumerate(astr)

print("list_obj type:", type(list_obj))
print("str_obj type:", type(str_obj))

print(list(enumerate(alist)) )
# Move the starting index to two from zero
print(list(enumerate(astr, 2)))
```

The output is:

```
list_obj type: <class 'enumerate'>
str_obj type: <class 'enumerate'>
[(0, 'apple'), (1, 'mango'), (2, 'orange')]
[(2, 'b'), (3, 'a'), (4, 'n'), (5, 'a'), (6, 'n'), (7, 'a')]
```

251. What Is The Use Of Globals() Function In Python?

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

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

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

```
Signature: globals()

Arguments: None
# Example: globals() function
x = 9
def fn():
    y = 3
    z = y + x
    # Calling the globals() method
    z = globals()['x'] = z
    return z

# Test Code
ret = fn()
print(ret)
```

The output is:

```
12
```

252. Why Do You Use The Zip() Method In Python?

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

```
Signature:
 zip(*iterators)
Arguments:
 Python iterables or collections (e.g., list, string, etc.)
Returns:
 A single iterator object with combined mapped values
# Example: zip() function

emp = [ "tom", "john", "jerry", "jake" ]
age = [ 32, 28, 33, 44 ]
dept = [ 'HR', 'Accounts', 'R&D', 'IT' ]

# call zip() to map values
out = zip(emp, age, dept)

# convert all values for printing them as set
out = set(out)

# Displaying the final values
print ("The output of zip() is : ",end="")
print (out)
```

The output is:

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

253. What Are Class Or Static Variables In Python Programming?

Ans: In Python, all the objects share common class or static variables.
But the instance or non-static variables are altogether different for different objects.
The programming languages like C++ and Java need to use the static keyword to make a variable as the class variable. However, Python has a unique way to declare a static variable.
All names initialized with a value in the class declaration becomes the class variables. And those which get assigned values in the class methods becomes the instance variables.

```
# Example
class Test:
    aclass = 'programming' # A class variable
    def __init__(self, ainst):
        self.ainst = ainst # An instance variable

# Objects of CSStudent class
test1 = Test(1)
test2 = Test(2)

print(test1.aclass)
print(test2.aclass)
print(test1.ainst)
print(test2.ainst)

# A class variable is also accessible using the class name
print(Test.aclass)
```

The output is:

```
programming
programming
1
2
programming
```

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

254. How Does The Ternary Operator Work In Python?

Ans: The ternary operator is an alternative for the conditional statements. It combines true or false values with a statement that you need to test.
The syntax would look like the one given below.
[onTrue] if [Condition] else [onFalse]

```
x, y = 35, 75
smaller = x if x < y else y
print(smaller)
```

255. What Does The "Self" Keyword Do?

Ans: The **self** is a Python keyword which represents a variable that holds the instance of an object.
In almost, all the object-oriented languages, it is passed to the methods as a hidden parameter.

256. What Are The Different Methods To Copy An Object In Python?

Ans: There are two ways to copy objects in Python.

- **copy.copy() function**
- It makes a copy of the file from source to destination.
- It'll return a shallow copy of the parameter.
- **copy.deepcopy() function**

- It also produces the copy of an object from the source to destination.
- It'll return a deep copy of the parameter that you can pass to the function.

257: What Is The Purpose Of Docstrings In Python?

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

258. Which Python Function Will You Use To Convert A Number To A String?

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

259. How Do You Debug A Program In Python? Is It Possible To Step Through The Python Code?

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

260. List Down Some Of The PDB Commands For Debugging Python Programs?

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

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

261. What Is The Command To Debug A Python Program?

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

```
$ python -m pdb python-script.py
```

262. How Do You Monitor The Code Flow Of A Program In Python?

Ans: In Python, we can use **the sys** module's **settrace()** method to setup trace hooks and monitor the functions inside a program.
You need to define a trace callback method and pass it to the **settrace()** function. The callback should specify three arguments as shown below.

```
import sys

def trace_calls(frame, event, arg):
    # The 'call' event occurs before a function gets executed.
    if event != 'call':
        return
    # Next, inspect the frame data and print information.
    print 'Function name=%s, line num=%s' % (frame.f_code.co_name,
frame.f_lineno)
    return

def demo2():
    print 'in demo2()'

def demo1():
```

```
    print 'in demo1()'
    demo2()

