155 Python Programming Questions For Beginers

1. Data Types and Variables

1. **What are the different data types in Python?**

```
python
Copy code
# Integer
x = 5

# Float
y = 3.14

# String
name = "John"

# List
fruits = ["apple", "banana", "cherry"]

# Tuple
coords = (1, 2)

# Dictionary
person = {"name": "Alice", "age": 25}

# Set
unique_numbers = {1, 2, 3}
```

2. **How do you create a variable in Python?**

```
python
Copy code
a = 10
```

3. **What is a mutable and immutable type in Python?**

Mutable: List, Dictionary, Set
Immutable: String, Tuple, Integer

4. **How do you convert a string to an integer in Python?**

```
python
Copy code
num_str = "123"
num = int(num_str)
```

5. **What is the difference between `is` and `==`?**

```
python
Copy code
# 'is' checks object identity
a = [1, 2]
```

```
b = a
print(a is b)  # True

# '==' checks value equality
a = [1, 2]
b = [1, 2]
print(a == b)  # True
```

6. **How do you find the data type of a variable?**

```
python
Copy code
x = 10
print(type(x))  # <class 'int'>
```

7. **Explain the concept of `None` in Python.**

```
python
Copy code
a = None
print(type(a))  # <class 'NoneType'>
```

8. **What is the difference between a list and a tuple in Python?**

List: Mutable, defined by `[]`

Tuple: Immutable, defined by `()`

```
python
Copy code
# List
lst = [1, 2, 3]
lst[0] = 10  # Works

# Tuple
tup = (1, 2, 3)
# tup[0] = 10  # Throws TypeError
```

---

2. Control Flow

9. **What is an `if` statement in Python?**

```
python
Copy code
x = 10
if x > 5:
    print("x is greater than 5")
```

10. **How do you write an `if-else` statement in Python?**

```
python
Copy code
x = 3
if x > 5:
    print("x is greater than 5")
```

```
else:
    print("x is less than or equal to 5")
```

11. What is a `for` loop and how is it different from a `while` loop?

`for` loop iterates over a sequence.
`while` loop runs as long as a condition is `True`.

```
python
Copy code
# For loop
for i in range(5):
    print(i)  # Prints 0 to 4

# While loop
i = 0
while i < 5:
    print(i)  # Prints 0 to 4
    i += 1
```

12. How do you write a `while` loop in Python?

```
python
Copy code
x = 0
while x < 5:
    print(x)
    x += 1
```

13. Explain the `break` and `continue` statements in Python.

```
python
Copy code
# break
for i in range(10):
    if i == 5:
        break  # Exit loop when i is 5
    print(i)

# continue
for i in range(10):
    if i == 5:
        continue  # Skip iteration when i is 5
    print(i)
```

14. What is a `pass` statement used for?

```
python
Copy code
# pass is a placeholder, used when a statement is required syntactically
but you don't want to execute any code.
def function():
    pass
```

15. What is a `try-except` block in Python?

```
python
Copy code
try:
```

```
    x = 10 / 0
except ZeroDivisionError:
    print("Cannot divide by zero")
```

---

3. Functions

16.How do you define a function in Python?

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

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

17.What is the difference between a function and a lambda function?

```
python
Copy code
# Regular function
def square(x):
    return x * x

# Lambda function
square_lambda = lambda x: x * x
```

18.What are default arguments in Python functions?

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

print(greet())  # Hello, Guest!
print(greet("Alice"))  # Hello, Alice!
```

19.How do you return multiple values from a Python function?

```
python
Copy code
def get_coordinates():
    return 1, 2

x, y = get_coordinates()
```

20.What is the purpose of the `global` keyword in Python?

```
python
Copy code
x = 5

def change_global():
    global x
    x = 10

change_global()
print(x)  # 10
```

21. What is the purpose of the `nonlocal` keyword in Python?

```
python
Copy code
def outer():
    x = 5
    def inner():
        nonlocal x
        x = 10
    inner()
    print(x)  # 10

outer()
```

---

4. Collections

22. What is a list in Python?

```
python
Copy code
lst = [1, 2, 3]
```

23. How do you add elements to a list in Python?

