**Find and read 20 interview questions for Data Types, Operators, Conditional Statements, Looping Statements, Functions.**

**1. What is Python?**

* Python is a **high-level**, **interpreted**, interactive, and **object-oriented**scripting language. It uses English keywords frequently. Whereas, other languages use punctuation, Python has fewer syntactic constructions.
* Python is designed to be highly**readable** and **compatible** with different platforms such as Mac, Windows, Linux, Raspberry Pi, etc.

### ****2. Python is an interpreted language. Explain.****

An interpreted language is any programming language that executes its statements line by line. Programs written in Python run directly from the source code, with no intermediary compilation step.

**3. What is the difference between lists and tuples?**

|  |  |
| --- | --- |
| **Lists** | **Tuples** |
| Lists are mutable, i.e., they can be edited | Tuples are immutable (they are lists that cannot be edited) |
| Lists are usually slower than tuples | Tuples are faster than lists |
| Lists consume a lot of memory | Tuples consume less memory when compared to lists |
| Lists are less reliable in terms of errors as unexpected changes are more likely to occur | Tuples are more reliable as it is hard for any unexpected change to occur |
| Lists consist of many built-in functions. | Tuples do not consist of any built-in functions. |
| Syntax:  list\_1 = [10, ‘Intellipaat’, 20] | Syntax:  tup\_1 = (10, ‘Intellipaat’ , 20) |

**4. What is pep 8?**

PEP in Python stands for Python Enhancement Proposal. It is a set of rules that specify how to write and design Python code for maximum readability.

### ****5. What are the Key features of Python?****

* Python is an interpreted language, so it doesn’t need to be compiled before execution, unlike [languages such as C](https://intellipaat.com/blog/tutorial/c-tutorial/).
* Python is dynamically typed, so there is no need to declare a variable with the data type. Python Interpreter will identify the data type on the basis of the value of the variable.
* Python follows an **object-oriented programming** paradigm with the exception of having access specifiers. Other than access specifiers (public and private keywords), Python has classes, inheritance, and all other usual OOPs concepts.
* Python is a **cross-platform language**, i.e., a Python program written on a Windows system will also run on a Linux system with little or no modifications at all.
* Python is literally a **general-purpose language**, i.e., Python finds its way in various domains such as web application development, automation, Data Science, Machine Learning, and more

**6. How is Memory managed in Python?**

* Memory in Python is managed by Python private heap space. All Python objects and data structures are located in a private heap. This private heap is taken care of by Python Interpreter itself, and a programmer doesn’t have access to this private heap.
* Python memory manager takes care of the allocation of Python private heap space.
* Memory for Python private heap space is made available by Python’s in-built garbage collector, which recycles and frees up all the unused memory.

### ****7. What is PYTHONPATH?****

**PYTHONPATH** has a role similar to PATH. This variable tells Python Interpreter where to locate the module files imported into a program. It should include the Python source library directory and the directories containing Python source code. PYTHONPATH is sometimes preset by Python Installer.

**8. What are Python Modules?**

Files containing Python codes are referred to as [**Python Modules**](https://intellipaat.com/blog/tutorial/python-tutorial/python-modules/). This code can either be classes, functions, or variables and saves the programmer time by providing the predefined functionalities when needed. It is a file with “.py” extension containing an executable code.

Commonly used built modules are listed below:

* os
* sys
* data time
* math
* random
* JSON

### ****10. Explain Inheritance in Python****

* **Single inheritance**: When a class inherits only one superclass
* **Multiple inheritance**: When a class inherits multiple superclasses
* **Multilevel inheritance**: When a class inherits a superclass, and then another class inherits this derived class forming a ‘parent, child, and grandchild’ class structure
* **Hierarchical inheritance**: When one superclass is inherited by multiple derived classes

### ****12. What is a dictionary in Python?****

Python dictionary is one of the supported [data types in Python](https://intellipaat.com/blog/tutorial/python-tutorial/python-datatypes/). It is an unordered collection of elements. The elements in dictionaries are stored as key-value pairs. Dictionaries are indexed by keys.

For example, below we have a dictionary named ‘dict’. It contains two keys, Country and Capital, along with their corresponding values, India and New Delhi.

**Syntax:**

dict={‘Country’:’India’,’Capital’:’New Delhi’, }

**Output**: Country**:** India, Capital**:** New Delhi

### ****13. What are functions in Python?****

A function is a block of code which is executed only when a call is made to the function. **def** keyword is used to define a particular function as shown below:

def function():

print("Hi, Welcome to Intellipaat")

function(); # call to the function

**Output:**  
Hi, Welcome to Intellipaat

### ****14. What is \_\_init\_\_ in Python?****

Equivalent to constructors in OOP terminology, \_\_init\_\_ is a reserved method in[Python classes](https://intellipaat.com/blog/tutorial/python-tutorial/python-classes-and-objects/). The \_\_init\_\_ method is called automatically whenever a new object is initiated. This method allocates memory to the new object as soon as it is created. This method can also be used to initialize variables.

