PYTHON

A programming language with objects, modules, threads, exceptions and automatic memory management. The benefits of Python are simplicity, portability, extensibility, build-in data structures, and open-source nature.

- ***Interpreted** language – unlike C and its variants, Python does not need to be compiled before its run. It runs directly from the source code by converting it into an intermediate language and translating it into an executable machine-level code. Essentially, all interpreted programming languages are not machine-level code before runtime (other examples could be PHP and Ruby).*
- ***Dynamically typed –** you don't need to state the types of variables when you declare them. Having the built-in data types saves programming time and effort, as well as makes it much easier to learn for beginners (e.g. as opposed to C). This comes at a cost - **writing code is quick,** but **running is slower** than for compiled languages. Even though, Python allows the inclusion of C based extensions so bottlenecks can be optimized away (e.g., NumPy).*
- ***Well-suited for OOP –** allows the definition of classes along with composition and inheritance. Does not have access specifiers (like C++s public or private).*
- ***Functions** are **first-class objects** – can be assigned to variables, returned from other functions, and passed to other functions. Classes are also first class objects.*
- ***Use in many applications –** web apps, automation, scientific modeling, big data applications and many more. Often used as "glue" code to get other languages and components to play nice.*
- ***Huge standard library** for most Internet platforms like Email, HTML, etc.*
- *D**oes not require explicit memory management** - the interpreter itself allocates/frees the memory to/from variables.*
- *It is **not a scripting language**, but is **capable** of being used like it*

PEP & Identation

***PEP (Python Enhancement Protocol) 8 –** a coding convention, a set of recommendations about how to write Python code to be more readable.*

***Identation** – Specifies a block of code. All code within loops, classes, gunctions, etc. is specified within an indented block (usually 4 space characters/tab). If the code is not idented, it will not execute & accurately and throw errors.*

Memory Management

*Python memory is managed by the **Python private heap space**. The programmer does not have an access to it, the interpreter takes care of it instead.*

*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 has an inbuilt **garbage collector,** which recycles all the unused memory and frees the memory to make it available for the heap space.*

Multithreading

1. *Python has a multi-threading package but if you want to multi-thread to speed your code up, then it is 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 some 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, using the threading package often is not a good idea.*

General

Unittest – *a unit testing framework. Supports sharing of setups, automation testing, shutdown code for tests, aggregation of tests into collection, etc.*

Finding Bugs | Performing Static Analysis – *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 conding standard.*

Docstring *– A python documentation string, a way of documenting python.*

Variables

Local variables: *if a variable is assigned a new value anywhere within the function body, it is assumed to be local.*

Global variables: *Variables declared outside a function (only referenced inside it) or in global space. Can be accessed by any function in the program.*

Namespace *- In Python, every name introduced has a place where it lives and can be hooked for, known as namespace, which makes sure that names are unique to avoid naming conflicts. It is like a box where a variable name is mapped to some object. Whenever the variable is searched out, this box will be searched to get the corresponding object.*

How can you share global variables across modules?
To share global variables across modules within a single program, create a special module and import it within all modules of your application. The module will be available as a global variable across modules.

How are arguments passed – by value or by reference?
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 they are mutable.

Operators

The use of // operator
It is a Floor Division operator, which is used for dividing two operands with the result as quotient showing only digits before the decimal point. For intance, 10//5 = 2 and 10.0//5.0 = 2.0.

What is the purpose of ***is, not,*** *and* ***in*** *operators?*
They are special functions that take one or more values and produce a corresponding result:

- ***is –*** *returns true when 2 operands are true (a is a)*
- ***not –*** *returns the inverse of the boolean value*
- ***in –*** *checks if some element is present in some sequence*

Data Types

Python built-in data types: *Integers, Floating-point, Complex numbers, Strings, Boolean, Built-in functions.*

- ***Mutable*** *(can be edited)* ***–*** *List, Set, Dictionary*
- ***Immutable*** *(cannot be edited)* ***–*** *String, Tuple, Number*

What is the difference between lists and tuples?
Lists are mutable but slower, whereas tuples are immutable but faster.

