Programming Skills - Python

Learn the basics of Python Programming

Day - 3

Functions

Agenda

3.1. Brief Recap

A brief recap of what we discussed yesterday.

3.2. Introduction to Functions

What they are, why they are used, and how to use them.

3.3. In-built Functions

Definition and examples.

3.4. Custom Functions

Definition and defining a custom function.

Agenda

3.5. Parameters

What they are and why they are integral to functions.

3.6. Return Values

What they are and why they are integral to functions.

3.7. Scope

Defining, understanding, and using scope.

3.8. Modular Code

Understanding the importance of modular code.

Agenda

3.9. Recursion

Did you mean: ***recursion***

3.1. Brief Recap

3.2. Introduction to Functions

Functions

What are they?

A block of organized, reusable code that performs a single action.

Why are they used?

Helps avoid repetition and makes programs modular and maintainable.

How to use it?

```
def function_name():
    # block of code
    # under the function
```

Replit →

3.3. In-built functions

In-built functions

Functions built into the language ready to be used by developers.

Examples:

```
input()
print()
```

3.4. Custom Functions

Custom Functions

Allows developers to create reusable blocks of code.

Example:

```
def greet():
    print(“Hello!”)
```

Replit →

3.5. Parameters

Parameters

Variables that are passed to the function to be used inside the function.

Example:

```
def greet(name):
    print(“Hello!,”, name)
```

Replit →

3.6. Return Values

Return Values

Value returned by the function that can be stored in another variable.

Example:

```
def find_area(l, b):
    return l * b

area = find_area(10, 5)
print(area) # 50
```

Replit →

3.7. Scope

Scope

Scope refers to the visibility or lifetime of variables in a program.

It determines where a variable can be accessed by a program.

Local and Global Variables

Local - defined and accessible only inside the function it is present in.

Global - defined outside all functions and accessible everywhere in the file.

Example:

```
x = 10 # global variable
def change_value():
    x = 5 # local variable
    print(x) # outputs 5

print(x) # 10
```

Replit →

3.8. Modular Code

Modular Code

Breaking down a complex program into smaller, manageable functions, such that each function performs a single task.

Why:

- Easier debugging
- Reusable code
- Better maintainance

3.9. Recursion

Recursion

A function that calls itself. Useful for solving problems that can be broken down into smaller, identical subproblems.

Example:

```
def factorial(n):
  if n == 1:
    return 1
  else:
    return n * factorial(n - 1)

print(factorial(5))
```

Replit →

Q&A

Thank You!