```
python
Copy code
lst = [1, 2, 3]
lst.append(4)
print(lst)  # [1, 2, 3, 4]
```

24. What is a dictionary in Python?

```
python
Copy code
person = {"name": "Alice", "age": 25}
```

25. How do you remove an item from a dictionary in Python?

```
python
Copy code
person = {"name": "Alice", "age": 25}
person.pop("age")
print(person)  # {'name': 'Alice'}
```

26. What is a set in Python and how is it different from a list?

- Set: Unordered, no duplicates.

```
python
Copy code
my_set = {1, 2, 3}
```

27. How do you merge two lists in Python?

```
python
Copy code
lst1 = [1, 2]
lst2 = [3, 4]
```

```
lst3 = lst1 + lst2
print(lst3)  # [1, 2, 3, 4]
```

28.How do you access elements in a list, dictionary, and set?

```
python
Copy code
# List
lst = [1, 2, 3]
print(lst[0])  # 1

# Dictionary
person = {"name": "Alice", "age": 25}
print(person["name"])  # Alice

# Set (use iteration or conversion to list)
my_set = {1, 2, 3}
for item in my_set:
    print(item)
```

29.What are list comprehensions in Python?

```
python
Copy code
lst = [x for x in range(5)]
print(lst)  # [0, 1, 2, 3, 4]
```

30.What is a nested list in Python?

```
python
Copy code
nested_lst = [[1, 2], [3, 4], [5, 6]]
```

31.How do you sort a list in Python?

```
python
Copy code
lst = [3, 1, 2]
lst.sort()
print(lst)  # [1, 2, 3]
```

---

5. String Operations

32.How do you concatenate strings in Python?

```
python
Copy code
str1 = "Hello"
str2 = "World"
result = str1 + " " + str2
print(result)  # Hello World
```

33.How do you split a string in Python?

```
python
Copy code
text = "Hello World"
words = text.split()
```

```
print(words)  # ['Hello', 'World']
```

34.How do you check if a string contains a certain substring in Python?

```
python
Copy code
text = "Hello World"
print("World" in text)  # True
```

35.How do you convert a string to uppercase or lowercase in Python?

```
python
Copy code
text = "Hello World"
print(text.upper())  # HELLO WORLD
print(text.lower())  # hello world
```

36.How do you remove whitespace from the beginning and end of a string?

```
python
Copy code
text = "  Hello World  "
print(text.strip())  # Hello World
```

37.What is string slicing in Python?

```
python
Copy code
text = "Hello World"
print(text[0:5])  # Hello
print(text[6:])   # World
```

38.How do you find the length of a string in Python?

```
python
Copy code
text = "Hello World"
print(len(text))  # 11
```

---

6. Classes and Objects

39.How do you define a class in Python?

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

person = Person("Alice", 25)
print(person.name)  # Alice
```

40.What is a constructor in Python?

```
python
Copy code
# Constructor is the __init__ method that initializes the object
```

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

person = Person("Bob", 30)
print(person.name)  # Bob
```

41. What is inheritance in Python?

```
python
Copy code
class Animal:
    def speak(self):
        print("Animal speaks")

class Dog(Animal):
    def speak(self):
        print("Dog barks")

dog = Dog()
dog.speak()  # Dog barks
```

42. How do you create an object of a class in Python?

```
python
Copy code
class Person:
    def __init__(self, name):
        self.name = name

person1 = Person("Alice")
print(person1.name)  # Alice
```

43. What is method overriding in Python?

```
python
Copy code
class Animal:
    def sound(self):
        print("Animal sound")

class Dog(Animal):
    def sound(self):
        print("Bark")

dog = Dog()
dog.sound()  # Bark
```

44. What is the difference between `__str__()` and `__repr__()` in Python?

```
python
Copy code
class Person:
    def __init__(self, name):
        self.name = name

    def __str__(self):
        return f"Person: {self.name}"

    def __repr__(self):
```

```
        return f"Person({self.name})"

person = Person("Alice")
print(str(person))   # Person: Alice
print(repr(person))  # Person(Alice)
```

45.How do you create a static method in Python?