Type conversion in Python:

- ***int()*** *- any data type into integer type*
- ***float()*** *- any data type into float type*
- ***ord()*** *- characters into integer*
- ***hex()*** *- integers to hexadecimal*
- ***oct()*** *- integers to octal*
- ***tuple()*** *- to a tuple*
- ***set()*** *- to a set*
- ***list()*** *- to a list*
- ***dict()*** *- tuple of order (key, value) to a dictionary*
- ***str()*** *- integer to string*
- ***complex(real,imag)*** *– real to complex(real,imag) number*

NumPy Arrays (nested) vs. Python Lists:

- ***Lists*** *are efficient general-purpose containers supporting optimized insertion, deletion, and similar procedures. Python's list comprehension makes them easy to construct and manipulate. However, they do not support effective "vectorized" operations, such as elementwise addition and multiplication (these are performed slowly). Essentially, due to the fact that lists allow to store objects of different types, Python needs to store type information for each of the elements separately, and execute type dispatching code when operating on them, which brings a large computational overhead when working with large arrays of numbers.*
- ***NumPy*** *is more efficient and convenient when working with large number arrays, provding lots of optimized vector and matrix operations, letting alone more complex procedures like FFTs, convolutions, fast searching, basic statistics, linear algebra, histograms, etc.*

How to add/remove values to a python array?
Elements can be added to an existing array using ***append(), extend(), insert(i, x)*** *functions, and removed using* ***pop()****,* ***remove()*** *methods (the former returns the deleted value, whereas the later does not).*

INDEXING – *Python sequences can be indexed in positive and negative numbers, where 0 is the first element, and 1 is the last (first from the end).*

SLICING – *a mechanism to select a range of items from sequence types like list, tuple, strings, etc.*

[::-1] *is used to reverse the order of an array or a sequence, but only makes a representation, the array or list remains unchanged if no pass to variable is defined.*

Control Flow

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.*

Iterators & Generators

Dict and List comprehension *– Syntax constructions to ease the creation of a Dictionary or List based on existing iterable.*

Iterators – *objects which can be traversed through or iterated upon.*

Generators – *functions that return an iterable set of items*

Random generators (import random)

- ***random.random()*** *- returns a the floating point number that is in the range [0,1)*
- ***randrange(a,b)*** *– choosen an integer and the range in-between [a,b). Returns the elements by selecting it randomly from the range that is specified, but does not build a range object.*
- ***uniform(a,b)*** *– chooses a floating point number that is defined in the range of [a,b).*
- ***normalvariate(mean, sdev)*** *– used for the normal distribution where the mu is a meand and the sdev is a sigma that is used for standard deviation.*
- *The* ***Random class*** *that is used and instantiated creates an independent multiple random number generators.*

Randomize *the items of a* ***list in place****?*
from random import shuffle
shuffle(list)

Xrange vs. Range
Both generate a list of integers, but Xrange returns the xrange object whilst range returns Python list. That is, xrange does not actually generate a static list at run-time, but creates the values as you need them with a special technique called yielding.

- *If you have a 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 memory sensitive system such as cell phones, because range will use as much memory as it can to create your array of integers, possibly resulting in a Memory Error and/or application crash.*

Functions & Methods

Functions – *Blocks of code which are executed only when called, defined using the* ***def*** *statement.*
Lambda – *It is a single expression anonymous function, often used to make new function objects and return them runtime. These function can have any number of parameters, but just a single statement.*
Map – *Executes the function given as the first argument on 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.*

What is the use of ***help()*** *and* ***dir()*** *functions in Python?*
They are both accessible from the Python interpreter and used for viewing a consolidated dump of built-in functions.

- ***help()*** *- display the documentation string and the help related to modules, keywords, attributes, etc.*
- ***dir()*** *- used to display the defined symbols.*

****args **kwargs***

- ****args*** *is used when we are not 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.*

These identifiers are a convention.

Regex *module functions:*

- ***re.split() -*** *uses a regex pattern to "split" a given string into a list.*
- ***re.sub() -*** *finds all substrings where the regex pattern matches and then replaces them with a different string.*
- ***re.subn() -*** *similar to sub(), but returns a new string along with the number of replacements.*

String processing functions:

Split function – *breaks a string into shorter strings using the defined separator, returning a list of all tokens present.*

Capitalize *the first letter of a* ***string****? Convert string to* ***lowercase****?*

- *capitalize()*
- *lower()*

Classes

__init__ *A method or constructor in Python, which is automatically called to allocate memory when a new object/instance of a class is created. All classes have the __init__ method.*

self - *An instance or an object of a class, which helps to differentiate between the methods and attributes of a class. The self variable in the init method refers to the newly created object, whereas in other methods it refers to the object whose method was called. In Python, this is explicitly included as the first parameter (optional in Java).*

