Salvadora, Angelle D.
CS3C

Functions

● **Defining a Function**

— is a block of reusable code that performs a specific task. It allows you to organize your code into logical blocks, making it easier to read, understand, and maintain. Functions can take inputs (called parameters or arguments), perform operations, and return outputs.

● **Reasons of Using Functions**

— It is easier to use and it makes our codes easier to read and understand.

● **Types of Functions in Python**

— Some examples are print(), len(), range(), max(), min(), abs(), sum(), etc.

● **Advantages of User – Defined Function**

— The advantages of a User-defined function allow our code organization, reusability, and abstraction. Simplifies the usage of complex functionality, and allows us to check our code in isolation, making it easier to identify and fix bugs.

● **Rules in Declaring a Function in Python**

1. **Use the def keyword:** Functions in Python are declared using the def keyword, followed by the function name and parentheses containing any parameters the function accepts.
   Ex.
   ```
   def my_function():
       print("Hello from a function")

   my_function()
   ```
2. **Function names** must be valid identifiers, meaning they cannot be a reserved keyword, and they must follow the same naming rules as variable names. Function names should be descriptive and indicate the purpose of the function.
3. **Parameters** are optional, but if present, they are enclosed within parentheses (). Parameters are separated by commas. Parameters can have default values, making them optional in function calls.
4. **Indentation:** The function body must be indented, typically by four spaces or a tab. This is Python's way of defining a code block. All statements within the function must be indented at the same level.
5. **Function Docstring:** It's a good practice to include a docstring, which is a string literal that describes the purpose of the function, its parameters, and its return value. Docstrings are **enclosed within triple quotes (“““ ”””)** and are placed immediately after the function definition.
   Ex.
   ```
   def my_function(param1, param2):
   """
   This function does something.

    Parameters:
    param1 (int): Description of param1.
   param2 (str): Description of param2.

   Returns:
   bool: Description of the return value.
   """
   pass
   ```
6. **Return Statement:** Functions can optionally return a value using the return statement. The return statement terminates the function and passes a value (if specified) back to the caller.

Ex.

```
def add_numbers(x, y):
    return x + y
```

7. **Global vs. Local Variables:** Variables defined within the function are considered **local** and only accessible within that function's scope. Variables defined outside of any function (in the **global scope**) are accessible throughout the program, including within functions. To modify a global variable within a function, you need to use the global keyword.

- **Python Function Syntax**

```
def add_numbers(x, y):
"""Function to add two numbers."""
result = x + y
return result
```

- **Function Argument and Parameter**

```
def function_name parameter1, parameter2=default_value, ...):
# Function body
pass
```

- **The Return Statement**

```
def function_name(parameters):
#Function body
#Code to perform the task
return [expression]  # Optional return statement
```