**Why Is Python Asked in Data Science Interviews?**

Python has ascended as the preeminent language in data science, overshadowing R, Julia, Spark, and Scala. This prominence is largely due to its extensive array of data science libraries and its supportive community.

Python’s versatility extends across the entire data science stack, simplifying tasks from exploratory data analysis and visualization to model building and deployment.

**Basic Python Interview Questions**

**1) What built-in data types are used in Python?**

Python utilizes several built-in data types, including:

* Number (int, float, complex)
* String (str)
* Tuple (tuple)
* Range (range)
* List (list)
* Set (set)
* Dictionary (dict)

**2) How are data analysis libraries used in Python? Name some common ones.**

Python’s popularity in data science is driven by its extensive library collection, which includes:

* Pandas
* NumPy
* SciPy
* TensorFlow
* SciKit
* Seaborn
* Matplotlib

These libraries provide tools for data processing, analysis, visualization, and more.

**3) How is a negative index used in Python?**

Negative indexing in Python allows access to list elements from the end. For instance, n-1 retrieves the last item, while n-2 fetches the second-to-last.

**4) What is the difference between lists and tuples in Python?**

* Syntax: Lists use square brackets [ ], while tuples use parentheses ( ).
* Mutability: Lists are mutable; tuples are immutable.
* Operations: Lists support more operations, such as insert and pop.
* Performance: Tuples, being immutable, are faster and consume less memory.

**5) Which library would you prefer for plotting: Seaborn or Matplotlib?**

Seaborn, built on top of Matplotlib, offers more customization and faster implementation for many common tasks. Matplotlib is better suited for fine-tuning.

**6) Is Python an object-oriented programming language?**

Python integrates features from both object-oriented programming (OOP) and aspect-oriented programming. However, it lacks strong encapsulation, a core OOP feature.

**7) What is the difference between a series and a data frame in Pandas?**

* Series: One-dimensional array with axis labels (index).
* Data Frame: Two-dimensional, tabular data structure with labeled axes (rows and columns).

**8) How would you find duplicate values in a dataset in Python?**

Use Pandas’ duplicated() method to check for duplicates, returning a Boolean series indicating duplicate entries.

**9) What is a lambda function in Python?**

Lambda functions, or anonymous functions, are defined using the lambda keyword and can take multiple parameters but are restricted to a single expression.

**10) Is memory de-allocated when you exit Python?**

Not always. Modules with circular references might not be freed, and some memory reserved by the C library may remain.

**11) What is a compound datatype?**

Compound data structures represent multiple values:

* Lists: An ordered collection of values.
* Tuples: Ordered sequence of values.
* Sets: Unordered collection of unique values.

**12) What is list comprehension in Python? Provide an example.**

List comprehension provides a concise way to create lists. For example:

rletters = [letter for letter in 'retain']  
print(rletters) # Output: ['r', 'e', 't', 'a', 'i', 'n']

**13) What is tuple unpacking and why is it important?**

Tuple unpacking assigns elements of a tuple to multiple variables, useful for swapping variables without a temporary variable:

x, y = 20, 30  
x, y = y, x  
print(f"x: {x}, y: {y}") # Output: x: 30, y: 20

**14) What’s the difference between ‘/’ and ‘//’ in Python?**

* / performs float division (e.g., 9 / 2 returns 4.5).
* // performs floor division, returning the largest integer less than or equal to the division result (e.g., 9 // 2 returns 4).

**15) How do you convert integers to strings in Python?**

The str() function converts integers to strings. Alternatives include f-strings and the .format() method.

**16) What are arrays in Python?**

Arrays store multiple values in a single variable, e.g.,

faang = ["Facebook", "Apple", "Amazon", "Netflix", "Google"]  
print(faang) # Output: ['Facebook', 'Apple', 'Amazon', 'Netflix', 'Google']

**17) What’s the difference between mutable and immutable objects?**

* **Mutable:** Values can change (e.g., lists, sets, dictionaries).
* **Immutable:** Values cannot change (e.g., tuples, strings).

**18) What are some limitations of Python?**

* **Speed:** Slower than languages like Java and C.
* **Mobile Development:** Less effective for mobile apps.
* **Memory Consumption:**High memory usage.
* **Python 2 vs Python 3:**Incompatibilities between versions.

