



# Python Interview Questions

## for Data Analytics

Mastering Python is essential for aspiring data analysts. By preparing for common Python interview questions, data analysts can showcase their proficiency in the language and demonstrate their ability to solve real-world data problems.

### Python Core Questions

#### **Question 1: What Is Python?**

It is very open, and there is no perfect answer. This question is principally used to demonstrate your ability to summarize high-level concepts in a few seconds. Here's a possible short answer:

*Python is a high-level and interpreted programming language that supports multi-paradigms and is used for generic purposes.*

#### **Question 2: Which Python Built-in Functions Do You Know?**

Python's built-in functions are always available for use. These functions are available in the official Python documentation.

Here are some built-in functions:

- `int()`: Converts a value to an integer.

- `range()`: Generates a sequence of numbers within a specified range.
- `min()`: Finds the smallest element in an iterable object (such as a list).
- `max()`: Finds the greatest element in an iterable object.
- `sum()`: Gets the sum of elements in an iterable object.
- `abs()`: Returns the absolute value of a number.
- `round()`: Returns a rounded number.
- `pow()`: Returns a number to a given power.
- `hex()`: Converts an integer number to a lowercase hexadecimal string.
- `len()`: Returns the length of an object.

### Question 3: How Do You Import External Libraries into Python?

Importing an external library is something you'll do multiple times a day! Let's see an example using the pandas library:

```
import pandas as pd
dataset = { 'users': ["John", "Frida", "Katlyn"],
            'age': [31, 27, 42] }
dataframe = pd.DataFrame(dataset)
```

This will import the entire pandas library, allowing you to use all its functions and classes, here, you use the DataFrame function. The alias `pd` to make it easier to reference this dataframe in the code.

### Question 4: How Are Errors and Exceptions Managed in Python?

In Python, errors and exceptions are easily handled using the try-except block. This allows you to catch and handle exceptions that occur during the execution of your code, preventing them from causing your program to terminate unexpectedly.

Let's see an example of handling a classic error: division by zero.

```
try: # Code that may raise an exception

x = 10 / 0 # This will raise a ZeroDivisionError

except ZeroDivisionError:

# Handle the specific exception

print("Error: Division by zero!")

except:

# Handle any other exception

print("An unexpected error occurred!")

finally:

# Cleanup code

print("Exiting try-except block")
```

In the example above, the try block encloses code that may raise an exception. The except block catches and handles exceptions raised within the try block. The type of exception we want to catch can be specified; here, it's ZeroDivisionError.

A generic except block can be used to catch any exception. The finally block is optional and is used to execute cleanup code (such as closing files or releasing resources) regardless of whether an exception occurred. The code in the finally block is always executed, even if an exception is raised and not caught.

# **Data Analysis Questions**

## **Question 5: Calculate a Mean in Python**

The mean is undoubtedly one of the most-used functions in data analysis. Also known as the average, the mean is calculated by summing up all the values in a dataset and dividing the sum by the total number of values.

If you aspire to a data analyst job, you should know how to calculate the mean in Python. Python's built-in functions allow us to do it in a very simple way. We can see this in the following example:

```
data = [2, 3, 4, 5, 6]
```

```
mean = sum(data) / len(data)
```

Thanks to the built-in functions `sum()` and `len()`, we can calculate the sum of the values of the dataset and the total number of values. Another great way to do it is by using the function `mean()` from the module `statistics`:

```
import statistics
```

```
data = [2, 3, 4, 5, 6]
```

```
mean = statistics.mean(data)
```

## **Question 6: Describe a Typical Data Analysis Process**

The data analysis process involves several steps aimed at understanding, interpreting, and deriving insights from data. The general data analysis process typically includes the following main steps:

- **Data Collection:** Gather relevant data from various sources. In Python, you can use the requests library to retrieve data from web servers by sending HTTP GET and POST. You can also use web scraping using libraries like BeautifulSoup or Selenium.
- **Data Cleaning:** Remove errors, duplicates, and inconsistencies from the dataset. If you need to perform complex operations as you clean your data, you can use a Python data-cleaning library like NumPy or pandas.
- **Exploratory Data Analysis (EDA):** Explore the dataset to understand its structure, patterns, and relationships. Python libraries such as pandas, NumPy, and Scikit-learn are excellent tools for performing EDA on a dataset.
- **Modeling:** Apply statistical techniques or machine learning algorithms to analyze the data. Python libraries like Scikit-learn, TensorFlow, and PyTorch provide a set of tools and algorithms for data modeling and machine learning tasks.
- **Visualization:** Present findings visually through charts, graphs, or dashboards. You can use Python data visualization libraries like Matplotlib or Seaborn to create plots and charts.

