DATA SCIENCE COURSE TUTORIAL # 30

3.18 Functions

What is a Function?

A **function** in Python is a reusable block of code that performs a specific task. Functions help make code modular, organized, and easier to maintain. Instead of repeating the same code multiple times, you can define it once and call it whenever needed.

Functions can take input (called **parameters**) and can return output using the **return** statement.

Example:

```
def greet():
    print("Hello, welcome to Python!")

greet()  # Function call
```

Benefits of Using Functions

- Reduces code repetition.
- Improves readability and organization.
- Easier debugging and maintenance.
- Enables modular programming.

Types of Functions

Python provides two main types of functions:

1. **Built-in Functions:** Predefined functions provided by Python such as `len()`, `sum()`, `max()`, `min()`, `print()`, and `type()`.
2. **User-defined Functions:** Functions that are created by programmers using the `def` keyword.

Creating a Function

You can define a function using the `def` keyword, followed by the function name and parentheses.

Syntax:

```
def function_name():
    # Code block
```

Example:

```python
def display():
    print("This is a simple function.")

display()
```

Function with Parameters

Functions can accept inputs known as **parameters**. Parameters allow functions to process dynamic data.

Example:

```python
def greet_user(name):
    print("Hello, " + name + "!")

greet_user("Ali")
```

Function with Multiple Parameters

You can pass multiple parameters separated by commas.

Example:

```python
def add_numbers(a, b):
    result = a + b
    print("Sum:", result)

add_numbers(5, 3)
```

Return Statement

The `return` statement allows a function to send a value back to the caller. Once `return` is executed, the function ends immediately.

Example:

```python
def multiply(x, y):
    return x * y

result = multiply(4, 5)
print(result)
```

Function with Default Parameters

You can assign default values to parameters. If no argument is passed, the default value will be used.

Example:

```python
def greet(name="Guest"):
    print("Welcome,", name)

greet("Sara")
greet()  # Uses default value 'Guest'
```

Keyword Arguments

You can specify arguments by their parameter names when calling a function.

Example:

```python
def student_info(name, age):
    print("Name:", name)
    print("Age:", age)

student_info(age=22, name="Ali")
```

---

*_Arbitrary Arguments (args)_

If you do not know how many arguments will be passed, use `*args`. It allows a function to accept any number of positional arguments.

Example:

```python
def show_numbers(*nums):
    for n in nums:
        print(n)

show_numbers(10, 20, 30, 40)
```

Arbitrary Keyword Arguments (kwargs**)

If you want to accept any number of keyword arguments, use `**kwargs`. It stores arguments as a dictionary.

Example:

```python
def show_details(**info):
    for key, value in info.items():
        print(key, ":", value)

show_details(name="Ali", age=22, country="Pakistan")
```

Nested Functions

You can define one function inside another. Such functions are called **nested functions**.

Example:

```python
def outer_function():
    print("This is the outer function.")

    def inner_function():
        print("This is the inner function.")

    inner_function()

outer_function()
```

Scope of Variables

Scope refers to the region where a variable can be accessed.

- **Local Scope:** Variables defined inside a function are local to that function.
- **Global Scope:** Variables defined outside all functions are accessible globally.

Example:

```python
x = 10  # Global variable

def my_function():
    y = 5  # Local variable
    print(y)

my_function()
print(x)
```

Global Keyword

The `global` keyword allows you to modify a global variable inside a function.

Example:

```python
count = 0

def increment():
    global count
    count += 1

increment()
print(count)
```

Docstrings (Documentation Strings)

A **docstring** is a string written just below the function definition to describe what the function does. It can be accessed using the `__doc__` attribute.

Example:

```python
def square(num):
    """This function returns the square of a number."""
    return num ** 2

print(square.__doc__)
```

Pass Statement in Functions

If a function body is empty, use the `pass` statement as a placeholder to avoid errors.

Example:

```python
def future_function():
    pass  # Placeholder for future code
```

Anonymous (Unnamed) Functions and Recursive Functions

An **anonymous function** is created using the `lambda` keyword. However, we will discuss **lambda functions** and **recursive functions** separately in the next tutorial.