# PYTHON

# Top 150 Python Interview Questions & Answers

---

## 1. What is the Difference Between a Shallow Copy and a Deep Copy?

**Deep copy** creates a different object and populates it with the child objects of the original object. Therefore, changes in the original object are not reflected in the copy. `copy.deepcopy()` creates a deep copy.

**Shallow copy** creates a different object and populates it with the references of the child objects within the original object. Therefore, changes in the original object are reflected in the copy. `copy.copy()` creates a shallow copy.

---

## 2. How Is Multithreading Achieved in Python?

Multithreading usually implies that multiple threads are executed concurrently. The Python Global Interpreter Lock doesn't allow more than one thread to hold the Python interpreter at that particular point of time. So, multithreading in Python is achieved through context switching. It is quite different from multiprocessing which opens up multiple processes across multiple threads.

---

## 3. Discuss Django Architecture.

Here you can also find a comprehensive guide on Python Django Tutorial that is very easy to understand.

Django is a web service used to build your web pages. Its architecture is as shown:

- **Template**: The frontend of the web page
- **Model**: The backend where the data is stored
- **View**: It interacts with the model and template and maps it to the URL
- **Django**: Serves the page to the user

---

## 4. What Advantage Does the Numpy Array Have over a Nested List?

Numpy is written in C so that all its complexities are backed into a simple-to-use module. Lists, on the other hand, are dynamically typed. Therefore, Python must check the data type of each element every time it uses it. This makes Numpy arrays much faster than lists.

Numpy has a lot of additional functionality that the list doesn't offer; for instance, a lot of things can be automated in Numpy.

## 5. What are Pickling and Unpickling?

If you just created a neural network model, you can save that model to your hard drive, pickle it, and then unpickle it to bring it back into another software program or use it at a later time.

| **Pickling** | **Unpickling** |
| --- | --- |
| Converting a Python object hierarchy to a byte stream is called pickling | Converting a byte stream to a Python object hierarchy is called unpickling |
| Pickling is also referred to as serialization | Unpickling is also referred to as deserialization |

## 6. How is Memory Managed in Python?

Python has a private heap space that stores all the objects. The Python memory manager regulates various aspects of this heap, such as sharing, caching, segmentation, and allocation. The user has no control over the heap; only the Python interpreter has access.

## 7. Are Arguments in Python Passed by Value or by Reference?

Arguments are passed in Python by reference. This means that any changes made within a function are reflected in the original object.

Consider two sets of code shown below:

```python
# Example 1
def func(l):
    l[0] = 3

l = [1, 2, 3, 4]
func(l)
print(l)  # Output: [3, 2, 3, 4]

# Example 2
def func(l):
    l = [3, 2, 3, 4]

l = [1, 2, 3, 4]
func(l)
print(l)  # Output: [1, 2, 3, 4]
```

In the first example, we only assigned a value to one element of 'l', so the output is [3, 2, 3, 4].

In the second example, we have created a whole new object for 'l'. But, the values [3, 2, 3, 4] don't show up in the output as it is outside the definition of the function.

---

## 8. How Would You Generate Random Numbers in Python?

To generate random numbers in Python, you must first import the random module. The `random()` function generates a random float value between 0 & 1.

```python
random.random()
```

The `randrange()` function generates a random number within a given range.

**Syntax:** `randrange(beginning, end, step)`

**Example:**

```python
random.randrange(1, 10, 2)
```

---

## 9. What Does the // Operator Do?

In Python, the `/` operator performs division and returns the quotient in float.

**Example:** `5 / 2` returns `2.5`

The `//` operator, on the other hand, returns the quotient as an integer.

**Example:** `5 // 2` returns `2`

---

## 10. What Does the 'is' Operator Do?

The `is` operator compares the id of the two objects.

```python
```

```python
list1 = [1, 2, 3]
list2 = [1, 2, 3]
list3 = list1

list1 == list2  # True
list1 is list2  # False
list1 is list3  # True
```

---

## 11. What Is the Purpose of the Pass Statement?

The pass statement is used when there's a syntactic but not an operational requirement.

**Example:** The program below prints a string ignoring the spaces.

```python
var = "Simple learn"
for i in var:
    if i == " ":
        pass
    else:
        print(i, end="")
```

Here, the pass statement refers to 'no action required.'

---

## 12. How Will You Check If All the Characters in a String Are Alphanumeric?

Python has an inbuilt method `isalnum()` which returns true if all characters in the string are alphanumeric.

**Example:**

```python
"abcd123".isalnum()
# Output: True

"abcd@123#".isalnum()
# Output: False
```

Another way is to use regex as shown:

```python
```

```
import re
bool(re.match('[A-Za-z0-9]+$', 'abcd123'))
# Output: True

bool(re.match('[A-Za-z0-9]+$', 'abcd@123'))
# Output: False
```

---

## 13. How Will You Merge Elements in a Sequence?

There are three types of sequences in Python:

- Lists
- Tuples
- Strings

**Example of Lists:**

```
python

l1 = [1, 2, 3]
l2 = [4, 5, 6]
l1 + l2
# Output: [1, 2, 3, 4, 5, 6]
```

**Example of Tuples:**

```
python

t1 = (1, 2, 3)
t2 = (4, 5, 6)
t1 + t2
# Output: (1, 2, 3, 4, 5, 6)
```

**Example of String:**

```
python

s1 = "Simpli"
s2 = "learn"
s1 + s2
# Output: 'Simplilearn'
```

---

## 14. How Would You Remove All Leading Whitespace in a String?

Python provides the inbuilt function `lstrip()` to remove all leading spaces from a string.

```python
" Python".lstrip()
# Output: 'Python'
```

---

## 15. How Would You Replace All Occurrences of a Substring with a New String?

The `replace()` function can be used with strings for replacing a substring with a given string.

**Syntax:** `str.replace(old, new, count)`

`replace()` returns a new string without modifying the original string.

