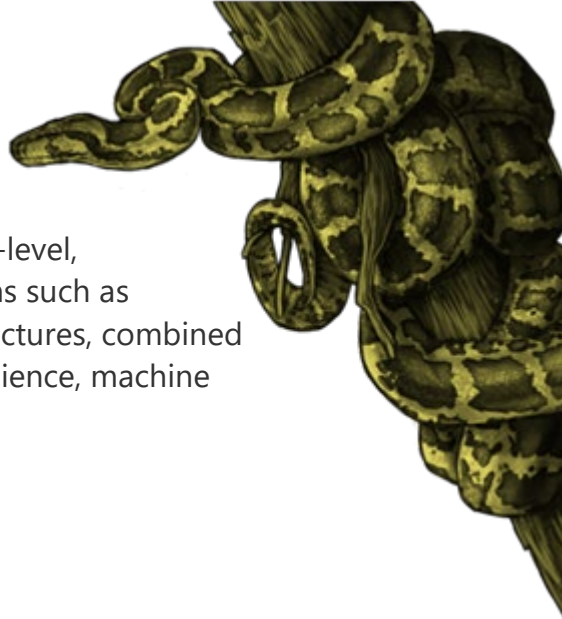
Python Interview Questions

Beginner to Advanced Level

Curated by Adnan Manna



1) What is Python?

Python was created by **Guido van Rossum**, and released in **1991**.
It is a general-purpose computer programming language. It is a high-level, object-oriented language which can run equally on different platforms such as Windows, Linux, UNIX, and Macintosh. Its high-level built-in data structures, combined with dynamic typing and dynamic binding. It is widely used in data science, machine learning and artificial intelligence domain.

It is easy to learn and require less code to develop the applications.

It is widely used for:

- Web development (server-side).
- Software development.
- Mathematics.
- System scripting.

---

2) Why Python?

- Python is an interpreted, object-oriented, high-level programming language with dynamic semantics.
- Python is compatible with different platforms like **Windows, Mac, Linux, Raspberry Pi,** etc.
- Python has a simple syntax as compared to other languages.
- Python allows a developer to write programs with fewer lines than some other programming languages.
- Python runs on an interpreter system, means that the code can be executed as soon as it is written. It helps to provide a prototype very quickly.
- Python can be described as a procedural way, an object-orientated way or a functional way.
- The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed.

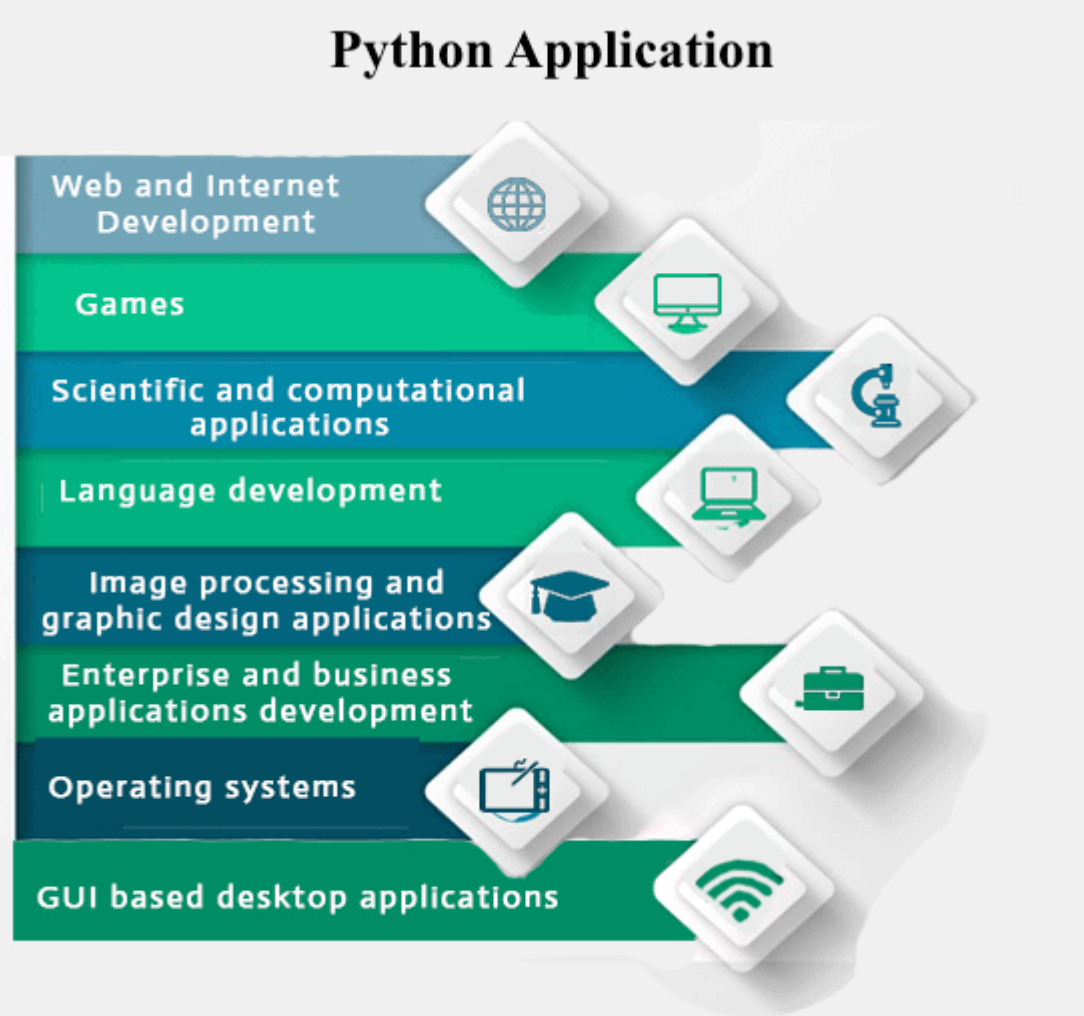
---

3) What are the applications of Python?

