Day 6: Functions in Python

Topics Covered:

- *Introduction to Functions*
- *Defining Functions*
- *Function Parameters*
- *Return Statements*
- *Variable Scope*
- *Lambda Functions*
- *Recursion*

1. Introduction to Functions:

In Python, a function is a block of reusable code that is used to perform a specific task. Functions help in organizing code, promoting reuse, and reducing redundancy.

2. Defining Functions:

Functions are defined using the `def` keyword, followed by the function name and parameters (if any). The body of the function contains the code to be executed when the function is called.

Example:

```
def greet():
    print("Hello, World!")
greet()  # Output: Hello, World!
```

3. Function Parameters:

Functions can take parameters (also known as arguments) that allow us to pass data into the function. Parameters are specified inside the parentheses when defining a function.

Example:

```
def greet(name):
    print(f"Hello, {name}!")
greet("Alice")  # Output: Hello, Alice!
```

4. Return Statements:

A function can return a value using the `return` keyword. The `return` statement allows the function to produce a result that can be used elsewhere in the code.

Example:

```
def add(a, b):
    return a + b
result = add(3, 5)
print(result)  # Output: 8
```

5. Variable Scope:

Variables defined inside a function are local to that function, meaning they cannot be accessed outside of it. Variables defined outside of any function are global and can be accessed anywhere in the code.

Example of local scope:

```
def example():
    x = 5  # Local variable
    print(x)
example()  # Output: 5
```

Example of global scope:

```
x = 10  # Global variable
def example():
    print(x)
example()  # Output: 10
```

6. Lambda Functions:

Lambda functions are small anonymous functions defined using the `lambda` keyword. They can have any number of input parameters, but only one expression.

Example:

```
add = lambda a, b: a + b
result = add(3, 5)
print(result)  # Output: 8
```

7. Recursion:

Recursion occurs when a function calls itself. Recursive functions are typically used to solve problems that can be broken down into smaller subproblems of the same type.

Example of Recursion:

```
def factorial(n):
    if n == 0:
        return 1
    else:
        return n * factorial(n - 1)
result = factorial(5)
print(result)  # Output: 120
```

Summary of Day 6:

Today, we learned about functions in Python. We explored how to define functions, use parameters, and return values. We also covered the scope of variables, lambda functions for concise function expressions, and recursion for solving problems with repetitive structures.