Python Programming Basics: Series 3

Functions and Dictionary Iteration

Table of contents

1 Introduction 1

2 Functions Revisited 2

3 Iterating Through Dictionaries 4

4 Function Examples with Dictionaries 7

5 Conclusion 9

6 Additional Resources 10

7 Acknowledgments 10

1 Introduction

Welcome to Series 3 of our Python programming journey! In this series, we'll dive deeper into:

- **Functions**: Understanding how to define and call them in scripts, with detailed explanations and examples.
- **Dictionary Iteration**: Learning how to iterate through dictionaries using `.keys()`, `.values()`, and `.items()` in `for` loops.

We'll build upon what we've learned in Series 2 and explore these concepts through theory, examples, and exercises.

Let's get started!

---

2 Functions Revisited

A **function** is a reusable block of code that performs a specific task. Functions help us organize our code, make it more readable, and avoid repetition.

2.1 Defining a Function

To define a function in Python, we use the `def` keyword, followed by the function name and parentheses `()` which may include parameters.

```
def function_name(parameters):
    # Function body
    return result
```

- **`def`**: Keyword to define a function.
- **`function_name`**: Name of the function (should be descriptive).
- **`parameters`**: Inputs to the function (optional).
- **`return`**: Keyword to return a value from the function (optional).

2.2 Calling a Function

To use a function, we **call** it by writing its name followed by parentheses, passing any required arguments.

```
# Calling the function
result = function_name(arguments)
```

2.3 Example: Function Definition and Call

```
# Defining a function to add two numbers
def add_numbers(a, b):
    sum = a + b
    return sum

# Calling the function in the main script
number1 = 5
number2 = 10
result = add_numbers(number1, number2)

print(f"The sum is: {result}")  # Output: The sum is: 15
```

```
The sum is: 15
```

2.3.1 Explanation

- **Defining the Function**:
  - `def add_numbers(a, b):` defines a function named `add_numbers` with parameters `a` and `b`.
  - Inside the function, we calculate the sum and `return` it.
- **Calling the Function**:
  - We assign values to `number1` and `number2`.
  - We call `add_numbers(number1, number2)` and store the result in `result`.
  - We print the result.

2.4 Functions in Scripts

When writing scripts, it's common to define functions at the top and then call them in the main part of the script.

```python
# Function definitions
def greet(name):
    return f"Hello, {name}!"

def square(number):
    return number ** 2

# Main script
if __name__ == "__main__":
    name = "Alice"
    print(greet(name))  # Output: Hello, Alice!

    num = 4
    print(f"The square of {num} is {square(num)}")  # Output: The square of 4 is 16
```

```
Hello, Alice!
The square of 4 is 16
```

2.4.1 Explanation

- **`if __name__ == "__main__":`**: This condition checks if the script is being run directly (not imported as a module). It's a common practice to include this in Python scripts.
- **Function Definitions**: We define `greet` and `square` functions.
- **Main Script**: We call the functions with appropriate arguments and print the results.

2.5 Exercise 1: Writing and Calling Functions

Question

Create a function named `multiply_numbers` that:

- Accepts two parameters, `x` and `y`.
- Returns the product of `x` and `y`.

Then, in the main part of your script:

- Assign the values `7` and `8` to variables `a` and `b`.
- Call the `multiply_numbers` function with `a` and `b`.
- Print the result.

Hints:

- Remember to define the function before calling it.

```
# Define the function multiply_numbers
def multiply_numbers(x, y):
    # Function body
    ?

# Main script
if __name__ == "__main__":
    # Assign values to a and b
    a = ?
    b = ?

    # Call the function and print the result
    result = ?
    print(f"The product of {a} and {b} is: {result}")
```

Expected Output:

```
The product of 7 and 8 is: 56
```

---

3 Iterating Through Dictionaries

Dictionaries store data in key-value pairs. Sometimes, we need to loop through a dictionary to access its keys, values, or both.