Python is used in various software domains some application areas are given below.

- Web and Internet Development

- Games
- Scientific and computational applications
- Language development
- Image processing and graphic design applications
- Enterprise and business applications development
- Operating systems
- GUI based desktop applications



Python provides various web frameworks to develop web applications. The popular python web frameworks are **Django, Pyramid, Flask**.

Python's standard library supports for E-mail processing, FTP, IMAP, and other Internet protocols.

Python's **SciPy** and **NumPy** helps in scientific and computational application development.

Python's **Tkinter** library supports to create a desktop based GUI applications.

4) What are the advantages of Python?

Advantages of Python are:

- Python is **Interpreted** language

Interpreted: Python is an interpreted language. It does not require prior compilation of code and executes instructions directly.

- It is Free and open source

Free and open source: It is an open-source project which is publicly available to reuse. It can be downloaded free of cost.

- It is **Extensible**

Extensible: It is very flexible and extensible with any module.

- Object-oriented

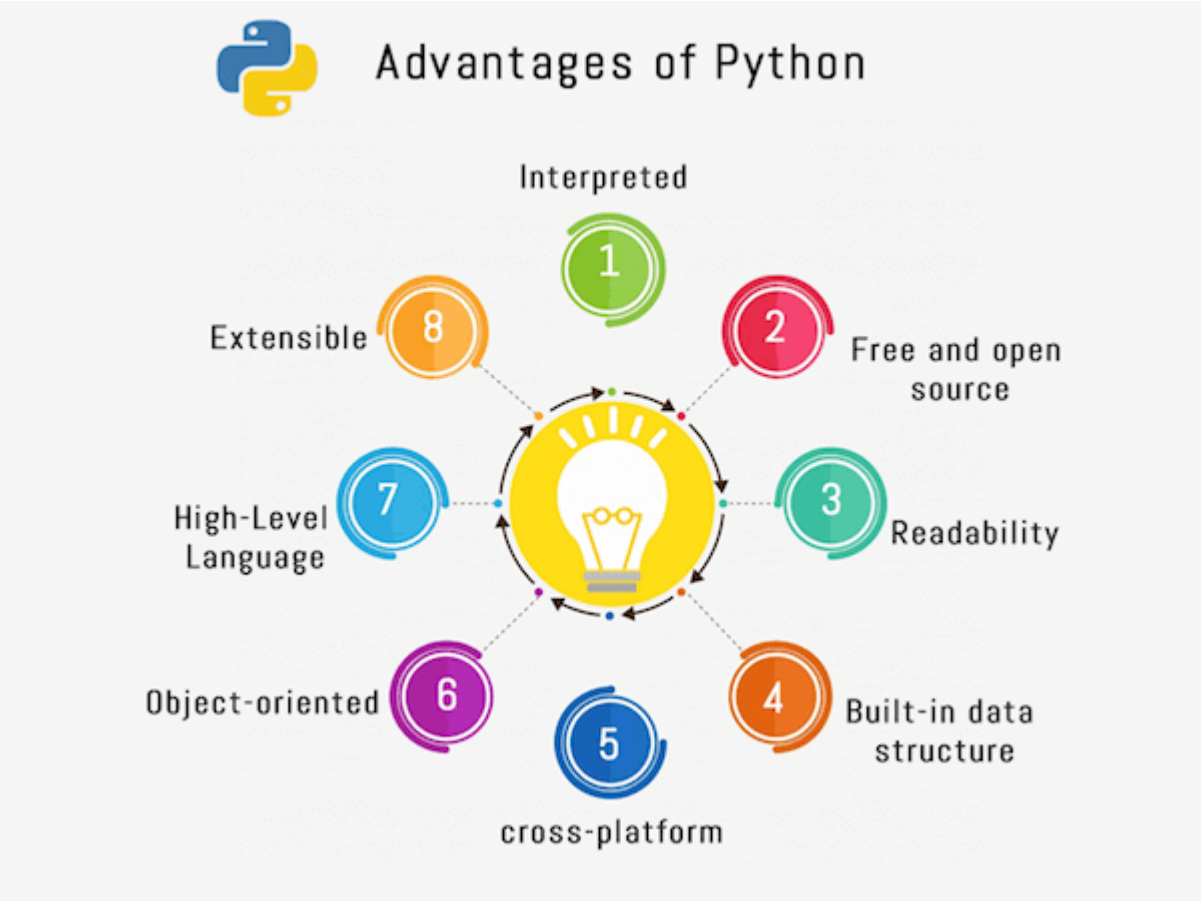
Object-oriented: Python allows to implement the Object-Oriented concepts to build application solution.

- It has Built-in data structure

Built-in data structure: Tuple, List, and Dictionary are useful integrated data structures provided by the language.

- Readability
- High-Level Language
- Cross-platform

Portable: Python programs can run on cross platforms without affecting its performance.



5) What is PEP 8?

PEP 8 stands for **Python Enhancement Proposal**, it can be defined as a document that helps us to provide the guidelines on how to write the Python code. It is basically a set of rules that specify how to format Python code for maximum readability. It was written by Guido van Rossum, Barry Warsaw and Nick Coghlan in 2001.

6) What do you mean by Python literals?

Literals can be defined as a data which is given in a variable or constant. Python supports the following literals:

String Literals

String literals are formed by enclosing text in the single or double quotes. For example, string literals are string values.

Example:

```
1. # in single quotes
```

```
2. single = 'AdnanManna'
3. # in double quotes
4. double = " AdnanManna"
5. # multi-line String
6. multi = '''Adnan
7.           _
8.              Manna'''
9.
10. print(single)
11. print(double)
12. print(multi)
```

Output:

```
AdnanManna
AdnanManna
Adnan
           _
              Manna
```

Numeric Literals

Python supports three types of numeric literals integer, float and complex.