```
python
Copy code
class MyClass:
    @staticmethod
    def static_method():
        print("Static method called")

MyClass.static_method()  # Static method called
```

46.How do you create a class method in Python?

```
python
Copy code
class MyClass:
    @classmethod
    def class_method(cls):
        print("Class method called")

MyClass.class_method()  # Class method called
```

---

7. File Handling

47.How do you open a file in Python?

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

48.How do you read from a file in Python?

```
python
Copy code
file = open("example.txt", "r")
content = file.read()
print(content)
file.close()
```

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

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

50.How do you close a file in Python?

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

```
file.close()
```

51. What is the difference between `read()` and `readlines()`?

```
python
Copy code
# read() reads the entire content as a single string
file = open("example.txt", "r")
content = file.read()
print(content)
file.close()

# readlines() reads the content as a list of lines
file = open("example.txt", "r")
lines = file.readlines()
print(lines)
file.close()
```

52. How do you handle file exceptions in Python?

```
python
Copy code
try:
    file = open("example.txt", "r")
    content = file.read()
except FileNotFoundError:
    print("File not found")
finally:
    file.close()
```

---

8. Error Handling

53. What is an exception in Python?

```
python
Copy code
try:
    x = 10 / 0
except ZeroDivisionError as e:
    print(f"Error: {e}")
```

54. What is the difference between `try-except` and `try-except-finally` blocks?

```
python
Copy code
# try-except
try:
    x = 10 / 0
except ZeroDivisionError:
    print("Cannot divide by zero")

# try-except-finally
try:
    x = 10 / 0
except ZeroDivisionError:
    print("Cannot divide by zero")
finally:
    print("This will always execute")
```

55.How do you raise an exception in Python?

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

print(divide(10, 0))  # Raises ValueError
```

56.What is a custom exception in Python?

```
python
Copy code
class CustomError(Exception):
    pass

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

57.What is the purpose of the `assert` statement in Python?

```
python
Copy code
x = 10
assert x == 10  # No output, passes
assert x == 5   # Raises AssertionError
```

---

9. Modules and Packages

58.How do you import a module in Python?

```
python
Copy code
import math
print(math.sqrt(16))  # 4.0
```

59.What is the difference between `import` and `from ... import`?

```
python
Copy code
# import
import math
print(math.sqrt(16))  # 4.0

# from ... import
from math import sqrt
print(sqrt(16))  # 4.0
```

60.What is the purpose of the `__init__.py` file in a package?

The `__init__.py` file is used to mark a directory as a Python package, enabling it to contain submodules.

61. How do you create a package in Python?

Create a directory and add an `__init__.py` file inside.

62. What is the difference between `os` and `sys` modules?

`os`: Provides functions for interacting with the operating system.
`sys`: Provides access to system-specific parameters and functions.

```
python
Copy code
import os
print(os.getcwd())  # Get current working directory

import sys
print(sys.version)  # Python version
```

63. How do you install a package using `pip`?

Use the command `pip install package_name` in the terminal.

---

10. Decorators and Generators

64. What is a decorator in Python?

```
python
Copy code
def my_decorator(func):
    def wrapper():
        print("Before the function call")
        func()
        print("After the function call")
    return wrapper

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

say_hello()
```

65. How do you pass arguments to a decorator in Python?

```
python
Copy code
def my_decorator(func):
    def wrapper(*args, **kwargs):
        print("Before the function call")
        func(*args, **kwargs)
        print("After the function call")
    return wrapper

@my_decorator
def say_hello(name):
    print(f"Hello, {name}!")

say_hello("Alice")
```

66. What is a generator in Python?

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

gen = my_generator()
for value in gen:
    print(value)
```

67. What is the difference between a generator and a normal function?

A generator uses the `yield` statement and returns an iterator, whereas a normal function returns a single result using `return`.

