1.What are lambda functions in Python? How are they different from regular functions?

Lambda functions are known as anonymous functions or single-expression functions.

Lambda functions	Regular functions
• They are defined using lambda keyword followed by function arguments and colon, all in single line	• They are defined using the def keyword, followed by the function name, parentheses for arguments, a colon, and an indented block of code.
• They are single expression function to perform single task	• They can perform multiple statement which can perform complex task.

add = lambda val1,val2: val1+val2
val1 = int(input("Enter the value1: "))
val2 = int(input("Enter the value2: "))
print('The sum of two numbers are ',add(val1,val2))
	def add(val1,val2):
	  sumOfNumbers = val1+val2
	  return sumOfNumbers
	val1 = int(input("Enter the value 1"))
	val2 = int(input("Enter the value 2"))
	addition = add(val1,val2)
	print(f'Sum of two numbers are \ {addition}')

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

Yes, a lambda function can have multiple arguments in python. To define and use it in python we simply define commas in between the arguments and below will be the following example.

Snippet:

```
multiplication = lambda x,y: x*y
x = int(input('Enter the value for x: '))
y = int(input('Enter the value for y: '))
print(multiplication(x,y))
```

Output:

```
Enter the value for x: 10
Enter the value for y: 20
The multiplication of two numbers will be:  200
```

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

Lambda functions in python are typically used for the simple functions that we use only once without defining a separate 'def' statement. They are small functions without a name. They are also known as anonymous functions.

1. Use case

Sorting a list of tuples based on specific element

Snippet:

```
my_list = [('surya',23),('darshini',24),('vicky',23),('abhi',23),('aishu',24)]
my_list.sort(key = lambda x:x[1])
print(my_list)
```

Output:

```
[('surya', 23), ('vicky', 23), ('abhi', 23), ('darshini', 24), ('aishu', 24)]
```

2. Use case

Sorting a list of dictionaries based on specific element

Snippet:

```
my_dict = [{'name':'surya','age':23},{'name':'dood','age':24},{'name':'aishu','age':25},
{'name':'abhi','age':23}]
my_dict.sort(key=lambda x:x['age'])
print(my_dict)
```

Output:

```
[{'name': 'surya', 'age': 23}, {'name': 'abhi', 'age': 23}, {'name': 'dood', 'age': 24},
{'name': 'aishu', 'age': 25}]
```

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

ADVANTAGES OF LAMBDA

1.Reduced line of code

2.No additional variables are added.

3.Easy to execute.

4.Avoids ambiguity.

LIMITATIONS:

1.Difficult to understand and unfamiliar.

2. Lambda expressions can only contain one statement, so some readable language features, such as tuple unpacking, cannot be used with them.

3.Can’t perform high end operations.

5. Are lambda functions in Python able to access variables defined outside of their own scope? Explain with an example. Yes, Lambda functions in python can access variables that are defined outdise of their own scope. It can access variables even after the scope is ended. This is known as lexical scoping.

Snippet

```
def accessing variable():
    print_var=lambda name: f"Welcome, (name)!"
    return print_var
name = input('Enter the name: ")
```

```
greeting accessing_variable()

print(greeting(name))
```

Output:

```
Enter the name: surya

Welcome, surya!
```

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

Snippet:

```
sqr=lambda x:x*x
x=int(input("enter the number:"))
print(f'Square of {x} is: ',sqr(x))
```

Output:

```
enter the number:2

Square of 2 is:  4
```

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

Method 1:

Snippet:

```
my_list =[]
val=int(input("enter the no of values:"))
print('Enter the elements: ')
for i in range(val):
   elements = int(input())
   my_list.append(elements)
print(my_list)
maximum_numbers = max(map(lambda x:x,my_list))
print('The maximum number in the list is',maximum_numbers)
```

Output:

```
Enter the elements: 10
20
30
40
50
[10, 20, 0, 40, 50]
The maximum number in the list is 50
```

Method 2:

Snippet:

```
my_list =[]
print('Enter the elements: ')
for i in range(5):
   elements = int(input())
```

```
    my_list.append(elements)
print(my_list)
maximum_numbers = max(map(lambda x:x,my_list))
print('The maximum number in the list is',maximum_numbers)
```