**19) Explain the ‘zip’ and ‘enumerate’ functions.**

* **enumerate():** Returns indexes and items from an iterable.
* **zip():** Combines multiple iterables into tuples.

**20) Define PYTHONPATH.**

PYTHONPATH tells the Python interpreter where to locate module files, akin to the PATH variable in the operating system.

**String Manipulation Python Interview Questions**

String parsing is common in data science interviews, especially for text-heavy companies like Twitter, LinkedIn, Indeed, or Netflix [Netflix Technology Blog](https://medium.com/u/c3aeaf49d8a4?source=post_page---user_mention--ae5bf9a9948c--------------------------------). These questions test your ability to clean and transform text data.

**21) Write a function that returns a list of bigrams from a string.**

def bigrams(sentence):  
 words = sentence.split()  
 return [words[i] + ' ' + words[i+1] for i in range(len(words) - 1)]  
  
print(bigrams("Have free hours and love children"))  
# Output: ['Have free', 'free hours', 'hours and', 'and love', 'love children']

**22) Given two strings, determine if one can be shifted to become the other.**

def can\_shift(A, B):  
 return len(A) == len(B) and B in A + A  
  
print(can\_shift("abcde", "cdeab")) # Output: True  
print(can\_shift("abc", "acb")) # Output: False

**23) Determine if there is a one-to-one character mapping between two strings.**

def is\_one\_to\_one(string1, string2):  
 if len(string1) != len(string2):  
 return False  
 mapping = {}  
 for char1, char2 in zip(string1, string2):  
 if char1 in mapping:  
 if mapping[char1] != char2:  
 return False  
 elif char2 in mapping.values():  
 return False  
 else:  
 mapping[char1] = char2  
 return True  
  
print(is\_one\_to\_one("qwe", "asd")) # Output: True  
print(is\_one\_to\_one("donut", "fatty")) # Output: False

**24) Return the first recurring character in a string.**

def first\_recurring\_char(s):  
 seen = set()  
 for char in s:  
 if char in seen:  
 return char  
 seen.add(char)  
 return None  
  
print(first\_recurring\_char("interviewquery")) # Output: 'i'

**25) Check if one string is a subsequence of another.**

def is\_subsequence(string1, string2):  
 it = iter(string2)  
 return all(char in it for char in string1)  
  
print(is\_subsequence("abc", "ahbgdc")) # Output: True  
print(is\_subsequence("axc", "ahbgdc")) # Output: False

**Python Statistics and Probability Interview Questions**

These questions assess your ability to apply statistical and probability concepts using Python.

**26) Generate N samples from a normal distribution and plot them.**

import numpy as np  
import matplotlib.pyplot as plt  
  
def plot\_normal\_distribution(N):  
 samples = np.random.randn(N)  
 plt.hist(samples, bins=30, alpha=0.5, edgecolor='black')  
 plt.title('Histogram of Normal Distribution')  
 plt.xlabel('Value')  
 plt.ylabel('Frequency')  
 plt.show()  
  
plot\_normal\_distribution(1000)

**27) How do you handle missing data in a dataset?**

**Common methods include:**

* Dropping Missing Values: Using dropna() in Pandas.
* Imputation: Replacing missing values with the mean, median, or mode using fillna().

**28) Calculate the mean, median, and mode of a dataset in Python.**

from scipy import stats  
  
data = [1, 2, 2, 3, 4, 5, 5, 5, 6]  
  
mean = np.mean(data)  
median = np.median(data)  
mode = stats.mode(data)  
  
print(f"Mean: {mean}, Median: {median}, Mode: {mode.mode[0]}")

**29) Perform a t-test to compare the means of two samples.**

from scipy.stats import ttest\_ind  
  
sample1 = np.random.randn(100)  
sample2 = np.random.randn(100)  
  
t\_stat, p\_value = ttest\_ind(sample1, sample2)  
  
print(f"T-statistic: {t\_stat}, P-value: {p\_value}")

**30) How do you calculate the Pearson correlation coefficient in Python?**

from scipy.stats import pearsonr  
  
data1 = np.random.randn(100)  
data2 = np.random.randn(100)  
  
corr, \_ = pearsonr(data1, data2)  
  
print(f"Pearson correlation coefficient: {corr}")

**Python Pandas Interview Questions**

Pandas is a must-know library for any data science interview, encompassing data wrangling and pre-processing skills.