**Example:**

```python
"Hey John. How are you, John?".replace("john", "John", 1)
# Output: "Hey John. How are you, John?"
```

---

## 16. What Is the Difference Between Del and Remove() on Lists?

Here is an example to understand the two statements:

```python
lis = ['a', 'b', 'c', 'd']
del lis[1:3]
print(lis)
# Output: ["a", "d"]

lis = ['a', 'b', 'b', 'd']
lis.remove('b')
print(lis)
# Output: ['a', 'b', 'd']
```

| **del** | **remove()** |
|---|---|
| del removes all elements of a list within a given range | remove() removes the first occurrence of a particular character |
| Syntax: `del list[start:end]` | Syntax: `list.remove(element)` |

**Note:** In the range 1:3, the elements are counted up to 2 and not 3.

---

## 17. How Do You Display the Contents of a Text File in Reverse Order?

You can display the contents of a text file in reverse order using the following steps:

1. Open the file using the `open()` function
2. Store the contents of the file in a list
3. Reverse the contents of the list
4. Run a for loop to iterate through the list

---

## 18. Differentiate Between append() and extend().

| **append()** | **extend()** |
|---|---|
| append() adds an element to the end of the list | extend() adds elements from an iterable to the end of the list |
| **Example:**<br>`lst = [1, 2, 3]`<br>`lst.append(4)`<br>`print(lst)`<br>Output: `[1, 2, 3, 4]` | **Example:**<br>`lst = [1, 2, 3]`<br>`lst.extend([4, 5, 6])`<br>`print(lst)`<br>Output: `[1, 2, 3, 4, 5, 6]` |

---

## 19. What Is the Output of the Below Code? Justify Your Answer.

```python
def addToList(val, list=[]):
    list.append(val)
    return list

list1 = addToList(1)
list2 = addToList(123, [])
list3 = addToList('a')

print("list1 = %s" % list1)
print("list2 = %s" % list2)
print("list3 = %s" % list3)
```

**Output:**

```
list1 = [1, 'a']
list2 = [123]
list3 = [1, 'a']
```

Note that list1 and list3 are equal. When we passed the information to the addToList, we did it without a second value. If we don't have an empty list as the second value, it will start with an empty list, which we then append. For list2, we appended the value to an empty list, so its value becomes [123].

For list3, we're adding 'a' to the list. Because we didn't designate the list, it is a shared value. It means the list doesn't reset and we get its value as [1, 'a'].

Remember that a default list is created only once during the function definition and not during its call number.

---

## 20. What Is the Difference Between a List and a Tuple?

Lists are mutable while tuples are immutable.

**Example:**

**List:**

```python
lst = [1, 2, 3]
lst[2] = 4
print(lst)
# Output: [1, 2, 4]
```

**Tuple:**

```python
tpl = (1, 2, 3)
tpl[2] = 4
print(tpl)
# Output: TypeError: 'tuple' object does not support item assignment
```

There is an error because you can't change the tuple 123 into 124. You have to completely reassign the tuple to a new value.

---

## 21. What Is Docstring in Python?

This is one of the most frequently asked Python interview questions.

Docstrings are used in providing documentation to various Python modules, classes, functions, and methods.

**Example:**

```python
```

```
def add(a, b):
    """This function adds two numbers."""
    sum = a + b
    return sum

sum = add(10, 20)
print("Accessing docstring method 1:", add.__doc__)
print("Accessing docstring method 2:", end="")
help(add)
```

**Output:**

```
Accessing docstring method 1: This function adds two numbers.
Accessing docstring method 2: Help on function add in module __main__:

add(a, b)
    This function adds two numbers.
```

---

## 22. How Do You Use Print() Without the Newline?

The solution to this depends on the Python version you are using.

**Python v2:**

```
python

print("Hi."),
print("How are you?")
# Output: Hi. How are you?
```

**Python v3:**

```
python

print("Hi", end=" ")
print("How are you?")
# Output: Hi How are you?
```

---

## 23. How Do You Use the Split() Function in Python?

The `split()` function splits a string into several strings based on a specific delimiter.

**Syntax:** `string.split(delimiter, max)`

Where:

- **delimiter** is the character based on which the string is split. By default it is space.
- **max** is the maximum number of splits

**Example:**

```python
var = "Red,Blue,Green,Orange"
lst = var.split(",", 2)
print(lst)
# Output: ['Red', 'Blue', 'Green,Orange']
```

Here, we have a variable var whose values are to be split with commas. Note that '2' indicates that only the first two values will be split.

---

## 24. Is Python Object-oriented or Functional Programming?

Python is considered a multi-paradigm language.

**Python follows the object-oriented paradigm:**

- Python allows the creation of objects and their manipulation through specific methods
- It supports most of the features of OOP such as inheritance and polymorphism

**Python follows the functional programming paradigm:**

- Functions may be used as first-class objects
- Python supports Lambda functions which are characteristic of the functional paradigm

---

## 25. Write a Function Prototype That Takes a Variable Number of Arguments.

The function prototype is as follows:

```python
```

```
def function_name(*list):
    pass

def fun(*var):
    for i in var:
        print(i)

fun(1)
fun(1, 25, 6)
```

In the above code, `*` indicates that there are multiple arguments of a variable.

---

## 26. What Are *args and *kwargs?

***args:**

- It is used in a function prototype to accept a varying number of arguments.
- It's an iterable object.
- Usage: `def fun(*args)`

****kwargs:**

- It is used in a function prototype to accept the varying number of keyworded arguments.
- It's an iterable object
- Usage: `def fun(**kwargs):`
  - Example call: `fun(colour="red", units=2)`

---

## 27. "In Python, Functions Are First-class Objects." What Do You Infer from This?

It means that a function can be treated just like an object. You can assign them to variables, or pass them as arguments to other functions. You can even return them from other functions.

---

## 28. What Is the Output Of: Print(name)? Justify Your Answer.

`__name__` is a special variable that holds the name of the current module. Program execution starts from the main or code with 0 indentations. Thus, `__name__` has a value `__main__` in the above case. If the file is imported from another module, `__name__` holds the name of this module.

---

## 29. What Is a Numpy Array?

A numpy array is a grid of values, all of the same type, and is indexed by a tuple of non-negative integers. The number of dimensions determines the rank of the array. The shape of an array is a tuple of integers giving the size of the array along each dimension.

