# Chapter 6: Functions in Python

Connect with me: Youtube | LinkedIn | WhatsApp Channel | Web | Facebook | Twitter

- Download PDF
- To access the updated lecture notes, please click on the following link: https://yasirbhutta.github.io/python/docs/functions.html

> "The only way to do great work is to love what you do."
> – Steve Jobs

## 6.1 What is a Function?

A function is a block of reusable code that performs a specific task. It's reusable, which means you can call it multiple times in your program. This helps to organize your code, make it more readable, and avoid repetition.

**Why Do We Use Functions? **

We use functions in Python for several reasons:

- **Code Reusability:** You can call a function multiple times instead of repeating code. This saves time and effort.
- **Modularity:** Breaking down a large program into smaller, manageable chunks (functions) makes it easier to understand and maintain.
- **Avoiding Repetition:** Functions prevent you from writing the same code over and over, reducing errors and improving efficiency."

## 6.2 How to Write a Function

To define a function, you use the `def` keyword followed by the function name, parentheses for parameters, and a colon. The code block that defines the function is indented.

**Syntax:**

```
def function_name(parameters):
  # Function body
  # Code to be executed
```

### Example 6.1: Defining and Calling a Function

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

# Calling the function
greet("Ahmad")  # Output: Hello, Ahmad!
```

**Explanation:**

- `def greet(name):` defines a function named `greet` that takes one parameter, `name`.
- `print("Hello,", name + "!")` is the function body, which prints a greeting message using the provided name.

- `greet("Ahmad")` calls the function with the argument "Ahmad".

**Key Points:**

- **Parameters:** These are variables passed to the function when it's called. For more details, See Appendix B: Parameters and Arguments
- **Return Value:** A function can optionally return a value using the `return` statement.
- **Docstrings:** It's good practice to include a docstring (a string that explains the function's purpose) after the function definition.

## 6.3 **Return Statement**

- Functions can return values using the `return` keyword.

### Example 6.2: Function with a Return Value

```
def add(x, y):
  return x + y

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

### Task 6.1: Create a Function to Calculate the Area of a Rectangle

**Function Requirements:**

1. Define a function named `calculate_area` that takes two parameters: `length` and `width`.
2. The function should calculate the area of the rectangle (Area = Length × Width) and return the result.

**Input:**

- Length (a positive float or integer)
- Width (a positive float or integer)

**Output:**

- The area of the rectangle (a float)

**Expected Output**

```
The area of the rectangle with length 5 and width 3 is: 15
```

**Additional Test Cases**

1. **Input:** `length = 7`, `width = 2`
    - **Output:** `The area of the rectangle with length 7 and width 2 is: 14`
2. **Input:** `length = 10.5`, `width = 4.2`
    - **Output:** `The area of the rectangle with length 10.5 and width 4.2 is: 44.1`

### Task 6.2: Create a Function to Check if a Number is Even or Odd

**Function Requirements:**

1. Define a function named `is_even` that takes one parameter: `number`.
2. The function should determine if the number is even or odd.
3. It should return the string `"Even"` if the number is even, and `"Odd"` if the number is odd.

**Input:**

- A single integer (positive or negative)

**Output:**

- A string: either `"Even"` or `"Odd"`

**Expected Output**

```
The number 4 is: Even
```

### Additional Test Cases

Encourage beginners to test the function with various numbers:

1. **Input:** `num = 7`
    - **Output:** `The number 7 is: Odd`
2. **Input:** `num = -2`
    - **Output:** `The number -2 is: Even`
3. **Input:** `num = 0`
    - **Output:** `The number 0 is: Even`

## 6.4 **Default Arguments**

- You can assign default values to parameters, which makes them optional when calling the function.
- Video: Learn How to Use Default Parameters in Function Definition

### Example 6.3:

```
def greet(name, message="Hello"):
    print(f"{message}, {name}!")
```

```
greet("Alice")           # Uses default message "Hello"
greet("Alice", "Hi")      # Overrides default with "Hi"
```

## 6.5 **Keyword Arguments**

- Python allows you to specify arguments by name, making your code more readable.
- Example:

Example 6.4:

```
def multiply(a, b):
    return a * b

result = multiply(b=3, a=5)  # You can specify arguments in any order
```

Task 6.3: Basic Default Argument Task

- **Task:** Write a function `greet` that takes a name as an argument and a greeting message with a default value of "Hello". If no greeting is provided, the function should use "Hello."
- **Example:** `greet("Alice")` should output `"Hello, Alice!"` and `greet("Alice", "Good morning")` should output `"Good morning, Alice!"`

Task 4: Create a Function with Multiple Defaults and Modify One

- **Task:** Write a function `calc_price` that accepts `price`, `tax=0.05`, and `discount=0`. Calculate the final price after applying tax and discount. Test with various keyword arguments to see how changes affect the result.
- 
- **Example:** `calc_price(100)`, `calc_price(100, discount=0.1)`, `calc_price(100, tax=0.07, discount=0.1)`

video: Guard Statements in Python: Explained Simply

Python Quiz - Functions

## Fix the Errors

1. Fixing Errors in Function Calls and Assignments

```
def greet():
    print("Hello World!")
```

```
greeting = greet
```

