Wexdi SoftSkills

Functions in Python

Creating reusable code with function

- Functions allow you to organize code in modular blocks and make code reusable. The more your Python code grows the more difficult it can get to manage it if you don't use functions. Once you define a function you can call it multiple times in your code without having to rewrite that code again (avoid code duplication).

```
def function_name(parameters):
    """docstring"""
    function_body
```

Creating reusable code with function


```
def function_name(parameters):
    """docstring"""
    function_body
```

- The components used to define a Python function are:
  - header: this is made of the def keyword that starts the definition of the function, the function name, parameters enclosed within parentheses, and the colon symbol. Parameters are optional, a function can also have no parameters. Also, a function can have any number of parameters.
  - docstring: provides documentation about the function.
  - body: this is a sequence of Python statements and can end with an optional return statement if the function returns a value.

Built-in vs user-defined functions

Wexdi SoftSkills

Built-in Functions

Python comes with a vast collection of built-in functions that cover a wide range of tasks, from basic operations to advanced functionalities. These functions are readily available and optimized for efficiency, making them convenient for common programming tasks.

```
# Examples of built-in functions
print(len([1, 2, 3]))  # Output: 3
print(max(4, 7, 2, 9))  # Output: 9
print(sorted([3, 1, 4, 1, 5]))  # Output: [1, 1, 3, 4, 5]
```

User-Defined Functions

User-defined functions are created by the developer to encapsulate custom logic or operations tailored to specific requirements. They provide flexibility, modularity, and reusability, allowing developers to organize code more effectively and solve complex problems with ease.

```
# Example of a user-defined function
def greet(name):
    return f"Hello, {name}!"

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