1. What is a lambda function in Python, and how does it differ from a regular function?

**Ans.1**

* A lambda function is a small, anonymous function defined using the lambda keyword.
* It can have any number of arguments but can only have one expression.
* Lambda functions are often used for short, simple operations.

2. Can a lambda function in Python have multiple arguments? If yes, how can you define and use them?

**Ans.2**

* Yes, a lambda function in Python can have multiple arguments.
* You can define and use multiple arguments by separating them with commas.

Example of a Lambda Function with Multiple Arguments:

add = lambda x, y: x + y

result = add(3, 5) # Result is 8

3. How are lambda functions typically used in Python? Provide an example use case.

**Ans.3**

* Lambda functions are typically used when a small, simple function is needed for a short duration, such as in functional programming constructs like’ **map’,’ filter’**, and ‘**sorted’.**
* They are also used for on-the-fly operations, especially for sorting and filtering.

4. What are the advantages and limitations of lambda functions compared to regular functions in Python?

**Ans.4**

Advantages:

* + Concise and easy to define.
  + Useful for short, simple operations.
  + Don't require a separate def statement.
* Limitations:
  + Can only contain one expression.
  + Limited in terms of complexity compared to regular functions.
  + Can make the code less readable if overused.

5. Are lambda functions in Python able to access variables defined outside of their own scope? Explain with an example.

Ans.5

Lambda functions can access variables defined outside of their scope.

outside\_var = 10

func = lambda x: x + outside\_var

result = func(5)  **# Result is 15**

6. Write a lambda function to calculate the square of a given number.

**Ans.6**

**Lambda Function to Calculate the Square:**

square = lambda x: x \*\* 2

result = square(4) # Result is 16

7. Create a lambda function to find the maximum value in a list of integers.

**Ans.7**

**Lambda Function to Find Maximum Value in a List:**

numbers = [8, 2, 6, 12, 4]

max\_value = max(numbers, key=lambda x: x)

8. Implement a lambda function to filter out all the even numbers from a list of integers

**Ans.8**

**Lambda Function to Filter Even Numbers:**

numbers = [1, 2, 3, 4, 5, 6, 7, 8]

even\_numbers = list(filter(lambda x: x % 2 == 0, numbers))

9. Write a lambda function to sort a list of strings in ascending order based on the length of each string.

**Ans.9**

**Lambda Function to Sort Strings by Length:**

strings = ["apple", "banana", "cherry", "date", "fig"]

sorted\_strings = sorted(strings, key=lambda x: len(x))

10. Create a lambda function that takes two lists as input and returns a new list containing the common elements between the two lists.

**Ans.10**

**Lambda Function to Find Common Elements in Two Lists:**

list1 = [1, 2, 3, 4, 5]

list2 = [3, 4, 5, 6, 7]

common\_elements = list(filter(lambda x: x in list1, list2))

11. Write a recursive function to calculate the factorial of a given positive integer.

**Ans.11**

**Recursive Function for Factorial:**

**def factorial(n):**

**if n == 0:**

**return 1**

**else:**

**return n \* factorial(n - 1)**

12. Implement a recursive function to compute the nth Fibonacci number.

**Ans.12**

**Recursive Function for Fibonacci:**

def fibonacci(n):

if n <= 1:

return n

else:

return fibonacci(n - 1) + fibonacci(n - 2)

13. Create a recursive function to find the sum of all the elements in a given list.

**Ans.13**

**Recursive Function for Sum of List Elements:**

def list\_sum(1st):

if not 1st:

return 0

else:

return lst[0] + list\_sum(lst[1:])

14. Write a recursive function to determine whether a given string is a palindrome.

**Ans.14**

**Recursive Function to Check for Palindrome:**

def is\_palindrome(s):

s = s.lower().replace(" ", "")

if len(s) <= 1:

return True

elif s[0] != s[-1]:

return False

else:

return is\_palindrome(s[1:-1])

15. Implement a recursive function to find the greatest common divisor (GCD) of two positive integers.

**Ans.15**

**Recursive Function for GCD (Greatest Common Divisor):**

def gcd(a, b):

if b == 0:

return a

else:

return gcd(b, a % b)