Liezel Cielo D. Tiqui
CS3C

Defining a Function:
A function in Python is a segment of reusable code that carries out a particular activity. You can use it to organize code based on common needs and run it at any time. The "def" keyword, the function name, and a pair of parentheses are how you define your own functions in Python.

Reasons for Using Functions:
Python's functions have many benefits. By allowing a code block to be defined once and reused numerous times inside the program, they promote code reusability. Furthermore, functions encourage modularity by decomposing complex issues into smaller, easier-to-manage parts, improving the readability and maintainability of code. They also offer abstraction, which frees developers to focus on higher-level reasoning by removing the need to worry about implementation specifics. Functions also help to organize code by dividing it into logical sections, which makes the code easier to understand and navigate.

Types of Functions in Python:
Built-in functions and user-defined functions are the two primary categories of functions in Python. The Python language has built-in functions that are instantly available for use in a variety of activities. Prominent instances are print(), len(), and range(). On the other hand, programmers create user-defined functions to carry out customized activities. Using the "def" keyword to create a function structure, these functions enable developers to create custom features.

Advantages of User-Defined Functions:
When it comes to software development, functions have many advantages. First of all, they reduce code redundancy by enabling you to use a function multiple times in your program, increasing reusability. Furthermore, functions encourage modularity by decomposing complex issues into smaller, easier-to-manage chunks, improving the readability and organization of the code. They also help with abstraction by hiding implementation details, so you can focus on logic at a higher level. Additionally, functions provide the isolation and testing of particular code areas, which makes problem identification and resolution easier. This helps with debugging and maintenance. Finally, functions enable parallel work on various software components and promote code reuse and collaboration among several programmers.

Rules in Declaring a Function in Python:
In Python, it's important to follow some rules when declaring functions. First and foremost, function names should be meaningful and follow naming guidelines. Generally, this means using lowercase letters with underscores between words. It's important to take care not to interfere with any built-in Python methods or keywords. The function name and parentheses come after the "def" keyword in the function definition. Any parameters are defined in these parenthesis. The code that will be run when the function is invoked is contained in the function's body, which

is appropriately indented. The "return" statement is used to specify the value to be returned when the function is intended to return a value.

Python Function Syntax:
The function declaration begins with the function name and parentheses () after the def keyword. It accepts any parameters enclosed in parenthesis. The function body is indented below the declaration. A docstring can be included to explain the function's goal, and a return statement may be included to return a value.
example:

```
def greet(name):
   # This function greets the user by name.
   print("Hello, " + name + "!")

# Calling the function
greet("Liezel")
```

Function Argument and Parameter:
Function arguments are the values passed to a function when called, allowing it to accept input and perform operations. These arguments are specified in the function definition, usually after the function name.
example:

```
def add_numbers(a, b):
  # This function adds two numbers and prints the result.
   sum = a + b
   print("The sum of", a, "and", b, "is", sum)

# Calling the function with arguments
add_numbers(3, 5)
```

The Return Statement in Python:
The return statement in Python specifies the value a function should return to its caller, allowing it to provide an output or result back to the code that called it. It is typically placed at the end of the function and can return any valid Python expression or value. For example, the calculate_sum function calculates the sum of two numbers, sending the result back as the function's output.
example:

```
def calculate_sum(a, b):
   # This function calculates the sum of two numbers and returns the result.
   sum = a + b
   return sum

# Calling the function and storing the returned value
result = calculate_sum(4, 5)
print("The sum is:", result)
```