1. **What is a lambda function in Python, and how does it differ from a regular function?**
   ANSWER: Lambda function is an anonymous function (i.e., defined without a name) that can take any number of arguments but, unlike normal functions, evaluates and returns only one expression. In regular functions we surround the parameters within parenthesis but in Lambda function we do not surround the parameters within parenthesis.

```
In [1]: # regular function
        def myfunc(x,y):
            return x+y

        myfunc(2,3)

Out[1]: 5

In [2]: # lambda function
        r=lambda x,y: x+y
        print(r(2,3))

        5
```

2. **Can a lambda function in Python have multiple arguments? If yes, how can you define and use them?**
   ANSWER: Python lambda can be used with multiple arguments and these arguments are used in evaluating an expression to return a single value.

```
In [30]: r=(lambda x,y,z:x+y+z)(10,20,30) #Lambda function with 3 arguments
         print(r)

         60
```

3. **How are lambda functions typically used in Python? Provide an example use case.**
   ANSWER: 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 filter(), map(), and reduce(), etc.
   Example:

```
In [32]: l1=[1,2,3,4]
         l1_square=map(lambda x: x*x, l1)
         print(list(l1_square))

         [1, 4, 9, 16]
```

4. **What are the advantages and limitations of lambda functions compared to regular functions in Python?**
   ANSWER: Advantages: They can be immediately passed around (no variable needed). They return automatically.

   Disadvantages: They can only have a single line of code within them. They can't have a docstring and they don't have a name.

5. **Are lambda functions in Python able to access variables defined outside of their own scope? Explain with an example.**
   ANSWER: Lambda functions have their own local namespace and cannot access variables other than those in their parameter list and those in the global namespace.

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

```
In [4]: # lambda function
        x=int(input("Enter the num"))
        r=lambda x: x**2
        print(r(x))

        Enter the num3
        9
```

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

```
In [8]: from functools import reduce

        l1=[3,6,5,8,2]
        r=reduce(lambda x,y: x if x>y else y, l1)
        print(r)

        8
```

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

```
In [9]: l1=[1,2,3,4,5,6,7,8,9,10]
        r=list(filter(lambda x: x%2==0, l1))
        print(r)

        [2, 4, 6, 8, 10]
```

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

ANSWER:

```
In [14]: l1=["Jerry", "Tom", "Hercules", "Jo", "Timothy"]
         r=sorted(l1, key=lambda x:len(x))
         print(r)

         ['Jo', 'Tom', 'Jerry', 'Timothy', 'Hercules']
```

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

ANSWER:

```
In [16]: import itertools

         def common_member(a, b):
             a_set = set(a)
             b_set = set(b)
             #Using filterfalse
             not_in_a = set(itertools.filterfalse(lambda x: x in a_set, b_set))
             not_in_b = set(itertools.filterfalse(lambda x: x in b_set, a_set))

             result = set(a_set).intersection(set(b_set) - not_in_a).union(set(b_set).intersection(set(a_set) - not_in_b))
             return list(result)
         # Example usage:

         a = [1, 2, 3, 4, 5]
         b = [3, 4, 5, 6, 7]
         print("The common elements in the two lists are: ")
         print(common_member(a, b))
         # Example usage:


         The common elements in the two lists are:
         [3, 4, 5]
```

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

ANSWER:

```
In [24]: def rec(n):
             if n<1:
                 return 1
             else:
                 return n*rec(n-1)

         rec(5)

Out[24]: 120
```

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

ANSWER:

```
In [26]: def Fibonacci(n):
             if n<= 0:
                 print("Incorrect input")
             # First Fibonacci number is 0
             elif n == 1:
                 return 0
             # Second Fibonacci number is 1
             elif n == 2:
                 return 1
             else:
                 return Fibonacci(n-1)+Fibonacci(n-2)


         print(Fibonacci(6))

         5
```

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

ANSWER:

```
In [28]: l1=[1,2,3,4,5]
         def sum(l1):
             a=0
             for i in l1:
                 a=a+i
             return a
         print(sum(l1))

         15
```

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

ANSWER:

```
In [32]: def myfunc(string):
             if string==string[::-1]:
                 return "string is a palindrome"
             else:
                 return "string is not a palindrome"

         myfunc("madam")

Out[32]: 'string is a palindrome'
```

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

ANSWER:

```
In [35]: def myfunc(n1,n2):
             gcd=1
             for i in range(1,min(n1,n2)):
                 if n1%i==0 and n2%i==0:
                     gcd=i
             print("Gcd of", n1, "and", n2, "is", gcd)

In [36]: myfunc(36,60)

         Gcd of 36 and 60 is 12
```