- # Theory Ouestion

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

function-

A function is a block of reusable code that is dehned using def and is not bound to an object.

It can be called independently

Example :-def greet(name): return f"HeIlo, (name)!"

print(greet("Alice"))

Method- A method is a function that is defined inside a class and operates on instances of that class (objects).

It is bound to the object, and often takes self as its first parameter to access the instance

Example :-class Greeter: def greet(self, name): return f"HeIlo, (name)!"

g = Greeter() print(g.greet("Alice"))

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

-Parameter :-A variable in a function definition. It acts as a placeholder for the values that will be passed into the function.

-Argument :-The actual value you pass into a function when calling it .

Type of agruments or parameter -1. Positional Parameters

2. Keyword Parameters
3. Default Parameters
4. Arbitrary Positional Arguments (*args) -Used when you're not sure how many arguments you'll receive
5. Arbitrary Keyword Arguments (**kwargs) -Used to accept any number of keyword arguments

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

-In Python, functions can be defined and called in several ways depending on how flexible or structured you want your code to be. Below are the main ways to define and call functions, including some advanced options.

1. Basic Function Dehnition and Call

-def greet(): print("Hello!")

2. Function with Parameters

-def greet(name): print(f"HeIlo, (name)!")

3. Function with Default Parameters

-def greet(name, message="Hi"): print(f"(message}, (name)!")

4. Function with Arbitrary Positional Arguments (*args) -

-def add(*numbers): return sum(numbers)

5. Lambda (Anonymous) Functions

-square = lambda x: x * x

4) What is the purpose of the re u n statement in a Python function?

-The return statement in a Python function is used to send a result back to the caller and exit the function

1. When a function finishes executing, return lets it hand back a value

def add(a, b):

```
return a + b
```

result = add(2, 3) # result is now 5

2. End function execution

return also immediately ends the function. Any code after it won't run:

def example():

```
return "Done"
print("This will never be printed")
```

3. Return multiple values (as a tuple)

Python functions can return more than one value using commas, and Python packs them into a tuple

def stats(numbers):

```
return min(numbers), max(numbers), sum(numbers)
```

low, high, total = stats([1, 2, 3])

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

-Iterators : An iterator is an object that produces one item at a time from an iterable on demand, using the next() method. It remembers its state between items.

You get an iterator from an iterable by calling iter() on it

example :-my list = [10, 20, 30] it = iter(my list) # it is an iterator

print(next(it)) # 10 print(next(it)) # 20 print(next(it)) # 30

Think of an iterable as a book and an iterator as a bookmark:

The book (iterable) contains all the pages (data).

The bookmark (iterator) lets you keep track of which page you're on.

You can create new bookmarks (iterators) to start over

An iterable is like a container you can loop through, while an iterator is the tool that actually performs the looping, one item at a time

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

-Generator:

A generator is a special type of iterator that yields values one at a time, only when requested.

Instead of returning all values at once like a list, a generator pauses after each value is produced.

It remembers its state, so it can resume where it left off

They are defined (using yield)

A generator function looks like a normal function, but it uses the yield keyword instead of return.

Example :-def count_up_to(max):

```
count  1
while count <= max:
    yield count
    count += 1
```

calling the function

gen = count up to(3):

print(next(gen)) # 1

print(next(gen)) # 2

print(next(gen)) # 3

7) What are the advantages of using generators over regular functions

-ADVANTAGES :-

1. Memory Efficiency
2. Lazy Evaluation
3. Improved Performance
4. Infinite Sequences Support
5. Cleaner Code for Iteration

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

- Lambda Function

A lambda function is a small, anonymous function defined with the lambda keyword.

It can take any number of arguments but only one expression.

The expression is evaluated and returned automatically.

Lambda functions do not have a name unless you assign them to a variable

Syntax : lambda arguments: expression

Example :add = lambda x, y: x + y

print(add(3, 5)) # Output: 8

when it is used :

Short, throwaway functions used temporarily and defined inline.