Example:

```
1. # Integer literal
2. a = 10
3. #Float Literal
4. b = 12.3
5. #Complex Literal
6. x = 3.14j
7. print(a)
8. print(b)
9. print(x)
```

Output:

```
10
12.3
3.14j
```

Boolean Literals

Boolean literals are used to denote Boolean values. It contains either True or False.

Example:

```
p = (1 == True)
q = (1 == False)
r = True + 3
s = False + 7

print("p is", p)
print("q is", q)
print("r:", r)
print("s:", s)
```

Output:

```
p is True
q is False
r: 4
s: 7
```

Special literals

Python contains one special literal, that is, **'None'**. This special literal is used for defining a null variable. If 'None' is compared with anything else other than a 'None', it will return false.

Example:

```
word = None
print(word)
```

Output:

```
None
```

---

7) Explain Python Functions?

A function is a section of the program or a block of code that is written once and can be executed whenever required in the program. A function is a block of self-contained statements which has a valid name, parameters list, and body. Functions make programming more functional and modular to perform modular tasks. Python provides several built-in functions to complete tasks and also allows a user to create new functions as well.

There are three types of functions:

- **Built-In Functions:** copy(), len(), count() are the some built-in functions.

- **User-defined Functions:** Functions which are defined by a user known as user-defined functions.
- **Anonymous functions:** These functions are also known as lambda functions because they are not declared with the standard def keyword.

Example: A general syntax of user defined function is given below.

```
def function_name(parameters list):
    #--- statements---
    return a_value
```

---

8) What is zip() function in Python?

Python **zip()** function returns a zip object, which maps a similar index of multiple containers. It takes an iterable, convert into iterator and aggregates the elements based on iterables passed. It returns an iterator of tuples.

Signature

```
zip(iterator1, iterator2, iterator3 ...)
```

Parameters

iterator1, iterator2, iterator3: These are iterator objects that are joined together.

Return

It returns an iterator from two or more iterators.

Note: If the given lists are of different lengths, zip stops generating tuples when the first list ends. It means two lists are having 3, and 5 lengths will create a 3-tuple.

---

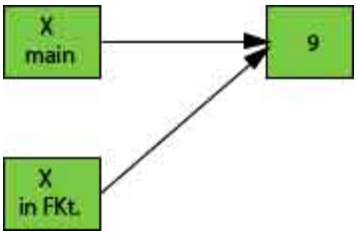
9) What is Python's parameter passing mechanism?

There are two parameters passing mechanism in Python:

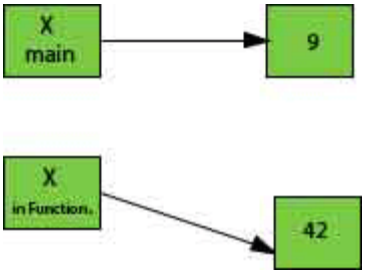
- Pass by references
- Pass by value

By default, all the parameters (arguments) are passed "by reference" to the functions. Thus, if you change the value of the parameter within a function, the change is reflected in the calling function as

well. It indicates the original variable. For example, if a variable is declared as a = 10, and passed to a function where it's value is modified to a = 20. Both the variables denote to the same value.



The pass by value is that whenever we pass the arguments to the function only values pass to the function, no reference passes to the function. It makes it immutable that means not changeable. Both variables hold the different values, and original value persists even after modifying in the function.



Python has a default argument concept which helps to call a method using an arbitrary number of arguments.

---

10) How to overload constructors or methods in Python?

Python's constructor: _init__ () is the first method of a class. Whenever we try to instantiate an object __init__() is automatically invoked by python to initialize members of an object. We can't overload constructors or methods in Python. It shows an error if we try to overload.

Example:

```
class student:
    def __init__(self, name):
        self.name = name
    def __init__(self, name, email):
        self.name = name
        self.email = email

# This line will generate an error
#st = student("rahul")
```

```
# This line will call the second constructor
st = student("rahul", "rahul@gmail.com")
print("Name: ", st.name)
print("Email id: ", st.email)
```

Output:

```
Name:  rahul
Email id:  rahul@gmail.com
```

---

11) What is the difference between remove() function and del statement?

The user can use the remove() function to delete a specific object in the list.

Example:

```
list_1 = [ 3, 5, 7, 3, 9, 3 ]
print(list_1)
list_1.remove(3)
print("After removal: ", list_1)
```

Output:

```
[3, 5, 7, 3, 9, 3]
After removal: [5, 7, 3, 9, 3]
```

If you want to delete an object at a specific location (index) in the list, you can either use **del** or **pop**.

Example:

```
list_1 = [ 3, 5, 7, 3, 9, 3 ]
print(list_1)
del list_1[2]
print("After deleting: ", list_1)
```

Output:

```
[3, 5, 7, 3, 9, 3]
After deleting: [3, 5, 3, 9, 3]
```

Note: You don't need to import any extra module to use these functions for removing an element from the list.

We cannot use these methods with a tuple because the tuple is different from the list.

---

12) What is swapcase() function in the Python?

It is a string's function which converts all uppercase characters into lowercase and vice versa. It is used to alter the existing case of the string. This method creates a copy of the string which contains all the characters in the swap case. If the string is in lowercase, it generates a small case string and vice versa. It automatically ignores all the non-alphabetic characters. See an example below.

Example:

```
1. string = "IT IS IN LOWERCASE."
2. print(string.swapcase())
3.
4. string = "it is in uppercase."
5. print(string.swapcase())
```

Output:

```
it is in lowercase.
IT IS IN UPPERCASE.
```

---

13) How to remove whitespaces from a string in Python?

To remove the whitespaces and trailing spaces from the string, Python providies strip([str]) built-in function. This function returns a copy of the string after removing whitespaces if present. Otherwise returns original string.

