Functions in Python

1. What is a Function in Python?

A **function** is a reusable block of code that performs a specific task. Functions help break down large programs into smaller, manageable pieces.

name of the function

```
def hello_world():
    print("Hello, I'm a data scientist!")
```

code to be executed

keyword

indentation

You can call (execute) the function using:

```
hello_world()
```

Hello, I'm a data scientist!

2. Why Are Functions Important?

- ① **Modularity:** Breaks complex tasks into smaller, manageable pieces.
- ② **Reusability:** Write once and reuse the same logic across multiple parts of your program.
- ③ **Readability:** Code becomes easier to read and maintain.
- ④ **Debugging:** Easier to isolate and fix issues.

3. Function with Parameters and Return Value

Functions can accept input values (called parameters) and return output values.

first second argument

```
def add(a, b):
    return a + b

result = add(5, 3)
print(result)  # Output: 8
```

= 8

return

8

4. Default Arguments

You can provide default values for function parameters.

```
def power(base, exponent=2):
    return base**exponent

print(power(3))      # 9
print(power(3, 3))   # 27
```

exponentation

power(3) = 3^2 = 9

power(3,3) = 3^3 = 27

5. Keyword and Positional Arguments

Positional arguments are matched by order. **Keyword arguments** are matched by name.

name = Oscar   role = data scientist

```
def info(name, role):
    print(f"{name} is a {role}.")

info("Oscar", "Data Scientist")                 # Positional
info(role="Data Scientist", name="Oscar")   # Keyword
```

→ Oscar is a data Scientist

→ Oscar is a data scientist

6. Applications in Data Science

- 1 **Data Preprocessing:** Create reusable functions to clean and transform data.
- 2 **Feature Engineering:** Encapsulate logic for new feature creation.
- 3 **Model Training and Evaluation:** Define modular training and evaluation pipelines.
- 4 **Visualization:** Reuse plotting functions for consistency.

Example: Normalize a Column

→ column is a column from a pandas dataframe

```
def normalize(column):
    return (column - column.mean()) / column.std()
```

Example: Evaluate Model Accuracy

```
def evaluate_accuracy(y_true, y_pred):
    correct = sum(y_true == y_pred)
    return correct / len(y_true)
```

7. Anonymous Functions (Lambda)

`lambda` functions are small, one-line anonymous functions.

```
square = lambda x: x**2
print(square(4))  # Output: 16
```

8. Summary

Functions are foundational to Python programming and essential for any junior data scientist. They improve code modularity, reusability, and clarity. Mastering functions will help you write cleaner and more efficient analytical workflows.