Python Functions

What is a Function in Python?

- A function is a reusable block of code that performs a specific task
- Functions help us organize code and avoid repetition
- Python has built-in functions and allows us to create our own

```
print("Hello, World!")  # print() is a built-in function
```

Why Use Functions?

- **Code Reusability**: Write once, use many times
- **Modularity**: Break complex problems into smaller parts
- **Readability**: Make code easier to understand
- **Maintainability**: Easier to fix and update

Defining a Function

```
def greet():
    """This function prints a greeting message."""
    print("Hello, welcome to Python functions!")

# Calling the function
greet()
```

- `def` keyword starts the function definition
- Function name follows naming rules (letters, numbers, underscores)
- Parentheses `()` may contain parameters
- Colon `:` ends the function header
- Indented block contains the function body
- Docstring (optional but recommended) describes the function

Function Parameters

Parameters allow functions to receive input values

```
def greet(name):
    """Greet a person by name."""
    print(f"Hello, {name}!")

# Calling with an argument
greet("Alice")  # Output: Hello, Alice!
greet("Bob")    # Output: Hello, Bob!
```

Multiple Parameters

Functions can have multiple parameters

```
def describe_pet(animal_type, pet_name):
    """Display information about a pet."""
    print(f"I have a {animal_type} named {pet_name}.")

# Calling with positional arguments
describe_pet("dog", "Rex")  # Output: I have a dog named Rex.

# Calling with keyword arguments
describe_pet(pet_name="Whiskers", animal_type="cat")
# Output: I have a cat named Whiskers.
```

Default Parameter Values

You can set default values for parameters

```
def greet(name, greeting="Hello"):
    """Greet a person with a customizable greeting."""
    print(f"{greeting}, {name}!")

greet("Alice")              # Output: Hello, Alice!
greet("Bob", "Good day")    # Output: Good day, Bob!
```

Return Values

Functions can send back results using the `return` statement

```
def add(a, b):
    """Add two numbers and return the result."""
    return a + b

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

# You can use the result directly
print(add(10, 20) + add(30, 40))  # Output: 100
```

Multiple Return Values

Functions can return multiple values as a tuple

```
def get_dimensions():
    """Return width and height."""
    return 1920, 1080

width, height = get_dimensions()
print(f"Width: {width}, Height: {height}")
# Output: Width: 1920, Height: 1080
```

Practice

- Write a function to calculate the volume of a cube from its side length s
- Write a function that prints the odd numbers between two integers n_1 and n_2, inclusive