## True/False (Mark T for True and F for False)

## Multiple Choice (Select the best answer)

1. What is the output of the following code?

```
def myfunction(val):
    return val % 4 == 0
print(myfunction (13) or myfunction (8))
```

- A) 0
- B) 13
- C) False
- D) True
- E) 3.5
- **Watch the Video Tutorial for the Answer:** https://youtu.be/laKpsLlq60I

2. **What is the output of the following code?** Python Quiz #88

```
def greet(name="User"):
  return "Hello, " + name
print(greet("Ahmad"))
```

```
A) `Hello, User`
B) `Hello, Ahmad`
C) `Hello`
D) `Error`
```

3. **What is the output of the following code?** Python Quiz #89

```
def my_function():
  pass
print(my_function())
```

```
- A) `None`
- B) `0`
- C) `True`
- D) `Error`
```

4. **What is the output of the following code?** Python Quiz #90

```
def my_func(a, b=2, c=3):
    return a + b + c
print(my_func(5, c=4))
```

   - A) `11`
   - B) `12`
   - C) `10`
   - D) `Error`

5. **Which of the following function calls is invalid for this function definition?** [Python Quiz #93]

```
def my_func(a, b, c=3):
    return a + b + c
```

- A) `my_func(1, 2)`
- B) `my_func(1, 2, 4)`
- C) `my_func(a=1, b=2, c=5)`
- D) `my_func(1, c=4, b=2, 5)`

6. **What is the output of the following code?** [Python Quiz #91]

```
def change_value(x):
    x = 10

num = 5
change_value(num)
print(num)
```

   - A) `5`
   - B) `10`
   - C) `Error`
   - D) `None`

7. **What is the output of the following code?** [Python Quiz #96]

```
def greet(name: str) -> str:
    return "Hello, " + name + "!"
```

```
result = greet(5)
print(result)
```

```
- A) Hello, 5!
- B) TypeError
- C) None
- D) Hello, !
```

8. **What will be the output of this code?** [Python Quiz #87]

```
def func(x, y=2):
    return x * y
print(func(3))
```

   - A) 2
   - B) 6
   - C) 3
   - D) Error

9. **What is the main purpose of a function in Python?**

- A) To group a set of related code into a single unit
- B) To create a new type of data
- C) To write a program in a single line
- D) To change the value of global variables

10. **What is the purpose of the return statement in a function in Python?**

- A) To print the output of the function
- B) To exit the function and return a value
- C) To execute the function without returning anything
- D) To stop the function and start a new one

11. **What is the correct way to define a function in Python?**

- A) function my_function():
- B) def my_function():
- C) define my_function():
- D) my_function() {

12. **What happens if you don't include a return statement in a function?**

- A) The function will return None.
- B) The function will cause an error.
- C) The function will return 0.
- D) The function will return the last variable used.

## Exercises

### Beginner

1. Write a Python program that takes two numbers as input and prints their sum.

- [**Watch the Solution Now** ✨](https://www.youtube.com/watch?v=CQHXsGnUns0&list=PLKYRx0Ibk7Vi-CC7ik98qT0VKK0F7ikja&index=24

2. Write a function `sum3(num1,num3,num3)` that takes three numbers as input and returns the sum.
3. Write a function `SumNum(num1)` that takes a number as input and returns the sum of numbers from 1 to that number (num1).
4. Write a function `sumSquares(x)` that takes a list of numbers as input and returns the sum of their squares.
5. Write a function `order_food` that accepts a `main_dish`, an optional `side_dish` with a default value of `"fries"`, and an optional `drink` with a default of `"water"`. Call this function using both positional and keyword arguments.
    - **Example:** `order_food("burger")`, `order_food("pizza", drink="soda")`, and `order_food("salad", side_dish="breadsticks", drink="juice")`
6. Create a function `introduce` that takes three parameters: `name`, `age`, and `city`. Set default values for `age` to 18 and `city` to "Unknown". Test calling the function with different combinations of arguments.
    - **Example:** `introduce("John")`, `introduce("John", 25)`, `introduce("John", 25, "New York")`
7. Define a function `student_profile` that accepts `name`, `grade`, and `subject` with a default value of `"Math"`. Use keyword arguments to call this function in different orders.
    - **Example:** `student_profile(grade="A", name="Emma")` and `student_profile("Sophia", "B", subject="History")`
8. Write a function `add_numbers` that takes two numbers and returns their sum.
    - **Example:** `add_numbers(3, 5)` should return `8`
9. Write a function `circle_area` that calculates the area of a circle given its radius. Use the formula: area = π * radius² (you can use `3.14` for π).
    - **Example:** `circle_area(3)` should return approximately `28.26`
10. Write a function `celsius_to_fahrenheit` that takes a temperature in Celsius and converts it to Fahrenheit using the formula: Fahrenheit = Celsius * (9/5) + 32.