sys.settrace(trace_calls)
demo1()
```

263. How long can an identifier be in Python?

Ans: According to the official Python documentation, an identifier can be of any length. However, PEP 8 suggests that you should limit all lines to a maximum of 79 characters. Also, PEP 20 says 'readability counts'. So, a very long identifier will violate PEP-8 and PEP-20.
Apart from that, there are certain rules we must follow to name one:
According to the official Python documentation, an identifier can be of any length. However, PEP 8 suggests that you should limit all lines to a maximum of 79 characters. Also, PEP 20 says 'readability counts'. So, a very long identifier will violate PEP-8 and PEP-20.
Apart from that, there are certain rules we must follow to name one:

- It can only begin with an underscore or a character from A-Z or a-z.
- The rest of it can contain anything from the following: A-Z/a-z/_/0-9.
- Python is case-sensitive, as we discussed in the previous question.
- 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	nonlocal

264. How would you convert a string into lowercase?

Ans: We use the lower() method for this.

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

265. 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

266. Explain help() and dir() functions in Python?

Ans:
The help() function displays the documentation 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__', '__getattribute__', '__globals__', '__gt__', '__hash__', '__init__', '__init_subclass__', '__kwdefaults__', '__le__', '__lt__', '__module__', '__name__', '__ne__', '__new__', '__qualname__', '__reduce__', '__reduce_ex__', '__repr__', '__setattr__', '__sizeof__', '__str__', '__subclasshook__']

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

Ans: 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'])

268. How will you check if all characters in a string are alphanumeric?

Ans: For this, we use the method isalnum().

269. With Python, how do you find out which directory you are currently in?

Ans: To find this, we use the function/method getcwd(). We import it from the module os.

1. >>> import os
2. >>> os.getcwd()
   'C:\\Users\\lifei\\AppData\\Local\\Programs\\Python\\Python36-32'

1. >>> type(os.getcwd)
   <class 'builtin_function_or_method'>
   We can also change the current working directory with chdir().

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

270. How do you insert an object at a given index in Python?

Ans: Let's build a list first.

1 >>> a=[1,2,4]
Now, we use the method insert. The first argument is the index at which to insert, the second is the value to insert.

1. >>> a.insert(2,3)
2. >>> a
   [1, 2, 3, 4]

271. how do you reverse a list?

Ans: Using the reverse() method.

1. >>> a.reverse()
2. >>> a
   [4, 3, 2, 1]
   You can also do it via slicing from right to left:

1. >>> a[::-1]
2. >>> a
   [1, 2, 3, 4]
   This gives us the original list because we already reversed it once. However, this does not modify the original list to reverse it.

272. **How does a function return values?**

Ans: A function uses the 'return' keyword to return a value. Take a look:

1. >>> def add(a,b):
2. return a+b

273. **How would you define a block in Python?**

Ans: For any kind of statements, we possibly need to define a block of code under them. However, Python does not support curly braces. This means we must end such statements with colons and then indent the blocks under those with the same amount.

1. >>> if 3>1:
2. print("Hello")
3. print("Goodbye")
   Hello
   Goodbye

274. Will the do-while loop work if you don't end it with a semicolon?

Ans: Trick question! Python does not support an intrinsic do-while loop. Secondly, to terminate do-while loops is a necessity for languages like C++.

275. **In one line, show us how you'll get the max alphabetical character from a string.?**

Ans: For this, we'll simply use the max function.

1. >>> max('flyiNg')
   'y'
   The following are the ASCII values for all the letters of this string-
   f- 102
   l- 108
   y- 121
   i- 105
   N- 78
   g- 103
   By this logic, try to explain the following line of code-
1. >>> max('fly{}iNg')
   '}'
   (Bonus: } – 125)

276. **What is Python good for?**

Ans: Python is a jack of many trades, check out Applications of Python to find out more. Meanwhile, we'll say we can use it for:

- Web and Internet Development
- Desktop GUI
- Scientific and Numeric Applications
- Software Development Applications
- Applications in Education
- Applications in Business
- Database Access
- Network Programming
- Games, 3D Graphics
- Other Python Applications

277. Can you name ten built-in functions in Python and explain each in brief?

Ans: Ten Built-in Functions, you say? Okay, here you go.
complex()- Creates a complex number.

1. >>> complex(3.5,4)
   (3.5+4j)

eval()- Parses a string as an expression.

1. >>> eval('print(max(22,22.0)-min(2,3))')
   20

filter()- Filters in items for which the condition is true.

1. >>> list(filter(lambda x:x%2==0,[1,2,0,False]))
   [2, 0, False]

format()- Lets us format a string.

1. >>> print("a={0} but b={1}".format(a,b))
   a=2 but b=3

hash()- Returns the hash value of an object.

1. >>> hash(3.7)
   644245917

hex()- Converts an integer to a hexadecimal.

1. >>> hex(14)
   '0xe'

input()- Reads and returns a line of string.

1. >>> input('Enter a number')
   Enter a number7
1. '7'

len()- Returns the length of an object.

1. >>> len('Ayushi')
   6

locals()- Returns a dictionary of the current local symbol table.

1. >>> locals()
   {'__name__': '__main__', '__doc__': None, '__package__': None, '__loader__': <class '_frozen_importlib.BuiltinImporter'>, '__spec__': None, '__annotations__': {}, '__builtins__': <module 'builtins' (built-in)>, 'a': 2, 'b': 3}

open()- Opens a file.

1. >>> file=open('tabs.txt')

278. How will you convert a list into a string?

Ans: We will use the join() method for this.

1. >>> nums=['one','two','three','four','five','six','seven']
2. >>> s=' '.join(nums)
3. >>> s
   o/p= 'one two three four five six seven'

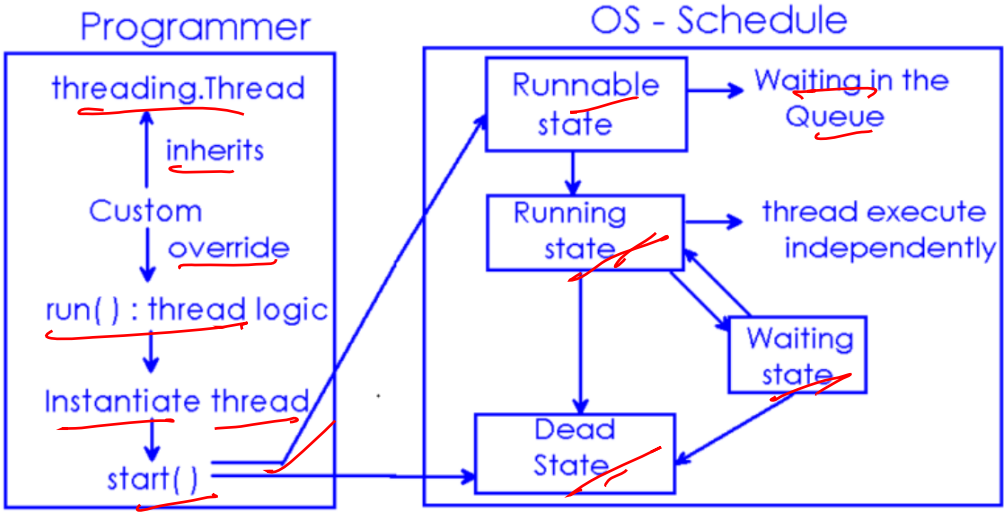
279. How will you remove a duplicate element from a list?

Ans: We can turn it into a set to do that.

1. >>> list=[1,2,1,3,4,2]
2. >>> set(list)
   {1, 2, 3, 4}

280. Can you explain the life cycle of a thread?

Ans:



- To create a thread, we create a class that we make override the run method of the thread class. Then, we instantiate it.
- A thread that we just created is in the new state. When we make a call to start() on it, it forwards the threads for scheduling. These are in the ready state.
- When execution begins, the thread is in the running state.
- Calls to methods like sleep() and join() make a thread wait. Such a thread is in the waiting/blocked state.
- When a thread is done waiting or executing, other waiting threads are sent for scheduling.
- A running thread that is done executing terminates and is in the dead state.

281. Explain the //, %, and ** operators in Python?

Ans: 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

282. **What are membership operators?**

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

1. >>> 'me' in 'disappointment'
   True

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

283. **Explain identity operators in Python?**