68. How do you create an infinite generator in Python?

```
python
Copy code
def infinite_gen():
    num = 0
    while True:
        yield num
        num += 1

gen = infinite_gen()
for _ in range(5):
    print(next(gen))  # Prints 0, 1, 2, 3, 4
```

---

11. Lambda Functions and Map/Filter/Reduce

69. What is a lambda function in Python?

```
python
Copy code
square = lambda x: x * x
print(square(5))  # 25
```

70. How do you use `map()` in Python?

```
python
Copy code
numbers = [1, 2, 3, 4]
squared = map(lambda x: x * x, numbers)
print(list(squared))  # [1, 4, 9, 16]
```

71. How do you use `filter()` in Python?

```
python
Copy code
numbers = [1, 2, 3, 4, 5]
even_numbers = filter(lambda x: x % 2 == 0, numbers)
print(list(even_numbers))  # [2, 4]
```

72. What is `reduce()` in Python?

```
python
Copy code
from functools import reduce

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

---

12. Comprehensions

73. What is a list comprehension in Python?

```
python
Copy code
numbers = [1, 2, 3, 4, 5]
squares = [x * x for x in numbers]
print(squares)  # [1, 4, 9, 16, 25]
```

74. How do you create a dictionary comprehension in Python?

```
python
Copy code
numbers = [1, 2, 3, 4, 5]
square_dict = {x: x * x for x in numbers}
print(square_dict)  # {1: 1, 2: 4, 3: 9, 4: 16, 5: 25}
```

75. How do you create a set comprehension in Python?

```
python
Copy code
numbers = [1, 2, 3, 4, 5]
square_set = {x * x for x in numbers}
print(square_set)  # {1, 4, 9, 16, 25}
```

76. How do you create a generator expression in Python?

```
python
Copy code
numbers = [1, 2, 3, 4, 5]
square_gen = (x * x for x in numbers)
for square in square_gen:
    print(square)
```

---

13. Regular Expressions

77. How do you use regular expressions in Python?

```
python
Copy code
import re

text = "The price is $100"
pattern = r"\d+"
result = re.findall(pattern, text)
```

```
print(result)  # ['100']
```

78.How do you match a string using a regular expression in Python?

```
python
Copy code
import re

text = "The price is $100"
pattern = r"\$[0-9]+"
match = re.search(pattern, text)
print(match.group())  # $100
```

79.How do you replace a part of a string using regular expressions in Python?

```
python
Copy code
import re

text = "The price is $100"
pattern = r"\$[0-9]+"
new_text = re.sub(pattern, "$200", text)
print(new_text)  # The price is $200
```

---

14. Concurrency and Parallelism

80.What is multithreading in Python?

```
python
Copy code
import threading

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

thread = threading.Thread(target=print_numbers)
thread.start()
thread.join()  # Wait for the thread to finish
```

81.What is multiprocessing in Python?

```
python
Copy code
from multiprocessing import Process

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

process = Process(target=print_numbers)
process.start()
process.join()  # Wait for the process to finish
```

82.How do you use `concurrent.futures` for parallelism in Python?

```
python
Copy code
from concurrent.futures import ThreadPoolExecutor
```

```
def print_numbers(i):
    print(i)

with ThreadPoolExecutor() as executor:
    executor.map(print_numbers, range(5))
```

---

15. Collections and Iterators

83.What is an iterator in Python?

```
python
Copy code
numbers = [1, 2, 3]
iterator = iter(numbers)
print(next(iterator))  # 1
print(next(iterator))  # 2
print(next(iterator))  # 3
```

84.What is a deque in Python?

```
python
Copy code
from collections import deque

queue = deque([1, 2, 3])
queue.append(4)
queue.appendleft(0)
print(queue)  # deque([0, 1, 2, 3, 4])
```

85.How do you remove an element from a deque?

```
python
Copy code
from collections import deque

queue = deque([1, 2, 3])
queue.pop()
queue.popleft()
print(queue)  # deque([2])
```

---

16. Sorting and Searching

86.How do you sort a list of dictionaries by a key in Python?

```
python
Copy code
people = [{'name': 'Alice', 'age': 25}, {'name': 'Bob', 'age': 30},
{'name': 'Charlie', 'age': 20}]
sorted_people = sorted(people, key=lambda x: x['age'])
print(sorted_people)  # [{'name': 'Charlie', 'age': 20}, {'name': 'Alice',
'age': 25}, {'name': 'Bob', 'age': 30}]
```

87.How do you perform binary search in Python?

