PYTHON FUNCTION (ASSIGNMENT)

Theory Questions:

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

Answer :--

Function: A block of code that is defined using def and can be called independently.

Method: A function that is associated with an object and called using dot (.) notation. **Example**

In [ ]:

```
def greet():  # Function
    print("Hello")

"hello".upper()  # Method (associated with string object)
```

Out[ ]:

```
'HELLO'
```

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

Answer:--

Parameters: Variables listed in a function's definition.

Arguments: Actual values passed to the function when calling it.

Example:

In [ ]:

```
def add(x, y):  # x, y are parameters
    return x + y

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

Out[ ]:

```
8
```

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

Answer :--

Define: Using def, or lambda for anonymous functions.

PYTHON FUNCTION (ASSIGNMENT)

Call: By function name followed by parentheses and arguments.

Example:

In [ ]:

```
def square(x): return x * x
print(square(4))

double = lambda x: x * 2
print(double(5))
```

```
16
10
```

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

Answer :--

It sends a value back to the caller and exits the function.

Example:

In [ ]:

```
def add(x, y):
    return x + y

result = add(2, 3)
print(result)
```

```
5
```

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

Answer:--

Iterable: Any object that can return an iterator (e.g., list, tuple).

Iterator: An object with **iter**() and **next**() methods.

Example:

In [ ]:

PYTHON FUNCTION (ASSIGNMENT)

```
nums = [1, 2, 3]       # Iterable
it = iter(nums)        # Iterator
print(next(it))        # Output: 1
```

Practical Questions:

1. Write a Python function that takes a list of numbers as input and returns the sum of all even numbers in the list.

```
In [8]:

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

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

```
12
```

2. Create a Python function that accepts a string and returns the reverse of that string.

```
In [14]:

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

print(reverse_string("hello"))
```

```
olleh
```

3. Implement a Python function that takes a list of integers and returns a new list containing the squares of each number.

```
In [15]:

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

print(square_numbers([1, 2, 3, 4]))
```

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

PYTHON FUNCTION (ASSIGNMENT)

4. Write a Python function that checks if a given number is prime or not from 1 to 200.

In [16]:

```
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

primes = [i for i in range(1, 201) if is_prime(i)]
print(primes)
```

```
[2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71,
73, 79, 83, 89, 97, 101, 103, 107, 109, 113, 127, 131, 137, 139, 149, 151,
157, 163, 167, 173, 179, 181, 191, 193, 197, 199]
```

5. Create an iterator class in Python that generates the Fibonacci sequence up to a specified number of terms.

In [17]:

```
class FibonacciIterator:
    def __init__(self, n):
        self.n = n
        self.a, self.b = 0, 1
        self.count = 0

    def __iter__(self):
        return self

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

fib = FibonacciIterator(7)
print(list(fib))
```

```
[0, 1, 1, 2, 3, 5, 8]
```

PYTHON FUNCTION (ASSIGNMENT)

6. Write a generator function in Python that yields the powers of 2 up to a given exponent.

In [18]:

```
def powers_of_two(max_exponent):
    for i in range(max_exponent + 1):
        yield 2 ** i

print(list(powers_of_two(4)))
```

```
[1, 2, 4, 8, 16]
```

7. Implement a generator function that reads a file line by line and yields each line as a string.

In [20]:

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

for line in read_lines("sample.txt"):
    print(line)
```

8. Use a lambda function in Python to sort a list of tuples based on the second element of each tuple.

In [21]:

```
tuples = [(1, 3), (4, 1), (2, 2)]
sorted_tuples = sorted(tuples, key=lambda x: x[1])
print(sorted_tuples)
```

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

9. Write a Python program that uses map() to convert a list of temperatures from Celsius to Fahrenheit.

In [22]:

```
celsius = [0, 20, 30, 40]
fahrenheit = list(map(lambda c: (c * 9/5) + 32, celsius))
```

PYTHON FUNCTION (ASSIGNMENT)

```
print(fahrenheit)
```

```
[32.0, 68.0, 86.0, 104.0]
```

10. Create a Python program that uses filter() to remove all the vowels from a given string.

```
In [23]:
```

```
def remove_vowels(s):
    return ''.join(filter(lambda x: x.lower() not in 'aeiou', s))

print(remove_vowels("Interview"))
```

```
ntrvw
```

11) Imagine an accounting routine used in a book shop. It works on a list with sublists, which look like this:

Order Number	Book Title and Author	Quantity	Price per Item
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

Write a Python program, which returns a list with 2-tuples. Each tuple consists of the order number and the product of the price per item and the quantity. The product should be increased by 10,- € if the value of the order is smaller than 100,00 €.

Write a Python program using lambda and map.

```
In [24]:
```

```
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], round(order[2] * order[3] + (10 if order[2] * order[3] < 100
else 0), 2)), orders))

print(result)
```

PYTHON FUNCTION (ASSIGNMENT)

```
[(34587, 163.8), (98762, 284.0), (77226, 108.85), (88112, 84.97)]
```

Explanation:

- Each entry is processed by map and checked if the Quantity × Price is less than 100.
- If it is, 10 is added.
- The result is rounded to 2 decimal places and returned as a tuple with the order number.

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

Answer :--

A generator is a function that yields values one at a time using the yield keyword.

They are memory efficient and lazy.

Example:

In [ ]:

```
def count_up_to(n):
    for i in range(n):
        yield i

gen = count_up_to(3)
print(next(gen))  # Output: 0
```

```
0
```

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

Answer :--

Memory Efficient: Generates items one at a time.

Faster for large data: Doesn't compute entire result at once.

Pause/Resume: Maintains state between yields.

Example:

In [ ]:

PYTHON FUNCTION (ASSIGNMENT)

```
def infinite():
    n = 0
    while True:
        yield n
        n += 1
```

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

Answer :--

An anonymous function defined using lambda.

Useful for short, throwaway functions.

Example:

In [ ]:

```
square = lambda x: x * x
print(square(4))
```

```
16
```

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

Answer :--

Applies a function to each item in an iterable.

Returns a map object (needs to be converted to list for display).

Example:

In [ ]:

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

```
[1, 4, 9]
```

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

PYTHON FUNCTION (ASSIGNMENT)

Answer :--

map(): Applies a function to all elements.

filter(): Selects elements where function returns True.

reduce(): Applies a rolling computation to pairs (needs functools).

Example:

In [ ]:

```
from functools import reduce

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

```
[2, 4, 6, 8]
[2, 4]
10
```

11. Pen & Paper: Show reduce() sum mechanism for list [47, 11, 42, 13]

Answer :--