Python Interview Questions & Answers Compilation -:

Creating a comprehensive list of over 100 Python questions and answers covering a variety of topics is quite a task, but I'll provide a structured list divided by key topics such as basic syntax, data structures, functions, OOP, modules, file handling, exceptions, and advanced concepts. Here's a start to such a list:

1. Basic Syntax

Q1: What is Python? A1: Python is a high-level, interpreted programming language known for its readability and versatility. It supports multiple programming paradigms, including procedural, object-oriented, and functional programming.

Q2: How do you write a comment in Python? A2: Single-line comments start with a `#`. Multi-line comments can be written using triple quotes (`''' comment '''` or `""" comment """`).

Q3: How do you print a message in Python? A3: Use the `print()` function. For example:

```python
print("Hello, World!")
```

Q4: How do you get input from the user in Python? A4: Use the `input()` function. For example:

```python
name = input("Enter your name: ")
```

2. Data Types and Variables

Q5: What are the basic data types in Python? A5: The basic data types include integers (`int`), floating-point numbers (`float`), strings (`str`), and booleans (`bool`).

Q6: How do you declare a variable in Python? A6: Simply assign a value to a variable name. For example:

```python
x = 5
name = "Alice"
```

Q7: How do you check the type of a variable? A7: Use the `type()` function. For example:

```python
print(type(x))  # Output: <class 'int'>
```

3. Data Structures

Q8: What are lists in Python? A8: Lists are ordered, mutable collections of items. They are defined using square brackets. For example:

```python
my_list = [1, 2, 3, 4, 5]
```

Q9: How do you add an item to a list? A9: Use the `append()` method. For example:

```python
my_list.append(6)
```

Q10: How do you remove an item from a list? A10: Use the `remove()` method or the `pop()` method. For example:

```python
my_list.remove(3)
# or
my_list.pop(2)  # Removes the item at index 2
```

Q11: What are tuples in Python? A11: Tuples are ordered, immutable collections of items. They are defined using parentheses. For example:

```python
my_tuple = (1, 2, 3)
```

Q12: What are dictionaries in Python? A12: Dictionaries are unordered collections of key-value pairs. They are defined using curly braces. For example:

```python
my_dict = {"name": "Alice", "age": 25}
```

Q13: How do you access a value in a dictionary? A13: Use the key inside square brackets. For example:

```python
print(my_dict["name"])  # Output: Alice
```

4. Control Flow

Q14: How do you write an if-else statement in Python? A14: Use the `if`, `elif`, and `else` keywords. For example:

```python
x = 10
if x > 0:
    print("Positive")
elif x == 0:
    print("Zero")
else:
    print("Negative")
```

Q15: How do you write a for loop in Python? A15: Use the `for` keyword followed by an iterable. For example:

python

```python
for i in range(5):
    print(i)
```

Q16: How do you write a while loop in Python? A16: Use the `while` keyword followed by a condition. For example:

python

```python
count = 0
while count < 5:
    print(count)
    count += 1
```

5. Functions

Q17: How do you define a function in Python? A17: Use the `def` keyword followed by the function name and parentheses. For example:

python

```python
def greet(name):
    return f"Hello, {name}!"
```

Q18: How do you call a function in Python? A18: Use the function name followed by parentheses. For example:

python

```python
print(greet("Alice"))
```

Q19: What are default arguments in a function? A19: Default arguments are specified by assigning a value in the function definition. For example:

python

```python
def greet(name, msg="Hello"):
    return f"{msg}, {name}!"
```

Q20: What are *args and **kwargs? A20:** `*args` allows a function to accept any number of positional arguments, and `**kwargs` allows a function to accept any number of keyword arguments. For example:

python

```python
def func(*args, **kwargs):
    print(args)
    print(kwargs)

func(1, 2, 3, a=4, b=5)
```

6. Object-Oriented Programming (OOP)

Q21: What is a class in Python? A21: A class is a blueprint for creating objects. It defines a set of attributes and methods. For example:

python

```python
class Dog:
    def __init__(self, name):
        self.name = name

    def bark(self):
        return f"{self.name} says woof!"
```

Q22: How do you create an object of a class? A22: Use the class name followed by parentheses. For example:

python

```python
my_dog = Dog("Buddy")
print(my_dog.bark())
```

Q23: What is inheritance in Python? A23: Inheritance allows a class to inherit attributes and methods from another class. For example:

python

```python
class Animal:
    def __init__(self, name):
        self.name = name

class Dog(Animal):
    def bark(self):
        return f"{self.name} says woof!"
```

