

### 3.2 Guided Practice Python Functions with AI

#### Question 1: What is a function in Python?

- **Syntax:** Defined with the `def` keyword, followed by the function name and parentheses.
- **Parameters:** Functions can accept inputs (arguments) to customize behavior.
- **Return values:** Functions can send back results using the `return` statement.
- **Indentation:** Code inside a function must be indented to indicate scope.
- **Calling:** Functions are executed by writing their name followed by parentheses.

Sources: [Programiz](#) +2

#### Example

Python ^

Copy

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

# Calling the function
print(greet("Alice"))
print(greet("Bob"))
```

👉 Output:

Code ^

Copy

```
Hello, Alice!
Hello, Bob!
```



Help

Message Copilot



Smart (GPT-5) ▾



Question 2: How do you define a Python function named `add`, which adds two numbers together? It receives two parameters.

You can define a Python function named `add` that takes two parameters and returns their sum like this:

Python ^

 Copy

```
# Define the function
def add(x, y):
    return x + y

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

## 💡 Explanation

- `def add(x, y):` → Defines a function called `add` with two parameters (`x` and `y`).
- `return x + y` → Returns the sum of the two parameters.
- `add(5, 3)` → Calls the function with arguments `5` and `3`.
- `print(result)` → Displays the result (`8`).

This is a **user-defined function** that demonstrates how Python functions can accept inputs and produce outputs.

Would you like me to also show you how to write this same `add` function as a **lambda function** (a shorter, anonymous version)?

     |  Edit in a page

Message Copilot



Smart (GPT-5) ▾



Question 3: What happens when you call a function?

- **Execution jumps to the function:** Python transfers control to the block of code defined inside the function.
- **Arguments are passed in:** Any values you provide in the parentheses are assigned to the function's parameters.
- **Code inside runs:** The indented statements inside the function body execute in order.
- **Return value (if any):** If the function has a `return` statement, Python sends that value back to the caller. If not, it returns `None` by default.
- **Control returns to the caller:** After the function finishes, Python resumes execution at the point where the function was called.

## Example

Python ^

 Copy

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

result = add(5, 3)    # Function call
print(result)
```

👉 Step-by-step:

1. `add(5, 3)` is called.
2. `x` becomes `5`, `y` becomes `3`.
3. Inside the function, `x + y` is computed → `8`.
4. `return 8` sends the result back.



5  

Message Copilot



Smart (GPT-5) ▾



Question 4: What is the return statement in a Python function?

## 🔑 Key Points About `return`

- **Ends function execution:** Once `return` is executed, the function stops running immediately.
- **Sends back a value:** The expression after `return` is passed back to the caller.
- **Optional:** If no `return` is used, the function automatically returns `None`.
- **Multiple returns:** A function can have multiple `return` statements, but only one is executed per call depending on conditions.
- **Return types:** Can return any Python object — numbers, strings, lists, dictionaries, even other functions.

## 📘 Example 1: Returning a Value

Python ^

Copy

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

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

Here, `return x + y` sends the sum back to the caller.

## 📘 Example 2: Returning Nothing

Message Copilot



Smart (GPT-5) ▾

0

Question 5: How do you pass arguments to a function?

## 💡 Ways to Pass Arguments

- **Positional arguments**

Values are matched to parameters in the order they are given.

Python ^

Copy

```
def greet(name, age):
    print(f"Hello {name}, you are {age} years old.")

greet("Alice", 25) # name="Alice", age=25
```

- **Keyword arguments**

You specify which parameter each value belongs to.

Python ^

Copy

```
greet(age=30, name="Bob") # Explicitly matched
```

- **Default arguments**

Parameters can have default values if no argument is provided.

Python ^

Copy

```
def greet(name, age=18):
    print(f"Hello {name}, you are {age} years old.")

greet("Charlie") # Uses default age=18
```

- **Variable-length arguments (`*args`)**

Collects extra positional arguments into a tuple.

Python ^



Copy

Message Copilot



Smart (GPT-5) ▾