### Question 7: What File Types Are Commonly Used in Python Data Analysis?

In data analysis, various file types are used to store and manipulate data. Your answer to this question will indicate to the interviewer how much experience you have in manipulating data. Some common file types include:

- **CSV (Comma-Separated Values):** CSV is a plain text format where each line represents a row of data and columns are separated by commas (or another character, such as a semi-colon). CSV files are widely used for storing tabular data and are compatible with many software tools.
- **Excel spreadsheets (.xlsx, .xls):** Microsoft Excel files are commonly used for storing tabular data, performing calculations, and creating visualizations.
- **JSON (JavaScript Object Notation):** JSON is a lightweight data-interchange format that's easy for humans to read and write and easy for machines to parse and generate. JSON files store data in key-value pairs and are commonly used for web APIs and configuration files.

- **XML (eXtensible Markup Language):** XML is a markup language that defines a set of rules for encoding documents in a human-readable and machine-readable format. An XML file is a text file that stores data in this format.
- **SQL (Structured Query Language) databases:** SQL databases store structured data in tables with rows and columns.

## String Questions

### Question 8: How Do You Print a Variable in a String in Python?

There are many ways to print a variable in a Python string. The more experience with Python you have, the more ways you will share with the interviewer.

There are three ways to print a variable: concatenation, string formatting, and f-strings.

```
language = "Python"  
print("My favorite language is " + language)
```

The syntax in the example above is called string concatenation: two strings are merged into a single string using the + operator.

```
print("My favorite language is {}".format(language))
```

In this example, string formatting is used. The format() method is called on a string that contains replacement fields delimited by braces {}. The replacement fields will be replaced by the parameters passed through the format() function – here, language.

```
print(f"My favorite language is {language}")
```

The example above uses an f-string, which is a formatted string prefixed with f. The parameters we want to include in the string come between the braces {}.

## Question 9: How Do You Slice a String?

Slicing a string in Python refers to extracting a substring from a given string by specifying a range of indices. In data analysis, string slicing is essential for cleaning and preprocessing textual data. It helps handle data inconsistencies, remove unwanted characters, and standardize data formatting.

String slicing uses three optional parameters: start, end, and step. The syntax is as follows:

```
String[start:end:step]
```

Here is what each parameter does:

- **start (optional):** The index from which the slicing begins. It indicates the position of the first character to be included in the slice. If omitted, slicing starts from the beginning of the string (index 0).
- **end (optional):** The index up to which slicing occurs; however, it excludes the character at this position. If omitted, slicing extends to the end of the string.
- **step (optional):** The step or increment value for selecting characters within the specified range. If omitted, the default value is 1.

Below are some examples of string slicing:

```
s = "hello world"
```

```
print(s[:5]) # Output: "hello" (extract the first 5 characters)
```

```
print(s[6:]) # Output: "world" (extract from index 6 to the end)
```

```
print(s[::2]) # Output: "hlowrd" (extract every other character)
```

```
print(s[::-1]) # Output: "dlrow olleh" (reverse the string)
```

## Pandas Question

### Question 10: How Do You Read a CSV File with pandas?

This question seems very basic, but it demonstrates the candidate's knowledge of the pandas library.

```
import pandas as pd

df = pd.read_csv('dataset.csv')
```

With pandas, a CSV file can be easily loaded using the function `read_csv()`. The path of the CSV file is indicated as a parameter.

## Data Structure and Algorithm Questions

### Question 11: Describe Python's Main Data Structures

Manipulating data in Python implies that you can choose the right data structure to correctly store and perform operations on your data. If you have Python programming experience, you should know the “fantastic four” data structures: list, tuple, dictionary, and set. Name them and provide a short explanation of each one in your answer.

- **Lists** are widely used during data collection and data cleaning. They offer a combination of flexibility, efficiency, and ease of use that makes them suitable for storing and manipulating data.
- **Tuples** are lightweight data structures. Using tuples instead of lists for storing data that will never be changed is a good practice; in terms of performance, tuples are more efficient than lists.



- **Dictionaries** provide a powerful way to organize and access data using key-value pairs. For example, you can use a dictionary to store the total sales revenue for each product category, with product categories as keys and total revenues as values.
- **Sets** are generally used during data cleaning to ensure data integrity and accuracy; they are particularly useful for efficiently removing duplicate elements from a dataset. Converting a list or some other collection to a set automatically eliminates duplicates, leaving only unique elements behind.