```
python
```

```
Copy code
import bisect

numbers = [1, 2, 3, 4, 5]
index = bisect.bisect_left(numbers, 3)
print(index)  # 2
```

88. How do you sort a list in descending order in Python?

```
python
Copy code
numbers = [5, 3, 1, 4, 2]
numbers.sort(reverse=True)
print(numbers)  # [5, 4, 3, 2, 1]
```

---

17. Miscellaneous

89. How do you create a shallow copy of a list in Python?

```
python
Copy code
original = [1, 2, 3]
copy = original.copy()
```

90. What is a `set` in Python, and how do you perform set operations?

```
python
Copy code
set1 = {1, 2, 3}
set2 = {3, 4, 5}

print(set1 & set2)  # Intersection: {3}
print(set1 | set2)  # Union: {1, 2, 3, 4, 5}
print(set1 - set2)  # Difference: {1, 2}
```

91. How do you get unique elements from a list using a set in Python?

```
python
Copy code
numbers = [1, 2, 3, 3, 4, 5, 5]
unique_numbers = list(set(numbers))
print(unique_numbers)  # [1, 2, 3, 4, 5]
```

92. How do you merge two dictionaries in Python?

```
python
Copy code
dict1 = {'a': 1, 'b': 2}
dict2 = {'b': 3, 'c': 4}
merged = {**dict1, **dict2}
print(merged)  # {'a': 1, 'b': 3, 'c': 4}
```

93. What is the `zip()` function in Python?

```
python
Copy code
list1 = [1, 2, 3]
list2 = ['a', 'b', 'c']
```

```
zipped = zip(list1, list2)
print(list(zipped))  # [(1, 'a'), (2, 'b'), (3, 'c')]
```

94. How do you check for membership in a list in Python?

```
python
Copy code
my_list = [1, 2, 3, 4]
print(3 in my_list)  # True
print(5 in my_list)  # False
```

---

18. Advanced Data Structures

95. What is a heap in Python, and how do you use it?

```
python
Copy code
import heapq

heap = []
heapq.heappush(heap, 10)
heapq.heappush(heap, 5)
heapq.heappush(heap, 20)

print(heap)  # [5, 10, 20]
print(heapq.heappop(heap))  # 5
```

96. How do you create a priority queue in Python?

```
python
Copy code
import heapq

priority_queue = []
heapq.heappush(priority_queue, (2, 'task 2'))
heapq.heappush(priority_queue, (1, 'task 1'))
heapq.heappush(priority_queue, (3, 'task 3'))

print(heapq.heappop(priority_queue))  # (1, 'task 1')
```

97. What is a `defaultdict` in Python?

```
python
Copy code
from collections import defaultdict

dd = defaultdict(int)
dd['apple'] += 1
dd['banana'] += 2

print(dd)  # defaultdict(<class 'int'>, {'apple': 1, 'banana': 2})
```

98. What is a `Counter` in Python, and how do you use it?

```
python
Copy code
from collections import Counter

items = ['apple', 'banana', 'apple', 'orange', 'banana', 'banana']
```

```
counter = Counter(items)

print(counter)  # Counter({'banana': 3, 'apple': 2, 'orange': 1})
```

99. **How do you implement a stack using a list in Python?**

```
python
Copy code
stack = []
stack.append(1)  # Push
stack.append(2)  # Push
print(stack.pop())  # Pop: 2
print(stack)  # [1]
```

100. **How do you implement a queue using a deque in Python?** ```python from collections import deque

```
c
Copy code
queue = deque()

perl
Copy code
queue.append(1)  # Enqueue
queue.append(2)  # Enqueue
print(queue.popleft())  # Dequeue: 1
print(queue)  # deque([2])
```

---

19. Working with Dates and Times

101. **How do you get the current date and time in Python?** `

python from datetime import datetime

```
makefile
Copy code
now = datetime.now()

perl
Copy code
print(now)  # e.g., 2024-12-25 14:30:00.123456
```

102. **How do you format a date in Python?** ```python from datetime import datetime