Q24: What is polymorphism in Python? A24: Polymorphism allows different classes to be treated as instances of the same class through a common interface. For example:

python

```python
class Cat:
    def speak(self):
        return "Meow"

class Dog:
    def speak(self):
        return "Woof"

def make_sound(animal):
    print(animal.speak())

make_sound(Cat())
make_sound(Dog())
```

7. Modules and Packages

Q25: How do you import a module in Python? A25: Use the `import` statement. For example:

python

```python
import math
print(math.sqrt(16))
```

Q26: How do you install a package in Python? A26: Use the `pip` command. For example:

```sh
pip install numpy
```

Q27: How do you import specific functions from a module? A27: Use the `from` keyword. For example:

```python
from math import sqrt, pi
print(sqrt(16))
print(pi)
```

8. File Handling

Q28: How do you open a file in Python? A28: Use the `open()` function. For example:

```python
file = open("example.txt", "r")
```

Q29: How do you read from a file? A29: Use the `read()` method. For example:

```python
content = file.read()
print(content)
file.close()
```

Q30: How do you write to a file? A30: Use the `write()` method. For example:

```python
file = open("example.txt", "w")
file.write("Hello, World!")
file.close()
```

Q31: How do you handle file exceptions? A31: Use a `try-except` block. For example:

```python
```

```
try:
    file = open("example.txt", "r")
except FileNotFoundError:
    print("File not found")
```

9. Exception Handling

Q32: How do you handle exceptions in Python? A32: Use the `try-except` block. For example:

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

Q33: How do you catch multiple exceptions? A33: Use multiple `except` blocks. For example:

```python
try:
    x = int("abc")
except ValueError:
    print("ValueError")
except TypeError:
    print("TypeError")
```

Q34: How do you use the `finally` block? A34: The `finally` block executes regardless of whether an exception occurs. For example:

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

```
finally:
    print("This will always execute")
```

10. Advanced Concepts

Q35: What are decorators in Python? A35: Decorators are functions that modify the behavior of other functions. For example:

```python
def decorator_func(func):
    def wrapper():
        print("Something is happening before the function is called.")
        func()
        print("Something is happening after the function is called.")
    return wrapper

@decorator_func
def say_hello():
    print("Hello!")

say_hello()
```

Q36: What are generators in Python? A36: Generators are functions that return an iterable set of items, one at a time, in a special way using the `yield` keyword. For example:

```python
def my_generator():
    yield 1
    yield 2
    yield 3

gen = my_generator()
print(next(gen))
print(next(gen))
print(next(gen))
```

Q37: What is a lambda function in Python? A37: Lambda functions are small anonymous functions defined with the `lambda` keyword. For example:

python

```python
add = lambda x, y: x + y
print(add(2, 3))
```

Q38: What is list comprehension? A38: List comprehension provides a concise way to create lists. For example:

python

```python
squares = [x**2 for x in range(10)]
print(squares)
```

Q39: What is a context manager in Python? A39: Context managers are used to manage resources, such as file streams. They are implemented using the `with` statement. For example:

python

```python
with open("example.txt", "r") as file:
    content = file.read()
    print(content)
```

11. Testing

Q40: What is unit testing in Python? A40: Unit testing is a method where individual units or components of the software are tested. Python's `unittest` module is commonly used for this. For example:

python

```python
import unittest

def add(x, y):
    return x + y
```

```
class TestMath(unittest.TestCase):
    def test_add(self):
        self.assertEqual(add(2, 3), 5)

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

12. Regular Expressions

Q41: What are regular expressions in Python? A41: Regular expressions are patterns used to match character combinations in strings. Python provides the `re` module for working with regular expressions. For example:

python

```
import re

pattern = r"\d+"
text = "There are 123 numbers in this text."
matches = re.findall(pattern, text)
print(matches)
```

This is a subset of the questions you might encounter when learning Python. To reach a total of 100 questions and answers, you would continue to delve deeper into each topic and introduce additional topics such as working with external libraries, advanced data structures, concurrency, and more.

Would you like to continue with more questions, or focus on any particular section in more detail?

more

Sure! Let's continue with more questions across various topics, aiming for a total of 100.

1. Basic Syntax (Continued)

Q42: What is the difference between `print()` and `return` in Python? A42: `print()` outputs data to the console, while `return` exits a function and optionally passes back an expression to the caller.

Q43: How do you write a multi-line string in Python? A43: Use triple quotes, either `'''` or `"""`. For example:

python