Ans: 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

284. **Finally, tell us about bitwise operators in Python?**
Ans:

Python Bitwise Operators

Operator	Description
&	Binary AND
|	Binary OR
^	Binary XOR
~	Binary One's Complement
<<	Binary Left-Shift
>>	Binary Right-Shift

These operate on values bit by bit.
AND (&) This performs & on each bit pair.

1. >>> 0b110 & 0b010
   2
   OR (|) This performs | on each bit pair.

1. >>> 3|2

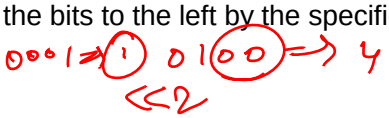
3
XOR (^) This performs an exclusive-OR operation on each bit pair.

1. >>> 3^2
1

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

1. >>> ~2
-3

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.
Binary Right-Shift (>>)

1. >>> 4>>2
1

285. What data types does Python support?

Ans: Python provides us with five kinds of data types:
Numbers – Numbers use to hold numerical values.

1. >>> a=7.0
2. >>>

Strings – A string is a sequence of characters. We declare it using single or double quotes.

1. >>> title="Ayushi's Book"

Lists – A list is an ordered collection of values, and we declare it using square brackets.

1. >>> colors=['red','green','blue']
2. >>> type(colors)
<class 'list'>

Tuples – A tuple, like a list, is an ordered collection of values. The difference. However, is that a tuple is immutable. This means that we cannot change a value in it.

1. >>> name=('Ayushi','Sharma')
2. >>> name[0]='Avery'
Traceback (most recent call last):
File "<pyshell#129>", line 1, in <module>
name[0]='Avery'
TypeError: 'tuple' object does not support item assignment

Dictionary – A dictionary is a data structure that holds key-value pairs. We declare it using curly braces.

1. >>> squares={1:1,2:4,3:9,4:16,5:25}
2. >>> type(squares)
<class 'dict'>
1. >>> type({})
<class 'dict'>

We can also use a dictionary comprehension:

1. >>> squares={x:x**2 for x in range(1,6)}
2. >>> squares
{1: 1, 2: 4, 3: 9, 4: 16, 5: 25}

286. How would you convert a string into an int in Python?

Ans: If a string contains only numerical characters, you can convert it into an integer using the int() function.

1. >>> int('227')
   227
   Let's check the types:

1. >>> type('227')
   <class 'str'>

1. >>> type(int('227'))
   <class 'int'>

287. How do you take input in Python?

Ans: For taking input from the user, we have the function input(). In Python 2, we had another function raw_input().
The input() function takes, as an argument, the text to be displayed for the task:

1. >>> a=input('Enter a number')
   Enter a number7
   But if you have paid attention, you know that it takes input in the form of a string.

1. >>> type(a)
   <class 'str'>
   Multiplying this by 2 gives us this:

1. >>> a*=2

2. >>> a
   '77'
   So, what if we need to work on an integer instead?
   We use the int() function for this.

1. >>> a=int(input('Enter a number'))
   Enter a number7
   Now when we multiply it by 2, we get this:

1. >>> a*=2

2. >>> a
   14

288. What is a function?

Ans: When we want to execute a sequence of statements, we can give it a name. Let's define a function to take two numbers and return the greater number.

1. >>> def greater(a,b):

2. return a if a>b else b

3. >>> greater(3,3.5)
   3.5

289. What is recursion?

Ans: When a function makes a call to itself, it is termed ***recursion***. But then, in order for it to avoid forming an infinite loop, we must have a base condition.

Let's take an example.

1. >>> def facto(n):
2. if n==1: return 1
3. return n*facto(n-1)
4. >>> facto(4)
   24

290. What do you know about relational operators in Python?

Ans:

Python Relational Operators

Operator	Description
<	Less than
>	Greater than
<=	Less than or equal to
>=	Greater than or equal to
=	Equal to
!=	Not equal to

Relational operators compare values.
Less than (<) If the value on the left is lesser, it returns True.

1. >>> 'hi'<'Hi'
   False
   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.
   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
   Greater than or equal to (>=) If the value on the left is greater than or equal to, it returns True.
1. >>> True>=False
   True
   Equal to (==) If the two values are equal, it returns True.
1. >>> {1,3,2,2}=={1,2,3}
   True
   Not equal to (!=) If the two values are unequal, it returns True.
1. >>> True!=0.1
   True
1. >>> False!=0.1
   True

291. **What are assignment operators in Python?**

Ans:

Python Assignment Operators

Operator	Description
=	Assign
+=	Add and Assign
-=	Subtract and Assign
*=	Multiply and Assign
/=	Divide and Assign
%=	Modulus and Assign
**=	Exponent and Assign
//=	Floor-Divide and Assign

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
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

292. **Explain logical operators in Python.?**

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

1. >>> False and True
   False

1. >>> 7<7 or True
   True

1. >>> not 2==2
   False

293. What does the function zip() do?

Ans: One of the less common functions with beginners, zip() returns an iterator of tuples.

1. >>> list(zip([‘a’,‘b’,‘c’],[1,2,3]))
   [(‘a’, 1), (‘b’, 2), (‘c’, 3)]
   Here, it pairs items from the two lists and creates tuples with those. But it doesn’t have to be lists.

1. >>> list(zip((‘a’,‘b’,‘c’),(1,2,3)))
   [(‘a’, 1), (‘b’, 2), (‘c’, 3)]

294. How can you declare multiple assignments in one statement?

Ans: There are two ways to do this:
First –

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

1. >>> a=b=c=3 #This assigns 3 to a, b, and c

295. If you are ever stuck in an infinite loop, how will you break out of it?

Ans: For this, we press Ctrl+C. This interrupts the execution. Let’s create an infinite loop to demonstrate this.

1. >>> def counterfunc(n):
2. while(n==7):print(n)
3. >>> counterfunc(7)
   7
   7
   7
   7
   7
   7
   7
   7
   7
   7
   7
   7
   7
   7
   7
   7
   7
   7
   Traceback (most recent call last):
   File “<pyshell#332>”, line 1, in <module>
   counterfunc(7)
   File “<pyshell#331>”, line 2, in counterfunc
   while(n==7):print(n)

KeyboardInterrupt

296. How is a .pyc file different from a .py file?

Ans: While both files hold bytecode, **.pyc** is the compiled version of a Python file. It has platform-independent bytecode. Hence, we can execute it on any platform that supports the **.pyc** format. Python automatically generates it to improve performance(in terms of load time, not speed).

297. How many types of objects does Python support?

Ans: Immutable objects- Those which do not let us modify their contents. Examples of these will be tuples, booleans, strings, integers, floats, and complexes. Iterations on such objects are faster.

1. >>> tuple=(1,2,4)
2. >>> tuple
   (1, 2, 4)

1. >>> 2+4j
   (2+4j)

Mutable objects – Those that let you modify their contents. Examples of these are lists, sets, and dicts. Iterations on such objects are slower.

1. >>> [2,4,9]
   [2, 4, 9]

1. >>> dict1={1:1,2:2}
2. >>> dict1
   {1: 1, 2: 2}

While two equal immutable objects' reference variables share the same address, it is possible to create two mutable objects with the same content.

298. When is the else part of a try-except block executed?

Ans: In an if-else block, the else part is executed when the condition in the if-statement is False. But with a try-except block, the else part executes only if no exception is raised in the try part.

