Python Functions Guide

What is a Function?

A function is like a recipe. You give it ingredients (inputs) and it makes something for you (output). Functions help you avoid writing the same code over and over.

How to Make a Function

Use the `def` keyword to create a function. **Important**: Press Tab or use 4 spaces before each line inside the function:

```
def say_hello():
    print("Hello!")

# Call the function
say_hello()
```

Output:

```
Hello!
```

Functions with Inputs

You can give functions inputs called **parameters**. Remember to indent with Tab:

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

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

Output:

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

Functions that Return Values

Functions can give you something back using `return`. Use Tab to indent:

```
def add_numbers(a, b):
    result = a + b
    return result

answer = add_numbers(5, 3)
print(answer)  # Output: 8
```

Output:

```
8
```

Different Value Types in Functions

Numbers

```
def multiply(x, y):
    return x * y

result = multiply(4, 7)
print(result)  # Output: 28
```

Output:

```
28
```

Strings (Text)

```
def make_full_name(first, last):
    return first + " " + last

name = make_full_name("John", "Doe")
print(name)  # Output: John Doe
```

Output:

```
John Doe
```

Lists

```
def get_first_item(my_list):
    return my_list[0]

fruits = ["apple", "banana", "orange"]
first_fruit = get_first_item(fruits)
print(first_fruit)  # Output: apple
```

Output:

```
apple
```

Booleans (True/False)

```
def is_adult(age):
    return age >= 18

print(is_adult(25))  # Output: True
print(is_adult(15))  # Output: False
```

Output:

```
True
False
```

Functions with Multiple Parameters

```
def calculate_area(length, width):
    return length * width

area = calculate_area(5, 3)
print(f"Area is: {area}")  # Output: Area is: 15
```

Output:

```
Area is: 15
```

Default Values

You can set default values for parameters:

```
def greet_person(name, greeting="Hello"):
    return f"{greeting}, {name}!"

print(greet_person("Alice"))           # Output: Hello, Alice!
print(greet_person("Bob", "Hi"))       # Output: Hi, Bob!
```

Output:

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

Print Function Examples

The `print()` function shows text on screen:

Basic Printing

```
print("Hello World")
print(42)
print(3.14)
```

Output:

```
Hello World
42
3.14
```

Printing Variables

```
name = "Sarah"
age = 20
print(name)
print(age)
```

Output:

```
Sarah
20
```

Printing Multiple Things

```
print("Name:", name, "Age:", age)
print("I am", age, "years old")
```

Output:

```
Name: Sarah Age: 20
I am 20 years old
```

Using f-strings (Easy Way)

```
name = "Mike"
score = 95
print(f"Student: {name}, Score: {score}")
```

Output:

```
Student: Mike, Score: 95
```

Print with Separators

```
print("apple", "banana", "orange", sep=", ")
# Output: apple, banana, orange
```

Output:

```
apple, banana, orange
```

Print on Same Line

```
print("Loading", end="")
print(".", end="")
print(".", end="")
print(".")
# Output: Loading...
```

Output:

```
Loading...
```

Print Different Types Together

```
student = "Emma"
grade = 85
passed = True

print(f"Student {student} got {grade}% - Passed: {passed}")
```

Output:

```
Student Emma got 85% - Passed: True
```

Quick Tips

1. **Function names** should describe what they do
2. **Parameters** go in parentheses `()`
3. **Return** gives back a value
4. **Call** a function by using its name with `()`
5. **Indent** your code inside functions with Tab or 4 spaces

Practice Example

```
def calculate_grade(score):
    if score >= 90:
        return "A"
    elif score >= 80:
        return "B"
    elif score >= 70:
        return "C"
    else:
        return "F"

# Test the function
student_score = 85
letter_grade = calculate_grade(student_score)
print(f"Score: {student_score}, Grade: {letter_grade}")
```

Output:

```
Score: 85, Grade: B
```

Functions make your code cleaner and easier to use. Practice making your own!