Python Programming: Concepts II

Oliver Bonham-Carter

Table of contents

Welcome to MORE Python Programming!

> Tip
>
> **Learning smaller parts of code will help you to implement specific types of functionalities in larger projects!**

What are literals, Again??

Let's Return to Literals (e.g., numbers, strings, booleans, etc.) and work with them in interesting ways using `print()`.

> Note
>
> In Python, print statements with an `f` prefix before the opening quotation mark denote f-strings, also known as formatted string literals. F-strings provide a concise and readable way to embed expressions inside string literals for formatting output.

Python Code Sample with F-strings

> **Note**
>
> Curiously, both of the below print statements print out the same results, but the code is not the same ...

```
firstName = "Robert"
lastName = "Paulson"
print(f"His name is: {firstName} {lastName}!") # f for auto formatting
print("His name is:",firstName, lastName,"!") # original structure
```

Output:

His name is: Robert Paulson!

His name is: Robert Paulson ! # A space appears?!

Operators

Operators let you do math, compare values, and more!

> **Important**
>
> **Examples:**
>
> - Arithmetic: `+`, `-`, `*`, `/`, `**` (power)
> - Comparison: `==`, `!=`, `<`, `>`
> - Assignment: `=`, `+=`, `-=`

Python Code Sample

```
x = 5
y = 2
sum = x + y
product = x * y
is_equal = (x == y)
x += 1
```

Interesting Application

Variables and operators are the backbone of calculators, games, and simulations!

Review Python Loops and Conditionals

Loops

Loops repeat actions. Use `for` and `while` loops.

Conditionals

Conditionals let your code make decisions using `if`, `elif`, and `else`.

Examples:

```
for i in range(5):
    print(i)

count = 0
while count < 3:
    print("Counting:", count)
    count += 1
    # Do not forget to increment
    # the counter, or you will create an infinite loop!

    if x > y:
        print("x is greater!")
    elif x == y:
        print("x equals y!")
    else:
        print("x is less!")
```

Sort of Interesting Application

Loops and conditionals can be used to determine positive, negative numbers, or zeros. Neat-O!

Examples:

```
num = 10
if num > 0:
    print("Positive number")
elif num == 0:
    print("Zero")
else:
    print("Negative number")
```

Output:

Positive

Nested Conditionals

You can nest loops, one inside the other.

Examples:

```
num = 15
if num >= 0:
    if num == 0:
        print("Zero")
    else:
        print("Positive number")
else:
    print("Negative number")
```

Output:

Positive

Logical Operator

Logical operators use `True` and `False` to determine outcomes.

Examples:

```
a = 10
b = 20
if a > 5 and b > 15:
    print("Both conditions are true")
if a > 5 or b < 15:
```

```
    print("At least one condition is true")
if not a < 5:
    print("a is not less than 5")
```

Output:

Both conditions are true At least one condition is true a is not less than 5

String Manipulation

We now look at how we can manipulate strings

Examples:

```
text = "Hello, World!"
print(text.lower())  # hello, world!
print(text.upper())  # HELLO, WORLD!
print(text.replace("World", "Python"))  # Hello, Python!
print(text.split(", "))  # ['Hello', 'World!']
print(text.strip("!"))  # Hello, World
```

Lists

- A list is a fundamental data structure in Python used to store an ordered collection of items.
- Lists maintain the order of elements, with each item accessible by its index (starting from 0).
- Lists are mutable, so their contents can be changed after creation (items can be added, removed, or modified).
- Lists can hold items of different data types, including other lists.
- Lists are created using square brackets, with elements separated by commas.

Demo of Lists

Examples:

```python
my_list = [1, 2, 3, 4, 5] # definition
print(my_list[0])  # Access first element
my_list.append(6)  # Add an element to the end
print(my_list)  # [1, 2, 3, 4, 5, 6]
my_list.remove(3)  # Remove element with value 3
print(my_list)  # [1, 2, 4, 5, 6]
print(len(my_list))  # Length of the list
```

Output:

1

[1, 2, 3, 4, 5, 6]

[1, 2, 4, 5, 6]

5

Dictionaries

A dictionary is a built-in data structure in Python used to store key-value pairs.

Dictionaries are unordered collections (as of Python 3.6+, they maintain insertion order), where each value is accessed using its unique key.

Dictionaries are mutable, so their contents can be changed after creation (items can be added, removed, or modified).

Keys in a dictionary must be unique and immutable (such as strings, numbers, or tuples), while values can be of any data type.

Dictionaries are created using curly braces {}, with key-value pairs separated by commas and a colon between each key and value.

Demo of Dictionaries

Examples:

```python
my_dict = {"name": "Alice", "age": 25, "is_student": True}  # definition
print(my_dict["name"])  # Access value by key
my_dict["age"] = 26  # Modify value
my_dict["city"] = "New York"  # Add new key-value pair
print(my_dict)  # {'name': 'Alice', 'age': 26, 'is_student': True, 'city': 'New York'}
del my_dict["is_student"]  # Remove a key-value pair
```

```
print(my_dict)  # {'name': 'Alice', 'age': 26, 'city': 'New York'}
print(len(my_dict))  # Number of key-value pairs
```

Output:

Alice

{‘name’: ‘Alice’, ‘age’: 26, ‘is_student’: True, ‘city’: ‘New York’}

{‘name’: ‘Alice’, ‘age’: 26, ‘city’: ‘New York’}

3

Coding Challenges

1. Variables and Operators

Challenge:
Create two variables, `a` and `b`, assign them integer values, and print their sum, difference, product, and quotient.

Solution:

```
# TODO
```

---

2. Working with Lists

Challenge:
Create a list of five numbers. Print the largest number in the list.

Solution:

```
# TODO
```

---

3. Dictionaries

Challenge:
Create a dictionary with three key-value pairs representing a person's name, age, and city. Print each key and its value.

Solution:

```
# TODO
```

---

4. For Loops

Challenge:
Use a for loop to print all even numbers from 1 to 10.

Solution:

```
# TODO
```

---

5. While Loops and Conditionals

Challenge:
Use a while loop to keep asking the user to enter a number until they enter a negative number.

Solution:

```
# TODO
```