```python
multi_line_string = """This is a multi-line string."""
```

Q44: What is PEP 8? A44: PEP 8 is the Python Enhancement Proposal that provides guidelines and best practices on how to write Python code.

2. Data Types and Variables (Continued)

Q45: What is the difference between `=` and `==` in Python? A45: `=` is the assignment operator, while `==` is the equality operator used to compare two values.

Q46: How do you convert a string to an integer in Python? A46: Use the `int()` function. For example:

python

```python
num = int("123")
```

Q47: How do you convert an integer to a string in Python? A47: Use the `str()` function. For example:

python

```python
num_str = str(123)
```

3. Data Structures (Continued)

Q48: How do you create a set in Python? A48: Use curly braces or the `set()` function. For example:

python

```
my_set = {1, 2, 3}
# or
my_set = set([1, 2, 3])
```

Q49: What is the difference between a list and a tuple in Python? A49: A list is mutable, meaning it can be changed, while a tuple is immutable, meaning it cannot be changed once created.

Q50: How do you perform a union of sets in Python? A50: Use the `union()` method or the `|` operator. For example:

```python
set1 = {1, 2, 3}
set2 = {3, 4, 5}
union_set = set1.union(set2)
# or
union_set = set1 | set2
```

Q51: How do you perform an intersection of sets in Python? A51: Use the `intersection()` method or the `&` operator. For example:

```python
intersection_set = set1.intersection(set2)
# or
intersection_set = set1 & set2
```

Q52: How do you create a dictionary comprehension in Python? A52: Use curly braces with a key-value expression. For example:

```python
squares = {x: x**2 for x in range(5)}
print(squares)  # Output: {0: 0, 1: 1, 2: 4, 3: 9, 4: 16}
```

4. Control Flow (Continued)

Q53: What is the purpose of the `break` statement in Python? A53: The `break` statement exits the nearest enclosing loop immediately.

Q54: What is the purpose of the `continue` statement in Python? A54: The `continue` statement skips the current iteration and moves to the next iteration of the loop.

Q55: How do you use a `for` loop with an `else` clause? A55: The `else` clause in a `for` loop executes after the loop completes normally (without encountering a `break` statement). For example:

```python
for i in range(5):
    print(i)
else:
    print("Loop completed")
```

5. Functions (Continued)

Q56: How do you define a recursive function in Python? A56: A recursive function is a function that calls itself. For example:

```python
def factorial(n):
    if n == 1:
        return 1
    else:
        return n * factorial(n-1)
```

Q57: How do you use the `map()` function in Python? A57: The `map()` function applies a given function to all items in an input list. For example:

```python
def square(x):
    return x * x

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

```
squared_numbers = list(map(square, numbers))
print(squared_numbers)  # Output: [1, 4, 9, 16]
```

Q58: How do you use the `filter()` function in Python? A58: The `filter()` function constructs a list from elements of an iterable for which a function returns true. For example:

python

```
def is_even(x):
    return x % 2 == 0

numbers = [1, 2, 3, 4, 5, 6]
even_numbers = list(filter(is_even, numbers))
print(even_numbers)  # Output: [2, 4, 6]
```

6. Object-Oriented Programming (OOP) (Continued)

Q59: What is encapsulation in Python? A59: Encapsulation is the bundling of data and methods that operate on the data within one unit, such as a class. It restricts direct access to some of the object's components.

Q60: What is an abstract class in Python? A60: An abstract class is a class that cannot be instantiated and typically includes one or more abstract methods that must be implemented by subclasses. Use the `abc` module to define abstract classes. For example:

python

```
from abc import ABC, abstractmethod

class Animal(ABC):
    @abstractmethod
    def make_sound(self):
        pass

class Dog(Animal):
    def make_sound(self):
        return "Woof"
```

Q61: What is method overloading in Python? A61: Python does not support method overloading directly. Instead, you can achieve similar functionality by using default arguments or by checking types within a method.

Q62: What is method overriding in Python? A62: Method overriding occurs when a subclass provides a specific implementation of a method that is already defined in its superclass.