---

## 30. What Is the Difference Between Matrices and Arrays?

| Matrices | Arrays |
|---|---|
| A matrix comes from linear algebra and is a two-dimensional representation of data | An array is a sequence of objects of similar data type |
| It comes with a powerful set of mathematical operations that allow you to manipulate the data in interesting ways | An array within another array forms a matrix |

---

## 31. How Do You Get Indices of N Maximum Values in a Numpy Array?

python

```
import numpy as np
arr = np.array([1, 3, 2, 4, 5])
print(arr.argsort()[-N:][::-1])
```

---

## 32. How Would You Obtain the Res_set from the Train_set and the Test_set from Below?

python

```
train_set = np.array([1, 2, 3])
test_set = np.array([[0, 1, 2], [1, 2, 3]])
# Res_set = [[1, 2, 3], [0, 1, 2], [1, 2, 3]]
```

Choose the correct option:

1. `res_set = train_set.append(test_set)`
2. `res_set = np.concatenate([train_set, test_set])`
3. `resulting_set = np.vstack([train_set, test_set])`
4. None of these

Here, options a and b would both do horizontal stacking, but we want vertical stacking. So, option c is the right statement.

**Answer:** `resulting_set = np.vstack([train_set, test_set])`

## 33. How Would You Import a Decision Tree Classifier in Sklearn? Choose the Correct Option.

1. `from sklearn.decision_tree import DecisionTreeClassifier`
2. `from sklearn.ensemble import DecisionTreeClassifier`
3. `from sklearn.tree import DecisionTreeClassifier`
4. None of these

**Answer:** 3. `from sklearn.tree import DecisionTreeClassifier`

---

## 34. You Have Uploaded the Dataset in csv Format on Google Spreadsheet and Shared It Publicly. How Can You Access This in Python?

We can use the following code:

```python
link = 'https://docs.google.com/spreadsheets/d/...'
source = StringIO.StringIO(requests.get(link).content)
data = pd.read_csv(source)
```

---

## 35. What Is the Difference Between the Two Data Series Given Below?

`df['Name']` and `df.loc[:, 'Name']`, where:

```python
df = pd.DataFrame(['aa', 'bb', 'xx', 'uu'], [21, 16, 50, 33],
                  columns=['Name', 'Age'])
```

Choose the correct option:

1. 1 is the view of the original dataframe and 2 is a copy of original dataframe
2. 2 is the view of the original dataframe and 1 is a copy of original dataframe
3. Both are copies of original dataframe
4. Both are views of original dataframe

**Answer:** 3. Both are copies of the original dataframe.

---

## 36. You Get the Error "temp.csv" While Trying to Read a File Using Pandas. Which of the Following Could Correct It?

**Error:**

```
Traceback (most recent call last):
  File "<input>", line 1, in <module>
UnicodeEncodeError: 'ascii' codec can't encode a character
```

Choose the correct option:

1. `pd.read_csv("temp.csv", compression='gzip')`
2. `pd.read_csv("temp.csv", dialect='str')`
3. `pd.read_csv("temp.csv", encoding='utf-8')`
4. None of these

The error relates to the difference between utf-8 coding and Unicode. So option 3. `pd.read_csv("temp.csv", encoding='utf-8')` can correct it.

---

## 37. How Do You Set a Line Width in the Plot Given Below?

```python
import matplotlib.pyplot as plt
plt.plot([1, 2, 3, 4])
plt.show()
```

Choose the correct option:

1. In line two, write `plt.plot([1, 2, 3, 4], width=3)`
2. In line two, write `plt.plot([1, 2, 3, 4], line_width=3)`
3. In line two, write `plt.plot([1, 2, 3, 4], lw=3)`
4. None of these

**Answer:** 3. In line two, write `plt.plot([1, 2, 3, 4], lw=3)`

---

## 38. How Would You Reset the Index of a Dataframe to a Given List? Choose the Correct Option.

1. `df.reset_index(new_index)`

2. `df.reindex(new_index)`
3. `df.reindex_like(new_index)`
4. None of these

**Answer:** 2. `df.reindex(new_index)`

---

## 39. How Can You Copy Objects in Python?

The functions used to copy objects in Python are:

- `copy.copy()` for shallow copy
- `copy.deepcopy()` for deep copy

---

## 40. What Is the Difference Between range() and xrange() Functions in Python?

| **range()** | **xrange()** |
|---|---|
| range returns a Python list object | xrange returns an xrange object |

**Note:** xrange is only available in Python 2.x. In Python 3.x, range() behaves like xrange() from Python 2.x.

---

## 41. How Can You Check Whether a Pandas Dataframe Is Empty or Not?

The attribute `df.empty` is used to check whether a pandas data frame is empty or not.

```python
import pandas as pd
df = pd.DataFrame({'A': []})
df.empty
# Output: True
```

---

## 42. Write a Code to Sort an Array in Numpy by the (N-1)th Column.

This can be achieved by using `argsort()` function. Let us take an array X; the code to sort the (n-1)th column will be `x[x[:, n-2].argsort()]`

The code is as shown below:

```python
```

```python
import numpy as np
X = np.array([[1, 2, 3], [0, 5, 2], [2, 3, 4]])
X[X[:, 1].argsort()]
# Output: array([[1, 2, 3], [2, 3, 4], [0, 5, 2]])
```

---

## 43. How Do You Create a Series from a List, Numpy Array, and Dictionary?

The code is as shown:

```python
# Input
import numpy as np
import pandas as pd

mylist = list('abcedfghijklmnopqrstuvwxyz')
myarr = np.arange(26)
mydict = dict(zip(mylist, myarr))

# Solution
ser1 = pd.Series(mylist)
ser2 = pd.Series(myarr)
ser3 = pd.Series(mydict)
print(ser3.head())
```

---

## 44. How Do You Get the Items Not Common to Both Series A and Series B?

```python
# Input
import pandas as pd
ser1 = pd.Series([1, 2, 3, 4, 5])
ser2 = pd.Series([4, 5, 6, 7, 8])

# Solution
ser_u = pd.Series(np.union1d(ser1, ser2))  # union
ser_i = pd.Series(np.intersect1d(ser1, ser2))  # intersect
ser_u[~ser_u.isin(ser_i)]
```