299. Explain join() and split() in Python?

Ans:

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

1. >>> ','.join('12345')
   '1,2,3,4,5'

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

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

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

Ans:

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

a. filter()

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

1. >>> list(filter(lambda x:x>5,range(8)))
   [6, 7]
   b. map()
   Map applies a function to every element in an iterable.

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

1. [0, 1, 4, 9, 16, 25, 36, 49]
   c. reduce()
   Reduce repeatedly reduces a sequence pair-wise until we reach a single value.

1. >>> from functools import reduce

2. >>> reduce(lambda x,y:x-y,[1,2,3,4,5])
   -13

301. Is del the same as remove()? What are they?

Ans:
del and remove() are methods on lists/ ways to eliminate elements.

1. >>> list=[3,4,5,6,7]

2. >>> del list[3]

3. >>> list
   [3, 4, 5, 7]

1. >>> list.remove(5)

2. >>> list
   [3, 4, 7]
   While del lets us delete an element at a certain index, remove() lets us remove an element by its value.

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

Ans: Sometimes, when we want to iterate over a list, a few methods come in handy.
a. filter()
Filter lets us filter in some values based on conditional logic.

1. >>> list(filter(lambda x:x>5,range(8)))
   [6, 7]
   b. map()
   Map applies a function to every element in an iterable.

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

1. [0, 1, 4, 9, 16, 25, 36, 49]
   c. reduce()
   Reduce repeatedly reduces a sequence pair-wise until we reach a single value.

1. >>> from functools import reduce

2. >>> reduce(lambda x,y:x-y,[1,2,3,4,5])
   -13

304. How do you open a file for writing?

Ans: Let's create a text file on our Desktop and call it tabs.txt. To open it to be able to write to it, use the following line of code-

1. >>> file=open('tabs.txt','w')
   This opens the file in writing mode. You should close it once you're done.
1. >>> file.close()

305. Difference between the append() and extend() methods of a list.

Ans: The methods append() and extend() work on lists. While append() adds an element to the end of the list, extend adds another list to the end of a list.
Let's take two lists.

1. >>> list1,list2=[1,2,3],[5,6,7,8]
   This is how append() works:
1. >>> list1.append(4)
2. >>> list1
   [1, 2, 3, 4]
   And this is how extend() works:
1. >>> list1.extend(list2)
2. >>> list1
   [1, 2, 3, 4, 5, 6, 7, 8]

306. What are the different file-processing modes with Python?

We have the following modes-

- read-only – 'r'
- write-only – 'w'
- read-write – 'rw'
- append – 'a'
  We can open a text file with the option 't'. So to open a text file to read it, we can use the mode 'rt'. Similarly, for binary files, we use 'b'.

307. What does the map() function do?

Ans: map() executes the function we pass to it as the first argument; it does so on all elements of the iterable in the second argument. Let's take an example, shall we?

1. >>> for i in map(lambda i:i**3, (2,3,7)):
2. print(i)
   8
   27
   343
   This gives us the cubes of the values 2, 3, and 7.

308. Is there a way to remove the last object from a list?

Yes, there is. Try running the following piece of code-

1. >>> list=[1,2,3,4,5

2. >>> list.pop(–1)
   5

1. >>> list
   [1, 2, 3, 4]

309. How will you convert an integer to a Unicode character?

Ans: This is simple. All we need is the chr(x) built-in function. See how.

1. >>> chr(52)
   '4'

1. >>> chr(49)
   '1'

1. >>> chr(67)
   'C'

310. So does recursion cause any trouble?

Ans: Sure does:

- Needs more function calls.
- Each function call stores a state variable to the program stack- consumes memory, can cause memory overflow.
- Calling a function consumes time.

311. What good is recursion?

Ans: With ***recursion***, we observe the following:

- Need to put in less efforts.
- Smaller code than that by loops.
- Easier-to-understand code.

312. Can you remove the whitespaces from the string "aaa bbb ccc ddd eee"?

Ans: I can think of three ways to do this.
Using join-

1. >>> s='aaa bbb ccc ddd eee'
2. >>> s1=''.join(s.split())
3. >>> s1
   'aaabbbcccdddeee'

Using a ***list comprehension***–

1. >>> s='aaa bbb ccc ddd eee'
2. >>> s1=str(''.join(([i for i in s if i!=' '])))
3. >>> s1
   'aaabbbcccdddeee'

Using replace()-

1. >>> s='aaa bbb ccc ddd eee'
2. >>> s1 = s.replace(' ','')
3. >>> s1
   'aaabbbcccdddeee'

313. How do you get the current working directory using Python?

Ans: Working on software with Python, you may need to read and write files from various directories. To find out which directory we're presently working under, we can borrow the getcwd() method from the os module.

1. >>> import os
2. >>> os.getcwd()
   'C:\\Users\\Raj\\AppData\\Local\\Programs\\Python\\Python37-32'

314. What are the file-related modules we have in Python?

Ans: We have the following libraries and modules that let us manipulate text and binary files on our file systems-

os
os.path
shutil

315. Explain the uses of the modules sqlite3, ctypes, pickle, traceback, and itertools.

- sqlite3- Helps with handling databases of type SQLite
- ctypes- Lets create and manipulate C data types in Python
- pickle- Lets put any data structure to external files
- traceback- Allows extraction, formatting, and printing of stack traces
- itertools– Supports working with permutations, combinations, and other useful iterables.

316. How will you print the contents of a file?

1. >>> try:

2. with open('tabs.txt','r') as f:
3. print(f.read())
4. except IOError:
5. print("File not found")

317. What is Virtualenv in Python?

Ans: virtualenv is a tool to create isolated Python environments. virtualenv creates a folder which contains all the necessary executables to use the packages that a Python project would need. It can be used standalone, in place of Pipenv.
Install virtualenv via pip: $ pip install **virtualenv.**

318. What is the function of "self"?

Ans: Self is a variable that represents the instance of the object to itself. In most of the object oriented programming language, this is passed to the methods as a hidden parameters that is defined by an object. But, in python, it is declared and passed explicitly. It is the first argument that gets created in the instance of the class A and the parameters to the methods are passed automatically. It refers to separate instance of the variable for individual objects.
Let's say you have a class `ClassA` which contains a method `methodA` defined as:
`def methodA(self, arg1, arg2): #do something`
and `ObjectA` is an instance of this class.
Now when `ObjectA.methodA(arg1, arg2)` is called, python internally converts it for you as:
`ClassA.methodA(ObjectA, arg1, arg2)`
The `self` variable refers to the object itself.

319. What does the Python nonlocal statement do (in Python 3.0 and later)?

Ans: In short, it lets you assign values to a variable in an outer (but non-global) scope.
The `nonlocal` statement causes the listed identifiers to refer to previously bound variables in the nearest enclosing scope excluding globals.
For example the counter generator can be rewritten to use this so that it looks more like the idioms of languages with closures.

