Training Day 5 Report

Date: 29 June 2025

Topic: Functions in Python

Overview:

On the fifth day of AI/ML training at Sensations Software Technology, the focus was on **functions in Python**. Functions allow us to **reuse code, organize logic, and make programs modular**. Both normal functions (using def) and anonymous functions (using lambda) were covered with multiple examples.

Topics Covered:

1. Creating and using functions with def
2. Returning values from functions
3. Factorial using functions
4. Lambda functions (single-line anonymous functions)
5. Passing lists to functions
6. Even/Odd number check
7. Checking if a string starts with a particular letter using lambda
8. Prime number checking function
9. Lambda with multiple conditions

Examples:

1. Normal Function Example:

```
def multiply_four(a, b, c, d):
    return a * b * c * d

print(multiply_four(2, 3, 4, 5))  # Output: 120
```

2. Factorial Function:

```
def fac(n):
    result = 1
    for i in range(1, n + 1):
        result *= i
    return result

print(fac(5))  # Output: 120
```

3. Lambda Function:

```
x = lambda a, b: a + b
print(x(2, 3))  # Output: 5
```

4. Average Using Function:

```
lis = [1, 2, 3, 4]

def myfunc(a):
    x = sum(a) / len(a)
    print(x)

myfunc(lis)  # Output: 2.5
```

5. Even/Odd Check:

```
def even_odd(num):
    if num % 2 == 0:
        print(num, "is Even")
    else:
        print(num, "is Odd")

even_odd(6)  # 6 is Even
even_odd(9)  # 9 is Odd
```

6. String Condition with Lambda:

```
starts_with_A = lambda s: s[0] == 'A'
print(starts_with_A("Apple"))   # True
print(starts_with_A("Banana"))  # False
```

7. Prime Number Check (Improved):

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

print(is_prime(20))  # False
print(is_prime(7))   # True
```

8. Lambda with Multiple Conditions:

```
c = lambda a: a % 3 == 0 and a % 5 == 0
print(c(5))   # False
print(c(15))  # True
```

Hands-On Practice:

- Defined **custom functions** for multiplication, factorial, and average.
- Learned to use **lambda functions** for simple, one-line tasks.
- Checked for **even/odd numbers** using both normal and lambda functions.
- Wrote a **prime number function** and debugged errors.
- Practiced **conditional lambda functions** for multiple checks.

Learning Outcome:

By the end of the session, I understood how to:
✔ Write reusable and modular functions.
✔ Use **lambda functions** for concise operations.
✔ Implement condition checks using both def and lambda.
✔ Apply functions in solving problems like factorial, average, even/odd, and prime number detection.