# Functions in Python Exercises

**Exercise Set 1: Understanding the Basics of Functions**

**Exercise 1.1: Hello Function**

* **Goal:** Create a simple function that prints a greeting.
* **Instructions:**
  1. Define a function called greet that takes no arguments.
  2. Inside the function, print the message "Hello, Python functions!".
  3. Call the greet function.

**Exercise 1.2: Personalized Greeting**

* **Goal:** Create a function that takes an argument and uses it in its output.
* **Instructions:**
  1. Define a function called personal\_greet that takes one argument, name.
  2. Inside the function, print a personalized message like "Hello, [name]! Welcome to functions.".
  3. Call personal\_greet twice, once with your own name and once with "Alice".

**Exercise 1.3: Adding Two Numbers**

* **Goal:** Create a function that performs a simple calculation and returns a value.
* **Instructions:**
  1. Define a function called add\_numbers that takes two arguments, num1 and num2.
  2. Inside the function, calculate the sum of num1 and num2.
  3. Return the sum.
  4. Call add\_numbers with 5 and 3, and print the result.
  5. Call add\_numbers with 10 and 20, and store the result in a variable called total, then print total.

**Exercise 1.4: Simple Subtraction**

* **Goal:** Another basic arithmetic function to reinforce returning values.
* **Instructions:**
  1. Define a function called subtract\_numbers that takes two arguments, num1 and num2.
  2. The function should return the result of num1 minus num2.
  3. Test your function with a few different pairs of numbers and print the results.

**Exercise Set 2: Function Arguments and Scope**

**Exercise 2.1: Default Greeting**

* **Goal:** Understand default arguments.
* **Instructions:**
  1. Modify your personal\_greet function from Exercise 1.2.
  2. Make the name argument optional, with a default value of "Guest".
  3. Call the function once without providing a name.
  4. Call the function once with your name.

**Exercise 2.2: Keyword Arguments**

* **Goal:** Understand how to use keyword arguments.
* **Instructions:**
  1. Define a function called describe\_pet that takes two arguments: animal\_type and pet\_name.
  2. Inside the function, print a sentence like "I have a [animal\_type] named [pet\_name]."
  3. Call describe\_pet using positional arguments (e.g., describe\_pet("dog", "Buddy")).
  4. Call describe\_pet using keyword arguments (e.g., describe\_pet(pet\_name="Max", animal\_type="cat")).

**Exercise 2.3: Return Multiple Values (Optional/Advanced)**

* **Goal:** Show how functions can logically return multiple pieces of information, often as a tuple.
* **Instructions:**
  1. Define a function called calculate\_area\_perimeter that takes length and width as arguments.
  2. It should calculate both the area (length \* width) and the perimeter (2 \* (length + width)).
  3. Return both the area and the perimeter.
  4. Call the function with some dimensions, and unpack the returned values into two separate variables, then print them.

**Exercise Set 3: Practical Applications**

**Exercise 3.1: Even or Odd Checker**

* **Goal:** Use conditional logic inside a function.
* **Instructions:**
  1. Define a function called is\_even that takes one argument, number.
  2. The function should return True if the number is even, and False if it's odd. (Hint: Use the modulo operator %).
  3. Test your function with a few numbers (e.g., 4, 7, 0, 15) and print the results.

**Exercise 3.2: Find the Maximum**

* **Goal:** Find the larger of two numbers using a function.
* **Instructions:**
  1. Define a function called find\_maximum that takes two arguments, a and b.
  2. It should return the larger of the two numbers.
  3. Test with various pairs of numbers.

**Exercise 3.3: Grade Calculator**

* **Goal:** A slightly more complex conditional logic example.
* **Instructions:**
  1. Define a function called get\_grade that takes one argument, score (an integer between 0 and 100).
  2. It should return a letter grade based on the following scale:
     + 90-100: "A"
     + 80-89: "B"
     + 70-79: "C"
     + 60-69: "D"
     + Below 60: "F"
  3. Test your function with several different scores and print the grades.

**Exercise Set 4: Functions and Lists (Slightly More Advanced)**

**Exercise 4.1: Sum of List Elements**

* **Goal:** Iterate through a list inside a function.
* **Instructions:**
  1. Define a function called sum\_list that takes one argument, numbers (a list of numbers).
  2. It should calculate and return the sum of all numbers in the list.
  3. Test with a sample list like [1, 2, 3, 4, 5].

**Exercise 4.2: Count Vowels**

* **Goal:** Iterate through a string and apply conditions.
* **Instructions:**
  1. Define a function called count\_vowels that takes one argument, word (a string).
  2. It should return the number of vowels (a, e, i, o, u, case-insensitive) in the word.
  3. Test with words like "hello", "Python", "AEIOU".

**Exercise 4.3: Reverse a List**

* **Goal:** Practice list manipulation within a function.
* **Instructions:**
  1. Define a function called reverse\_list that takes one argument, my\_list.
  2. It should return a new list that is the reverse of my\_list. (Avoid using the built-in reverse() method or [::-1] for this exercise; try to do it manually with a loop).
  3. Test with a list of numbers or strings.

## Marking Rubrics for Exercise set 4

**Exercise 4.1**

| Criteria | Excellent | Good | Needs Improvement | No Attempt |
| --- | --- | --- | --- | --- |
| Function Definition | Correctly defines sum\_list(numbers) with proper parameter. | Function defined but parameter naming unclear. | Function defined incorrectly (e.g., missing parameter). | No function defined. |
| Logic / Correctness | Correctly iterates through list and returns accurate sum. | Minor error in summation (e.g., prints instead of returns). | Incorrect logic (e.g., only adds first element). | No attempt at logic. |
| Code Style | Clear, readable, uses loop properly. | Somewhat clear but inconsistent style. | Poor readability or unnecessary complexity. | No code. |

**Exercise 4.2**

| Criteria | Excellent | Good | Needs Improvement | No Attempt |
| --- | --- | --- | --- | --- |
| Function Definition | Correctly defines count\_vowels(word) with proper parameter. | Function defined but parameter naming unclear. | Function defined incorrectly. | No function defined. |
| Logic / Correctness | Correctly counts vowels (both upper & lower case). | Counts vowels but case sensitivity issue. | Attempts but incorrect logic (e.g., counts consonants). | No attempt at logic. |
| Code Style | Clear, efficient, uses loop/condition well. | Works but slightly inefficient or unclear. | Hard to follow or redundant code. | No code. |

**Exercise 4.3**

| Criteria | Excellent | Good | Needs Improvement | No Attempt |
| --- | --- | --- | --- | --- |
| Function Definition | Correctly defines reverse\_list(my\_list) with proper parameter. | Function defined but parameter naming unclear. | Function defined incorrectly. | No function defined. |
| Logic / Correctness | Correctly reverses list manually using loop. | Reverses list but uses built-in methods ([::-1] or .reverse()). | Attempted but incorrect reversal logic. | No attempt at logic. |
| Code Style | Clear, efficient, well-structured. | Works but slightly unclear or redundant. | Hard to follow or poor structure. | No code. |