Functions in Python

- **What are Functions?**
  - Functions in Python are a set of instructions that take an input and return an output. They help organize code and make it reusable.
  - Example: The `print` function takes a value and outputs it, such as printing the string `"Hello, World"`.
- **Declaring a Function**
  - A function is declared using the `def` keyword, followed by the function name, optional parameters inside parentheses, and the task (logic) to complete.
    - Example:

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

    - This function takes two arguments `x` and `y` and returns their sum.
- **Function Demonstration: Calculating Tax**
  - A practical example of using a function to calculate tax for a bill is demonstrated:
    - Variables are declared to store the `bill` (a float) and `tax_rate` (an integer).
    - The tax is calculated as `bill * tax_rate / 100`.
    - The total tax is printed out using the `print` function.
- **Making the Code Reusable with Functions**
  - The previous tax calculation logic is turned into a reusable function called `calculate_tax`.
    - The function accepts arguments: `bill` (total value of the bill) and `tax_rate` (tax rate percentage).
    - Inside the function, the tax is calculated and returned using the formula `bill * tax_rate / 100`.
    - Example of the function:

```
def calculate_tax(bill, tax_rate):
    return bill * tax_rate / 100
```

- **Calling the Function**
  - To use the function, you call it with specific arguments (e.g., `calculate_tax(175, 15)`), which returns the tax value.
  - The result for a bill of 175 and a tax rate of 15% is `26.25`.

- **Function Reusability**
  - The function can be called multiple times with different arguments to calculate taxes for different bills and tax rates.
    - Example:

```
print(calculate_tax(175, 15))  # Outputs 26.25
print(calculate_tax(164.33, 22))  # Outputs 36.15
```

- **Using `round()` to Format Output**
  - The result can be formatted using the `round()` function to limit the number of decimal places (e.g., `round(36.1526, 2)` will return `36.15`).
- **Benefits of Functions**
  - Functions make code more modular and easier to maintain.
  - Any changes made to the function are automatically reflected wherever the function is called in the code.
- **Conclusion**
  - Functions are an essential tool for organizing and reusing code in Python. They can take inputs, return outputs, and simplify complex tasks by breaking them into smaller, reusable pieces.