Python Interview Q&A and Coding Questions

Q1: What is Python? List some features.
A1: Python is a high-level, interpreted programming language known for its simplicity and readability. Features
include dynamic typing, garbage collection, wide library support, and cross-platform compatibility.

Q2: How do you handle missing values in a dataset using Python?
A2: Handle missing values using df.dropna() to remove them, df.fillna(value) to replace them, or use interpolation with df.interpolate().

Q3.What is the difference between a list and a tuple in Python?
A3.
1. Lists are mutable (can be changed), tuples are immutable.
2. Lists are defined using square brackets, tuples with parentheses.
3. Tuples are faster and consume less memory than lists.

Q4. What is a Python dictionary? How is it different from a list?
A4:
1. A dictionary stores key-value pairs, while a list stores a sequence of values.
2. Dictionaries are unordered (until Python 3.7), lists are ordered.
3. Dictionary elements are accessed by keys; list elements are accessed by index.

Q5. Explain the concept of data types in Python.
A5: Python has several built-in data types:

- Numeric types: `int`, `float`, `complex`
- Sequence types: `list`, `tuple`, `range`
- Text type: `str`
- Mapping type: `dict`
- Boolean type: `bool`

Q6. What is the role of indentation in Python?
A6:Indentation in Python defines the block of code of loops, conditionals, functions, and classes. It replaces the use of braces like `{}` in other languages.

Q7. How would you explain Python’s popularity for data science?
A7: Python’s popularity for data science can be explained as

- Rich ecosystem of libraries (e.g., pandas, NumPy, scikit-learn).
- Simple, readable syntax.
- Strong community support.
- Easy integration with other tools (SQL, Excel, web apps).

Q8: What are lambda functions in Python and where would you use them?
A8: Anonymous, single-expression functions used for short simple operations, often in map(), filter(), or sorted().

Q9: Explain *args and **kwargs.
A9: *args collects extra positional arguments as a tuple, **kwargs collects extra keyword arguments as a dictionary in functions.

Q10. What is the role of the apply() function in pandas?
A10. To apply custom functions to rows or columns in a DataFrame or elements in a Series.

Q11: How do you ensure code quality in collaborative Python projects?
A11: Code reviews, linting tools (flake8, black), unit tests (pytest), CI/CD pipelines.6

Q12: How does Python handle memory management and garbage collection?
A12: Python uses reference counting and generational garbage collection (via gc module).

Coding Questions:

Q1. Implement a function to check if a number is prime.

```
def is_prime(n):
    if n <= 1:
        return False
    for i in range(2, int(n**0.5)+1):
        if n % i == 0:
            return False
    return True

# Example
print(is_prime(17))  # Output: True
```

Q2. Use list comprehension to extract even numbers from a list.

```
numbers = [1, 2, 3, 4, 5, 6]
even_numbers = [n for n in numbers if n % 2 == 0]
print(even_numbers)  # Output: [2, 4, 6]
```

Q3. Create a class in Python with __init__, __str__ and one method.

```
class Person:
    def __init__(self, name, age):
        self.name = name
        self.age = age

    def greet(self):
        return f"Hello, my name is {self.name}."

    def __str__(self):
        return f"{self.name}, {self.age} years old"

# Example
p = Person("Alice", 30)
print(p)        # Output: Alice, 30 years old
print(p.greet()) # Output: Hello, my name is Alice.
```

Q4. Build a simple linear regression using scikit-learn.

```
from sklearn.linear_model import LinearRegression
import numpy as np

X = np.array([[1], [2], [3], [4]])
y = np.array([2, 4, 6, 8])

model = LinearRegression()
model.fit(X, y)
print("Coefficient:", model.coef_)
print("Intercept:", model.intercept_)
print("Prediction for 5:", model.predict([[5]]))
```

Q5. Implement k-means clustering from scratch.

```
import numpy as np

X = np.array([[1, 2], [1, 4], [1, 0], [10, 2], [10, 4], [10, 0]])

# Initialize centroids randomly
centroids = X[np.random.choice(len(X), 2, replace=False)]

for _ in range(5):
    clusters = [[] for _ in range(len(centroids))]
    for point in X:
        distances = [np.linalg.norm(point - c) for c in centroids]
        cluster_idx = np.argmin(distances)
        clusters[cluster_idx].append(point)

    centroids = [np.mean(cluster, axis=0) for cluster in clusters]

print("Final centroids:", centroids)
```

Q6. Use NumPy to normalize a dataset.

```
import numpy as np

data = np.array([[1, 2], [2, 3], [3, 4]])
normalized = (data - data.mean(axis=0)) / data.std(axis=0)
print(normalized)
```

Q7. Build a data pipeline that reads, cleans, transforms, and stores data.

```
import pandas as pd

df = pd.read_csv("data.csv")
df.dropna(inplace=True)
df["column"] = df["column"].apply(lambda x: x.strip().lower())
df.to_csv("cleaned_data.csv", index=False)
```

Q8. Deploy a trained model using Flask or FastAPI.

```
from flask import Flask, request, jsonify
import joblib

model = joblib.load("model.pkl")
app = Flask(__name__)

@app.route('/predict', methods=['POST'])
def predict():
    data = request.get_json()
    prediction = model.predict([data['features']])
    return jsonify({'prediction': prediction.tolist()})

# Run using: flask run
```

Q9. Reverse a string without using built-in methods.

```
def reverse_string(s):
    result = ""
    for char in s:
        result = char + result
    return result

# Example
print(reverse_string("hello"))  # Output: "olleh".
```

Q10. Find the second largest number in a list.

```
def second_largest(numbers):
    unique_numbers = list(set(numbers))
    unique_numbers.sort()
    return unique_numbers[-2] if len(unique_numbers) >= 2 else None

# Example
print(second_largest([10, 20, 4, 45, 99, 99]))  # Output: 45
```

Q11. Write a program to count vowels in a string.

```
def count_vowels(s):
    vowels = "aeiouAEIOU"
    return sum(1 for char in s if char in vowels)

# Example
print(count_vowels("Data Science"))  # Output: 5
```

Q12. Use pandas to filter rows where age > 25.

```
import pandas as pd

data = {'name': ['Alice', 'Bob', 'Charlie'], 'age': [24, 27, 22]}
df = pd.DataFrame(data)
filtered_df = df[df['age'] > 25]
print(filtered_df)
```