---

## 45. How Do You Keep Only the Top Two Most Frequent Values as It Is and Replace Everything Else as 'other' in a Series?

```python
# Input
import pandas as pd
np.random.RandomState(100)
ser = pd.Series(np.random.randint(1, 5, [12]))

# Solution
print("Top 2 Freq:", ser.value_counts())
ser[~ser.isin(ser.value_counts().index[:2])] = 'Other'
print(ser)
```

---

## 46. How Do You Find the Positions of Numbers That Are Multiples of Three from a Series?

```python
# Input
import pandas as pd
ser = pd.Series(np.random.randint(1, 10, 7))

# Solution
print(ser)
np.argwhere(ser % 3 == 0)
```

---

## 47. How Do You Compute the Euclidean Distance Between Two Series?

The code is as shown:

```python
# Input
p = pd.Series([1, 2, 3, 4, 5, 6, 7, 8, 9, 10])
q = pd.Series([10, 9, 8, 7, 6, 5, 4, 3, 2, 1])

# Solution
sum((p - q)**2)**0.5

# Solution using function
np.linalg.norm(p - q)
```

You can see that the Euclidean distance can be calculated using two ways.

---

## 48. How Do You Reverse the Rows of a DataFrame?

```python
# Input
df = pd.DataFrame(np.arange(25).reshape(5, -1))

# Solution
df.iloc[::-1, :]
```

---

## 49. If You Split Your Data into Train/Test Splits, Is It Possible to Overfit Your Model?

Yes. One common beginner mistake is re-tuning a model or training new models with different parameters after seeing its performance on the test set.

---

## 50. Which Python Library Is Built on Top of Matplotlib and Pandas to Ease Data Plotting?

Seaborn is a Python library built on top of matplotlib and pandas to ease data plotting. It is a data visualization library in Python that provides a high-level interface for drawing statistical informative graphs.

---

## 51. What are the Important Features of Python?

- Python is a scripting language. Python, unlike other programming languages like C and its derivatives, does not require compilation before execution.
- Python is dynamically typed, which means you don't have to specify the kinds of variables when declaring them or anything.
- Python is well suited to object-oriented programming since it supports class definition, composition, and inheritance.

---

## 52. What Type of Language is Python?

Although Python can be used to write scripts, it is primarily used as a general-purpose programming language.

---

## 53. Explain How Python is an Interpreted Language.

Any programming language that is not in machine-level code before runtime is called an interpreted language.

Python is thus an interpreted language.

---

## 54. What is PEP 8?

PEP denotes Python Enhancement Proposal. It's a collection of guidelines for formatting Python code for maximum readability.

---

## 55. Explain the Python Namespace.

In Python, a namespace refers to the name that is assigned to each object.

---

## 56. What are Decorators in Python?

Decorators are used for changing the appearance of a function without changing its structure. Decorators are typically defined before the function they are enhancing.

---

## 57. How to Use Decorators in Python?

Decorators are typically defined before the function they are enhancing. To use a decorator, we must first specify its function. Then we write the function to which it is applied, simply placing the decorator function above the function to which it must be applied.

---

## 58. Differentiate Between .pyc and .py.

The `.py` files are the source code files for Python. The bytecode of the Python files is stored in `.pyc` files, which are created when code is imported from another source. The interpreters save time by converting the source `.py` files to `.pyc` files.

---

## 59. What is Slicing in Python?

Slicing is a technique for gaining access to specific bits of sequences such as strings, tuples, and lists.

---

## 60. How to Use the Slicing Operator in Python?

Slicing is a technique for gaining access to specific bits of sequences such as lists, tuples, and strings. The slicing syntax is `[start:end:step]`. This step can also be skipped. `[start:end]` returns all sequence items from the

start (inclusive) to the end-1 element. It means the ith element from the end of the start or end element is negative i. The step represents the jump or the number of components that must be skipped.

---

## 61. What are Keywords in Python?

In Python, keywords are reserved words with a specific meaning. They are commonly used to specify the type of variables. Variable and function names cannot contain keywords. Following are the 33 keywords of Python:

- Yield, For, Else, Elif, If
- Not, Or, And, Raise, Nonlocal
- None, Is, In, Import, Global
- From, Finally, Except, Del, Continue
- Class, Assert, With, Try, False
- True, Return, Pass, Lambda, Def
- As, Break, While

---

## 62. How to Combine Dataframes in Pandas?

This is one of the most commonly asked Python interview questions.

The following are the ways through which the data frames in Pandas can be combined:

- Concatenating them by vertically stacking the two dataframes
- Concatenating them by horizontally stacking the two dataframes
- Putting them together in a single column

---

## 63. What are the Key Features of the Python 3.9.0.0 Version?

- Zoneinfo and graphlib are two new modules
- Improved modules such as asyncio and ast
- Optimizations include improved idiom for assignment, signal handling, and Python built-ins
- Removal of erroneous methods and functions
- Instead of LL1, a new parser is based on PEG
- Remove Prefixes and Suffixes with New String Methods
- Generics with type hinting in standard collections

---

## 64. In Python, How is Memory Managed?

- Python's private heap space is in charge of memory management. A private heap holds all Python objects and data structures. This secret heap is not accessible to the programmer. Instead, the Python interpreter takes care of it.
- Python also includes a built-in garbage collector, which recycles all unused memory and makes it available to the heap space.
- Python's memory management is in charge of allocating heap space for Python objects. The core API allows programmers access to some programming tools.

---

## 65. Explain PYTHONPATH.

It's an environment variable that is used when you import a module. When a module is imported, PYTHONPATH is checked to see if the imported modules are present in various folders. It is used by the interpreter to determine which module to load.

---

## 66. Explain Global Variables and Local Variables in Python.

**Local Variables:** A local variable is any variable declared within a function. This variable exists only in local space, not in global space.

**Global Variables:** Global variables are variables declared outside of a function or in a global space. Any function in the program can access these variables.