```
def make_counter():
    count = 0
    def counter():
        nonlocal count
        count += 1
        return count
    return counter
```

320. What are the wheels and eggs? What is the difference?

Ans:
Wheel and **Egg** are both packaging formats that aim to support the use case of needing an install artifact that doesn't require building or compilation, which can be costly in testing and production workflows.
The Egg format was introduced by setuptools in 2004, whereas the Wheel format was introduced by PEP 427 in 2012.

Wheel is currently considered the standard for built and binary packaging for Python. Here's a breakdown of the important differences between Wheel and Egg.

- Wheel has an official PEP. Egg did not.
- Wheel is a distribution format, i.e a packaging format. 1 Egg was both a distribution format and a runtime installation format (if left zipped), and was designed to be importable.
- Wheel archives do not include .pyc files. Therefore, when the distribution only contains Python files (i.e. no compiled extensions), and is compatible with Python 2 and 3, it's possible for a wheel to be "universal", similar to an sdist.
- Wheel uses PEP376-compliant .dist-info directories. Egg used .egg-info.
- Wheel has a richer file naming convention. A single wheel archive can indicate its compatibility with a number of Python language versions and implementations, ABIs, and system architectures.
- Wheel is versioned. Every wheel file contains the version of the wheel specification and the implementation that packaged it.
- Wheel is internally organized by sysconfig path type, therefore making it easier to convert to other formats.

CSS Qn

321. What is webpack?

Ans: **Webpack** is a build tool that puts all of your assets, including Javascript, images, fonts, and CSS, in a dependency graph. Webpack lets you use `require()` in your source code to point to local files, like images, and decide how they're processed in your final Javascript bundle, like replacing the path with a URL pointing to a CDN.

322. Name some benefits of using webpack

Ans: Webpack and static assets in a dependency graph offers many benefits. Here's a few:

- **Dead asset elimination.** This is killer, especially for CSS rules. You only build the images and CSS into your `dist/` folder that your application actually needs.
- **Easier code splitting.** For example, because you know that your file `Homepage.js` only requires specific CSS files, Webpack could easily build a `homepage.css` file to greatly reduce initial file size.
- **You control how assets are processed.** If an image is below a certain size, you could base64 encode it directly into your Javascript for fewer HTTP requests. If a JSON file is too big, you can load it from a URL. You can `require('./style.less')` and it's automaticaly parsed by Less into vanilla CSS.
- **Stable production deploys.** You can't accidentally deploy code with images missing, or outdated styles.

- Webpack will slow you down at the start, but give you **great speed benefits** when used correctly. You get hot page reloading. True CSS management. CDN cache busting because Webpack automatically changes file names to hashes of the file contents, etc.
  Webpack is the main build tool adopted by the React community.

323. Name some plugins you think are very important and helpful?

Ans:

- **CommonsChunkPlugin** – creates a separate file (known as a chunk), consisting of common modules shared between multiple entry points.
- **DefinePlugin** – allows you to create global constants which can be configured at compile time.
- **HtmlWebpackPlugin** – simplifies creation of HTML files to serve your webpack bundles.
- **ExtractTextWebpackPlugin** – Extract text from a bundle, or bundles, into a separate file.
- **CompressionWebpackPlugin** – Prepare compressed versions of assets to serve them with Content-Encoding.

324. Webpack gives us a dependency graph. What does that mean?

Ans: Any time one file depends on another, webpack treats this as a *dependency*. This allows webpack to take non-code assets, such as images or web fonts, and also provide them as dependencies for your application.
Webpack lets you use require() on local "static assets":