7. Modules and Packages (Continued)

Q63: How do you create a module in Python? A63: A module is simply a Python file containing functions and variables. For example, create a file `mymodule.py`:

```python
# mymodule.py
def greet(name):
    return f"Hello, {name}!"
```

Q64: How do you use the `dir()` function in Python? A64: The `dir()` function returns a list of valid attributes for an object. For example:

```python
import math
print(dir(math))
```

Q65: How do you reload a module in Python? A65: Use the `importlib.reload()` function from the `importlib` module. For example:

```python
import mymodule
import importlib
importlib.reload(mymodule)
```

8. File Handling (Continued)

Q66: How do you read a file line by line in Python? A66: Use a `for` loop to iterate over the file object. For example:

```python
with open("example.txt", "r") as file:
    for line in file:
        print(line.strip())
```

Q67: How do you write multiple lines to a file in Python? A67: Use the `writelines()` method. For example:

```python
lines = ["First line\n", "Second line\n", "Third line\n"]
with open("example.txt", "w") as file:
    file.writelines(lines)
```

Q68: How do you append data to a file in Python? A68: Open the file in append mode using `"a"`. For example:

```python
with open("example.txt", "a") as file:
    file.write("Additional line\n")
```

9. Exception Handling (Continued)

Q69: How do you raise an exception in Python? A69: Use the `raise` keyword. For example:

```python
def divide(x, y):
    if y == 0:
        raise ValueError("Cannot divide by zero")
    return x / y
```

Q70: How do you define a custom exception in Python? A70: Create a new class that inherits from the built-in `Exception` class. For example:

```python
class CustomError(Exception):
    pass

try:
    raise CustomError("This is a custom error")
except CustomError as e:
    print(e)
```

10. Advanced Concepts (Continued)

Q71: What is a closure in Python? A71: A closure is a function object that has access to variables in its lexical scope, even after the outer function has finished executing. For example:

```python
def outer_func(msg):
    def inner_func():
        print(msg)
    return inner_func

closure = outer_func("Hello")
closure()  # Output: Hello
```

Q72: What are metaclasses in Python? A72: Metaclasses are classes of classes that define how a class behaves. A class is an instance of a metaclass. Use the `type` function to create metaclasses.

Q73: How do you use the `itertools` module in Python? A73: The `itertools` module provides various functions that work on iterators to produce complex iterators. For example:

```python
import itertools

perms = list(itertools.permutations([1, 2, 3]))
print(perms)
```

11. Testing (Continued)

Q74: What is the `assert` statement in Python? A74: The `assert` statement is used during debugging to check for conditions that should always be true. For example:

```python
def add(x, y):
    assert isinstance(x, int) and isinstance(y, int), "Inputs must be integers"
    return x + y
```

Q75: How do you use the `pytest` framework in Python? A75: `pytest` is a testing framework that makes it easy to write small tests. Install it using `pip install pytest` and write test functions prefixed with `test_`. For example:

```python
def add(x, y):
    return x + y

def test_add():
    assert add(2, 3) == 5
    assert add(-1, 1) == 0
```

Run the tests using the `pytest` command.

12. Regular Expressions (Continued)

Q76: How do you use the `re.search()` function in Python? A76: The `re.search()` function searches for a pattern within a string and returns a match object if found. For example:

```python
import re

pattern = r"\d+"
text = "The number is 123"
match = re.search(pattern, text)
```

```
if match:
    print(match.group())
```

Q77: How do you use the `re.sub()` function in Python? A77: The `re.sub()` function replaces occurrences of a pattern with a replacement string. For example:

python

```
import re

pattern = r"\d+"
text = "The number is 123"
new_text = re.sub(pattern, "456", text)
print(new_text)  # Output: The number is 456
```

13. Advanced Topics

Q78: What are decorators with arguments in Python? A78: Decorators with arguments allow you to pass arguments to the decorator. For example:

python

```
def repeat(n):
    def decorator(func):
        def wrapper(*args, **kwargs):
            for _ in range(n):
                func(*args, **kwargs)
        return wrapper
    return decorator

@repeat(3)
def say_hello():
    print("Hello!")

say_hello()
```

Q79: What are coroutines in Python? A79: Coroutines are a more generalized form of subroutines, used for cooperative multitasking. Use the `async` and `await` keywords. For example:

```python
import asyncio

async def greet():
    print("Hello")
    await asyncio.sleep(1)
    print("World")

asyncio.run(greet())
```

Q80: What is the `yield from` statement in Python? A80: The `yield from` statement is used to delegate part of a generator's operations to another generator. For example:

```python
def generator1():
    yield from range(5)

def generator2():
    yield from generator1()
    yield 5

for value in generator2():
    print(value)
```

Q81: How do you handle binary data in Python? A81: Use the `struct` module to work with binary data. For example:

```python
import struct

data = struct.pack('i', 1024)
print(data)
value = struct.unpack('i', data)
print(value)
```