**Syntax**

(for defining the \_\_init\_\_ method):

class Human:

# init method or constructor

def \_\_init\_\_(self, age):

self.age = age

# Sample Method

def say(self):

print('Hello, my age is', self.age)

h= Human(22)

h.say()

**Output:**

Hello, my age is 22

**15. What are the common built-in data types in Python?**

Python supports the below-mentioned built-in data types:

**Immutable data types:**

* Number
* String
* Tuple

**Mutable data types:**

* List
* Dictionary
* set

### ****16. What are local variables and global variables in Python?****

**Local variable**: Any variable declared inside a function is known as Local variable and it’s accessibility remains inside that function only.

**Global Variable**: Any variable declared outside the function is known as Global variable and it can be easily accessible by any function present throughout the program.

g=4                #global variable

def func\_multiply():

l=5       #local variable

m=g\*l

return m

func\_multiply()

### ****17. What is type conversion in Python?****

Python provides you with a much-needed functionality of converting one form of data type into the needed one and this is known as type conversion.

**Type Conversion is classified into types:**

1. Implicit Type Conversion: In this form of [**type conversion python**](https://intellipaat.com/blog/tutorial/python-tutorial/type-conversion-in-python/) interpreter helps in automatically converting the data type into another data type without any User involvement.

2. Explicit Type Conversion: In this form of Type conversion the data type inn changed into a required type by the user.

Various Functions of explicit conversion are shown below:

int() –  function converts any data type into integer.

float() –   function converts any data type into float.

ord() – function returns an integer representing the Unicode character

hex() –  function converts integers to hexadecimal strings.

oct() –   function converts integer to octal strings.

tuple() – function convert to a tuple.

set() – function returns the type after converting to set.

list() – function converts any data type to a list type.

dict() – function is used to convert a tuple of order (key,value) into a dictionary.

str() –  function used to convert integer into a string.

complex(real,imag) – function used to convert real numbers to complex(real,imag) numbers.

**19. What is the difference between Python Arrays and lists?**

|  |  |
| --- | --- |
| List | Array |
| Consists of elements belonging to different data types | Consists of only those elements having the same data type |
| No need to import a module for list declaration | Need to explicitly import a module for array declaration |
| Can be nested to have different type of elements | Must have all nested elements of the same size |
| Recommended to use for shorter sequence of data items | Recommended to use for longer sequence of data items |
| More flexible to allow easy modification (addition or deletion) of data | Less flexible since addition or deletion has to be done element-wise |
| Consumes large memory for the addition of elements | Comparatively more compact in memory size while inserting elements |
| Can be printed entirely without using looping | A loop has to be defined to print or access the components |
| **Syntax:** list = [1,”Hello”,[‘a’,’e’]] | **Syntax:** import array array\_demo = array.array(‘i’, [1, 2, 3]) (array  as integer type) |

### ****20. Is python case sensitive?****

Yes, Python is a case sensitive language. This means that Function and function both are different in pythons like SQL and Pascal.

### ****21. What does [::-1] do?****

[::-1] ,this is an example of slice notation and helps to reverse the sequence with the help of indexing.

[Start,stop,step count]

Let’s understand with an example of an array:

import array as arr

Array\_d=arr.array('i',[1,2,3,4,5])

Array\_d[::-1]          #reverse the array or sequence

**Output:** 5,4,3,2,1

### ****22. What are Python packages?****

A Python package refers to the collection of different sub-packages and modules based on the similarities of the function.

### ****23. What are decorators in Python?****

In Python, decorators are necessary functions that help add functionality to an existing function without changing the structure of the function at all. These are represented by **@decorator\_name** in Python and are called in a bottom-up format.

Let’s have a look how it works:

def decorator\_lowercase(function):   # defining python decorator

def wrapper():

func = function()

input\_lowercase = func.lower()

return input\_lowercase

return wrapper

@decorator\_lowercase    ##calling decoractor

def intro():                        #Normal function

return 'Hello,I AM SAM'

hello()

**Output:**‘hello,i am sam’

### ****24. Is indentation required in Python?****

Indentation in Python is compulsory and is part of its syntax.

All programming languages have some way of defining the scope and extent of the block of codes. In Python, it is indentation. Indentation provides better readability to the code, which is probably why Python has made it compulsory.

### ****25. How does break, continue, and pass work?****

These statements help to change the phase of execution from the normal flow that is why they are termed loop control statements.

