python def function

In Python, `def` is the keyword used to **define a function**. A function is a reusable block of code that performs a specific task.

Here is the basic syntax and examples.

Basic Syntax

This is the template for creating any function in Python:

```
Python

def function_name(parameter1, parameter2):
    """
    This is the docstring. It explains what the function does.
    """
    # Code block: The indented body of the function
    # ...
    return some_value # Optional: sends a value back
```

Key Parts Explained

1. `def` : The keyword that tells Python you are about to **def**ine a function.
2. `function_name` : The name you give to your function. It should be descriptive and follow Python's `snake_case` naming convention (all lowercase with underscores).
3. `()` : Parentheses. This is where you list any **parameters** (inputs) the function needs. If the function takes no inputs, the parentheses are still required, but they will be empty.
4. `:` (colon) : Marks the end of the function's header.
5. **Indented Block**: The lines of code inside the function must be indented (usually by 4 spaces). This is the function's "body."
6. `"""Docstring"""` (Optional but recommended): A string literal that explains the function's purpose, what its parameters are, and what it returns.
7. `return` (Optional): The keyword used to send a value back from the function. If you don't include a `return` statement, the function automatically returns a special value called `None` .

Examples

Here are three common types of functions:

1. Simple Function (No inputs, no return value)

This function just performs an action (like printing to the screen).

```
Python

# --- Definition ---
def greet():
    """Prints a simple greeting."""
    print("Hello, world!")

# --- How to use (call) it ---
greet()
```

Output:

```
Hello, world!
```

2. Function with Parameters (Takes inputs)

This function takes an input (a `name`) and uses it.

```
Python

# --- Definition ---
def greet_person(name):
    """Prints a personalized greeting."""
    print(f"Hello, {name}! Welcome.")

# --- How to use (call) it ---
greet_person("Alice")
greet_person("Bob")
```

Output:

```
Hello, Alice! Welcome.
Hello, Bob! Welcome.
```

3. Function with Parameters and a Return Value

This function takes inputs, processes them, and **returns** a result that you can store in a variable.

```
Python

# --- Definition ---
def add_numbers(num1, num2):
    """Adds two numbers together and returns the sum."""
    total = num1 + num2
    return total

# --- How to use (call) it ---
sum_result = add_numbers(5, 10)

print(f"The result is: {sum_result}")
print(f"10 + 15 is: {add_numbers(10, 15)}")
```

Output:

```
The result is: 15
10 + 15 is: 25
```

Would you like to know about more advanced function features, like default parameter values or `*args` and `**kwargs` ?

what is no patameter no return etc

You're asking about the four basic types of functions, based on whether they **take inputs** (parameters) or **give back outputs** (return values).

Think of a function like a machine:

- **Parameters** are the raw materials you put *into* the machine.
- **Return Values** are the finished products that come *out* of the machine.

Here are the four combinations.

1. No Parameters, No Return Value

This type of function is like a simple button. You call it, and it just *does something* (an action). It doesn't need any input from you, and it doesn't hand anything back.

- **Analogy:** Pressing a button on a coffee machine that's pre-programmed to make one specific drink. You give it no new info, and it just makes the drink (it doesn't hand you a *value*, it just performs the action).
- **Purpose:** To run the exact same block of code every time.

```
Python

def show_menu():
    """Prints a fixed menu to the screen."""
    print("--- MENU ---")
    print("1. File")
    print("2. Edit")
    print("3. View")

# --- How to use it ---
show_menu()
```

Output:

```
--- MENU ---
1. File
2. Edit
3. View
```

---

2. With Parameters, No Return Value

This function *takes input* (parameters) and uses that input to *do something*. It doesn't hand a value back to your main code.

- **Analogy:** A printer. You give it a document (a parameter), and it performs an action (prints the document). It doesn't give you anything *back* (except the physical paper, but it doesn't return data to your program).
- **Purpose:** To perform an action that can be customized based on the input.

```
Python

def greet_person(name):
    """Takes a name and prints a custom greeting."""
    print(f"Hello, {name}. Welcome to the team!")

# --- How to use it ---
greet_person("Maria")
greet_person("David")
```

Output:

```
Hello, Maria. Welcome to the team!
Hello, David. Welcome to the team!
```

---

3. No Parameters, With Return Value

This function doesn't need any input from you. It runs, does some work (like getting data from a sensor, checking the time, or rolling a die), and then *hands a value back* to your code.

- **Analogy:** A digital thermometer on the wall. You don't give it any input, you just *ask* it for the temperature, and it *returns* a value (like "22°C").
- **Purpose:** To get a value that might change, like the current time, a random number, or user input.

```
Python

import random

def get_random_dice_roll():
    """Returns a random number between 1 and 6."""
    roll = random.randint(1, 6)
    return roll

# --- How to use it ---
# The 'return' value is assigned to the variable
my_roll = get_random_dice_roll()
print(f"You rolled a: {my_roll}")

if get_random_dice_roll() == 4:
    print("You rolled a 4!")
```

Output (will be random):

```
You rolled a: 4
```

---

4. With Parameters, With Return Value

This is the most common type of function. It takes *input* (parameters), processes that input, and *returns a new value* as a result.

- **Analogy:** A calculator. You give it numbers and an operation (parameters, like `5` and `10`), and it *returns* a result (like `15`).
- **Purpose:** To transform data and give you the result.

```
Python

def add(num1, num2):
    """Adds two numbers together and returns the sum."""
    total = num1 + num2
    return total

# --- How to use it ---
sum_result = add(5, 10)
print(f"The result is: {sum_result}")
```

Output:

```
The result is: 15
```

Summary Table

Function Type	Takes Input? (Parameters)	Gives Output? (Return Value)
Type 1	❌ No	❌ No
Type 2	✅ Yes	❌ No
Type 3	❌ No	✅ Yes
Type 4	✅ Yes	✅ Yes

Export to Sheets

Would you like to see how to give parameters a default value, making them optional?