Example:

```
1. string = " AdnanManna "
2. string2 = "    AdnanManna       "
3. string3 = "      AdnanManna "
4. print(string)
5. print(string2)
6. print(string3)
7. print("After stripping all have placed in a sequence:")
8. print(string.strip())
9. print(string2.strip())
10. print(string3.strip())
```

Output:

```
AdnanManna
    AdnanManna
       AdnanManna
```

```
After stripping all have placed in a sequence:
AdnanManna
AdnanManna
AdnanManna
```

14) How to remove leading whitespaces from a string in the Python?

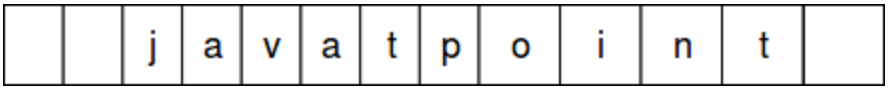
To remove leading characters from a string, we can use lstrip() function. It is Python string function which takes an optional char type parameter. If a parameter is provided, it removes the character. Otherwise, it removes all the leading spaces from the string.

Example:

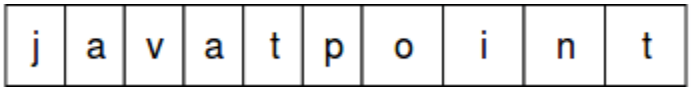
```
1. string = "  AdnanManna "
2. string2 = "    AdnanManna       "
3. print(string)
4. print(string2)
5. print("After stripping all leading whitespaces:")
6. print(string.lstrip())
7. print(string2.lstrip())
```

Output:

```
AdnanManna
    AdnanManna
After stripping all leading whitespaces:
AdnanManna
AdnanManna
```



After stripping, all the whitespaces are removed, and now the string looks like the below:



15) Why do we use join() function in Python?

The join() is defined as a string method which returns a string value. It is concatenated with the elements of an iterable. It provides a flexible way to concatenate the strings. See an example below.

Example:

```
1. str = "Rohan"
```

```
2. str2 = "ab"
3. # Calling function
4. str2 = str.join(str2)
5. # Displaying result
6. print(str2)
```

Output:

```
aRohanb
```

16) Give an example of shuffle() method?

This method shuffles the given string or an array. It randomizes the items in the array. This method is present in the random module. So, we need to import it and then we can call the function. It shuffles elements each time when the function calls and produces different output.

Example:

```
1. # import the random module
2. import random
3. # declare a list
4. sample_list1 = ['Z', 'Y', 'X', 'W', 'V', 'U']
5. print("Original LIST1: ")
6. print(sample_list1)
7. # first shuffle
8. random.shuffle(sample_list1)
9. print("\nAfter the first shuffle of LIST1: ")
10. print(sample_list1)
11. # second shuffle
12. random.shuffle(sample_list1)
13. print("\nAfter the second shuffle of LIST1: ")
14. print(sample_list1)
```

Output:

```
Original LIST1:
['Z', 'Y', 'X', 'W', 'V', 'U']

After the first shuffle of LIST1:
['V', 'U', 'W', 'X', 'Y', 'Z']

After the second shuffle of LIST1:
['Z', 'Y', 'X', 'U', 'V', 'W']
```

17) What is the use of break statement?

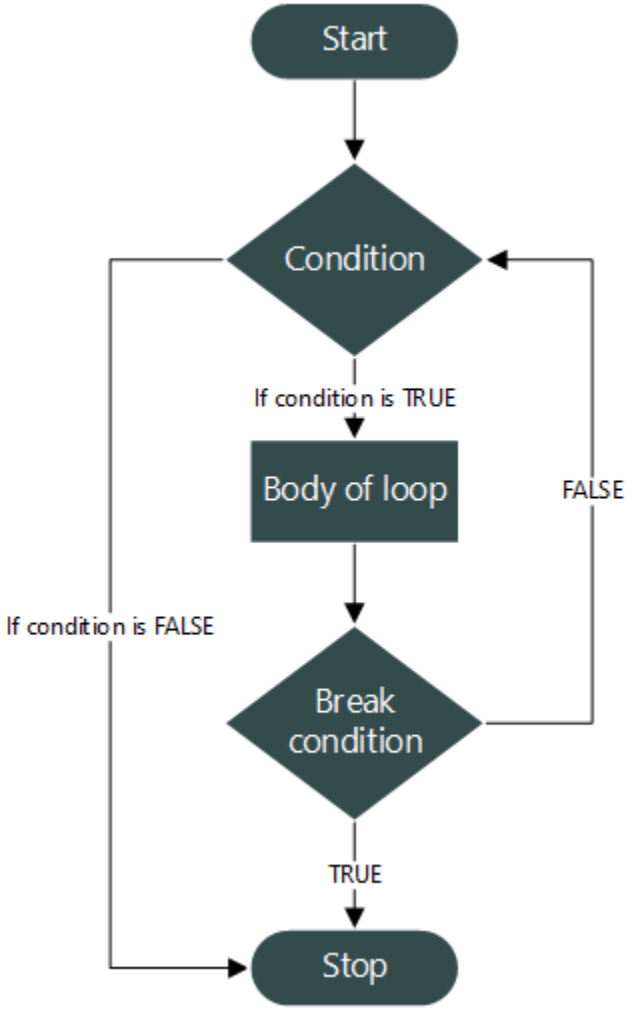
The break statement is used to terminate the execution of the current loop. Break always breaks the current execution and transfer control to outside the current block. If the block is in a loop, it exits from the loop, and if the break is in a nested loop, it exits from the innermost loop.

Example:

```
1. list_1 = ['X', 'Y', 'Z']
2. list_2 = [11, 22, 33]
3. for i in list_1:
4.     for j in list_2:
5.         print(i, j)
6.         if i == 'Y' and j == 33:
7.             print('BREAK')
8.             break
9.     else:
10.        continue
11.    break
```

Output:

```
2
X 11
X 22
X 33
Y 11
Y 22
Y 33
BREAK
```



Python Break statement flowchart.

