Training Day 3 Report

Date: 25 June 2025
Topic: Functions and Lambda Expressions in Python

Overview:

On the third day of AI/ML training at Sensations Software Technology, we focused on **functions** in Python, including normal functions, reusability, and **lambda (anonymous) functions**. Functions help in breaking programs into smaller modules, making code reusable and easier to maintain.

Topics Covered:

1. Defining functions with def
2. Returning values from functions
3. Factorial function example
4. Code reusability with functions
5. Lambda functions (anonymous functions)
6. Average calculation using functions
7. Even-Odd check using functions
8. Prime number check (debugging example)
9. Condition checks using lambda

Details:

1. Normal Function Example:

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

result = multiply_four(2, 3, 4, 5)
print("Multiplication result:", result)
```

Output:

Multiplication result: 120

2. Factorial of a Number:

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

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

3. Lambda Function Example:

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

Lambda is a **single-line anonymous function** without the def keyword.

4. Average Function Example:

```
lis = [1, 2, 3, 4]
def myfunc(a):
    x = sum(a) / len(a)
    print(x)

myfunc(lis)   # Output: 2.5
```

5. Even-Odd Function:

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

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

6. Lambda with Condition:

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

7. Prime Number Function (Debugging Example):

```
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. More Lambda Conditions:

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

Hands-On Practice:

- Created normal functions with def
- Used loops for factorial and prime number check
- Learned about **lambda functions** for short operations
- Implemented condition checks with lambda
- Debugged prime number function

Learning Outcome:

Gained practical understanding of **functions, lambda functions, and condition-based programming**. These concepts improve **code reusability, modularity, and efficiency** in Python programming — essential for building scalable AI/ML applications.