---

## 67. Is Python Case-sensitive?

Yes, Python is case-sensitive.

---

## 68. How to Install Python on Windows and Set Path Variables?

- Download Python from https://www.python.org/downloads/
- Install it on your computer. Using your command prompt, look for the location where PYTHON is installed on your computer by typing `cmd python`.
- Then, in advanced system settings, create a new variable called PYTHON_NAME and paste the copied path into it.
- Search the path variable, choose its value, and select 'edit'.
- If the value doesn't have a semicolon at the end, add one, and then type `%PYTHONHOME%`.

---

## 69. Is it Necessary to Indent in Python?

Indentation is required in Python. It designates a coding block. An indented block contains all of the code for loops, classes, functions, and so on. Typically, four space characters are used. Your code will not execute correctly if it is not indented, and it will also generate errors.

---

## 70. On Unix, How Do You Make a Python Script Executable?

The script file should start with `#!/usr/bin/env python`.

---

## 71. What is the Use of Self in Python?

Self is used to represent the class instance. In Python, you can access the class's attributes and methods with this keyword. It connects the attributes to the arguments. Self appears in a variety of contexts and is frequently mistaken for a term. Self is not a keyword in Python, unlike in C++.

---

## 72. What are the Literals in Python?

For primitive data types, a literal in Python source code indicates a fixed value.

---

## 73. What are the Types of Literals in Python?

For primitive data types, a literal in Python source code indicates a fixed value. Following are the 5 types of literal in Python:

- **String Literal:** A string literal is formed by assigning some text to a variable that is contained in single or double quotes. Assign the multiline text encased in triple quotes to produce multiline literals.
- **Numeric Literal:** They may contain numeric values that are floating-point values, integers, or complex numbers.
- **Character Literal:** It is made by putting a single character in double quotes.
- **Boolean Literal:** True or False
- **Literal Collections:** There are four types of literals such as list collections, tuple literals, set literals, dictionary literals, and set literals.

---

## 74. What are Python Modules? Name a Few Python Built-in Modules That Are Often Used.

Python modules are files that contain Python code. Functions, classes, or variables can be used in this code. A

Python module is a `.py` file that contains code that may be executed. The following are the commonly used built-in modules:

- JSON
- datetime
- random
- math
- sys
- OS

---

## 75. What is init?

`__init__` is a constructor or method in Python. This method is used to allocate memory when a new object is created.

---

## 76. What is the Lambda Function?

A lambda function is a type of anonymous function. This function can take as many parameters as you want, but just one statement.

---

## 77. Why Lambda is Used in Python?

Lambda is typically utilized in instances where an anonymous function is required for a short period. Lambda functions can be applied in two different ways:

- Assigning Lambda functions to a variable
- Wrapping the Lambda function inside another function

---

## 78. How Does Continue, Break, and Pass Work?

| Continue | Break | Pass |
| --- | --- | --- |
| When a specified condition is met, the control is moved to the beginning of the loop, allowing some parts of the loop to be skipped | When a condition is met, the loop is terminated and control is passed to the next statement | When you need a piece of code syntactically but don't want to execute it, use this. This is a null operation |

---

## 79. What are Python Iterators?

Iterators are objects that can be iterated through or traversed.

---

## 80. Differentiate Between range and xrange.

In terms of functionality, xrange and range are essentially the same. They both provide you the option of generating a list of integers to use whatever you want. The sole difference between range and xrange is that range produces a Python list object whereas xrange returns an xrange object. This is especially true if you are working with a machine that requires a lot of memory, such as a phone because range will utilize as much memory as it can to generate your array of numbers, which can cause a memory error and crash your program. It is a beast with a memory problem.

**Note:** xrange is only available in Python 2.x. In Python 3.x, range() works like xrange() from Python 2.x.

---

## 81. What are Unpickling and Pickling?

The Pickle module takes any Python object and converts it to a string representation, which it then dumps into a file using the dump method. This is known as pickling. Unpickling is the process of recovering original Python objects from a stored text representation.

---

## 82. What are Generators in Python?

Functions that return an iterable set of items are known as generators.

---

## 83. How Do You Copy an Object in Python?

The assignment statement (= operator) in Python does not copy objects. Instead, it establishes a connection between the existing object and the name of the target variable. The copy module is used to make copies of an object in Python. Furthermore, the copy module provides two options for producing copies of a given object:

**Deep Copy:** Deep Copy recursively replicates all values from the source to the destination object, including the objects referenced by the source object.

**Shallow Copy:** A bit-wise copy of an object is called a shallow copy. The values in the copied object are identical to those in the original object. If one of the values is a reference to another object, only its reference addresses are copied.

```
python
```

```
from copy import copy, deepcopy

list_1 = [1, 2, [3, 5], 4]

## Shallow copy
list_2 = copy(list_1)
list_2[3] = 7
list_2[2].append(6)
print(list_2)  # Output => [1, 2, [3, 5, 6], 7]
print(list_1)  # Output => [1, 2, [3, 5, 6], 4]

## Deep copy
list_3 = deepcopy(list_1)
list_3[3] = 8
list_3[2].append(7)
print(list_3)  # Output => [1, 2, [3, 5, 6, 7], 8]
print(list_1)  # Output => [1, 2, [3, 5, 6], 4]
```

---

## 84. In Python, are Arguments Provided by Value or Reference?

**Pass by value:** The actual item's copy is passed. Changing the value of the object's copy does not affect the original object's value.

**Pass by reference:** The actual object is passed as a reference. The value of the old object will change if the value of the new object is changed.

Arguments are passed by reference in Python.

```
python

def appendNumber(arr):
    arr.append(4)

arr = [1, 2, 3]
print(arr)  # Output: => [1, 2, 3]
appendNumber(arr)
print(arr)  # Output: => [1, 2, 3, 4]
```

---

## 85. How to Delete a File in Python?

Use the command `os.remove(file_name)` to delete a file in Python.

---

## 86. Explain join() and split() Functions in Python.

The `join()` function can be used to combine a list of strings based on a delimiter into a single string.