---

18) What is tuple in Python?

A tuple is a built-in data collection type. It allows us to store values in a sequence. It is immutable, so no change is reflected in the original data. It uses () brackets rather than [] square brackets to create a tuple. We cannot remove any element but can find in the tuple. We can use indexing to get elements. It also allows traversing elements in reverse order by using negative indexing. Tuple supports various methods like max(), sum(), sorted(), Len() etc.

To create a tuple, we can declare it as below.

Example:

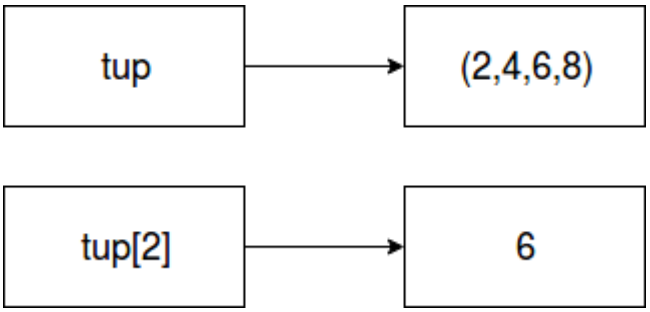
```
1. # Declaring tuple
2. tup = (2,4,6,8)
```

```
# Displaying value
print(tup)

# Displaying Single value
print(tup[2])
```

Output:

```
(2, 4, 6, 8)
6
```



It is immutable. So updating tuple will lead to an error.

Example:

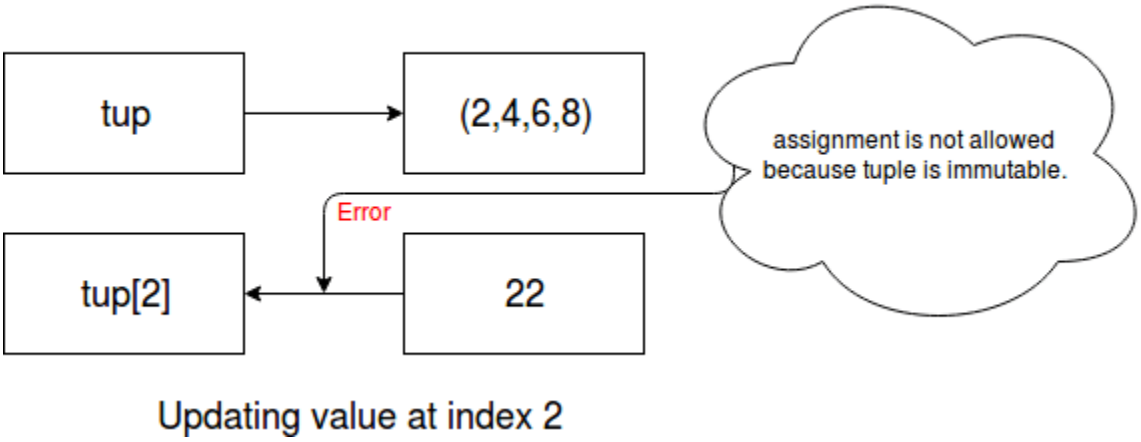
```
# Declaring tuple
tup = (2,4,6,8)
# Displaying value
print(tup)

# Displaying Single value
print(tup[2])

# Updating by assigning new value
tup[2]=22
# Displaying Single value
print(tup[2])
```

Output:

```
tup[2]=22
TypeError: 'tuple' object does not support item assignment
(2, 4, 6, 8)
```



Updating value at index 2

19) Which are the file related libraries/modules in Python?

The Python provides libraries/modules that enable you to manipulate text files and binary files on the file system. It helps to create files, update their contents, copy, and delete files. The libraries are os, os.path, and shutil.

Here, os and os.path - modules include a function for accessing the filesystem

while shutil - module enables you to copy and delete the files.

20) What are the different file processing modes supported by Python?

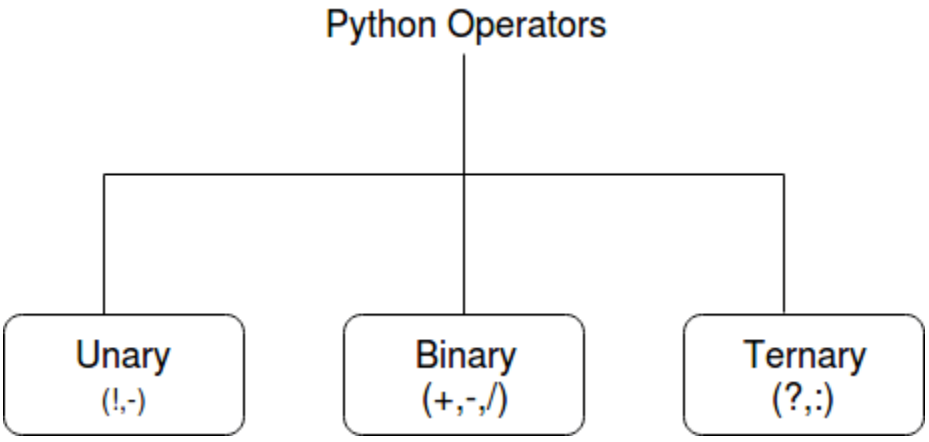
Python provides **four** modes to open files. The read-only (r), write-only (w), read-write (rw) and append mode (a). 'r' is used to open a file in read-only mode, 'w' is used to open a file in write-only mode, 'rw' is used to open in reading and write mode, 'a' is used to open a file in append mode. If the mode is not specified, by default file opens in read-only mode.

- Read-only mode (r): Open a file for reading. It is the default mode.
- Write-only mode (w): Open a file for writing. If the file contains data, data would be lost. Other a new file is created.
- Read-Write mode (rw): Open a file for reading, write mode. It means updating mode.
- Append mode (a): Open for writing, append to the end of the file, if the file exists.