*TIP : Knowing the theory of data structures is excellent, but it's not enough; you must also be able to provide practical examples of how to use each data structure in a real-life scenario.*

### **Question 12: How Do You Efficiently Retrieve a Dictionary Value Using a Key?**

This question will test your ability to write optimized code. Suppose you have a dictionary named `my_dict`, and you want to retrieve the value associated with the specific key `"the_key"`. If the dictionary does not contain the key, it will generate a fatal error during execution. Therefore, you need to be sure that the key exists when you access it.

A better way to answer the question is using the dictionary `get()` method, which allows us to retrieve a value for a specific key in a dictionary. It accepts a default value in case the key does not exist:

```
value = my_dict.get(the_key, 'default value')
```

### **Question 13: What Data Types Can Be a Python Dictionary Key?**

This is a trick question! Most candidates will answer `int` or `string` because those are the types commonly used as a key dictionary, but this is only partially correct. To answer this question correctly, you must understand the concept of *hashable*.

Dictionary keys are hashable, meaning that they have a hash value that does not change over time. This allows them to be efficiently stored and retrieved. Strings, integers, floats, tuples, and booleans (among others) can be used as dictionary keys. Mutable data types like lists, sets, and dictionaries themselves cannot be used as dictionary keys because they are not hashable.

### **Question 14: How Do You Sort Dataset Elements?**

Sorting is always an expensive operation in terms of performance. Writing a proper sort algorithm is not as easy as it may seem. Python provides a ready-made sorting function for you. And it's pretty easy to use:

```
list_name.sort()
```

Additionally, the `sort()` function supports the `reverse` parameter. By adding `reverse=True`, we get the elements sorted in descending order. Note that `reverse` is `False` by default – if you don't provide the argument, you'll always get your elements sorted in ascending order.

```
my_list = [9, 3, 5, 1]

my_list.sort()

print(my_list) #Print 1, 3, 5, 9

my_list.sort(reverse=True)

print(my_list) #Print 9, 5, 3, 1
```

You may also use a custom sorting function with `sort()`

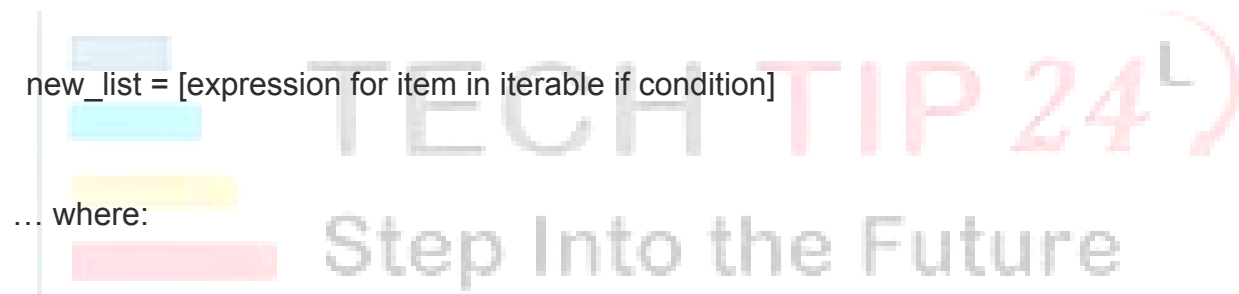
# Bonus Question!

## Question 15: Explain The Concept of List Comprehension in Python

The concept of list comprehension in Python is essential because it promotes clean, expressive, and efficient code. It is a fundamental tool in the Python programmer's toolkit and is widely used in various domains – including data analysis. A great answer to this question would be the following:

List comprehension is a concise and powerful way to create lists in Python. It provides a compact syntax for generating lists and applying operations or conditions to each element.

The general syntax of a list comprehension in Python is ...



```
new_list = [expression for item in iterable if condition]
```

... where:

- `expression` is the operation or transformation to apply to each element.
- `item` is the variable representing each element in the iterable.
- `iterable` is the existing sequence (e.g. list, tuple, range) over which to iterate.
- `condition` is an optional filtering condition. If specified, only elements for which the condition evaluates to True will be included in the new list.

If you are asked for an example, you can provide this:

*The following list comprehension squares each element of a given list, excluding the number 1:*

```
numbers = [1, 2, 3, 4, 5]

squared_numbers = [x**2 for x in numbers if x > 1]

print(squared_numbers) #Prints 4, 9, 16, 25
```

## **After Reviewing Python Interview Questions, Don't Forget to Practice!**

Good luck with your interviews, and may your Python skills help you unlock exciting opportunities in the world of data!