```
<img src={ require('../../assets/logo.png') } />
```

When webpack processes your application, it starts from a list of modules defined on the command line or in its config file. Starting from these entry points, webpack recursively builds a dependency graph that includes every module your application needs, then packages all of those modules into a small number of bundles – often, just one – to be loaded by the browser.
The `require('logo.png')` source code never actually gets executed in the browser (nor in Node.js). Webpack builds a new Javascript file, replacing `require()` calls with valid Javascript code, such as URLs. The bundled file is what's executed by Node or the browser.

325. What are metaclasses in Python?

Ans: A metaclass is the class of a class. A class defines how an instance of the class (i.e. an object) behaves while a metaclass defines how a class behaves. A class is an instance of a metaclass. You can call it a 'class factory'.

326. How to make a chain of function decorators?

Ans: How can I make two decorators in Python that would do the following?

```
@makebold
@makeitalic
def say():
```

```
    return "Hello"
```
which should return:
```
"<b><i>Hello</i></b>"
```
Answer:
Consider:
```
from functools import wraps

def makebold(fn):
    @wraps(fn)
    def wrapped(*args, **kwargs):
        return "<b>" + fn(*args, **kwargs) + "</b>"
    return wrapped

def makeitalic(fn):
    @wraps(fn)
    def wrapped(*args, **kwargs):
        return "<i>" + fn(*args, **kwargs) + "</i>"
    return wrapped

@makebold
@makeitalic
def hello():
    return "hello world"

@makebold
@makeitalic
def log(s):
    return s

print hello()       # returns "<b><i>hello world</i></b>"
print hello.__name__ # with functools.wraps() this returns "hello"
print log('hello')  # returns "<b><i>hello</i></b>"
```

327. What is the difference between @staticmethod and @classmethod?

Ans: A **staticmethod** is a method that knows nothing about the class or instance it was called on. It just gets the arguments that were passed, no implicit first argument. It's definition is immutable via inheritance.
```
class C:
    @staticmethod
    def f(arg1, arg2, ...): ...
```
A **classmethod**, on the other hand, is a method that gets passed the class it was called on, or the class of the instance it was called on, as first argument. Its definition follows Sub class, not Parent class, via inheritance.
```
class C:
    @classmethod
    def f(cls, arg1, arg2, ...): ...
```
If your method accesses other variables/methods in your class then use `@classmethod`.

328. What's the difference between a Python module and a Python package?

Ans: Any Python file is a **module**, its name being the file's base name without the .py extension.

```
import my_module
```

A **package** is a collection of Python modules: while a module is a single Python file, a package is a directory of Python modules containing an additional **init**.py file, to distinguish a package from a directory that just happens to contain a bunch of Python scripts. Packages can be nested to any depth, provided that the corresponding directories contain their own **init**.py file.

Packages are modules too. They are just packaged up differently; they are formed by the combination of a directory plus **init**.py file. They are modules that can contain other modules.