3.1 Iterating Over Keys

Use `.keys()` to get all the keys in a dictionary.

```python
student = {
    "name": "Bob",
    "age": 15,
    "grade": "10th"
}

for key in student.keys():
    print(key)

# Output:
# name
# age
# grade
```

```
name
age
grade
```

3.2 Iterating Over Values

Use `.values()` to get all the values.

```python
for value in student.values():
    print(value)

# Output:
# Bob
# 15
# 10th
```

```
Bob
15
10th
```

3.3 Iterating Over Key-Value Pairs

Use `.items()` to get both keys and values as pairs (tuples).

```
for key, value in student.items():
    print(f"{key}: {value}")

# Output:
# name: Bob
# age: 15
# grade: 10th
```

```
name: Bob
age: 15
grade: 10th
```

3.4 Exercise 2: Dictionary Iteration

Question

Given the dictionary `inventory`:

```
inventory = {
    "apples": 5,
    "oranges": 3,
    "bananas": 2
}
```

Write code to:

1. Print all the fruit names (keys).
2. Print all the quantities (values).
3. Print each fruit and its quantity in the format: `"We have X Y"`, where `X` is the quantity and `Y` is the fruit name.

Hint:

- Use `for` loops with `.keys()`, `.values()`, and `.items()`.

```
inventory = {
    "apples": 5,
    "oranges": 3,
    "bananas": 2
}

# 1. Print all the fruit names
```

```
for key in ?:
    print(?)

# 2. Print all the quantities
for value in ?:
    print(?)

# 3. Print each fruit and its quantity
for key, value in ?:
    print(f"We have {value} {key}")
```

Expected Output:

```
apples
oranges
bananas
5
3
2
We have 5 apples
We have 3 oranges
We have 2 bananas
```

---

4 Function Examples with Dictionaries

Functions can return dictionaries, accept dictionaries as parameters, or both.

4.1 Example: Function Returning a Dictionary

```
def create_student(name, age, grade):
    student = {
        "name": name,
        "age": age,
        "grade": grade
    }
    return student

# Main script
if __name__ == "__main__":
    student1 = create_student("Alice", 14, "9th")
    print(student1)
    # Output: {'name': 'Alice', 'age': 14, 'grade': '9th'}
```

```
{'name': 'Alice', 'age': 14, 'grade': '9th'}
```

4.2 Example: Function Processing a Dictionary

```
def total_inventory(inventory: dict) -> int:
    total = 0
    for quantity in inventory.values():
        total += quantity
    return total

# Main script
if __name__ == "__main__":
    inventory = {"apples": 5, "oranges": 3, "bananas": 2}
    total_items = total_inventory(inventory)
    print(f"Total items in inventory: {total_items}")
    # Output: Total items in inventory: 10
```

```
Total items in inventory: 10
```

4.3 Exercise 3: Function and Dictionary Practice

Question

Create a function named `count_vowels` that:

- Accepts a string `text`.
- Returns a dictionary where the keys are vowels (`'a'`, `'e'`, `'i'`, `'o'`, `'u'`) and the values are the counts of each vowel in `text`.

Then, in the main script:

- Define a variable `sample_text` with the value `"Programming is fun!"`.
- Call `count_vowels` with `sample_text`.
- Iterate through the resulting dictionary and print each vowel and its count.

Hints:

- Use `text.lower()` to handle uppercase letters.
- Initialize the dictionary with vowels set to zero.
- Iterate over each character in `text` using `for` loop and update counts.

```
# Define the function count_vowels
def count_vowels(text: str) -> dict:
    # Function body
    ?

# Main script
if __name__ == "__main__":
    # Define sample_text
    sample_text = ?

    # Call the function
    vowel_counts = ?

    # Iterate and print each vowel and its count
    for vowel, count in ?:
        print(f"{vowel}: {count}")
```

Expected Output:

```
a: 1
e: 0
i: 2
o: 2
u: 1
```

---

5 Conclusion

In this series, we've:

- **Revisited Functions**: Learned more about defining functions and calling them in scripts, with detailed examples.
- **Explored Dictionary Iteration**: Practiced iterating over dictionaries using `.keys()`, `.values()`, and `.items()`.

By working through these exercises, you've strengthened your understanding of functions and dictionaries in Python. These are fundamental skills that will help you as you continue to learn programming.

---

6 Additional Resources

- **Python Official Documentation**: Functions
- **Python Official Documentation**: Dictionaries
- **W3Schools Python Tutorial**: Python Functions
- **W3Schools Python Tutorial**: Python Dictionaries

---

7 Acknowledgments

We hope this series has helped you deepen your understanding of Python programming. Keep practicing, and don't hesitate to explore more about programming and problem-solving!