**Python break**: This statement helps terminate the loop or the statement and pass the control to the next statement.

**Python** **continue**: This statement helps force the execution of the next iteration when a specific condition meets, instead of terminating it.

**Python** **pass**: This statement helps write the code syntactically and wants to skip the execution. It is also considered a null operation as nothing happens when you execute the pass statement.

### ****28. What type of language is python? Programming or scripting?****

Generally, Python is an all purpose Programming Language ,in addition to that Python is also Capable to perform scripting.

**30. Explain split(), sub(), subn() methods of “re” module in Python?**

These methods belong to the [Python RegEx or ‘re’ module](https://intellipaat.com/blog/tutorial/python-tutorial/python-regex-regular-expressions/) and are used to modify strings.

* split(): This method is used to split a given string into a list.
* sub(): This method is used to find a substring where a regex pattern matches, and then it replaces the matched substring with a different string.
* subn(): This method is similar to the sub() method, but it returns the new string, along with the number of replacements.

### ****32. What is a map function in Python?****

The map() function in Python has two parameters, function and iterable. The map() function takes a function as an argument and then applies that function to all the elements of an iterable, passed to it as another argument. It returns an object list of results.

For example:

def calculateSq(n):

return n\*n

numbers = (2, 3, 4, 5)

result = map( calculateSq, numbers)

print(result)

### ****33. What are the generators in python?****

Generator refers to the function that returns an iterable set of items.

**36. What are Dict and List comprehensions?**

[Python comprehensions](https://intellipaat.com/blog/tutorial/python-tutorial/python-list-comprehension/) are like decorators, that help to build altered and filtered lists, dictionaries, or sets from a given list, dictionary, or set. Comprehension saves a lot of time and code that might be considerably more complex and time-consuming.

Comprehensions are beneficial in the following scenarios:

* Performing mathematical operations on the entire list
* Performing conditional filtering operations on the entire list
* Combining multiple lists into one
* Flattening a multi-dimensional list

**For example:**

my\_list = [2, 3, 5, 7, 11]

squared\_list = [x\*\*2 for x in my\_list]    # list comprehension

# output => [4 , 9 , 25 , 49 , 121]

squared\_dict = {x:x\*\*2 for x in my\_list}    # dict comprehension

# output => {11: 121, 2: 4 , 3: 9 , 5: 25 , 7: 49}

#### **1. What are the two types of functions in Python?**

There are two types of functions in Python: built-in functions and user-defined functions. Built-in functions are functions that are already defined in the Python language, such as the print() function. User-defined functions are functions that are created by the user, and they can be created to do anything that the user wants them to do.

#### **2. Can you explain what a call graph is? How do you create one?**

A call graph is a visual representation of the relationships between the various functions in a Python program. It can be used to help debug code, optimize code, and understand code flow. To create a call graph, you can use the pycallgraph library.

#### **3. When should you use anonymous functions and when should you use regular ones?**

There is no definitive answer to this question, as it depends on the specific situation and what you are trying to accomplish. However, a general rule of thumb is that anonymous functions are best used for simple tasks that can be easily expressed in a single line of code. Regular functions, on the other hand, are better suited for more complex tasks that require multiple lines of code.

#### **4. Do Python functions have return values? If yes, then how many can they have?**

Yes, Python functions can have return values. They can have a single return value, or they can have multiple return values.

#### **5. Why does Python support both positional and keyword arguments to its functions?**

Python supports both positional and keyword arguments in order to give developers more flexibility when designing their functions. Positional arguments are those that are passed in by position, without explicitly specifying the parameter name. Keyword arguments are those that are passed in by explicitly specifying the parameter name. Python allows for both types of arguments so that developers can choose the approach that makes the most sense for their particular function.

#### **6. Is it possible for a function’s code to read from or write to variables defined outside that function? If yes, then how?**

Yes, it is possible for a function’s code to read from or write to variables defined outside that function. This is known as “accessing global variables.” To do this, the function must first use the “global” keyword to declare which variables it is accessing. For example:

global var1

var1 = 5

def func():

print(var1)

func() # Prints 5

#### **7. Can you explain what a closure is in Python? How do you implement closures?**

A closure is a function that remembers the values from the enclosing scope even when the program flow is no longer in that scope. Closures are implemented by creating a function that takes in one or more values from the enclosing scope and then returning a new function that uses those values.

#### **8. Can you explain what decorators are in Python?**

Decorators are a way to dynamically alter the behavior of a function. They are usually used as a way to add functionality to an existing function without having to modify the code of the function itself. Decorators are typically written in the form of a wrapper function.

#### **9. Is there any way to define static methods in Python? If yes, then how?**

Static methods are defined in Python by using the @staticmethod decorator. This decorator can be applied to any method, and will cause the method to be treated as a static method, even if it is not defined as such.