21) What is an operator in Python?

An operator is a particular symbol which is used on some values and produces an output as a result. An operator works on operands. Operands are numeric literals or variables which hold some values.

Operators can be unary, binary or ternary. An operator which requires a single operand known as a **unary operator**, which require two operands known as a **binary operator** and which require three operands is called **ternary operator.**



Example:

```
1. # Unary Operator
2. A = 12
3. B = -(A)
4. print (B)
5. # Binary Operator
6. A = 12
7. B = 13
8. print (A + B)
9. print (B * A)
10. #Ternary Operator
11. A = 12
12. B = 13
13. min = A if A < B else B
14.
15. print(min)
```

Output:

```
# Unary Operator
-12
# Binary Operator
25
156
# Ternary Operator
12
```

22) What are the different types of operators in Python?

Python uses a rich set of operators to perform a variety of operations. Some individual operators like membership and identity operators are not so familiar but allow to perform operations.

- Arithmetic OperatorsRelational Operators
- Assignment Operators
- Logical Operators
- Membership Operators
- Identity Operators
- Bitwise Operators

Arithmetic operators perform basic arithmetic operations. For example "+" is used to add and "?" is used for subtraction.

Example:

```
# Adding two values
print(12+23)
# Subtracting two values
print(12-23)
# Multiplying two values
print(12*23)
# Dividing two values
print(12/23)
```

Output:

```
35
-11
276
0.5217391304347826
```

Relational Operators are used to comparing the values. These operators test the conditions and then returns a boolean value either True or False.

Examples of Relational Operators

Example:

```
a, b = 10, 12
print(a==b) # False
print(a<b) # True
print(a<=b) # True
print(a!=b) # True
```

Output:

```
False
True
True
True
```

Assignment operators are used to assigning values to the variables. See the examples below.

Example:

```
# Examples of Assignment operators
a=12
print(a) # 12
```

```
4. a += 2
5. print(a) # 14
6. a -= 2
7. print(a) # 12
8. a *=2
9. print(a) # 24
10. a **=2
11. print(a) # 576
```

Output:

```
12
14
12
24
576
```

Logical operators are used to performing logical operations like And, Or, and Not. See the example below.

Example:

```
1. # Logical operator examples
2. a = True
3. b = False
4. print(a and b) # False
5. print(a or b) # True
6. print(not b) # True
```

Output:

```
False
True
True
```

Membership operators are used to checking whether an element is a member of the sequence (list, dictionary, tuples) or not. Python uses two membership operators in and not in operators to check element presence. See an example.

Example:

```
1. # Membership operators examples
2. list = [2,4,6,7,3,4]
3. print(5 in list) # False
4. cities = ("india","delhi")
```

```
5. print("tokyo" not in cities) #True
```

Output:

```
False
True
```

Identity Operators (is and is not) both are used to check two values or variable which are located on the same part of the memory. Two variables that are equal does not imply that they are identical. See the following examples.

Example:

```
1. # Identity operator example
2. a = 10
3. b = 12
4. print(a is b) # False
5. print(a is not b) # True
```

Output:

```
False
True
```

Bitwise Operators are used to performing operations over the bits. The binary operators (&, |, OR) work on bits. See the example below.

Example:

```
1. # Identity operator example
2. a = 10
3. b = 12
4. print(a & b) # 8
5. print(a | b) # 14
6. print(a ^ b) # 6
7. print(~a) # -11
```

Output:

```
8
14
6
-11
```

---

23) How to create a Unicode string in Python?

In Python 3, the old Unicode type has replaced by "str" type, and the string is treated as Unicode by default. We can make a string in Unicode by using art.title.encode("utf-8") function.

Example:

```
1. unicode_1 = ("\u0123", "\u2665", "\U0001f638", "\u265E", "\u265F", "\u2168")
2. print (unicode_1)
```

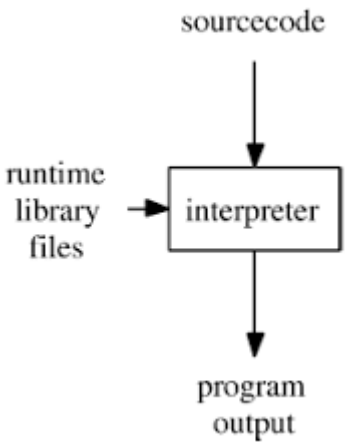
Output:

```
unicode_1: ('ģ', '♥', '😸', '♞', '♟', 'Ⅸ')
```

24) is Python interpreted language?

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

Unlike Java or C, Python does not require compilation before execution.



25) How is memory managed in Python?

Memory is managed in Python in the following ways:

- 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.
- 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.

- 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.

---

26) What is the Python decorator?

Decorators are very powerful and a useful tool in Python that allows the programmers to add functionality to an existing code. This is also called metaprogramming because a part of the program tries to modify another part of the program at compile time. It allows the user to wrap another function to extend the behaviour of the wrapped function, without permanently modifying it.

Example:

```
def function_is_called():
    def function_is_returned():
        print("AdnanManna")
    return function_is_returned
new_1 = function_is_called()
# Outputs "AdnanManna"
new_1()
```

Output:

```
AdnanManna
```

Functions vs. Decorators

A function is a block of code that performs a specific task whereas a decorator is a function that modifies other functions.

---

27) What are the rules for a local and global variable in Python?

Global Variables:

- Variables declared outside a function or in global space are called global variables.
- If a variable is ever assigned a new value inside the function, the variable is implicitly local, and we need to declare it as 'global' explicitly. To make a variable globally, we need to declare it by using global keyword.
- Global variables are accessible anywhere in the program, and any function can access and modify its value.

Example:

```
A = "AdnanManna"
def my_function():
  print(A)
my_function()
```

Output:

```
AdnanManna
```

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.
- If a variable is assigned a new value anywhere within the function's body, it's assumed to be a local.
- Local variables are accessible within local body only.

