Day 6: Understanding Python Functions

Functions are an essential part of Python programming. They allow us to reuse code, make programs more readable, and reduce errors.

In this notebook, we will cover:

1. Built-in Python functions
2. Why functions are important
3. How to define and use custom functions
4. Exercises to practice functions

1. Built-in Python Functions

Python comes with many **built-in functions** that you can use directly without having to define them. Some examples include:

- `print()`: Displays output on the screen
- `len()`: Returns the length of a sequence
- `type()`: Returns the type of an object
- `sum()`: Adds elements in a list or tuple

Here are a few examples:

```
# Example 1: Using built-in functions
print("Hello, World!")  # Prints a message
numbers = [1, 2, 3, 4, 5]
print("Length of list:", len(numbers))  # Finds the length of the list
print("Sum of numbers:", sum(numbers))  # Adds all numbers in the list
print("Type of numbers:", type(numbers))  # Returns the type of the
object
```

2. Why Functions are Important

Functions play a vital role in programming for several reasons:

1. **Reusability**: Write code once and reuse it multiple times.
2. **Readability**: Break down programs into smaller, understandable blocks.
3. **Maintainability**: Easier to update or debug specific parts of the program.
4. **Modularity**: Dividing code into functions makes it more organized.

For example, imagine you need to calculate the area of multiple rectangles. Instead of writing the formula multiple times, you can define a function and call it whenever needed.

```python
# Without a function
length1, width1 = 5, 10
length2, width2 = 7, 3

area1 = length1 * width1
area2 = length2 * width2

print("Area of rectangle 1:", area1)
print("Area of rectangle 2:", area2)

# With a function
def calculate_area(length, width):
    return length * width

# Reuse the function
print("Area of rectangle 1:", calculate_area(5, 10))
print("Area of rectangle 2:", calculate_area(7, 3))
```

3. Building Custom Functions

You can define your own functions in Python using the `def` keyword. The general syntax is:

```python
def function_name(parameters):
    # Code block
    return result
```

Example: A Simple Greeting Function

```python

# Defining a function
def greet(name):
    return f"Hello, {name}!"

# Calling the function
print(greet("Thapelo"))
print(greet("Python Learner"))
```

Example: Function with Default Parameters

```python

def greet(name="World"):
    return f"Hello, {name}!"

print(greet())  # Uses the default value
print(greet("Thapelo"))  # Overrides the default value
```

Example: Function with Multiple Parameters

```python
def calculate_area(length, width):
    return length * width

# Passing multiple parameters
print("Area:", calculate_area(10, 5))
```

4. Practice Exercises

1. Write a function `is_even(number)` that returns `True` if a number is even and `False` otherwise.

2. Write a function `convert_to_celsius(fahrenheit)` that converts a temperature from Fahrenheit to Celsius using the formula: [C = rac{5}{9} imes (F - 32)]

3. Write a function `factorial(n)` to calculate the factorial of a number. (Hint: Use a loop or recursion.)

```python

# Solution 1: Check if a number is even
def is_even(number):
    return number % 2 == 0

print(is_even(4))  # True
print(is_even(7))  # False


# Solution 2: Convert Fahrenheit to Celsius
def convert_to_celsius(fahrenheit):
    return (5/9) * (fahrenheit - 32)

print(convert_to_celsius(98.6))  # 37.0


# Solution 3: Factorial of a number
def factorial(n):
    result = 1
    for i in range(1, n + 1):
        result *= i
    return result

print(factorial(5))  # 120
```

Summary

- Python provides many built-in functions like `print()`, `len()`, and `sum()` for common tasks.
- Functions help to improve code reusability, readability, and maintainability.
- You can define custom functions using the `def` keyword.
- Practice writing functions to solidify your understanding.

Great work on completing Day 6! ▯