Q82: What is the `collections` module in Python? A82: The `collections` module provides specialized data structures such as `namedtuple`, `deque`, `Counter`, and `OrderedDict`. For example:

```python
from collections import Counter

counter = Counter(['a', 'b', 'a', 'c', 'a', 'b'])
print(counter)  # Output: Counter({'a': 3, 'b': 2, 'c': 1})
```

Q83: What is the `functools` module in Python? A83: The `functools` module provides higher-order functions that act on or return other functions. For example, use `functools.reduce()`:

```python
from functools import reduce

numbers = [1, 2, 3, 4]
product = reduce(lambda x, y: x * y, numbers)
print(product)  # Output: 24
```

Q84: How do you use the `json` module in Python? A84: The `json` module is used to work with JSON data. For example:

```python
import json

data = {"name": "John", "age": 30}
json_data = json.dumps(data)
print(json_data)  # Output: {"name": "John", "age": 30}

parsed_data = json.loads(json_data)
print(parsed_data)  # Output: {'name': 'John', 'age': 30}
```

Q85: How do you use the `datetime` module in Python? A85: The `datetime` module is used for manipulating dates and times. For example:

python

```
from datetime import datetime, timedelta

now = datetime.now()
print(now)

one_week_later = now + timedelta(weeks=1)
print(one_week_later)
```

Q86: What is the `pathlib` module in Python? A86: The `pathlib` module provides an object-oriented interface to working with filesystem paths. For example:

python

```
from pathlib import Path

path = Path("example.txt")
print(path.exists())
print(path.read_text())
```

14. Concurrency

Q87: What is the difference between threading and multiprocessing in Python? A87: `threading` is used for concurrent execution of code in multiple threads, sharing the same memory space. `multiprocessing` is used for concurrent execution of code in separate processes, each with its own memory space.

Q88: How do you create a thread in Python? A88: Use the `threading` module. For example:

python

```
import threading

def print_numbers():
    for i in range(5):
        print(i)

thread = threading.Thread(target=print_numbers)
```

```
thread.start()
thread.join()
```

Q89: How do you create a process in Python? A89: Use the `multiprocessing` module. For example:

```python
import multiprocessing

def print_numbers():
    for i in range(5):
        print(i)

process = multiprocessing.Process(target=print_numbers)
process.start()
process.join()
```

Q90: What is the Global Interpreter Lock (GIL) in Python? A90: The GIL is a mutex that protects access to Python objects, preventing multiple native threads from executing Python bytecodes at once. This can be a bottleneck in CPU-bound and multi-threaded code.

15. Network Programming

Q91: How do you create a TCP server in Python? A91: Use the `socket` module. For example:

```python
import socket

server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
server_socket.bind(('localhost', 8080))
server_socket.listen()

while True:
    client_socket, addr = server_socket.accept()
    print(f"Connection from {addr}")
```

Byom.cybertechie

```python
    client_socket.send(b"Hello, client!")
    client_socket.close()
```

Q92: How do you create a TCP client in Python? A92: Use the `socket` module. For example:

```python
import socket

client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
client_socket.connect(('localhost', 8080))
message = client_socket.recv(1024)
print(message.decode())
client_socket.close()
```

Q93: How do you perform HTTP requests in Python? A93: Use the `requests` module. For example:

```python
import requests

response = requests.get("https://api.example.com/data")
print(response.json())
```

Q94: How do you handle URLs in Python? A94: Use the `urllib` module. For example:

```python
import urllib.parse

url = "https://www.example.com/index.html?name=John&age=30"
parsed_url = urllib.parse.urlparse(url)
print(parsed_url)
```

16. Databases

Q95: How do you connect to a SQLite database in Python? A95: Use the `sqlite3` module. For example:

```python
import sqlite3

connection = sqlite3.connect('example.db')
cursor = connection.cursor()
cursor.execute('''CREATE TABLE IF NOT EXISTS users (id INTEGER PRIMARY KEY, name TEXT)''')
cursor.execute('''INSERT INTO users (name) VALUES ('John')''')
connection.commit()
connection.close()
```

Q96: How do you connect to a MySQL database in Python? A96: Use the `mysql-connector-python` module. For example:

```python
import mysql.connector

connection = mysql.connector.connect(
    host="localhost",
    user="yourusername",
    password="yourpassword",
    database="yourdatabase"
)

cursor = connection.cursor()
cursor.execute("SHOW TABLES")
for table in cursor:
    print(table)

connection.close()
```