Often passed as arguments to higher-order functions like map(), filter(), and sorted().

When you want to avoid the verbosity of defining a full function for a simple operation.

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

- map() Function

The map() function applies a given function to each item of an iterable (like a list, tuple, etc.) and returns an iterator with the results.

-Purpose

To transform all elements in an iterable by applying the same operation.

It's a concise way to replace explicit loops for element-wise operations

-Syntax map(function, iterable, ...)

function: A function (or lambda) to apply to each element.

iterable: One or more iterable(s) whose elements are passed to the function

1O)What is the difference between map(, edu e , and 1 e functions in Python?

-1. map(function, iterable, ...)

Transforms every element by applying a function.

Returns an iterator of the same length as the input.

Example :nums = [1, 2, 3]

squared = map(lambda x: x**2, nums)

print(list(squared)) # [1, 4, 9]

-2. filter(function, iterable)

Filters elements for which the function returns True.

Returns an iterator with only the elements that pass the test.

Example:nums = [1, 2, 3, 4, 5]

evens = filter(lambda x: x % 2 == 0, nums)

print(list(evens)) # [2, 4]

-3. reduce(function, iterable[, initializer])

Reduces the iterable to a single cumulative value by applying a binary function repeatedly.

Requires import: from functools import reduce

Example :from functools import reduce

nums = [1, 2, 3, 4]

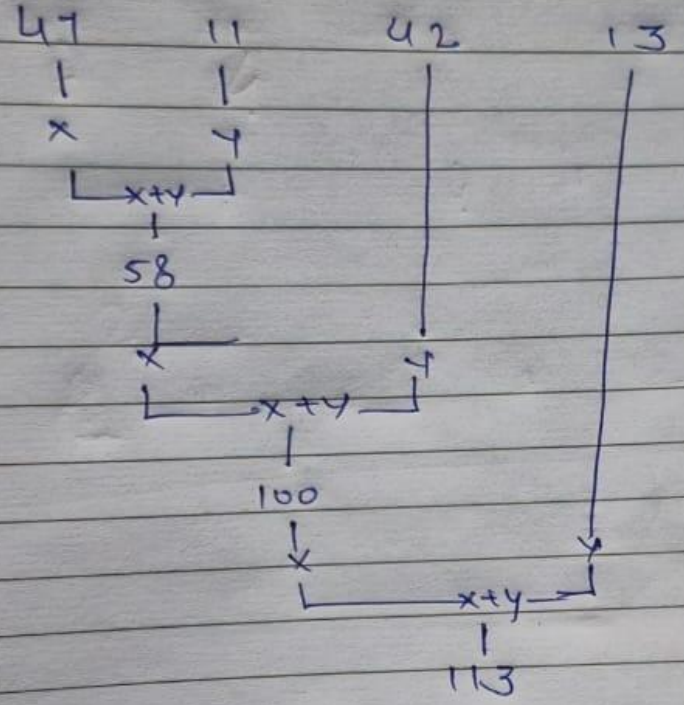
sum all = reduce(lambda x, y: x + y, nums)

