PYTHON LEVEL 2

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.

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.

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.

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
```

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__."

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.

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.

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.

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.

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.

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.

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.

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.

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.

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.

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.

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
```

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.

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
```

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.

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
```

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.

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.

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.

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)

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
```

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.

```
>>> def facto(n):

if n==1: return 1

return n*facto(n-1)

>>> facto(4)

24
```

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.

How will you print the contents of a file?

```
>>> try:

with open(‘tabs.txt’,‘r’) as f:
```

```
print(f.read())

except IOError:

print("File not found").
```

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.

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.

Name some plugins you think are very important and helpful?

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.

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.

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’.

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.

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.

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.