1. **What is Python?**
   Python is an interpreted, high-level, dynamically typed, and general-purpose programming language known for its readability.
2. **Explain Python's key features.**

Python supports object-oriented programming, is open-source, has extensive libraries, and includes dynamic typing and garbage collection.

3. **What are Python's data types?**
   Common types include int, float, str, bool, list, tuple, set, and dict.
4. **What is PEP 8?**
   PEP 8 is the official Python style guide for writing readable and consistent code.
5. **How is Python interpreted?**
   Python code is compiled into bytecode and then interpreted by the Python Virtual Machine (PVM).

Data Structures

6. **What is a list in Python?**
   list is a mutable, ordered collection of elements, defined using square brackets:
   my_list = [1, 2, 3].
7. **How is a tuple different from a list?**
   tuple is immutable, meaning it cannot be changed after creation.
8. **Explain a dictionary in Python.**
   dictionary is an unordered collection of key-value pairs, defined using {}:
   my_dict = {'key': 'value'}.
9. **What are Python sets?**
   Sets are unordered collections of unique elements: my_set = {1, 2, 3}.
10. **How do you reverse a list in Python?**
    Use list.reverse() or slicing: reversed_list = my_list[::-1].

Control Flow

11. **Explain Python's if statement.**
    It evaluates a condition and executes code if the condition is true:

```
if x > 5:
        print("x is greater than 5")
```

12. **What is a for loop in Python?**
    A loop that iterates over a sequence:

```
for item in [1, 2, 3]:
        print(item)
```

13. **How is a while loop used in Python?**
    It continues as long as the condition is true:

```
while x < 5:
        x += 1
```

14. **What is the purpose of the break statement?**
    It exits the nearest enclosing loop.
15. **What does the continue statement do?**
    It skips the current iteration and moves to the next.

Object-Oriented Programming

16. What is a class in Python?

A blueprint for creating objects, defined using class.

17. What are Python objects?

Instances of classes containing data (attributes) and behavior (methods).

18. What is inheritance in Python?

A mechanism where one class derives properties from another:

```
class Parent:
    pass

class Child(Parent):
    pass
```

19. Explain polymorphism in Python.

Different classes can define methods with the same name but different behavior.

20. What is encapsulation in Python?

Encapsulation restricts direct access to some of an object's components.

Functions and Modules

21. How do you define a function in Python?

Use the def keyword:

```
def greet():
    return "Hello!"
```

22. What are Python's built-in functions?

Examples include len(), print(), type(), range(), and input().

23. What is the difference between *args and **kwargs?

*args captures positional arguments; **kwargs captures keyword arguments.

24. What is the __init__ method?

It initializes a class instance.

25. How do you import a module in Python?

Use the import keyword: import math.

Error Handling

26. What is an exception?

An event that disrupts the normal execution flow, such as ZeroDivisionError.

27. How do you handle exceptions in Python?

Using try-except blocks:

```
try:
    x = 1 / 0
except ZeroDivisionError:
    print("Cannot divide by zero")
```

28. What is the finally block?

Code in the finally block always executes, regardless of exceptions.

29. What is the raise statement?

Used to explicitly raise an exception.

30. What are custom exceptions?

User-defined exceptions derived from the Exception class.

Advanced Concepts

31.What is Python's GIL?

The Global Interpreter Lock prevents multiple native threads from executing Python bytecodes simultaneously.

32.Explain decorators in Python.

Functions that modify another function's behavior:

```
def decorator(func):
    def wrapper():
        print("Before")
        func()
    print("After")
    return wrapper
```

33.What are Python generators?

Functions that yield values one at a time using yield.

34.What is a lambda function?

An anonymous function defined using the lambda keyword: lambda x: x + 1.

35.What are Python's comprehensions?

Compact syntax for creating sequences: [x**2 for x in range(5)].

Intermediate Topics

36.What is the difference between deepcopy and copy?

- copy.copy() creates a shallow copy, copying the object but not nested objects.
- copy.deepcopy() recursively copies all objects, including nested ones.

37.What is the difference between is and ==?

- is checks if two objects refer to the same memory location.
- == checks if the values of two objects are equal.

38.Explain Python's memory management.

Python uses reference counting and garbage collection to manage memory automatically.

39.What is the purpose of Python's with statement?

It simplifies resource management (e.g., file handling):

```
with open('file.txt', 'r') as f:
  data = f.read()
```

40.What are metaclasses in Python?

Metaclasses are classes that define the behavior of other classes.

41.What is the difference between @classmethod and @staticmethod?

- **@classmethod**: Method bound to the class, not an instance.
- **@staticmethod**: Method with no binding to class or instance.

42.What are magic methods in Python?

Special methods with __ (e.g., __init__, __str__) for operator overloading and custom behavior.

43.What is the purpose of __repr__ and __str__?

- **__repr__:** Debug representation of the object.
- **__str__:** User-friendly string representation of the object.

44.What is Python's super() function?

It allows you to call methods from a parent class:

```
class Child(Parent):
    def __init__(self):
        super().__init__()
```

45.What are Python properties?

Properties manage attribute access using @property decorators.

Libraries and Tools

46.What is NumPy?

A library for numerical computations in Python, particularly for arrays and matrices.

47.What is Pandas?

A library for data manipulation and analysis, especially with DataFrames.

48.What is Matplotlib?

A plotting library for creating static, interactive, and animated visualizations.

49.What is Flask?

A lightweight web framework for building Python web applications.

50.What is Django?

A high-level web framework for building scalable web applications.