```
perl
Copy code
now = datetime.now()
formatted_date = now.strftime("%Y-%m-%d %H:%M:%S")
print(formatted_date)  # e.g., 2024-12-25 14:30:00
```

103. **How do you parse a string into a date in Python?** ```python from datetime import datetime

```
perl
Copy code
date_str = "2024-12-25"
```

```
date_obj = datetime.strptime(date_str, "%Y-%m-%d")
print(date_obj)  # 2024-12-25 00:00:00
```

104. **How do you calculate the difference between two dates in Python?** ```python from datetime import datetime

```
scss
Copy code
date1 = datetime(2024, 12, 25)
date2 = datetime(2024, 12, 31)
difference = date2 - date1
print(difference.days)  # 6
```

105. **How do you add or subtract days from a date in Python?** ```python from datetime import datetime, timedelta

```
scss
Copy code
today = datetime.now()
tomorrow = today + timedelta(days=1)
yesterday = today - timedelta(days=1)

print(tomorrow)
print(yesterday)
```

---

20. Assertions and Testing

106. **What is the purpose of the `assert` keyword in Python?** `python x = 10 assert x == 10 # Passes assert x == 5 # Raises AssertionError`

107. **How do you write a simple unit test in Python?** ```python import unittest

```
css
Copy code
def add(a, b):

ruby
Copy code
    return a + b

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

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

108. **What is the `pytest` framework in Python?** ```python # In a file named test_sample.py def test_addition(): assert add(2, 3) == 5

```
perl
Copy code
# Run the test with the command: pytest test_sample.py
```

109. **How do you use `mock` to mock an object in Python testing?** ```python from unittest import mock

```
ruby
Copy code
def get_data():
    return "real data"

def fetch_data():
    return get_data()

# Mock the get_data function
with mock.patch('__main__.get_data', return_value="mocked data"):
    print(fetch_data())  # mocked data
```

---

21. Networking

110. **How do you send an HTTP GET request in Python?** ```python import requests

```
csharp
Copy code
response = requests.get("https://jsonplaceholder.typicode.com/posts")

perl
Copy code
print(response.text)  # The content of the response
```

111. **How do you send an HTTP POST request in Python?** ```python import requests

```
go
Copy code
data = {'name': 'Alice', 'age': 25}
response = requests.post("https://jsonplaceholder.typicode.com/posts",
json=data)
print(response.json())
```

112. **How do you handle exceptions when making HTTP requests?** ```python import requests

```
python
Copy code
try:
    response = requests.get("https://nonexistenturl.com")
    response.raise_for_status()
except requests.exceptions.RequestException as e:
    print(f"Error: {e}")
```

113. **How do you parse JSON data in Python?** ```python import json

```
scss
Copy code
json_str = '{"name": "Alice", "age": 25}'
data = json.loads(json_str)
```

```
print(data)  # {'name': 'Alice', 'age': 25}
```

114. **How do you convert a Python object to JSON?** ```python import json

```
go
Copy code
data = {'name': 'Alice', 'age': 25}
json_str = json.dumps(data)
print(json_str)  # {"name": "Alice", "age": 25}
```

---

22. Working with APIs

115. **How do you interact with REST APIs in Python?** ```python import requests

```
csharp
Copy code
response = requests.get("https://jsonplaceholder.typicode.com/posts")

perl
Copy code
print(response.json())  # Parse the response to JSON
```

116. **How do you use basic authentication with requests in Python?** ```python from requests.auth import HTTPBasicAuth

```
go
Copy code
response = requests.get("https://example.com", auth=HTTPBasicAuth('username',
'password'))
print(response.text)
```

117. **How do you handle timeouts in requests in Python?** ```python import requests

```
python
Copy code
try:
    response = requests.get("https://example.com", timeout=5)
    print(response.text)
except requests.exceptions.Timeout:
    print("The request timed out")
```

118. **How do you handle JSON data from a POST request in Python?** ```python import requests

```
perl
Copy code
data = {'name': 'Alice', 'age': 25}
response = requests.post("https://jsonplaceholder.typicode.com/posts",
json=data)
print(response.json())  # Parse the response JSON
```