The `split()` function can be used to split a string into a list of strings based on a delimiter.

```python
string = "This is a string."
string_list = string.split(' ')  # Delimiter is a 'space' character or ' '
print(string_list)  # Output: ['This', 'is', 'a', 'string.']
print(' '.join(string_list))  # Output: This is a string.
```

---

## 87. Explain **kwargs and *args.

***args:**

- The function definition uses the `*args` syntax to pass variable-length parameters.
- "*" denotes variable length, while "args" is the standard name. Any other will suffice.

****kwargs:**

- `**kwargs` is a special syntax for passing variable-length keyworded arguments to functions.
- When a variable is passed to a function, it is called a keyworded argument.
- "kwargs" is also used by convention here. You are free to use any other name.

---

## 88. What are Negative Indexes and Why are They Used?

- The indexes from the end of the list, tuple, or string are called negative indexes.
- `Arr[-1]` denotes the array's last element.

---

## 89. How Will You Capitalize the First Letter of a String?

The `capitalize()` function in Python capitalizes a string's initial letter. It returns the original text if the string already contains a capital letter at the beginning.

---

## 90. What Method Will You Use to Convert a String to All Lowercase?

The `lower()` function can be used to convert a string to lowercase.

---

## 91. In Python, How Do You Remark Numerous Lines?

Comments that involve multiple lines are known as multi-line comments. A `#` must prefix all lines that will be commented. You can also use a convenient shortcut to remark several lines. All you have to do is hold down the ctrl key and left-click anywhere you want a `#` character to appear, then input a `#` once. This will add a comment to every line where you put your cursor.

---

## 92. What are Docstrings?

Docstrings are documentation strings. Within triple quotations are these docstrings. They are not allocated to any variable and, as a result, they can also be used as comments.

---

## 93. What is the Purpose of 'not', 'is', and 'in' Operators?

Special functions are known as operators. They take one or more input values and output a result.

- **not** - returns the boolean value's inverse
- **is** - returns true when both operands are true
- **in** - determines whether a certain element is present in a series

---

## 94. What are the Functions help() and dir() Used for in Python?

Both `help()` and `dir()` are available from the Python interpreter and are used to provide a condensed list of built-in functions.

- **dir() function:** The `dir()` function displays the defined symbols.
- **help() function:** The `help()` function displays the documentation string and also allows you to access help for modules, keywords, attributes, and other items.

---

## 95. Why Isn't All the Memory De-allocated When Python Exits?

- When Python quits, some Python modules, especially those with circular references to other objects or objects referenced from global namespaces, are not necessarily freed or deallocated.
- Python would try to de-allocate/destroy all other objects on exit because it has its own efficient cleanup mechanism.
- It is difficult to de-allocate memory that has been reserved by the C library.

---

## 96. What is a Dictionary in Python?

A dictionary is one of Python's built-in datatypes. It establishes a one-to-one correspondence between keys and values. Dictionary keys and values are stored in pairs in dictionaries. Keys are used to index dictionaries.

---

## 97. In Python, How Do You Utilize Ternary Operators?

The Ternary operator is the operator for displaying conditional statements. This is made of true or false values and a statement that must be evaluated.

---

## 98. Explain the split(), sub(), and subn() Methods of the Python "re" Module.

Python's "re" module provides three ways for modifying strings. They are:

- **split():** A regex pattern is used to "separate" a string into a list
- **sub():** Identifies all substrings that match the regex pattern and replaces them with a new string
- **subn():** It works similarly to sub(), returning the new string as well as the number of replacements

---

## 99. What are Negative Indexes and Why Do We Utilize Them?

Python sequences are indexed, and they include both positive and negative values. Positive numbers are indexed with '0' as the first index and '1' as the second index, and so on.

The index for a negative number begins with '-1,' which is the last index in the sequence and ends with '-2,' which is the penultimate index, and the sequence continues like a positive number. The negative index is used to eliminate all new-line spaces from the string and allow it to accept the last character `S[:-1]`. The negative index can also be used to represent the correct order of the string.

---

## 100. Explain Python Packages.

Packages in Python are namespaces that contain numerous modules.

---

## 101. What are Built-in Types of Python?

Given below are the built-in types of Python:

- Built-in functions
- Boolean
- String
- Complex numbers

- Floating point
- Integers

---

## 102. What are the Benefits of NumPy Arrays over (Nested) Python Lists?

- Lists in Python are useful general-purpose containers. They allow for (relatively) quick insertion, deletion, appending, and concatenation, and Python's list comprehensions make them simple to create and operate.
- They have some limitations: they don't enable "vectorized" operations like elementwise addition and multiplication, and because they can include objects of different types, Python must maintain type information for each element and execute type dispatching code while working on it.
- NumPy arrays are faster, and NumPy comes with several features, including histograms, algebra, linear, basic statistics, fast searching, convolutions, FFTs, and more.

---

## 103. What is the Best Way to Add Values to a Python Array?

The `append()`, `extend()`, and `insert(i, x)` procedures can be used to add elements to an array.

---

## 104. What is the Best Way to Remove Values from a Python Array?

The `pop()` and `remove()` methods can be used to remove elements from an array. The difference between these two functions is that one returns the removed value while the other does not.

---

## 105. Is There an Object-oriented Programming (OOPs) Concept in Python?

Python is a computer language that focuses on objects. This indicates that by simply constructing an object model, every program can be solved in Python. Python, on the other hand, may be used as both a procedural and structured language.

---

## 106. Differentiate Between Deep and Shallow Copy.

When a new instance type is formed, a shallow copy is used to maintain the values that were copied in the previous instance. Shallow copy is used to copy reference pointers in the same way as values are copied. These references refer to the original objects, and any modifications made to any member of the class will have an impact on the original copy. Shallow copy enables faster program execution and is dependent on the size of the data being utilized.

Deep copy is a technique for storing previously copied values. The reference pointers to the objects are not copied during deep copy. It creates a reference to an object and stores the new object that is referenced to

another object. The changes made to the original copy will not affect any subsequent copies that utilize the item. Deep copy slows down program performance by creating many copies of each object that is called.