Example:

```
def my_function2():
    K = " AdnanManna Local"
    print(K)
my_function2()
```

Output:

```
AdnanManna Local
```

---

28) What is the namespace in Python?

The namespace is a fundamental idea to structure and organize the code that is more useful in large projects. However, it could be a bit difficult concept to grasp if you're new to programming. Hence, we tried to make namespaces just a little easier to understand.

A namespace is defined as a simple system to control the names in a program. It ensures that names are unique and won't lead to any conflict.

Also, Python implements namespaces in the form of dictionaries and maintains name-to-object mapping where names act as keys and the objects as values.

---

29) What are iterators in Python?

In Python, iterators are used to iterate a group of elements, containers like a list. Iterators are the collection of items, and it can be a list, tuple, or a dictionary. Python iterator implements __itr__ and next() method to iterate the stored elements. In Python, we generally use loops to iterate over the collections (list, tuple).

In simple words: Iterators are objects which can be traversed though or iterated upon.

---

30) What is a generator in Python?

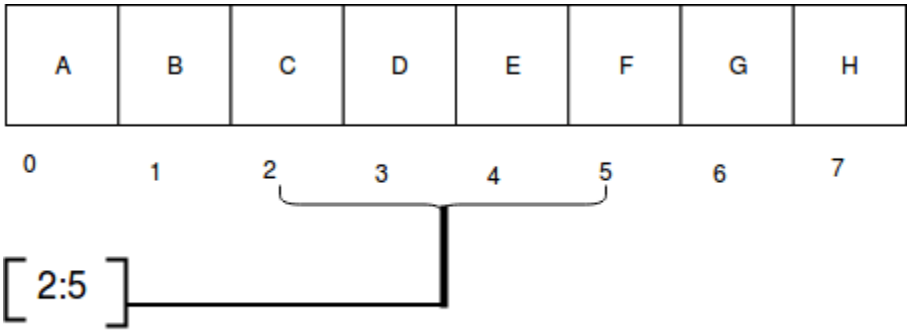
In Python, the generator is a way that specifies how to implement iterators. It is a normal function except that it yields expression in the function. It does not implements __itr__ and next() method and reduce other overheads as well.

If a function contains at least a yield statement, it becomes a generator. The yield keyword pauses the current execution by saving its states and then resume from the same when required.

---

31) What is slicing in Python?

Slicing is a mechanism used to select a range of items from sequence type like list, tuple, and string. It is beneficial and easy to get elements from a range by using slice way. It requires a : (colon) which separates the start and end index of the field. All the data collection types List or tuple allows us to use slicing to fetch elements. Although we can get elements by specifying an index, we get only single element whereas using slicing we can get a group of elements.



Example:

```
1. Q = "AdnanManna, Python Interview Questions!"
2. print(Q[2:25])
```

Output:

```
nanManna, Python Intervi
```

32) What is a dictionary in Python?

The Python dictionary is a built-in data type. It defines a one-to-one relationship between keys and values. Dictionaries contain a pair of keys and their corresponding values. It stores elements in key and value pairs. The keys are unique whereas values can be duplicate. The key accesses the dictionary elements.

Keys index dictionaries.

Example:

The following example contains some keys Country Hero & Cartoon. Their corresponding values are India, Modi, and Rahul respectively.

```
1. dict = {'Country': 'India', 'Hero': 'Modi', 'Cartoon': 'Rahul'}
2. print ("Country: ", dict['Country'])
3. print ("Hero: ", dict['Hero'])
4. print ("Cartoon: ", dict['Cartoon'])
```

Output:

```
Country:  India
Hero:  Modi
Cartoon:  Rahul
```

33) What is Pass in Python?

Pass specifies a Python statement without operations. It is a placeholder in a compound statement. If we want to create an empty class or functions, the pass keyword helps to pass the control without error.

Example:

```
1. class Student:
2.     pass # Passing class
3. class Student:
4.     def info():
5.         pass # Passing function
```

34) Explain docstring in Python?

The Python docstring is a string literal that occurs as the first statement in a module, function, class, or method definition. It provides a convenient way to associate the documentation.

String literals occurring immediately after a simple assignment at the top are called "attribute docstrings".

String literals occurring immediately after another docstring are called "additional docstrings".

Python uses triple quotes to create docstrings even though the string fits on one line.

Docstring phrase ends with a period (.) and can be multiple lines. It may consist of spaces and other special chars.

Example:

```
1. # One-line docstrings
2. def hello():
3.     """A function to greet."""
4.     return "hello"
```

---

35) What is a negative index in Python and why are they used?

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 go 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.

---

36) What is pickling and unpickling in Python?

The Python pickle is defined as a module which accepts any Python object and converts it into a string representation. It dumps the Python object into a file using the dump function; this process is called **Pickling**.

The process of retrieving the original Python objects from the stored string representation is called as **Unpickling**.

---

37) Which programming language is a good choice between Java and Python?

Java and Python both are object-oriented programming languages. Let's compare both on some criteria given below:

Criteria	Java	Python
Ease of use	Good	Very Good
Coding Speed	Average	Excellent
Data types	Static type	Dynamic type
Data Science and Machine learning application	Average	Very Good

---

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

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

Help() function: The help() function is used to display the documentation string and also facilitates us to see the help related to modules, keywords, and attributes.

Dir() function: The dir() function is used to display the defined symbols.

---

39) What are the differences between Python 2.x and Python 3.x?

Python 2.x is an older version of Python. Python 3.x is newer and latest version. Python 2.x is legacy now. Python 3.x is the present and future of this language.

The most visible difference between Python2 and Python3 is in print statement (function). In Python 2, it looks like print "Hello", and in Python 3, it is print ("Hello").

String in Python2 is ASCII implicitly, and in Python3 it is Unicode.

