Class 4 – Functions and Modules

Objective: Learn how to create and use functions, pass arguments, return values, and work with modules to write modular Python programs.

1. Warm-Up Discussion

Questions to ask:

- Why do we use functions in programming?
- How do large programs stay organized?

Explanation: Functions help in reusing code and organizing complex problems into smaller manageable tasks. They improve modularity and readability.

2. Understanding Functions

Let's begin with defining and calling simple functions.

```
# A simple function definition and call
def greet():
    print("Hello, welcome to Python!")

greet()
```

```
Hello, welcome to Python!
```

Explanation: This function `greet()` does not take any parameters. When we call `greet()`, it executes the print statement inside the function.

```
# Function with parameters and return value
def add_numbers(a, b):
    return a + b

result = add_numbers(5, 3)
print("Sum:", result)
```

```
Sum: 8
```

Explanation: This function `add_numbers` takes two arguments and returns their sum. The result is stored in a variable and printed.

```
# Function with default parameter
def greet(name="Guest"):
    print(f"Hello, {name}!")
```

```
greet()
greet("Alice")

Hello, Guest!
Hello, Alice!
```

Explanation: This function `greet` uses a default parameter. If no argument is passed, it uses 'Guest'.

3. Introduction to Lambda Functions

Lambda functions are small anonymous functions defined using the `lambda` keyword.

```
# Lambda function to square a number
square = lambda x: x * x
print(square(5))

25
```

Explanation: Lambda functions are one-liner functions. Here `lambda x: x * x` creates a function to square a number.

```
# Lambda with multiple arguments
add = lambda x, y: x + y
print(add(3, 7))

10
```

Explanation: This lambda function takes two arguments and returns their sum, similar to a regular function.

4. Understanding Modules

Modules are files containing Python code. They help in organizing code into separate files.

```
# my_module.py (Save this code in a separate Python file)
def greet(name):
    return f"Hello, {name}!"
```

Now let’s import and use that module.

```
# Importing user-defined module
import my_module

print(my_module.greet("Alice"))

Hello, Alice!
```

Explanation: We import our own module `my_module.py` and use the `greet` function defined in it.

```python
# Using built-in math module
import math
print(math.sqrt(16))
```

```
4.0
```

Explanation: Python provides many built-in modules like `math`. We use `math.sqrt()` to find the square root of a number.

5. Hands-On Activity

Factorial Function

```python
def factorial(n):
    result = 1
    for i in range(2, n + 1):
        result *= i
    return result

print(factorial(5))
```

```
120
```

Lambda to Find Maximum of Two Numbers

```python
max_func = lambda a, b: a if a > b else b
print(max_func(10, 20))
```

```
20
```

User-defined Math Module

Save the following in `basic_math.py`:

```python
# basic_math.py
def add(x, y):
    return x + y

def subtract(x, y):
    return x - y

def multiply(x, y):
    return x * y

def divide(x, y):
    if y != 0:
        return x / y
    else:
        return "Cannot divide by zero"
```

Now import and use it:

```
import basic_math
print(basic_math.add(5, 3))
print(basic_math.divide(10, 0))
```

```
8
Cannot divide by zero
```

6. Wrap-Up

Recap:

- Functions make code reusable and modular.
- Parameters make functions dynamic.
- Lambda functions are for short anonymous use.
- Modules help organize and reuse code.

Homework:

1. Add exponentiation function to `basic_math.py`:

```
def power(x, y):
    return x ** y
```

1. Write a function to find the largest number in a list:

```
def find_largest(numbers):
    largest = numbers[0]
    for num in numbers:
        if num > largest:
            largest = num
    return largest

print(find_largest([2, 10, 3, 55, 7]))
```

```
55
```