---

23. Performance Optimization

119.**How do you measure the execution time of a Python function?** ```python import time

```
ruby
Copy code
def slow_function():

lua
Copy code
    time.sleep(2)

start_time = time.time()
slow_function()
end_time = time.time()

print(f"Execution time: {end_time - start_time} seconds")
```

120. **How do you profile a Python program to check its performance?** ```python import cProfile

```
scss
Copy code
def slow_function():
    time.sleep(2)

cProfile.run('slow_function()')
```

121. **How do you use memoization to optimize recursive functions in Python?** ```python from functools import lru_cache

```
python
Copy code
@lru_cache(maxsize=None)
def fibonacci(n):
    if n <= 1:
        return n
    return fibonacci(n-1) + fibonacci(n-2)

print(fibonacci(100))  # Optimized version of Fibonacci sequence
```

122. **How do you implement caching in Python using `functools.lru_cache`?** ```python from functools import lru_cache

```
python
Copy code
@lru_cache(maxsize=100)
def expensive_computation(x):
    return x * x  # Simulate an expensive operation

print(expensive_computation(5))  # Cached result
```

---

24. File Handling

123. **How do you read a file line by line in Python?** `python with open('sample.txt', 'r') as file: for line in file: print(line.strip())`

124. **How do you write to a file in Python?** `python with open('output.txt', 'w') as file: file.write('Hello, World!\n') file.write('Python is awesome!')`

125. **How do you append to a file in Python?** `python with open('output.txt', 'a') as file: file.write('\nAppending new content.')`

126. **How do you check if a file exists in Python?** ```python import os

```
lua
Copy code
if os.path.exists('sample.txt'):

go
Copy code
    print("File exists")
else:
    print("File does not exist")
```

127. **How do you handle file exceptions in Python?** `python try: with open('nonexistentfile.txt', 'r') as file: content = file.read() except FileNotFoundError: print("File not found")`

---

25. Context Managers

128. **What is a context manager in Python?** ```python class MyContextManager: def **enter**(self): print("Entering the context") return self

def **exit**(self, exc_type, exc_val, exc_tb): print("Exiting the context")

```
csharp
Copy code
with MyContextManager():

go
Copy code
    print("Inside the context")
```

129. **How do you use `contextlib` to create a context manager?** ```python from contextlib import contextmanager

```
less
Copy code
@contextmanager
def my_context():
    print("Entering the context")
    yield
```

```
    print("Exiting the context")

with my_context():
    print("Inside the context")
```

---

26. Memory Management

130. **How do you manually free up memory in Python?** ```python import gc

```
bash
Copy code
# To free up memory manually

perl
Copy code
del some_object
gc.collect()  # Force garbage collection
```

131. **What is the gc module used for in Python?** ```python import gc

```
perl
Copy code
# Manually trigger garbage collection
gc.collect()
```

132. **What is reference counting in Python?** - Python uses reference counting to manage memory. When an object's reference count drops to zero, it is garbage collected.

133. **What is the purpose of del in Python?** `python a = [1, 2, 3] del a # Deletes the reference to the list object`

---

27. Working with JSON

134. **How do you load JSON data from a file in Python?** ```python import json

```
python
Copy code
with open('data.json', 'r') as file:

scss
Copy code
    data = json.load(file)
    print(data)
```

135. **How do you write JSON data to a file in Python?** ```python import json

```
kotlin
Copy code
data = {'name': 'Alice', 'age': 30}

with open('output.json', 'w') as file:
```

```
    json.dump(data, file)
```

136. **How do you pretty-print JSON in Python?** ```python import json

```
go
Copy code
data = {'name': 'Alice', 'age': 30}
print(json.dumps(data, indent=4))
```

137. **How do you handle JSON decoding errors?** ```python import json

```
python
Copy code
invalid_json = '{"name": "Alice", age: 30}'
try:
    data = json.loads(invalid_json)
except json.JSONDecodeError as e:
    print(f"JSON decoding error: {e}")
```

---

28. Command Line Arguments

138. **How do you pass command-line arguments to a Python script?** ```python import sys

```
bash
Copy code
# Run as python script.py arg1 arg2

scss
Copy code
print(sys.argv)  # ['script.py', 'arg1', 'arg2']
```