---

## 107. What are Python Libraries?

A Python library is a group of Python packages. Numpy, Pandas, Matplotlib, Scikit-learn, and many other Python libraries are widely used.

---

## 108. Why is split() Used?

In Python, the `split()` function is used to split a string.

---

## 109. How is Multithreading Achieved in Python?

- Although Python includes a multi-threading module, it is usually not a good idea to utilize it if you want to multi-thread to speed up your code.
- As this happens so quickly, it may appear to the human eye that your threads are running in parallel, but they are actually sharing the same CPU core.
- The Global Interpreter Lock is a Python concept (GIL). Only one of your 'threads' can execute at a moment, thanks to the GIL. A thread obtains the GIL, performs some work, and then passes the GIL to the following thread.

---

## 110. How are Classes Created in Python?

The class keyword in Python is used to construct a class.

---

## 111. What is a Pandas DataFrame?

A dataframe is a 2D changeable and tabular structure for representing data with rows and columns labeled.

---

## 112. Explain Monkey Patching in Python.

Monkey patches are solely used in Python to run-time dynamic updates to a class or module.

---

## 113. How is a Python Module Imported?

The import keyword can be used to import modules.

## 114. What is Inheritance in Python?

Inheritance allows one class to gain all of another class's members (for example, attributes and methods). Inheritance allows for code reuse, making it easier to develop and maintain applications.

---

## 115. What are the Different Types of Inheritance in Python?

The following are the various types of inheritance in Python:

- **Single inheritance:** The members of a single superclass are acquired by a derived class.
- **Multiple inheritance:** More than one base class is inherited by a derived class.
- **Multi-level inheritance:** D1 is a derived class inherited from base1 while D2 is inherited from base2.
- **Hierarchical Inheritance:** You can inherit any number of child classes from a single base class.

---

## 116. Is Multiple Inheritance Possible in Python?

A class can be inherited from multiple parent classes, which is known as multiple inheritance. In contrast to Java, Python allows multiple inheritance.

---

## 117. Explain Polymorphism in Python.

The ability to take various forms is known as polymorphism. For example, if the parent class has a method named ABC, the child class can likewise have a method named ABC with its own parameters and variables. Python makes polymorphism possible.

---

## 118. What is Encapsulation in Python?

Encapsulation refers to the joining of code and data. Encapsulation is demonstrated through a Python class.

---

## 119. In Python, How Do You Abstract Data?

Only the necessary details are provided, while the implementation is hidden from view. Interfaces and abstract classes can be used to do this in Python.

---

## 120. Are Access Specifiers Used in Python?

Access to an instance variable or function is not limited in Python. To imitate the behavior of protected and

private access specifiers, Python introduces the idea of prefixing the name of the variable, function, or method with a single or double underscore.

---

## 121. How to Create an Empty Class in Python?

A class that has no code defined within its block is called an empty class. The pass keyword can be used to generate it. You can, however, create objects of this class outside of the class. When used in Python, the PASS command has no effect.

---

## 122. What Does an object() Do?

It produces a featureless object that serves as the foundation for all classes. It also does not accept any parameters.

---

## 123. Write a Python Program to Generate a Star Triangle.

```python
def pyfunc(r):
    for x in range(r):
        print(' ' * (r - x - 1) + '*' * (2 * x + 1))

pyfunc(9)
```

**Output:**

```
        *
       ***
      *****
     *******
    *********
   ***********
  *************
 ***************
*****************
```

---

## 124. Write a Program to Produce the Fibonacci Series in Python.

```python
```

```python
# Enter the number of terms needed
# 0, 1, 1, 2, 3, 5....
a = int(input("Enter the terms: "))
f = 0  # First element of series
s = 1  # Second element of series

if a <= 0:
   print("The requested series is", f)
else:
   print(f, s, end=" ")
   for x in range(2, a):
      next = f + s
      print(next, end=" ")
      f = s
      s = next
```

**Output:**

```
Enter the terms: 5
0 1 1 2 3
```

---

## 125. Make a Python Program That Checks if a Sequence is a Palindrome.

```python
a = input("Enter sequence: ")
b = a[::-1]

if a == b:
   print("Palindrome")
else:
   print("Not a Palindrome")
```

**Output:**

```
Enter sequence: 323
Palindrome
```

---

## 126. Make a One-liner That Counts How Many Capital Letters are in a File.

Even if the file is too large to fit in memory, your code should work.

**Traditional approach:**

```python
with open(SOME_LARGE_FILE) as fh:
    count = 0
    text = fh.read()
    for character in text:
        if character.isupper():
            count += 1
```

**One-liner:**

```python
count = sum(1 for line in fh for character in line if character.isupper())
```

---

## 127. Can You Write a Sorting Algorithm with a Numerical Dataset?

```python
list = ["1", "4", "0", "6", "9"]
list = [int(i) for i in list]
list.sort()
print(list)
```

---

## 128. Check the Code Given Below, and List the Final Value of A0, A1 …An.

```python
A0 = dict(zip(('a', 'b', 'c', 'd', 'e'), (1, 2, 3, 4, 5)))
A1 = range(10)
A2 = sorted([i for i in A1 if i in A0])
A3 = sorted([A0[s] for s in A0])
A4 = [i for i in A1 if i in A3]
A5 = {i: i*i for i in A1}
A6 = [[i, i*i] for i in A1]
print(A0, A1, A2, A3, A4, A5, A6)
```

**Here's the answer:**

- A0 = {'a': 1, 'c': 3, 'b': 2, 'e': 5, 'd': 4} # The order may vary
- A1 = range(0, 10)

- A2 = []
- A3 = [1, 2, 3, 4, 5]
- A4 = [1, 2, 3, 4, 5]
- A5 = {0: 0, 1: 1, 2: 4, 3: 9, 4: 16, 5: 25, 6: 36, 7: 49, 8: 64, 9: 81}
- A6 = [[0, 0], [1, 1], [2, 4], [3, 9], [4, 16], [5, 25], [6, 36], [7, 49], [8, 64], [9, 81]]

---

## 129. What is Flask and Explain Its Benefits.

Flask is a Python web microframework based on the BSD license. Two of its dependencies are Werkzeug and Jinja2. This means it will have few if any, external library dependencies. It lightens the framework while reducing update dependencies and security vulnerabilities.