#### **10. Is it possible to pass a variable number of arguments to a function in Python? If yes, then how?**

Yes, it is possible to pass a variable number of arguments to a function in Python. This can be done using the \*args and \*\*kwargs parameters. \*args allows for a variable number of non-keyworded arguments to be passed to a function, while \*\*kwargs allows for a variable number of keyworded arguments to be passed.

#### **11. How would you convert JSON to Python objects?**

The json library in Python can help you convert JSON data into Python objects. The process is known as decoding. You can use the json.loads() function to decode JSON data. This function takes a JSON string and returns a Python object.

#### **12. Can you explain what recursion is and why it’s useful?**

Recursion is a function that calls itself. It’s useful because it allows you to break down a problem into smaller, more manageable pieces.

#### **1. What is the difference between a for loop and a while loop?**

A for loop is typically used when you know exactly how many times the loop needs to be repeated. A while loop is typically used when you don't know how many times the loop needs to be repeated.

A while loop repeats as long as its condition is true. For example, if a while loop says "while x == 5", then the line will execute as long as x equals five.

#### **2. What is the Python syntax for a for loop?**

The for loop is a programming construct that allows you to iterate over an arbitrary range of values, mapping them to the required actions.

It can be thought of as an extension of the mathematical notion of a for loop, which is defined as "a control structure that enables one to iterate (repeat) a process (such as counting or summing) while varying the process's start value (or its end point), step size, and/or direction."

#### **3. What are the advantages of using a for loop in Python?**

A for loop is a type of loop that executes a set of instructions repeatedly. It uses the following syntax: **for x in range(y): do something**

This type of loop has the following properties:

* For loops are good because they can make your code more readable and easier to follow.
* They also make sure that certain instructions are executed at least one time.

#### **4. What are the disadvantages of using a for loop in Python?**

The for-loop is one of the most basic constructs in programming. However, this construct has its own disadvantages.

1. If you don't know the upper-bound of your loop, you have to keep track of it by hand or break out of your loop early.
2. You can't use a break statement to escape from a nested loop.

#### **5. How can I use a break statement in my Python for loops?**

A break statement can be used to terminate a loop. It is often used when the programmer needs to stop iterating through the data and instead go back to checking for other conditions.

An example of such a situation is when we need to break out of a for loop if an exception occurs. This will allow us to take different paths depending on whether or not we want to handle that exception, or ignore it and continue with whatever else follows after the for-loop.

Code Example for Break Statement is as follows:

 animals = ["dog","cat","sheep","tiger","lion"]

 for x in animals:

 print (x)

 if x == "tiger":

 break;

**Q. What are local variables and global variables in Python?**

* **Global Variables**: Variables declared outside a function or in a 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

**Q. When to use a tuple vs list vs dictionary in Python?**

* Use a **tuple** to store a sequence of items that will not change.
* Use a **list** to store a sequence of items that may change.
* Use a **dictionary** when you want to associate pairs of two items.

**Q. Explain some benefits of Python**

* Python is a **dynamic-typed** language. It means that you don’t need to mention the data type of variables during their declaration.
* Python supports **object-orientated programming** as you can define classes along with the composition and inheritance.
* **Functions** in Python are like **first-class objects**. It suggests you can assign them to variables, return from other methods and pass them as arguments.
* Developing using Python is quick but running it is often slower than compiled languages.
* Python has several usages like web-based applications, test automation, data modeling, big data analytics, and much more.

**Q. What is Lambda Functions in Python?**

A **Lambda Function** is a small anonymous function. A lambda function can take *any* number of arguments, but can *only* have *one* expression.

**Q. How do I modify a string in python?**

You can’t because strings are *immutable in python*. In most situations, you should simply construct a new string from the various parts you want to assemble it from. Work with them as lists; turn them into strings only when needed.

**Q. What is a Negative Index in Python?**

Negative numbers mean that you count from the right instead of the left. So, **list[-1]** refers to the last element, **list[-2]** is the second-last, and so on.

**Q. How the string does get converted to a number?**Junior

* To convert the string into a number the built-in functions are used like int() a constructor. It is a data type that is used like int (‘1’) == 1.
* float() is also used to show the number in the format as float(‘1’) = 1.
* The number by default are interpreted as a decimal and if it is represented by int(‘0x1’) then it gives an error as ValueError. In this the int(string,base) the function takes the parameter to convert string to number in this the process will be like int(‘0x1’,16) == 16. If the base parameter is defined as 0 then it is indicated by octal and 0x indicates it as a hexadecimal number.
* There is function eval() that can be used to convert a string into number but it is a bit slower and present many security risks

**Q. What is docstring in Python?**Junior

* A documentation string or docstring is a multiline string used to document a specific code segment.