Output:

```
Enter the elements: 10
20
30
40
50
[10, 20, 0, 40, 50]
The maximum number in the list is 50
```

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

Snippet:

```
my_list =[]
val=int(input("enter the no of values:"))
print('Enter the elements: ')
for i in range(val):
   elements = int(input())
   my_list.append(elements)
print('The elements in the list are: ',my_list)
even_num = list(filter(lambda x:(x%2==0),my_list))
print('The even number in the list are: ',(even_num))
```

Output:

```
enter the no of values:5
Enter the elements:
1
2
3
4
5
The elements in the list are:  [1, 2, 3, 4, 5]
The even number in the list are: [2, 4]
```

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

Snippet:

```
names =[]
val=int(input("enter the no of employees:"))
for i in range(val):
   elements = input(f'enter the {i+1} name of the employee:')
   names.append(elements)
sortedList = sorted(names,key=lambda x: len(x))
print('The elements in the list are: ',sortedList)
```

Output:

```
enter the no of employees:4
enter the 1 name of the employee:surya
enter the 2 name of the employee:abhisheik
enter the 3 name of the employee:aiyswarya
```

```
enter the 4 name of the employee:darshini
The elements in the list are:  ['surya', 'darshini', 'abhisheik', 'aiyswarya']
```

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

Snippet:

```
names =[]
names1=[]
val=int(input("enter the no of employees:"))
for i in range(val):
   elements = input(f'enter the {i+1} name of the employee:')
   names.append(elements)
for j in range(val):
   elements = input(f'enter the {j+1} name of the employee:')
   names1.append(elements)
commonelements = list(filter(lambda x: x in names1, names))
print('The common elements in list are',commonelements)
```

Output:

```
enter the no of employees:3
enter the 1 name of the employee:surya
enter the 2 name of the employee:prakash
enter the 3 name of the employee:darshini
enter the 1 name of the employee:abhi
enter the 2 name of the employee:darshini
senter the 3 name of the employee:surya
The common elements in list are ['surya', 'darshini']
```

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

Snippet:

```
def fact(num):
        if num==0 or num ==1:
           return 1
        else:
           return num * fact(num-1)
num = int(input('Enter the number: '))
result = fact(num)
print(f"The factorial of {num} is {result}")
```

Output:

```
Enter the number: 5
The factorial of 5 is 120
```

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

Snippet:

```
def fib(n):
   if n<=1:
      return n
   else:
```

```
        return fib(n-1)+fib(n-2)
num = int(input('Enter the number:'))
for i in range(num):
   print(fib(i))
```

Output:

```
Enter the number:5
0
1
1
2
3
```

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

Snippet:

```
num=[]
val=int(input("enter no.of elements: "))
for i in range(val):
   element = int(input(f"enter the {i+1} element:"))
   num.append(element)
def sumOfList(num):
   return num[0] + sumOfList(num[1:]) if num else 0
print(f'The sum of numbers in the {num} are:',sumOfList(num))
```

Output:

```
enter no.of elements: 5
enter the 1 element:1
enter the 2 element:2
enter the 3 element:3
enter the 4 element:4
enter the 5 element:5
The sum of numbers in the [1, 2, 3, 4, 5] are: 15
```

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

1. With recursive function

Snippet:

```
def palindrome(word):
   if len(word)<=1:
      return True
   elif word[0]!=word[-1]:
      return False
   else:
      return palindrome(word[1:-1])
word = input('Enter the string: ')
if(palindrome(word)):
   print(f'{word} is a palindrome')
else:
   print(word + ' is not a palindrome')
```

2. Without recursive function

Snippet:

```
def palindrome(word):
    if word[0] == word[-1] or word[0] == word[-1] and word[1:-1]:
        print(f'{word} is a palindrome')
    else:
        print(word + ' is not a palindrome')
word = input('Enter the string: ')
palindrome(word)
```

Output:

```
Enter the string: mom
mom is a palindrome
```

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

Snippet:

```
def gcd(num1,num2):
    if num2==0:
        return num1
    else:
        return gcd(num2,num1%num2)
num1 = int(input('Enter the number1: '))
num2 = int(input('Enter the number2: '))
result = gcd(num1,num2)
print('The gcd of two numbers: ',result)
```

Output:

```
Enter the number1: 24
Enter the number2: 36
The gcd of two numbers:  12
```