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

**Ans1.** lambda functions are anonymous function it means the function is without a name. lambda function is created with keyword lambda.

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

**Ans2.** Alambda function can take any number of arguments

addition= lambda x,y,z : x+y+z

print(addition(9,8,7))

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

**Ans3.** Lambda function are frequently used with higher-order functions which takes one or more functions as argument or return one or more functions.

sq= lambda x : x\*\*2

print(sq(5))

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

Ans4. Below are the advantages and limitations:

Advantages:

* We can say It’s a one line of code with less code
* Lambda keyword is used to create functions
* Can take one or more functions as arguments or return one or more functions as argument
* Easy to write code
* Lambda functions executes fast because of inline code

Limitations:

* Tuff to understand the code
* Lambda function can have only one expression

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

Ans5. Lambda functions can access global define variables and variable which are in lambda expression scope or we can say their own scope.

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

Ans6. Square=lambda x:x\*\*2

Square(2)

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

Ans7.

lst=[1,4,2,6,9,7]

f = lambda l: l[0] if len(l) == 1 else l[0] if l[0] > f(l[1:]) else f(l[1:])

max\_number = f(lst)

print(max\_number)

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

Ans8.

lst=[1,4,2,6,9,7]

result=filter(lambda x : x%2==0, lst)

print(list(result))

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

Ans9.

str\_list=['one','three','hundred','four']

str\_list.sort(key=lambda el: len(el))

print(str\_list)

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

Ans10.

list1=[1,2,3,4,5,6]

list2=[1,6,7,8]

result=list(filter(lambda x : x in list1, list2))

print(result)

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

Ans11.

def factorial(num):

if num == 1:

return num

else:

return num\*factorial(num-1)

factorial(4)

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

Ans12.

lst=[]

def fab\_series(a,b):

c=a+b

if c == 1:

lst.append(a)

lst.append(b)

lst.append(c)

fab\_series(b,c)

elif c < 50:

lst.append(c)

fab\_series(b,c)

else:

print(lst)

fab\_series(0,1)

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

Ans13.

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

def sum\_of\_list(lists):

if len(lists)==0:

return 0

else:

return lists[0] + sum\_of\_list(lists[1:])

print(sum\_of\_list(lst))

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

Ans14.

def is\_palindrome(str):

if len(str) <1:

return True

else:

if str[0]==str[-1]:

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

else:

return False

print(is\_palindrome('radar'))

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

Ans15.

def gcd(a, b):

if a == b:

return a

elif a < b:

return gcd(b, a)

else:

return gcd(b, a - b)

print(gcd(15, 25))