**31) How do you read a CSV file in Pandas?**

import pandas as pd  
df = pd.read\_csv('data.csv')

**32) How do you handle missing values in a Data Frame?**

# Dropping rows with missing values  
df.dropna()  
# Filling missing values with the mean  
df.fillna(df.mean())

**33) How do you group data in a Data Frame?**

grouped = df.groupby('column\_name').agg({'other\_column': 'mean'})

**34) How do you merge two Data Frames in Pandas?**

merged\_df = pd.merge(df1, df2, on='common\_column')

**35) How do you create a pivot table in Pandas?**

pivot\_table = df.pivot\_table(index='column1',   
columns='column2', values='values\_column', aggfunc='mean')

**36) Explain how to use the ‘apply’ function in Pandas.**

df['new\_column'] = df['column'].apply(lambda x: x \* 2)

**37) How do you handle categorical data in Pandas?**

# Using pd.get\_dummies for one-hot encoding  
df = pd.get\_dummies(df, columns=['categorical\_column'])  
  
# Using LabelEncoder  
from sklearn.preprocessing import LabelEncoder  
le = LabelEncoder()  
df['encoded\_column'] = le.fit\_transform(df['categorical\_column'])

**38) How do you concatenate two Data Frames?**

concatenated\_df = pd.concat([df1, df2], axis=0)

**Python Data Manipulation Interview Questions**

These questions test your ability to transform data for analysis.

**39) How do you filter rows in a Data Frame?**

filtered\_df = df[df['column'] > value]

**40) How do you reshape data in a Data Frame?**

reshaped\_df = df.pivot(index='index\_column',   
columns='columns\_column', values='values\_column')

**41) How do you sort a Data Frame?**

sorted\_df = df.sort\_values(by='column')

**42) How do you handle time series data in Pandas?**

# Parsing dates while reading the CSV   
df = pd.read\_csv('data.csv', parse\_dates=['date\_column'])   
# Setting the date column as index   
df.set\_index('date\_column', inplace=True)   
# Resampling time series data   
resampled\_df = df.resample('M').mean() # Monthly resampling

**43) How do you add a new column to a Data Frame?**

df['new\_column'] = df['existing\_column'] \* 2

**Matrices and NumPy Python Interview Questions**

NumPy is essential for numerical computing and manipulating matrices.

**44) Create a 3x3 identity matrix using NumPy.**

import numpy as np  
identity\_matrix = np.eye(3)

**45) How do you perform matrix multiplication in NumPy?**

matrix1 = np.array([[1, 2], [3, 4]])   
matrix2 = np.array([[5, 6], [7, 8]])   
result = np.dot(matrix1, matrix2)

**46) How do you calculate the inverse of a matrix in NumPy?**

matrix = np.array([[1, 2], [3, 4]])   
inverse\_matrix = np.linalg.inv(matrix)

**47) How do you find the eigenvalues and eigenvectors of a matrix?**

matrix = np.array([[1, 2], [2, 3]])   
eigenvalues, eigenvectors = np.linalg.eig(matrix)

**48) How do you generate random numbers in NumPy?**

random\_numbers = np.random.rand(3, 3)   
# 3x3 matrix of random numbers

**Python Machine Learning Interview Questions**

These questions cover applying machine learning principles using Python.

**49) Implement the K-means algorithm from scratch.**

import numpy as np   
  
def kmeans(data, k, max\_iters=100):   
# Initialize centroids randomly from the data points   
centroids = data[np.random.choice(data.shape[0], k, replace=False)]   
  
for \_ in range(max\_iters):   
# Assign each data point to the nearest centroid   
distances = np.linalg.norm(data[:, np.newaxis] - centroids, axis=2)   
labels = np.argmin(distances, axis=1)   
  
# Recompute the centroids   
new\_centroids = np.array([data[labels == i].mean(axis=0) for i in range(k)])   
  
# Check for convergence   
if np.all(centroids == new\_centroids):   
break   
centroids = new\_centroids   
  
return labels, centroids   
  
# Example usage   
data = np.random.rand(100, 2)   
labels, centroids = kmeans(data, 3)