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

EMPTY CLASS – *A class that does not have any code defined within its block. Can be reached using the* ***pass*** *keyword. In python the pass command does nothing when it is executed (null statement).*

```
class a:
    pass
```

Modules | Packages | Filesystem

- ***Modules*** *are files containing Python code (.py files with executable code), serving as a way to structure program. This code can be either functions, classes, or variables. Some common built-in modules are os, sys, data time, JSON.*
- ***Packages*** *are namespaces containing multiple modules, or folders of Python programs that have multiple subfolders.*
- ***Libraries –*** *A collection of Python packages. Some of the majorly used python libraries are* ***Numpy, Pandas, Matplolib, Scikit-learn*** *and many more.*

IMPORTING MODULES – *using the* ***import*** *statement*

```
import array – importing using the original module name
import array as ar – importing using an alias name
from array import * - imports everything present in the array module.
```

PYTHONPATH – *An environment variable used when a module is imported (looked up to see if the imported module exists).*

PICKING / UNPICKLING – *Pickle module accepts any Python object and converts it into a string representation for dumping into a file (pickling). The original Python objects can then be retrieved from this stored representation (unpickling).*

How can you make a Python script executable on Unix?

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

How to delete file in Python?
os.remove(filename) or os.unlink(filename)

NUMPY vs. SCIPY
In ideal world, NumPy would contain nothing but the array data type and the most basic operations: indexing, sorting, reshaping, elementwise functions, etc., wheareas all numerical code would reside in SciPy. However:

- *NumPy aims to be widely compatible, so it retains most of the features supported by SciPy, e.g.: basic linear algebra.*
- *SciPy contains fully-featured versions of the linear algebra modules, as well as many other numerical algorithms.*
- *If you are doing scientific computing with Python, install both. Most new features reside in SciPy rather than NumPy.*

How do you make 3D plots/visualizations using NumPy/SciPy
Like 2D plotting, 3D graphics is beyond the scope of NumPy and SciPy, but just as in 2D case, packages exist that integrate with NumPy. Matplotlib provides basic 3D plotting in the mplot3d subpackages, whereas Mayavi provides a wide range of high-quality 3D visualization features, utilizing the powerful VTK engine.

PYTHON OOP

*What does an **object()** do?*
Returns a featureless object that is a base for all classes (does not take any parameters).

INHERITANCE

***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 ad 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. There are different types of inheritance supported by Python:*

- ***Single inheritance** – where a derived class acquires the members of a single super class.*
- ***Multi-level inheritance** – a derived class d1 is inherited from the base class base1 and d2 from base2.*
- ***Hierarchical inheritance** – from one base class you can inherit any number of child classes.*
- ***Multiple inheritance** – a derived class is inherited from more than one base class. Python does support this, Java does not.*

ENCAPSULATION

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

POLYMORPHISM

Polymorphism means the ability to take multiple forms. 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 this.

ACCESS SPECIFIERS

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 behaviour of protected and private access specifiers.

***MONKEY PATCHING –** dynamic modifications of a class or module at run-time. Let' say you have a file `m.py`:*

class MyClass:
def f(self):
print (“f()”)

we can then run the monkey patch testing like:

m.MyClass.f = monkey_f
obj = m.MyClass()
obj.f()
The output will be: monkey_f(). That is, we did make some changes in the behavior of f() in MyClass using the function we defined, monkey_f(), outside of the module m.

Basic Python Programs

BUBBLE SORT
def bubble_sort(integer_list):

b = len(integer_list) – 1 # -1 because we always compare 2 adjacent values

for x in range(b):

for y in range(b-x):

if integer_list[y]>integer_list[y+1]:

integer_list[y], integer_list[y+1] = integer_list[y+1], integer_list[y]

return a

*One-liner to **COUNT** the number of **CAPITAL LETTERS** in a **FILE** even if it is too big to fit in memory*

with open(SOME_LARGE_FILE) as fh:
count(sum(1 for line in fh for character in line if character.isupper())

Flask & Django

***FLASK –** a BSD licensed web micro framework for Python, primarily built for small applications, with simple requirements and little to no dependencies on external libraries. Flask is light, requires minimal updates, as has less security bugs.*

- *A session will allow you to remember information from one request to another. In Flask, it uses a signed cookie so the user can look at the session contents and modify it. This can be done IFF the user has the secret key Flask.secret_key*
- *The common way for a Flask script to work is to either make it the import path for your application or the path to a Python file. Essentially, Flask is a minimalistic framework that behaves same as an MVC model.*