139. **How do you parse command-line arguments with `argparse`?** ```python import argparse

```
python
Copy code
parser = argparse.ArgumentParser()
parser.add_argument('name', type=str, help='Your name')
args = parser.parse_args()

print(f"Hello, {args.name}")
```

140. **How do you set default values for command-line arguments?** ```python import argparse

```
go
Copy code
parser = argparse.ArgumentParser()
parser.add_argument('--age', type=int, default=30, help='Your age')
args = parser.parse_args()

print(f"Age: {args.age}")
```

---

29. Python 3.x Specific Features

141. **What is the `f-string` for string formatting in Python 3.6+?** `python name = 'Alice' age = 30 print(f"Name: {name}, Age: {age}")`

142. **How do you use type annotations in Python 3?** ```python def greet(name: str, age: int) -> str: return f"Hello {name}, you are {age} years old"

```
bash
Copy code
print(greet("Alice", 30))

go
Copy code
```

143. **What are the main differences between Python 2.x and Python 3.x?** - Python 3 uses `print()` as a function, while Python 2 uses it as a statement. - Python 3 has better Unicode support. - Integer division returns a float in Python 3.

---

30. Python Packaging and Distribution

144. **How do you create a virtual environment in Python?** `bash python -m venv myenv`

145. **How do you install packages from `requirements.txt` in Python?** `bash pip install -r requirements.txt`

146. **How do you create a `setup.py` file for a Python package?** ```python from setuptools import setup, find_packages

```
scss
Copy code
setup(

go
Copy code
    name='my_package',
    version='0.1',
    packages=find_packages(),
    install_requires=[
        'requests',
        'numpy'
    ],
)
```

147. **How do you upload a Python package to PyPI?** - Install `twine`: `bash pip install twine` - Build the package: `bash python setup.py sdist bdist_wheel` - Upload the package to PyPI: `bash twine upload dist/*`

---

31. Python in Web Development

148.**How do you create a simple web server with `Flask`?** ```python from flask import Flask

```
scss
Copy code
app = Flask(__name__)

less
Copy code
@app.route('/')
def hello_world():
    return 'Hello, World!'

if __name__ == '__main__':
    app.run(debug=True)
```

149. **How do you handle GET and POST requests with `Flask`?** ```python from flask import Flask, request

```
python
Copy code
app = Flask(__name__)

@app.route('/submit', methods=['POST'])
def submit():
    data = request.form['name']
    return f"Hello {data}"

if __name__ == '__main__':
    app.run(debug=True)
```

150. **How do you set up a RESTful API using `Flask`?** ```python from flask import Flask, jsonify

```
python
Copy code
app = Flask(__name__)

@app.route('/api/v1/resource', methods=['GET'])
def get_resource():
    return jsonify({"message": "Hello, API!"})

if __name__ == '__main__':
    app.run(debug=True)
```

---

32. Python Best Practices

151.**How do you ensure your Python code is readable and maintainable?** - Follow PEP 8 (Python's style guide). - Write modular code with functions and classes. - Use descriptive variable and function names.

152.**How do you use docstrings to document your code?** ```python def greet(name: str) -> str: """ Greet a person by their name.

Args: name (str): The name of the person.

Returns: str: A greeting message. """ return f"Hello, {name}!" ```

153. **What is the purpose of unit tests in Python?** - Unit tests verify that individual components of your code work correctly. - They help catch bugs early and ensure code changes don't break existing functionality.

---

33. Async Programming

154. **How do you create an asynchronous function in Python using `asyncio`?** ```python import asyncio

```
csharp
Copy code
async def hello():

scss
Copy code
    print("Hello")
    await asyncio.sleep(1)
    print("World")

asyncio.run(hello())
```

155. **How do you create and run multiple asynchronous tasks in Python?** ```python import asyncio

```
scss
Copy code
async def task1():
    await asyncio.sleep(1)
    print("Task 1")

async def task2():
    await asyncio.sleep(2)
    print("Task 2")

async def main():
    await asyncio.gather(task1(), task2())

asyncio.run(main())
```