print(sum_aII) # 10

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

Q-11 Internal mechanism for sum operation using reduce function

test : [47, 11, 42, 13]:



```
#pratical question**


#1) Write a Python function that takes a list of numbers as input and returns the
def sum_even_numbers(numbers):
    total = 0
    for num in numbers:
```

```
            if num % 2 == 0:
                total += num
        return total
```

```
nums =[1,2,3,4,5,6]
print(sum_even_numbers(nums))
```

```
12
```

```
#2)Create a Python function that accepts a string and returns the reverse of that

def reverse_string(text):
    return text[::-1]
```

```
print(reverse_string(“hello"))
print(reverse_string(“Python“))
```

```
olleh
nohtyP
```

```
#3. Implement a Python function that takes a list of integers and returns a new li
def square_list(numbers):
    squared = []
    for num in numbers:
        squared.append(num ** 2)
    return squared
```

```
nums = [1, 2, 3, 4, 5]
print(square_list(nums))
```

```
[1,  4,  9,  16,  25]
```

```
#4. Write a Python function that checks if a given number is prime or not from 1 ‹
def is_prime(n):
    if n < 2 or n > 200:
        return False  # Out of range or not prime
    for i in range(2, int(n ** 0.5) + 1):
        if n % i == 0:
            return False
    return True
```

```
print(is_prime(2))
print(is_prime(11))
print(is_prime(100))
print(is_prime(201))
```

```
True
True
False
False
```

```
#5. Create an iterator class in Python that generates the Fibonacci sequence up t‹
def fib(n):
    a   0
    b = 1
    for i in range(n): #if n=5, 0, 1, 2, 3, 4
        yield a
        a, b = b, a+b
```

```
f =fib(1000)
```

```
f
```

```
<generator object fib at 0x7c0bf4160040>
```

```
next(f)
```

```
0
```

```
next(f)
```

```
1
```

```
next(f)
```

```
1
```

```
next(f)
```

```
2
```

```
next(f)
```

```
3
```

```
next(f)
```

```
5
```

```
#6. Write a generator function in Python that yields the powers of 2 up to a giver
def power_of_two(n):
    for i in range(n + 1):
        yield 2 ** i
```

```
power =power_of_two(5)
```

```
power
```

```
<generator object power_of_tWo at 0x7c0bf4l7a5a0›
```

```
next(power)
```

```
1
```

```
next(power)
```

```
2
```

```
next(power)
```

```
4
```

```
next(power)
```

```
8
```

```
next(power)
```

```
16
```

```
next(power)
```

```
32
```

```
#7. Implement a generator function that reads a file line by line and yields each
def read_lines(filename):
    with open(filename, 'r') as file:
        for line in file:
            yield line.strip()
```

```
#8. Use a lambda function in Python to sort a list of tuples based on the second ‹
data = [(1, 5), (3, 2), (2, 8), (4, 1)]
sorted_data = sorted(data, key=lambda x: x[1])
print(sorted_data)
```

```
[(4, 1), (3, 2), (1, 5), (2, 8)]
```

```
#9. Write a Python program that uses 'map()' to convert a list of temperatures frs
celsius_temps = [0, 10, 20, 30, 40]
fahrenheit_temps = list(map(lambda c: (c * 9/5) + 32, celsius_temps))
```

```
print(“celsius:  “,celsius_temps)
print(“fahrenheit:  ”,fahrenheit_temps)
```

```
celsius:   [0, 10, 20, 30, 40]
fahrenheit:   [32.0, 50.0, 68.0, 86.0, 104.0]
```

```
#10. Create a Python program that uses filter() to remove all the vowels from a gi
def remove_vowels(string):
    vowels = 'aeiouAEIOU'
    return ''.join(filter(lambda x: x not in vowels, string))
```

```
input_string = “Hello, Python is awesome!”
result = remove_vowels(input_string)

print(“Original string:“, input_string)
print(“Without vowels: “, result)
```

```
Original string: Hello, Python is awesome!
Without vowels:  Hll, Pythn s wsm!
```

```
"'11) Imagine an accounting routine used in a book shop. It works on a list with
Write a Python program, which returns a list with 2-tuples. Each tuple consists o4
product of the price per item and the quantity. The product should be increased b)
```

```
order is smaller than 100,00 €.

Write a Python program using lambda and map."'

orders   [
    [34587, "Learning Python, Mark Lutz”, 4, 40.95],
    [98762, "Programming Python, Mark Lutz”, 5, 56.80],
    [77226, "Head First Python, Paul Barry”, 3, 32.95],
    [88112, "Einführung in Python3, Bernd Klein“, 3, 24.99]


# Use map and lambda
invoice = list(map(lambda order: (order[0], order[2] * order[3] if order[2] * ords

print(invoice)
```

```
[(34587, 163.8), (98762, 284.0), (77226, 108.85000000000001), (88112, 84.97)]
```

Start coding or generate with AI.