51.What is TensorFlow?

An open-source library for machine learning and deep learning.

52.What is Selenium?

A library for automating web browsers.

53.What is SQLAlchemy?

A library for database interaction and ORM (Object Relational Mapping).

54.What is Pytest?

A testing framework for writing simple to complex test cases.

55.What is the difference between pip and conda?

- **pip**: Python package manager for PyPI.
- **conda**: Manages Python environments and packages.

File Handling

56.How do you open a file in Python?

Use the open() function:

f = open('file.txt', 'r')

57.What are file modes in Python?

- 'r': Read
- 'w': Write
- 'a': Append
- 'b': Binary mode

58.How do you write to a file in Python?

```
with open('file.txt', 'w') as f:
    f.write("Hello")
```

59.How do you read from a file?

```
with open('file.txt', 'r') as f:
    data = f.read()
```

60.What is Python's os module?

It provides functionality to interact with the operating system, like file operations.

Multithreading and Multiprocessing

61.What is multithreading in Python?

Running multiple threads concurrently, often for I/O-bound tasks.

62.What is the Global Interpreter Lock (GIL)?

A mutex that ensures only one thread executes Python bytecode at a time.

63.What is the threading module?

A module to create and manage threads in Python.

64.What is multiprocessing in Python?

Running multiple processes concurrently for CPU-bound tasks.

65.How do you use Python's multiprocessing module?

from multiprocessing import Process

```
def worker():
    print("Hello from a process")

p = Process(target=worker)
p.start()
p.join()
```

Advanced

66.What is monkey patching?

Dynamically changing a class or module during runtime.

67.What is Python's ctypes library?

It allows interaction with C libraries.

68.What is the difference between async and await?

- **async:** Defines a coroutine.
- **await:** Pauses coroutine execution until the awaited task is done.

69.What is Python's __main__?

It's the entry point of a Python script:

```
if __name__ == "__main__":
    main()
```

70.What are Python annotations?

Syntax for adding metadata to function arguments and return types:

```
def add(x: int, y: int) -> int:
    return x + y
```

Advanced Topics (Continued)

71.What is a Python coroutine?

A coroutine is a special function declared with async def that can pause execution with await and resume later.

72.Explain Python's @property decorator.

Used to make methods act like attributes:

```
class MyClass:
    @property
    def value(self):
        return "Value"
```

73.What is a Python metaclass?

A metaclass defines the behavior of classes, controlling class creation and modification.

74.What is the difference between shallow and deep copies in Python?

- **Shallow Copy:** Copies the top-level object but references the inner objects.
- **Deep Copy:** Recursively copies all objects.

75.How does Python handle memory management?

Python uses reference counting and garbage collection for automatic memory management.

Python Libraries and Ecosystem

76.What is SciPy?

A library for scientific and technical computing, extending NumPy.

77.What is PyTorch?

A machine learning framework for building deep learning models.

78.What is OpenCV?

A library for image and video processing.

79.What is Scrapy?

A framework for web scraping in Python.

80.What is BeautifulSoup?

A library for parsing HTML and XML for web scraping.

Testing in Python

81.What is Python's unittest module?

A built-in testing framework to write and run test cases.

82.What is the purpose of a test fixture?

A setup needed for testing, like initializing objects or creating databases.

83.What is mocking in Python?

Mocking is creating fake objects to simulate real ones in testing using the unittest.mock library.

84.What are Python's assert statements?

Used to validate conditions in tests:

```
assert x == 5, "x should be 5"
```

85.What is the difference between unit tests and integration tests?

- **Unit Tests:** Test individual components.
- **Integration Tests:** Test how components work together.

Web Development in Python

86.What is WSGI?

Web Server Gateway Interface, a standard for Python web servers and frameworks to communicate.

87.What is Flask's route decorator?

Maps a URL to a view function:

```
@app.route("/")
def home():
  return "Welcome!"
```

88.What is Django ORM?

Django's Object-Relational Mapping maps database tables to Python classes.

89.What are Django middleware?

Components that process requests and responses globally in an application.

90.What is FastAPI?

A modern web framework for building fast APIs using Python and type hints.

Machine Learning and Data Science in Python

91.What is the difference between NumPy arrays and Python lists?

NumPy arrays are faster, consume less memory, and support element-wise operations.

92.What is a DataFrame in Pandas?

A 2D labeled data structure like a table or spreadsheet.

93.What is Scikit-learn?

A library for machine learning, offering tools for classification, regression, clustering, etc.

94. How do you handle missing data in Pandas?

- df.fillna(value) to fill missing values.
- df.dropna() to drop rows or columns with missing data.

95. What is the purpose of Matplotlib?

For data visualization using plots like line, bar, scatter, etc.

Miscellaneous

96. What is Python's pickle module?

Serializes and deserializes Python objects for storage or transfer.

97. What is the difference between join() and split() in strings?

- **join():** Combines a list into a string:

"-".join(["a", "b", "c"]) # "a-b-c"

- **split():** Breaks a string into a list:

"a-b-c".split("-") # ["a", "b", "c"]

98. What is the difference between mutable and immutable objects in Python?

- **Mutable:** Objects that can be changed (e.g., lists, dictionaries).
- **Immutable:** Objects that cannot be changed (e.g., strings, tuples).

99. What are Python namespaces?

A system to store variable names and avoid conflicts by mapping names to objects.

100. What is Python's zip() function?

Combines multiple iterables into tuples:

list(zip([1, 2], ['a', 'b'])) # [(1, 'a'), (2, 'b')]