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

**A: A lambda function is a small anonymous function. A lambda function can take any number of arguments, but can only have one expression.**

**Example:**

**Add 10 to argument a, and return the result:**

**x = lambda a : a + 10**

**print(x(5))**

**Output is 10**

**The lambda functions are different from regular functions like power of lambda is better shown when you use them as an anonymous function inside another function. you have a function definition that takes one argument, and that argument will be multiplied with an unknown number.**

**Example:**

**def myfunc(n):**

**return lambda a : a \* n**

**mydoubler = myfunc(2)**

**print(mydoubler(11))**

**Output is 22**

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

**A: The Lambda functions can have any number of arguments but they have only one expression. Firtsly, the expression is evaluated and then returned. We used Lambda functions wherever function objects are required.**

**Example:**

**def myfunc(n):**

**return lambda a : a \* n**

**mytripler = myfunc(3)**

**print(mytripler(11))**

**Output:**

**33**

**In the above code, lambda a: a \* n is the lambda function. Also, their ‘a’ is the argument and a \* n is the expression that gets evaluated and returned to the user.**

**This function has no name. It returns a function object which is assigned to the identifier double. we will now call it a standard function. The statement**

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

**A: We use lambda functions once we require a nameless function for a brief period of your time. In Python, we generally use Lambda Functions as an argument to a higher-order function a function that takes in other functions as arguments.**

**For Example: These are used together with built-in functions like len(), max(), and reduce(), etc.**

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

A: **Advantages of Lambda function:**

We’ll look at a few distinguishing qualities that distinguish Lambda functions from other types of functions.

* Ease of creating the function
* Variables with few options

## Limitations of Lambda function:

Let’s consider its drawbacks in the below section-

* Despite Python’s popularity as the world’s most popular programming language, the Lambda is not user-friendly.
* The Lambda function will be a problem if we need to build a complex function.
* Strictly limited to a single expression
* Although it cannot access a global variable, it can access the lone local variable.
* The simplicity of the Lambda function, which is a single statement, cannot be acceptable in all circumstances.
* For clarity, most Python functions and modules contain documentation, which is another drawback of the Lambda function.

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

**A: Lambda functions have their own local namespace and cannot access variables other than those in their parameter list and those in the global namespace.**

**Example:**

**sum = lambda num1, num2: num1 + num2;**

**# Now you can call sum as a function**

**print("value of total: ", sum(10, 20))**

**print("value of total: ", sum(20, 20))**

**input()**

**Output:**

**value of total: 30**

**value of total: 40**

**A Lambda function can take any number of arguments, but only have one expression**.

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

**A:**

**square = lambda x:x\*\*2**

**type(square)**

**square(5)**

**Output:**

**25**

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

**A:**

**f = lambda a,b: a if a>b else b**

**maximum = f(10, 14,)**

**print (maximum)**

**Output:**

**14**

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

**A:**

**numbers = [11, 22, 33, 44, 55, 66, 77, 88, 99, 100]**

**print("\nList of even numbers:")**

**evenNumbers = list(filter(lambda x: x%2 == 0, numbers))**

**print(evenNumbers)**

**Output:**

**List of even numbers:**

**[22, 44, 66, 88, 100]**

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

**A:**

**myList = ["rohan", "amy", "sapna", "muhammad", "aakash", "raunak", "chinmoy"]**

**sortedList = sorted(myList, key=lambda x: len(x))**

**print(sortedList)**

**Output:**

**['amy', 'rohan', 'sapna', 'aakash', 'raunak', 'chinmoy', 'muhammad']**

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

**A:**

**def intersection(lst1, lst2):**

**lst3 = [list(filter(lambda x: x in lst1, sublist)) for sublist in lst2]**

**return lst3**

**lst1 = [1, 6, 7, 10, 13, 28, 32, 41, 58, 63]**

**lst2 = [[13, 17, 18, 21, 32], [7, 11, 13, 14, 28], [1, 5, 6, 8, 15, 16]]**

**print(intersection(lst1, lst2))**

**Output:**

**[[13, 32], [7, 13, 28], [1, 6]]**

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

**A:**

**def recur\_factorial(n):**

**if n == 1:**

**return n**

**else:**

**return n\*recur\_factorial(n-1)**

**num = 8**

**# check if the number is negative**

**if num < 0:**

**print("Sorry, factorial does not exist for negative numbers")**

**elif num == 0:**

**print("The factorial of 0 is 1")**

**else:**

**print("The factorial of", num, "is", recur\_factorial(num))**

**Output:**

**The factorial of 8 is 40320**

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

**A:**

**def print\_fibonnaci(n):**

**a, b = 0, 1**

**for i in range(n):**

**print(a)**

**a, b = b, a + b**

**print\_fibonnaci(6)**

**Output:**

**0**

**1**

**1**

**2**

**3**

**5**

**def fibonnaci(n):**

**if n <=1:**

**return n**

**else:**

**return fibonnaci(n - 1) + fibonnaci(n - 2)**

**print(fibonnaci(6))**

**Output:**

**8**

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

**A:**

**def recursive\_sum(n):**

**if n == 1:**

**return 1**

**else:**

**return n + recursive\_sum(n - 1)**

**print(recursive\_sum(5)) # 5+4+3+2+1=15**

**Output:**

**15**

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

**A:**

**def is\_palindrome(l,r,S):**

**if l>=r:**

**return True**

**if S[l] != S[r]:**

**return False**

**return is\_palindrome(l+1,r-1,S)**

**my\_string = input('Enter your string: ')**

**l = 0**

**r = len(my\_string)-1**

**check = is\_palindrome(l,r,my\_string)**

**if check:**

**print(f"{my\_string} is palindrome")**

**else:**

**print(f"{my\_string} is not palindrome")**

**Output:**

**Enter your string: reffer**

**reffer is palindrome**

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

**A:**

**def gcd(a,b):**

**if a==b:**

**return a**

**elif a < b:**

**return gcd(b,a)**

**else:**

**return gcd(b, a - b)**

**a = 25**

**b = 45**

**print(gcd(a,b))**

**Output:**

5