```
from my_package.timing.danger.internets import function_of_love
```

329. What is GIL?

Ans: Python has a construct called the **Global Interpreter Lock** (GIL). The GIL makes sure that only one of your 'threads' can execute at any one time. A thread acquires the GIL, does a little work, then passes the GIL onto the next thread. This happens very quickly so to the human eye it may seem like your threads are executing in parallel, but they are really just taking turns using the same CPU core. All this GIL passing adds overhead to execution.

330. Is it a good idea to use multi-thread to speed your Python code?

Ans: Python doesn't allow multi-threading in the truest sense of the word. It has a multi-threading package but if you want to multi-thread to speed your code up, then it's usually not a good idea to use it.

Python has a construct called the Global Interpreter Lock (GIL). The GIL makes sure that only one of your 'threads' can execute at any one time. A thread acquires the GIL, does a little work, then passes the GIL onto the next thread. This happens very quickly so to the human eye it may seem like your threads are executing in parallel, but they are really just taking turns using the same CPU core. All this GIL passing adds overhead to execution.

331. How do I write a function with output parameters (call by reference)?

Ans: In Python arguments are passed *by assignment*. When you call a function with a parameter, a **new reference** is created that refers to the object passed in. This is separate from the reference that was used in the function call, so there's no way to update that reference and make it refer to a new object.

If you pass a *mutable* object into a method, the method gets a reference to that same object and you can mutate it to your heart's delight, but if you rebind the reference in the method (like `b = b + 1`), the outer scope will know nothing about it, and after you're done, the outer reference will still point at the original object.

So to achieve the desired effect your best choice is to return a tuple containing the multiple results:

```
def func2(a, b):
    a = 'new-value'        # a and b are local names
    b = b + 1              # assigned to new objects
    return a, b            # return new values

x, y = 'old-value', 99
x, y = func2(x, y)
print(x, y)
```

o/p 'new value', 100

covered

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

Ans: 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.

333. What is the purpose of the single underscore "_" variable in Python?

Ans: has 4 main conventional uses in Python:

1. To hold the result of the last executed expression(/statement) in an interactive interpreter session. This precedent was set by the standard CPython interpreter, and other interpreters have followed suit
2. For translation lookup in i18n (see the gettext documentation for example), as in code like: `raise forms.ValidationError(_("Please enter a correct username"))`
3. As a general purpose "throwaway" variable name to indicate that part of a function result is being deliberately ignored (Conceptually, it is being discarded.), as in code like: `label, has_label, _ = text.partition(':')`.
4. As part of a function definition (using either `def` or `lambda`), where the signature is fixed (e.g. by a callback or parent class API), but this particular function implementation doesn't need all of the parameters, as in code like: `callback = lambda _: True`

334. How is set() implemented internally?

I've seen people say that `set` objects in python have `O(1)` membership-checking. How are they implemented internally to allow this? What sort of data structure does it use? What other implications does that implementation have?

Ans:

Indeed, CPython's sets are implemented as something like dictionaries with dummy values (the keys being the members of the set), with some optimization(s) that exploit this lack of values.

So basically a **set** uses a **hashtable** as its underlying data structure. This explains the `O(1)` membership checking, since looking up an item in a hashtable is an `O(1)` operation, on average.

Also, it worth to mention when people say sets have `O(1)` membership-checking, they are talking about the average case. In the worst case (when all hashed values collide) membership-checking is `O(n)`.

335. What is MRO in Python? How does it work?

Ans: **Method Resolution Order (MRO)** it denotes the way a programming language resolves a method or attribute. Python supports classes inheriting from other classes. The class being

inherited is called the Parent or Superclass, while the class that inherits is called the Child or Subclass.
In Python,** method resolution order** *defines the order in which the base classes are searched when executing a method.* First, the method or attribute is searched within a class and then it follows the order we specified while inheriting. This order is also called Linearization of a class and set of rules are called MRO (Method Resolution Order). While inheriting from another class, the interpreter needs a way to resolve the methods that are being called via an instance. Thus we need the method resolution order.
Python resolves method and attribute lookups using the C3 linearisation of the class and its parents. The C3 linearisation is **neither depth-first nor breadth-first** in complex multiple inheritance hierarchies.

336. What is the difference between old style and new style classes in Python?

Ans: Declaration-wise:
New-style classes inherit from object, or from another new-style class.

```
class NewStyleClass(object):
    pass

class AnotherNewStyleClass(NewStyleClass):
    pass
```

Old-style classes don't.

```
class OldStyleClass():
    pass
```

Python 3 Note:
Python 3 doesn't support old style classes, so either form noted above results in a new-style class.
Also, **MRO (Method Resolution Order)** changed:

- Classic classes do a depth first search from left to right. Stop on first match. They do not have the **mro** attribute.
- New-style classes MRO is more complicated to synthesize in a single English sentence. One of its properties is that a Base class is only searched for once all its Derived classes have been.

  They have the **mro** attribute which shows the search order.

Some other notes:

- New style class objects cannot be raised unless derived from `Exception`.
- Old style classes are still marginally faster for attribute lookup.

337. Why Python (CPython and others) uses the GIL?

Ans: In CPython, the global interpreter lock, or GIL, is a mutex that prevents multiple native threads from executing Python bytecodes at once. This lock is necessary mainly because CPython's memory management is not thread-safe.
Python has a GIL as opposed to fine-grained locking for several reasons:

- It is faster in the single-threaded case.
- It is faster in the multi-threaded case for i/o bound programs.

- It is faster in the multi-threaded case for cpu-bound programs that do their compute-intensive work in C libraries.
- It makes C extensions easier to write: there will be no switch of Python threads except where you allow it to happen (i.e. between the Py_BEGIN_ALLOW_THREADS and Py_END_ALLOW_THREADS macros).
- It makes wrapping C libraries easier. You don't have to worry about thread-safety. If the library is not thread-safe, you simply keep the GIL locked while you call it

338. How are arguments passed by value or by reference in python?

Ans:

- **Pass by value:** Copy of the actual object is passed. Changing the value of the copy of the object will not change the value of the original object.
- **Pass by reference:** Reference to the actual object is passed. Changing the value of the new object will change the value of the original object.

In Python, arguments are passed by reference, i.e., reference to the actual object is passed.