The xrange() method has removed from Python 3 version. A new keyword as is introduced in Error handling.

40) How Python does Compile-time and Run-time code checking?

In Python, some amount of coding is done at compile time, but most of the checking such as type, name, etc. are postponed until code execution. Consequently, if the Python code references a user-defined function that does not exist, the code will compile successfully. The Python code will fail only with an exception when the code execution path does not exist.

41) What is the shortest method to open a text file and display its content?

The shortest way to open a text file is by using "with" command in the following manner:

Example:

```
with open("FILE NAME", "r") as fp:
    fileData = fp.read()
# To print the contents of the file
print(fileData)
```

Output:

```
"The data of the file will be printed."
```

42) What is the usage of enumerate () function in Python?

The enumerate() function is used to iterate through the sequence and retrieve the index position and its corresponding value at the same time.

Example:

```
list_1 = ["A","B","C"]
s_1 = "Javatpoint"
# creating enumerate objects
object_1 = enumerate(list_1)
object_2 = enumerate(s_1)

print ("Return type:",type(object_1))
print (list(enumerate(list_1)))
print (list(enumerate(s_1)))
```

Output:

```
Return type:
[(0, 'A'), (1, 'B'), (2, 'C')]
[(0, 'J'), (1, 'a'), (2, 'v'), (3, 'a'), (4, 't'), (5, 'p'), (6, 'o'), (7, 'i'), (8, 'n'),
(9, 't')]
```

43) Give the output of this example: A[3] if A=[1,4,6,7,9,66,4,94].

Since indexing starts from zero, an element present at 3rd index is 7. So, the output is 7.

44) What is type conversion in Python?

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.

45) How to send an email in Python Language?

To send an email, Python provides smtplib and email modules. Import these modules into the created mail script and send mail by authenticating a user.

It has a method SMTP(smtp-server, port). It requires two parameters to establish SMTP connection.

A simple example to send an email is given below.

Example:

```
import smtplib
# Calling SMTP
s = smtplib.SMTP('smtp.gmail.com', 587)
# TLS for network security
s.starttls()
# User email Authentication
s.login("sender@email_id", "sender_email_id_password")
# Message to be sent
message = "Message_sender_need_to_send"
# Sending the mail
s.sendmail("sender@email_id ", "receiver@email_id", message)
```

---

46) What is the difference between Python Arrays and lists?

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
User_Array = arr.array('i', [1,2,3,4])
User_list = [1, 'abc', 1.20]
print (User_Array)
print (User_list)
```

Output:

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

---

47) What is lambda function in Python?

The anonymous function in python is a function that is defined without a name. The normal functions are defined using a keyword "def", whereas, the anonymous functions are defined using the lambda function. **The anonymous functions are also called as lambda functions**.

---

48) Why do lambda forms in Python not have the statements?

Lambda forms in Python does not have the statement because it is used to make the new function object and return them in runtime.

---

49) What are functions in Python?

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 New_func():
    print ("Hi, Welcome to JavaTpoint")
New_func() #calling the function
```

Output:

```
Hi, Welcome to JavaTpoint
```

---

50) What is __init__?

The __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.

Example:

```
class Employee_1:
    def __init__(self, name, age,salary):
        self.name = name
        self.age = age
        self.salary = 20000
E_1 = Employee_1("pqr", 20, 25000)
# E1 is the instance of class Employee.
#__init__ allocates memory for E1.
print(E_1.name)
print(E_1.age)
print(E_1.salary)
```

Output:

```
pqr
20
25000
```

51) What is self in Python?

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.

---

52) How can you generate random numbers in Python?

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

1. **import** random
2. 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:

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.

uniform(a, b): it chooses a floating point number that is defined in the range of [a,b).lyt returns the floating point number

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.

The Random class that is used and instantiated creates independent multiple random number generators.

---

53) What is PYTHONPATH?

PYTHONPATH 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.

---

54) What are python modules? Name some commonly used built-in modules in Python?

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

---

55) What is the difference between range & xrange?

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.

---

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

- 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.
- 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.
- NumPy is not just more efficient; it is also more convenient. We get a lot of vector and matrix operations for free, which sometimes allow one to avoid unnecessary work. And they are also efficiently implemented.
- NumPy array is faster and we get a lot built in with NumPy, FFTs, convolutions, fast searching, basic statistics, linear algebra, histograms, etc.

---

57) Mention what the Django templates consist of.

The template is a simple text file. It 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.

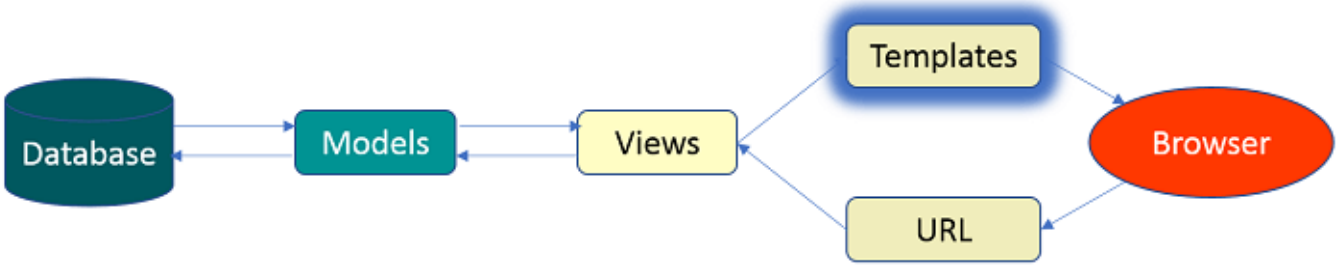


Figure: Python Interview Questions – Django Template

---

58) Explain the use of session in Django framework?

Django provides a session that lets the user 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.

Figure: Python Interview Questions – Django Framework

So, the data itself is not stored client side. This is good from a security perspective.