

# Python Programming: Concepts I

Guttag Chapter 2

Oliver Bonham-Carter

## Table of contents

Welcome to Python Programming!	1
Python Literals	2
Variables and Operators	3
Loops and Conditionals	3
Squaring Algorithms	4
Strings and Slicing	4
Python Challenge Exercises	5
Solutions to Challenge Exercises	6

## Welcome to Python Programming!



Tip

Let's make learning Python fun and practical!

## What are literals?

Literals are fixed values in your code. They can be numbers, strings, booleans, etc.

**i** Note

**Examples:** - 42 (integer) - 3.14 (float) - 'hello' (string) - True, False (boolean)

## Python Code Sample

```
age = 18
pi = 3.14159
greeting = "Hello, world!"
is_active = True
```

## Interesting Application

Use literals to set default values in games, apps, or data analysis scripts!

---

## Python Literals

Literals are fixed values written directly in your code and are used to assign values to variables or as standalone values in expressions.

```
"""
This example shows different types of literals.
"""
integer_literal = 42          # An integer literal
float_literal = 3.14          # A floating-point literal
string_literal = "hello"      # A string literal
boolean_literal = True        # A boolean literal

print(integer_literal, float_literal, string_literal, boolean_literal)
# Output: 42 3.14 hello True
```

This code demonstrates how to use different types of literals in Python.

---

## Variables and Operators

Variables store data for use in your program which can be manipulated by *operators* (e.g., addition, comparison, and assignment).

```
"""
This example shows how to use variables and operators.
"""
x = 10      # Assign 10 to x
y = 5       # Assign 5 to y
sum_xy = x + y      # Addition operator
diff_xy = x - y      # Subtraction operator
prod_xy = x * y      # Multiplication operator
is_equal = x == y    # Comparison operator
print(sum_xy, diff_xy, prod_xy, is_equal)
# Output: 15 5 50 False
```

This code uses variables and operators to do math and compare values.

---

## Loops and Conditionals

Loops repeat actions. Conditionals let your code make decisions.

```
"""
This example prints numbers and checks if they are even or odd.
"""

for i in range(1, 6): # Loop from 1 to 5
    if i % 2 == 0:     # Conditional: is i even?
        print(f"{i} is even")
    else:
        print(f"{i} is odd")
# Output:
# 1 is odd
# 2 is even
# and similar up to 5
```

This code loops through numbers and uses a conditional to check if each is even or odd.

---

## Squaring Algorithms

Squaring means multiplying a number by itself. There are several ways to do this in Python.

```
"""
This example shows three ways to square a number.
"""
def square(n):
    """Returns the square of n."""
    return n * n

num = 7
squ1 = num * num      # Multiplication
squ2 = num ** 2        # Exponentiation

print(f"squ1: {squ1}, squ2: {squ2}, square(num): {square(num)}")
# Output: squ1: 49, squ2: 49, square(num): 49
```

This code demonstrates three ways to square a number in Python.

---

## Strings and Slicing

Strings store text. Slicing lets you extract parts of a string.

```
"""
This example slices a string in different ways.
"""

text = "Python is awesome!"

first_word = text[:6]      # Get 'Python'
last_word = text[-8:]     # Get 'awesome!'
every_other = text[::2]    # Get every other character
```

```
print(first_word)    # Output: Python
print(last_word)     # Output: awesome!
print(every_other)   # Output: Pto saeo!
```

This code shows how to slice strings to get different parts or patterns.

---

## Python Challenge Exercises

Try these programming challenges to practice your Python skills!

---

### Challenge 1: Literal Mix-Up

Write code that uses at least three different types of literals (integer, float, string, boolean) and prints them in a single sentence.

```
# TODO
```

---

### Challenge 2: Variable Math

Create two variables, perform addition, subtraction, multiplication, and division, and print the results with clear labels.

```
# TODO
```

---

### Challenge 3: Loop & Conditional Fun

Write a loop that prints numbers from 1 to 10. For each number, print whether it is a multiple of 3 or not.

```
# TODO
```

---

### Challenge 4: Squaring Game

Write a function that takes a number and returns both its square and its cube. Print the results for the number 5.

```
# TODO
```

---

### Challenge 5: String Slicing Mystery

Given the string `mystery = "QuartoPythonRocks!"`, print:

- The first 6 characters
- The last 5 characters
- Every third character

```
# TODO
```

---

### Solutions to Challenge Exercises

---

### Solution 1: Literal Mix-Up

```
integer = 7
float_num = 2.5
text = "apples"
is_fresh = True
print(f"I bought {integer} {text}, each cost {float_num} dollars. Fresh? {is_fresh}")
```

This code uses integer, float, string, and boolean literals in a sentence.

---

### Solution 2: Variable Math

```
x = 12
y = 4
print(f"Addition: {x + y}")
print(f"Subtraction: {x - y}")
print(f"Multiplication: {x * y}")
print(f"Division: {x / y}")
```

This code performs math operations and prints results with labels.

---

### Solution 3: Loop & Conditional Fun

```
for n in range(1, 11):
    if n % 3 == 0:
        print(f"{n} is a multiple of 3")
    else:
        print(f"{n} is not a multiple of 3")
```

This code loops through numbers and checks for multiples of 3.

---

## Solution 4: Squaring Game

```
def square_and_cube(num):  
    """Returns the square and cube of num."""  
    return num ** 2, num ** 3  
  
sq, cu = square_and_cube(5)  
print(f"Square: {sq}, Cube: {cu}")
```

This function returns both the square and cube of a number.

---

## Solution 5: String Slicing Mystery

```
mystery = "QuartoPythonRocks!"  
print(mystery[:6])      # First 6 characters  
print(mystery[-5:])     # Last 5 characters  
print(mystery[::3])     # Every third character
```

This code slices the string in three different ways.