DJANGO** vs. **FLASK** vs. **PYRAMID

- ***Django and Flask** map the URL’s or addresses typed in the web browsers to functions in Python.*
- ***Flask** is much simpler than **Django** but it can’t do much for you if you do not specify the details, whereas Django does much without you needing to put much effort. It consists of prewritten code which the user will need to analyze.*
- ***Pyramid** is built for larger applications, providing flexibility and allowing the developers to use the right tools for their project. The developer can choose the database, URL, structure, and more, it is heavily configurable.*
- ***Django** can also be used for larger applications just like **Pyramid**, and it includes an ORM.*

Django Database Setup
You can use the command edit mysite/setting.py, it is a normal python module with module level representing Django settings. Django uses SQLite by default (stores data as a single file in filesystem) and does not require any other type of installation. In

the case your database choice is different, the following keys in the DATABASE 'default' item have to match your database connection settings.

DJANGO ARCHITECTURE

Django MVT Pattern – The developer provides the Model, the view and the template then just maps it to a URL and Django does the magic to serve it to the user.

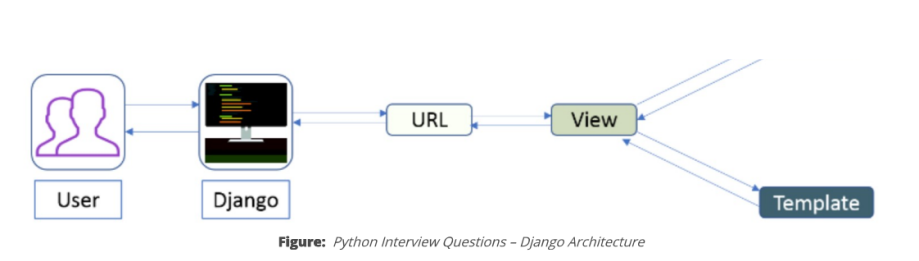

Figure: *Python Interview Questions – Django Architecture*

- ***Engines –*** *you can change the database by using `django.db.backends.sqlite3`, 'django.db.backends.mysql', 'django.db.backends.postgresql_psycopg2', 'django.db.backends.oracle', and etc.*
- ***Name –*** *the name of your database. In case you are using SQLite as your database, the database will be a file on your computer. Name should be a full abs path, including the file name of that file.*
- *If you are not choosing SQLite as your database then settings like Password, Host, User, etc. must be added.*

If you do have a database server – PostgreSQL, MySQL, Oracle, MSSQL – and want to use it instead of SQLite, use your database's administration tools to create a new database for your Django project. Either way, you also have to tell Django how to use your data via settings.py file:

```
DATABASES = {'default': {
                'ENGINE':'django.db.backends.sqlite3',
                'NAME':os.path.join(BASE_DIR, 'db.sqlite3')
                }}
```

DJANGO TEMPLATE

A simple text file, which can create any text-based format like XML, CSV, HTML, etc. A template contains variables that get replaced with values when the template is evaluated and tags (%tag%) that control the logic of the template.

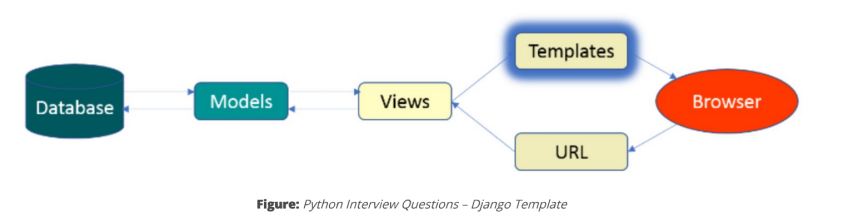

Figure: *Python Interview Questions – Django Template*

SESSION** in **DJANGO

Django provides a session that lets you store and retrieve data on a per-site-visitor basis. Django abstracts the process of sending and receiving cookies, by placing a session ID cookie on the client side, and storing all the related data on the server side (the data itself is not stored on the client side, which is a nice security trait).

INHERITANCE STYLES (DJANGO)

- ***Abstract Base Classes –*** *used when you only want parent's class to hold information that you don't want to type out for each child.*
- ***Multi-table Inheritance –*** *used when you are sub-classing an existing model and need each model to have its own database table.*
- ***Proxy models –*** *used if only want to modify the Python level behaviour of the model, without changing the model's fields*