Functions

1. What is the difference between a function and a method in Python?
    - A function in Python is a block of reusable code that is independent and can be called anywhere in the program. It is defined using the `def` keyword.
    - A method is a function that is associated with an object and belongs to a class. It is called on an instance of the class and can modify the object's attributes.
    - Key difference: Methods are functions within a class, and they always have at least one parameter (`self`) that refers to the instance, whereas functions are standalone.

2. Explain the concept of function arguments and parameters in Python?
    - Parameter: These are variable declared inside function parenthesis when function is created
      Example:
      def add(x,y): #here x,y are parameter
      return x+y
      add(4,5) output 9
    - Arguments: When values are passed/assigned during function fall, it is called argument.
      Example: def add(x,y):
      return x+y
      add(4,5) # here 4 and 5 are arguments, values passed to the parameter x=4,y=5

3. What are the different ways to define and call a function in Python?
      Function are block of organized, reusable code designed to perform specific task. There are different ways to define and call a function:
    - Regular function definition: "def" keyword is used to defined function
      Example:
      def greet():
      print("hello world")
      greet # hello world
    - Function with default arguments: You can specify default value in function parameter. When function is called and if you omit the value then the default value will be used automatically.
      Example:

```
def add(a,b=4):
return a+b
add(5) # output will be 9 where a=5 and for b the default value will be used b=4
```

- Lambda function: It is small anonymous function defined using lambda keyword. It can take any number arguments but can only have one expression.
  Example:
  ```
  sqr=lambda x:x**2
  sqr(5) # output will be 25
  ```
- Recursive function: A function can call itself, which is known as recursion. This is useful for solving problems that can be broken down into smaller, self-similar sub problems.
  Example:
  ```
  def factorial(n):
    if n == 0:
      return 1
    else:
      return n * factorial(n – 1)

  result = factorial(5)
  print(result) # output: 120
  ```
- Nested function: Functions can be defined inside other functions. The inner function has access to the outer function's variables (closure). To call a nested function, it must be called within the scope of the outer function.
  Example:
  ```
  def outer_function(x):
    def inner_function(y):
      return x + y
    return inner_function

  add_five = outer_function(5)
  result = add_five(3)
  print(result) # Output: 8
  ```
- Functions with Variable Arguments (*args and **kwargs)
  - *args: Used to pass a variable number of non-keyword arguments to a function. These arguments are received as a tuple.

- **kwargs: Used to pass a variable number of keyword arguments to a function. These arguments are received as a dictionary.

Example:

```
def print_arguments(*args, **kwargs):
  for arg in args:
    print("arg:", arg)
  for key, value in kwargs.items():
    print("key:", key, "value:", value)

print_arguments(1, 2, 3, name="Alice", age=30)
# Output:
# arg: 1
# arg: 2
# arg: 3
# key: name value: Alice
# key: age value: 30
```

4. What is the purpose of the `return` statement in a Python function?

The return statement in a Python function serves two primary purposes:

- Exiting the function: When a return statement is encountered, the execution of the function immediately stops, and control is transferred back to the caller. Any code following the return statement within the function is not executed.
- Returning a value: The return statement can optionally return a value to the caller. This value can be of any data type, including numbers, strings, lists, tuples, dictionaries, or even other functions. If no value is specified, the function returns None by default.

5. What are iterators in Python and how do they differ from iterables?

- An iterator in Python is an object that implements the __iter__() and __next__() methods, allowing sequential access to elements. It remembers its state and fetches the next item using next().
- An iterable is any object that can return an iterator, meaning it has an __iter__() method. Common iterables include lists, tuples, dictionaries, and sets.
- Key Difference:

- Iterables: Can be looped over but don't store iteration state (e.g., lists, tuples).
- Iterators: Produce elements one at a time and maintain state, using next() until exhaustion.

  Example:

```
my_list = [1, 2, 3]
iterator = iter(my_list)  # Converting iterable to iterator

print(next(iterator))  # Outputs: 1
print(next(iterator))  # Outputs: 2
```

6. Explain the concept of generators in Python and how they are defined?

- It is a special function that returns an iterator and produces value once at a time using 'yield' keyword. Unlike general function, it pauses after each yield and resume from same point when called again.
  Example: if we want square upto 1 crore then it will take memory and time. But using generator function can help me saving time, memory and verifying code also.

```
def sqr(n):
for i in range(n):
yield i**2
s=sqr(3)
next(s) # output 1: 1
next(s) # output 2: 4
next(s) # output 3: 9
```

7. What are the advantages of using generators over regular functions?

- Generators offer several advantages over regular functions, primarily concerning memory efficiency and lazy evaluation.
- Unlike regular functions that return a complete result immediately, generators yield values one at a time, saving memory, especially when dealing with large datasets or infinite sequences.
- They also allow for lazy evaluation, computing values only when needed, potentially improving performance.

8. What is a lambda function in Python and when is it typically used?

- A lambda function in Python is an anonymous, single-expression function created using the lambda keyword.
- Unlike regular functions defined with def, lambda functions don't require a name and are often used for short, simple operations where defining a separate function would be unnecessary.
- They are commonly used in cases like sorting lists using custom criteria, filtering data, or applying transformations within higher-order functions like map(), filter(), and reduce().
- Since lambda functions are concise and can be defined inline, they are particularly useful for scenarios requiring quick, temporary functionality without needing a full function definition.

9. Explain the purpose and usage of the `map()` function in Python?
    - The map() function in Python is used to apply a function to every item in an iterable (like a list or tuple) and return an iterator with the transformed values. It helps streamline operations that require modifying multiple elements efficiently.
    - Purpose of map()
        - Applies a function to each element of an iterable.
        - Improves performance by processing elements lazily (returns an iterator).
        - Reduces the need for loops, making code more readable and concise.
    - Example: map(function, iterable)
      lst=[1,2,3,4,5]
      list(map(lambda x:x**2, lst))
      output: [1,4,9,16,25]

10. What is the difference between `map()`, `reduce()`, and `filter()` functions in Python?
    - The map(), reduce(), and filter() functions are built-in higher-order functions in Python that operate on iterables. They each serve different purposes:
    - Map():Applies a given function to each item in an iterable (e.g., a list) and returns an iterator that yields the results.

- Filter():Filters elements from an iterable based on a given function (predicate) that returns True or False. It returns an iterator containing only the elements for which the function returns True.
- Reduce():Applies a function cumulatively to the items of an iterable, from left to right, to reduce the iterable to a single value. It is available in the functools module

11. Using pen & Paper write the internal mechanism for sum operation using reduce function on this given List:[47,11,42,13]