Functions

Note: For each theory Question, give at least one example.

1. What is the difference between a function and a method in Python?

    - A function is a standalone block of code defined independently. It performs a specific task and can be called directly by its name. e.g.

        To define a function
        Define <<Name of Function>> (Name)
        Return “Hello” Name

        To recall a function
        <<Name of Function>> (Enter <<Name>>)

    A method is a function that is associated with an object (specifically, an instance of a class). It operates on the data and attributes of that object and is called using dot notation on the object.

```
class Dog:
def __init__(self, name, breed, age, gender):
self.name = name
self.breed = breed
self.age = age
self.gender = gender

def bark(self):
print(f"{self.name} is barking!")

def show_info(self):
print(f"{self.breed}, {self.age}, {self.gender}, {self.name}")
```

---

2. Explain the concept of function arguments and parameters in Python.

    - Parameters are variables defined within the parentheses in a function definition, serving as placeholders for values the function will receive. Arguments are the actual values passed to a function when it's called, providing the data the function operates on.

```
def add(a,b):
        a and b are parameters
  add(7,63)
        7,63 are arguments
```

---

3. What are the different ways to define and call a function in Python?

    - Function can be defined by using Key word “def” followed by the function name, parentheses for parameters, a colon, and an indented block of code for the function body.
      A function is called by using the function name followed by parentheses, passing arguments if needed.

      To define a function

```
def greet(name):
    return "Hello, " + name
```

      To call a function

```
message = greet("World")
print(message)
```

---

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

    - The return statement in a Python function serves two primary purposes: to terminate the execution of the function and to send a value back

```
def greet(name):
    return "Hello, " + name
```

---

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

    - An iterator is an object representing a stream of data which return the data one by one.

```
for i in name :
    print(i)
```

      Python decided that name should be iterated through the concept of iterator
      An iterable is any python object/sequential structure/data structure that is capable of returning
      its members one at a time permitting it to be iterated over in a for loop.

      name = "Nilesh" string is iterable

---

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

    - A Python generator function allows you to declare a function that behaves like an iterator, providing a faster and easier way to create iterators. They can be used on an abstract container of data to turn it into an iterable object like lists, dictionaries and strings.
      A generator is defined as a function that returns an iterator that produces a sequence of values when Iterated over

      Example of a generator

```
def square_number_generators(n):
```

```
for i in range(n):
    yield i**2
```

---

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

   - Given below are advantages of generators over regular functions
     a. Memory efficiency
     b. Improves preformation as it computes only if required.
     c. Can be paused and resumed to do multitasking

---

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

   - A lambda function is a small, anonymous function that can take any number of arguments but can only have one expression. It's essentially a shorthand way to define a function
     add = lambda a, b: a+b (a, b are inputs and a+b defines the function to be done)

```
add(2, 5)
output = 7
```

---

9. Explain the purpose and usage of the `map()` function in Python.

   - A map function executes a specified function for each of item of an iterable. The map function serves to apply a given function to each item of an iterable and returns an iterator that yields the results. It avoids the need for explicit loops when performing element-wise operations.

```
A function is defined
    def square(x):
        return x ** 2

    numbers = [1, 2, 3, 4, 5]
    squared_numbers = list(map(square, numbers))
Here the map uses the defined function “square” and this performed for all
iterable of list “numbers”
```

---

10. What is the difference between `map()`, `reduce()`, and `filter()` functions in Python?

    - map - Applies a given function to each item in an iterable and returns a new iterable (map object) containing the results. It transforms each element individually.

```
def square(x):
    return x ** 2
numbers = [1, 2, 3, 4, 5]
squared_numbers = list(map(square, numbers))
Output = [1,4,9,16,25]
```

reduce - Applies a function cumulatively to the items of an iterable, from left to right, to reduce the iterable to a single value.

l = [2, 1, 3, 4, 4, 5, 6]
reduce(lambda x, y:x+y, l)
Output 25
In this first 2 numbers (x and y) are added and the sum becomes first number (x). This continues till all are added,

filter - Creates a new iterable (filter object) containing only the items from the original iterable for which a given function returns True. It selectively keeps elements based on a condition.

A = [2, 1, 3, 4, 4, 5, 6]
list(filter(lambda x: x%2 == 0, l))
output = [2,4,4,6]
In this the even numbers are filtered from the list of numbers.

---

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

(Attach paper image for this answer) in doc or colab notebook.

- list = [47,11,42,13]
  reduce (lambda x, y : x+ y, list)
  Output = 113