```
def appendNumber(arr):
    arr.append(4)


arr = [1, 2, 3]

print(arr)  #Output: => [1, 2, 3]
appendNumber(arr)
print(arr)  #Output: => [1, 2, 3, 4]
```

339. What is a boolean in Python?

Ans: Boolean is one of the built-in data types in Python, it mainly contains two values, and they are true and false.
Python bool() is the method used to convert a value to a boolean value.

```
1                Syntax for bool() method: bool([a])
```

340. What is Python String format and Python String replace?

Ans: **Python String Format:** The String format() method in Python is mainly used to format the given string into an accurate output or result.
Syntax for String format() method:

```
template.format(p0, p1, ..., k0=v0, k1=v1, ...)
```

Python String Replace: This method is mainly used to return a copy of the string in which all the occurrence of the substring is replaced by another substring.
Syntax for String replace() method:

```
str.replace(old, new [, count])
```

341. Name some of the built-in modules in Python?

Ans: The built-in modules in Python are:

- sys module
- OS module
- random module
- collection module
- JSON
- Math module

342. How do we convert the string to lowercase?

Ans: lower() function is used to convert string to lowercase.
Example:

```
str = 'XYZ'
print(str.lower())
```

Output:

```
xyz
```

343. How to remove values from a Python array?

Ans: The elements can be removed from a Python array using remove() or pop() function. The difference between pop() and remove() will be explained in the below example.
Example:

```
x = arr.array('d',  [ 1.0, 2.2, 3.4, 4.8, 5.2, 6.6, 7.3])
print(x.pop())
print(x.pop(3))
x.remove(1.0)
```

```
print(a)
```

Output:

```
7.3
4.8
array('d', [2.2, 3.4, 5.2, 6.6])
```

344. What is Try Block?

A block which is preceded by the try keyword is known as a try block
Syntax:

```
try{
   //statements that may cause an exception
}
```

345. How can we access a module written in Python from C?

Ans: We can access the module written in Python from C by using the following method.

```
Module == PyImport_ImportModule("<modulename>");
```

346. Write a program to count the number of capital letters in a file?

Ans:

```
with open(SOME_LARGE_FILE) as countletter:
count = 0
text = countletter.read()
for character in text:
if character.isupper():
count += 1
```

```
6
```

347. Write a program to display the Fibonacci sequence in Python?

Ans:

```
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
```

```
# Displaying Fibonacci sequence
n = 10
# first two terms
n0 = 0
n1 = 1
#Count
x = 0
# check if the number of terms is valid
if n <= 0:
   print("Enter positive integer")
elif n == 1:
   print("Numbers in Fibonacci sequence upto",n,":")
   print(n0)
else:
   print("Numbers in Fibonacci sequence upto",n,":")
   while x < n:
       print(n0,end=', ')
       nth = n0 + n1
       n0 = n1
       n1 = nth
       x += 1
```

Output:

```
1
```

```
0, 1, 1, 2, 3, 5, 8, 13, 21, 34,
```

348. Write a program in Python to produce Star triangle?

Ans: The code to produce star triangle is as follows:

```
def pyfun(r):
for a in range(r):
print(' '*(r-x-1)+'*'*(2*x+1))
pyfun(9)
```

Output:

```
        *
       ***
      *****
     *******
    *********
   ***********
  *************
 ***************
*****************
```

349. Write a program to check whether the given number is prime or not?

Ans: The code to check prime number is as follows:

```
# program to check the number is prime or not
n1 = 409
# num1 = int(input("Enter any one number: "))
# prime number is greater than 1
if n1 > 1:
# check the following factors
for x is in range of(2,num1):
if (n1 % x) == 0:
print(n1,"is not a prime number")
print(x,"times",n1//x,"is",num)
break
else:
print(n1,"is a prime number")
# if input number is smaller than
# or equal to the value 1, then it is not prime number
else:
```

```
            print(n1,"is not a prime number")
```

Output:

```
                        409 is a prime number
```

350. Write Python code to check the given sequence is a palindrome or not?

Ans:

```
# Python code to check a given sequence
#  is palindrome or not
my_string1 = 'MOM'
My_string1 = my_string1.casefold()
# reverse the given string
rev_string1  = reversed(my_string1)
# check whether the string is equal to the reverse of it or not
if list(my_string1) == list(rev_string1):
print("It is a palindrome")
else:
print("It is not a palindrome")
```

Output:

```
                        it is a palindrome
```

351. Write Python code to sort a numerical dataset?

Ans: The code to sort a numerical dataset is as follows:

```
list = [ "13", "16", "1", "5" , "8"]
list = [int(x) for x in the list]
list.sort()
print(list)
```

Output:

```
1, 5, 8, 13, 16
```

352. Differentiate between SciPy and NumPy?

Ans: The difference between SciPy and NumPy is as follows:

NumPy	SciPy
Numerical Python is called NumPy	Scientific Python is called SciPy
It is used for performing general and efficient computations on numerical data which is saved in arrays. For example indexing, reshaping, sorting, and so on	This is an entire collection of tools in Python mainly used to perform operations like differentiation, integration and many more.
There are some of the linear algebraic functions present in this module but they are not fully fledged.	For performing algebraic computations this module contain some of the fully fledged operations