- **Example:** `celsius_to_fahrenheit(25)` should return `77.0`

11. Write a function `is_even` that takes a number and returns `True` if the number is even, and `False` otherwise.

- **Example:** `is_even(4)` should return `True` and `is_even(7)` should return `False`

12. Write a function `max_of_three` that takes three numbers and returns the largest one.

- **Example:** `max_of_three(3, 7, 5)` should return `7`

13. Write a function `simple_interest` that calculates simple interest given `principal`, `rate`, and `time` using the formula: interest = (principal * rate * time) / 100.

- **Example:** `simple_interest(1000, 5, 2)` should return `100.0`

## Intermediate

14. Write a function `is_prime` that checks if a number is prime. A prime number has only two divisors: 1 and itself.

- **Example:** `is_prime(7)` should return `True` and `is_prime(8)` should return `False`

15. Write a function `factorial` that takes a number and returns its factorial. (Factorial of 5 is `5 * 4 * 3 * 2 * 1 = 120`)

- **Example:** `factorial(5)` should return `120`

16. Write a program with three functions:

17. **`isEven(n)`:** This function takes an integer `n` as input and returns `True` if `n` is even and `False` otherwise. You can use the modulo operator (`%`) to check for evenness.

18. **`printTable(n)`:** This function takes an integer `n` as input and prints its multiplication table. The table should show the product of `n` with each number from 1 to 10, formatted like `n * i = n * i`, where `i` is the current number in the loop.

19. **`main`:** The main program should:

    - Prompt the user to enter an integer.
    - Use the `isEven(n)` function to check if the entered number is even.
    - If the number is even, call the `printTable(n)` function to print its multiplication table.
    - If the number is odd, print a message indicating the number is odd and not eligible for printing a table.

**Example output:**

```
Enter an integer: 4
4 is even! Here's its multiplication table:
4 * 1 = 4
4 * 2 = 8
4 * 3 = 12
...
4 * 10 = 40
```

17. Write a function `fibonacci` that takes a number `n` and returns the `n`th number in the Fibonacci sequence.

- **Example:** `fibonacci(5)` should return `5` (sequence: 0, 1, 1, 2, 3, 5, ...)

### Advanced

## Projects

1. **Create a Number Guessing Game**

**Function Requirements:**

1. Define a function named `guess_number` that takes no parameters.
2. The function should randomly select a number between 1 and 100.
3. Prompt the user to guess the number, providing feedback on whether their guess is too high, too low, or correct.
4. The game should continue until the user guesses the correct number.
5. Once the user guesses correctly, the function should print a congratulatory message and the number of attempts it took.

**Input:**

- User input (guesses) from the console

**Output:**

- Feedback on each guess and a congratulatory message upon a correct guess

### Expected Output

When the user plays the game, the interaction might look like this:

```
Welcome to the Number Guessing Game!
Guess a number between 1 and 100.
Enter your guess: 50
Too low! Try again.
Enter your guess: 75
Too high! Try again.
Enter your guess: 60
Congratulations! You've guessed the number 60 in 3 attempts.
```

### Notes for Beginners

1. **Random Number Generation:** You can use the `random` module to select a random number.
2. **Input Handling:** Use `input()` to get the user's guess and convert it to an integer.
3. **Loops and Conditionals:** This task will help practice loops for continuous guessing and conditionals for feedback.

## Review Questions

## References and Bibliography

**Which of the following will cause a syntax error due to incorrect indentation in Python?**

A)

```
print("Hello World!")
```

B)

```
def my_function():
print("Hello World!")
```

C)

```
if x == 10:
    print("x is 10")
```

D)

```
x = 10
```

*Answer:* B

# Appendices

## Appendix A: Parameters and Arguments

Parameters are defined by the names that appear in a function definition, whereas arguments are the values actually passed to a function when calling it. Parameters define what kind of arguments a function can accept.

### Parameters

- **Definition:** Variables declared in a function's definition.
- **Purpose:** Act as placeholders for values that will be passed to the function when it's called.
- **Location:** Inside the function's parentheses.

### Arguments

- **Definition:** Actual values passed to a function when it's called.
- **Purpose:** Provide data for the function to work with.
- **Location:** Inside the function call parentheses.

See also the `FAQ` question of `Python Documentation` on the difference between arguments and parameters.

### Example 6.3: Defining a Function with Parameters and Passing Arguments

```
def greet(name):  # 'name' is a parameter
    print("Hello,", name + "!")

greet("Alice")  # "Alice" is an argument
```

In this example:

- `name` is a parameter in the function `greet`.
- `"Alice"` is an argument passed to the function when it's called.

**To summarize:**

- Parameters are defined *before* the function is called.
- Arguments are provided *when* the function is called.

**Think of it like this:**

- A parameter is like an empty box that expects a value.
- An argument is the value you put into the box.

Sure! Here's a simple task for beginners to practice writing functions in Python, along with input and output examples.