A session is just a way of remembering information from one request to the next. A session in a flask employs a signed cookie to allow the user to inspect and edit the contents of the session. If the user only has the secret key, he or she can change the session using `Flask.secret_key`.

---

## 130. Is Django Better as Compared to Flask?

Django and Flask map URLs or addresses entered into web browsers into Python functions. Flask is easier to use than Django, but it doesn't do much for you, so you will have to specify the specifics, whereas Django does a lot for you and you won't have to do anything. Django has prewritten code that the user must examine, whereas Flask allows users to write their code, making it easier to grasp. Both are technically excellent and have their own set of advantages and disadvantages.

---

## 131. Differentiate Between Pyramid, Django, and Flask.

- **Pyramid** is designed for larger apps. It gives developers flexibility and allows them to utilize the appropriate tools for their projects. The database, URL structure, templating style, and other options are all available to the developer. Pyramids can be easily customized.
- **Flask** is a "microframework" designed for small applications with straightforward needs. External libraries are required in a flask. The flask is now ready for use.
- **Django**, like Pyramid, may be used for larger applications. It has an ORM in it.

---

## 132. In NumPy, How Will You Read CSV Data into an Array?

This may be accomplished by utilizing the `genfromtxt()` method with a comma as the delimiter.

---

## 133. What is GIL?

The term GIL stands for Global Interpreter Lock. This is a mutex that helps thread synchronization by preventing deadlocks by limiting access to Python objects. GIL assists with multitasking (and not parallel computing).

---

## 134. What is PIP?

PIP denotes Python Installer Package. It is used to install various Python modules. It's a command-line utility that creates a unified interface for installing various Python modules. It searches the internet for the package and installs it into the working directory without requiring any user intervention.

---

## 135. What is the Use of Sessions in the Django Framework?

Django has a session feature that allows you to store and retrieve data for each site visitor. Django isolates the process of sending and receiving cookies by keeping all necessary data on the server side and inserting a session ID cookie on the client side.

---

## 136. Write a Program That Checks If All of the Numbers in a Sequence are Unique.

python

```python
def check_distinct(data_list):
    if len(data_list) == len(set(data_list)):
        return True
    else:
        return False


print(check_distinct([1, 6, 5, 8]))  # Prints True
print(check_distinct([2, 2, 5, 5, 7, 8]))  # Prints False
```

---

## 137. What is an Operator in Python?

An operator is a symbol that is applied to a set of values to produce a result. An operator manipulates operands. Numeric literals or variables that hold values are known as operands. Unary, binary, and ternary operators are all possible. The unary operator requires only one operand, the binary operator requires two operands, and the ternary operator requires three operands.

---

## 138. What are the Various Types of Operators in Python?

- Bitwise operators
- Identity operators
- Membership operators
- Logical operators
- Assignment operators
- Relational operators
- Arithmetic operators

---

## 139. How to Write a Unicode String in Python?

The old Unicode type has been replaced with the "str" type in Python 3, and the string is now considered Unicode by default. Using the `art.title.encode("utf-8")` function, we can create a Unicode string.

---

## 140. Explain the Differences Between Python 2.x and Python 3.x?

Python 2.x is an older version of the Python programming language. Python 3.x is the most recent version. Python 2.x is no longer supported. Python 3.x is the language's present and future.

In Python 2, a string is inherently ASCII, while in Python 3, it is Unicode.

---

## 141. How to Send an Email in Python Language?

Python includes the smtplib and email libraries for sending emails. Import these modules into the newly created mail script and send mail to users who have been authenticated.

---

## 142. Create a Program to Add Two Integers >0 Without Using the Plus Operator.

python

```python
def add_nums(num1, num2):
    while num2 != 0:
        data = num1 & num2
        num1 = num1 ^ num2
        num2 = data << 1
    return num1

print(add_nums(2, 10))
```

---

## 143. Create a Program to Convert Dates from yyyy-mm-dd to dd-mm-yyyy.

We can use this module to convert dates:

```python
import re

def transform_date_format(date):
    return re.sub(r'(\d{4})-(\d{1,2})-(\d{1,2})', '\\3-\\2-\\1', date)

date_input = "2021-08-01"
print(transform_date_format(date_input))
```

The datetime module can also be used, as demonstrated below:

```python
from datetime import datetime

new_date = datetime.strptime("2021-08-01", "%Y-%m-%d").strftime("%d:%m:%Y")
print(new_date)
```

---

## 144. Create a Program That Combines Two Dictionaries.

If you locate the same keys during combining, you can sum the values of these similar keys. Create a new dictionary.

```python
from collections import Counter

d1 = {'key1': 50, 'key2': 100, 'key3': 200}
d2 = {'key1': 200, 'key2': 100, 'key4': 300}
new_dict = Counter(d1) + Counter(d2)
print(new_dict)
```

---

## 145. Is There an Inherent Do-While Loop in Python?

No.

---

## 146. What Kind of Joins are Offered by Pandas?

There are four joins in Pandas: left, inner, right, and outer.

---

## 147. How are Dataframes in Pandas Merged?

The type and fields of the dataframes being merged determine how they are merged. If the data has identical fields, it is combined along axis 0, otherwise, it is merged along axis 1.

---

## 148. What is the Best Way to Get the First Five Entries of a Data Frame?

We may get the top five entries of a dataframe using the `head(5)` method. `df.head()` returns the top 5 rows by default. `df.head(n)` will be used to fetch the top n rows.

---

## 149. How Can You Access the Dataframe's Latest Five Entries?

We may get the last five entries of a dataframe using the `tail(5)` method. `df.tail()` returns the last 5 rows by default. `df.tail(n)` will be used to fetch the last n rows.

---

## 150. Explain Classifier.

Any data point's class is predicted using a classifier. Classifiers are hypotheses that are used to assign labels to data items based on their classification.

---

## End of Document

**Document prepared for Data Engineering Interview Preparation**

*All Python interview questions corrected and formatted for clarity*