Theory Questions:

Question 1. Difference between a function and a method in Python

Answer : A function is a block of reusable code that is defined using def and can be called independently.

A method is a function that is associated with an object and is called on that object.

Example:

```
def func():  # Function
    return "Hello"

class MyClass:
    def method(self):  # Method
        return "Hello from method"
```

Question 2. Function arguments and parameters in Python

Answer :Parameters are variables listed in the function definition.

Arguments are values passed when calling the function.

Example:

```
def add(a, b):  # a and b are parameters
    return a + b

result = add(5, 3)  # 5 and 3 are arguments
```

Question 3. Ways to define and call a function in Python

Answer :Using def, lambda functions, and function objects.

Example:

```
def greet():
```

```
    return "Hello"
print(greet())
```

Question 4. Purpose of the return statement

Answer :It is used to return a value from a function.

Example:

```
def square(n):
    return n * n
print(square(4))  # Output: 16
```

Question 5. Iterators vs. Iterables

Answer :An iterable is an object that can return an iterator (like lists, tuples).

An iterator is an object that produces elements one at a time using _next_().

Example:

```
my_list = [1, 2, 3]
my_iter = iter(my_list)  # Iterator
print(next(my_iter))  # Output: 1
```

Question 6. Concept and definition of generators

Answer :A generator is a function that yields values lazily using yield.

Example:

```
def gen():
    yield 1
```

```
    yield 2
```

Question 7. Advantages of generators over regular functions

Answer :Memory-efficient as they generate values one by one instead of storing them all at once.

Question 8. Lambda function and usage

Answer A small anonymous function defined using lambda.

Example:

```
square = lambda x: x * x
print(square(4))  # Output: 16
```

Question 9. Purpose of map() function

Answer :It applies a function to all items in an iterable.

Example:

```
nums = [1, 2, 3]
squares = list(map(lambda x: x * x, nums))
print(squares)  # Output: [1, 4, 9]
```

Question 10. Difference between map(), reduce(), and filter()

Answer :map(): Applies a function to all elements.

reduce(): Aggregates elements into a single value.

filter(): Filters elements based on a condition.

Example:

```
from functools import reduce
nums = [1, 2, 3, 4]
print(list(map(lambda x: x * 2, nums)))  # [2, 4, 6, 8]
print(reduce(lambda x, y: x + y, nums))  # 10
print(list(filter(lambda x: x % 2 == 0, nums)))  # [2, 4]
```

Question 11. Sum operation using reduce() on [47, 11, 42, 13]

Answer :Calculation: ((47 + 11) + 42) + 13 = 113

Example:

```
from functools import reduce
nums = [47, 11, 42, 13]
total = reduce(lambda x, y: x + y, nums)
print(total)  # Output: 113
```

Practical Questions:

```
1. #Function to return the sum of all even numbers in a list:

def sum_even_numbers(lst):
    return sum(num for num in lst if num % 2 == 0)

print(sum_even_numbers([1, 2, 3, 4, 5, 6]))  # Output: 12

2. #Function to reverse a string:

def reverse_string(s):
    return s[::-1]

print(reverse_string("hello"))  # Output: "olleh"
```

```
3. #Function to return a list of squares of numbers:

def square_numbers(lst):
    return [num ** 2 for num in lst]

print(square_numbers([1, 2, 3, 4]))  # Output: [1, 4, 9, 16]

4. #Function to check if a number is prime:

def is_prime(n):
    if n < 2:
        return False
    for i in range(2, int(n**0.5) + 1):
        if n % i == 0:
            return False
    return True

print(is_prime(17))  # Output: True
print(is_prime(18))  # Output: False

5. #Iterator class for Fibonacci sequence:

class Fibonacci:
    def _init_(self, max_terms):
        self.max_terms = max_terms
        self.a, self.b = 0, 1
        self.count = 0

    def _iter_(self):
        return self

    def _next_(self):
        if self.count >= self.max_terms:
            raise StopIteration
        value = self.a
        self.a, self.b = self.b, self.a + self.b
        self.count += 1
        return value

fib = Fibonacci(5)
```

```
print(list(fib))  # Output: [0, 1, 1, 2, 3]

6. #Generator function for powers of 2:

def powers_of_2(n):
    for i in range(n):
        yield 2 ** i

print(list(powers_of_2(5)))  # Output: [1, 2, 4, 8, 16]

7. #Generator function to read file line by line:

def read_file(filename):
    with open(filename, 'r') as file:
        for line in file:
            yield line.strip()

# Example usage
# for line in read_file("example.txt"):
#     print(line)

8. #Sorting a list of tuples by the second element using lambda:

data = [(1, 5), (2, 1), (3, 8)]
sorted_data = sorted(data, key=lambda x: x[1])
print(sorted_data)  # Output: [(2, 1), (1, 5), (3, 8)]

9. #Convert Celsius to Fahrenheit using map():

celsius = [0, 20, 30, 40]
fahrenheit = list(map(lambda c: (c * 9/5) + 32, celsius))
print(fahrenheit)  # Output: [32.0, 68.0, 86.0, 104.0]

10. #Remove vowels using filter():

def remove_vowels(s):
    return "".join(filter(lambda c: c.lower() not in "aeiou", s))

print(remove_vowels("hello world"))  # Output: "hello world"
```

```
11. #Accounting routine using map() and lambda:

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),
]

result = list(map(lambda order: (order[0], order[2] * order[3] + (10 if
order[2] * order[3] < 100 else 0)), orders))
print(result)
# Output: [(34587, 163.8), (98762, 284.0), (77226, 98.85 + 10), (88112,
74.97 + 10)]
```

Name: Dharmendra sharma
Email:sandysharma0783@gmail.com
Assignment Name:Functions
Github link : code