Course Coordinator

Python Programming

Table of contents

Preface **5**

Preface **5**

1 Basic Python Programming **7**

2 Sample Programs **9**
2.1 Introduction to Python 9
2.2 Data Types 9
2.2.1 Static Template 9
2.2.2 Interactive Cell 10
2.2.3 Variables 10
2.2.4 Input and Output Statements 11
2.2.5 Operators 11
2.2.6 Arithmetic Expressions 12
2.2.7 Operator Precedence 12
2.2.8 Evaluation of Expressions 12
2.2.9 Conditional Statements in Python 12

3 Summary **17**

References **19**

References **19**

Preface

This is a Quarto book.

To learn more about Quarto books visit https://quarto.org/docs/books.

1

Basic Python Programming

Dynamic evaluation of user provided code and data visualisation

This is a book created from markdown and executable code.

See Knuth (1984) for additional discussion of literate programming.

```
a=10
print(a)
```

2

Sample Programs

2.1 Introduction to Python

In this tutorial, we will cover the basics of Python programming, including data types, keywords, variables, input/output statements, operators, arithmetic expressions, operator precedence, and evaluation of expressions.

2.2 Data Types

Python supports several built-in data types. Let's explore some of the most common ones:

- **Integer (`int`)**: Represents whole numbers.
- **Floating Point (`float`)**: Represents decimal numbers.
- **String (`str`)**: Represents sequences of characters.
- **Boolean (`bool`)**: Represents `True` or `False`.

Example 1

2.2.1 Static Template

```
# Demonstrating different data types

# Integer
a = 10
print("Integer:", a, type(a))

# Float
b = 3.14
print("Float:", b, type(b))
```

```
# String
c = "Hello, Python!"
print("String:", c, type(c))

# Boolean
d = True
print("Boolean:", d, type(d))
```

```
Integer: 10 <class 'int'>
Float: 3.14 <class 'float'>
String: Hello, Python! <class 'str'>
Boolean: True <class 'bool'>
```

2.2.2 Interactive Cell

```
# Demonstrating different data types

# Integer
a = 10
print("Integer:", a, type(a))

# Float
b = 3.14
print("Float:", b, type(b))


# String
c = "Hello, Python!"
print("String:", c, type(c))

# Boolean
d = True
print("Boolean:", d, type(d))
```

2.2.3 Variables

Variables are used to store data in memory. A variable is created when you assign a value to it using the = operator.

```
# Variable assignment

x = 5
y = 2.5
z = x + y
```

```
print("x =", x)
print("y =", y)
print("z =", z)
```

2.2.4 Input and Output Statements

Python provides the input() function to take user input and the print() function to display output.

```
# Input and Output
name="justin"
age=32
#name = input("Enter your name: ")
#age = int(input("Enter your age: "))

print(f"Hello, {name}! You are {age} years old.")
```

2.2.5 Operators

Operators are special symbols used to perform operations on variables and values. Python supports several types of operators:

Arithmetic Operators: +, -, *, /, //, %, ** Comparison Operators: ==, !=, >, <, >=, <= Logical Operators: and, or, not Assignment Operators: =, +=, -=, *=, /=, //=, %=, **=

```
# Arithmetic Operations

a = 15
b = 4

addition = a + b
subtraction = a - b
multiplication = a * b
division = a / b
floor_division = a // b
modulus = a % b
exponentiation = a ** b

print("Addition:", addition)
print("Subtraction:", subtraction)
print("Multiplication:", multiplication)
print("Division:", division)
```

```
print("Floor Division:", floor_division)
print("Modulus:", modulus)
print("Exponentiation:", exponentiation)
```

2.2.6 Arithmetic Expressions

An arithmetic expression is a combination of numbers, operators, and variables that evaluates to a value.

```
# Evaluating arithmetic expressions

expression = (5 + 2) * (10 - 3) / 2 ** 2
print("Expression Result:", expression)
```

2.2.7 Operator Precedence

Operator precedence determines the order in which operations are performed in an expression. The following list shows the precedence from highest to lowest:

** (Exponentiation) *, /, //, % (Multiplication, Division, Floor Division, Modulus) +, - (Addition, Subtraction)

```
# Operator precedence

result = 5 + 3 * 2 ** 2 - 1
print("Operator Precedence Result:", result)
```

2.2.8 Evaluation of Expressions

Python evaluates expressions from left to right, following the precedence rules.

```
# Evaluation of expressions

value = (10 + 5) * 2 - 3 / 3
print("Evaluation Result:", value)
```

2.2.9 Conditional Statements in Python

Conditional statements in Python allow the execution of specific code blocks based on whether a condition is true or false. Let's explore various types of conditional statements.

2.2.9.1 The `if` Statement

The `if` statement tests a specific condition. If the condition is true, the code block under the `if` statement is executed.

Example

```
# Example of an if statement

number = 10

if number > 0:
    print(f"{number} is a positive number.")
```

Explanation The above program checks if number is greater than 0. Since 10 is greater than 0, the condition is true, and the message is printed.

2.2.9.2 The if-else Statement

The if-else statement allows you to execute one block of code if the condition is true and another block if it is false.

```
# Example of an if-else statement

number = -5

if number >= 0:
    print(f"{number} is a non-negative number.")
else:
    print(f"{number} is a negative number.")
```

Explanation In this example, the program checks if number is greater than or equal to 0. If true, it prints that the number is non-negative. Otherwise, it prints that the number is negative. Example 2

```
age = 20

if age >= 18:
    print("You are eligible to vote.")
else:
    print("You are not eligible to vote.")
```

Explanation This program checks if a person's age is greater than or equal to 18. If true, it prints that the person is eligible to vote. Otherwise, it states they are not eligible to vote. #### The elif Statement The elif statement, short for "else if," allows you to check multiple conditions sequentially. If one of the conditions is true, the corresponding block of code is executed.

```
# Example of an elif statement

number = 0

if number > 0:
    print(f"{number} is a positive number.")
elif number == 0:
    print(f"{number} is zero.")
else:
    print(f"{number} is a negative number.")
```

Explanation Here, the program checks three conditions: whether the number is positive, zero, or negative. The elif statement handles the case where number is exactly 0.

```
# Another example of an elif statement

marks = 85

if marks >= 90:
    grade = 'A'
elif marks >= 80:
    grade = 'B'
elif marks >= 70:
    grade = 'C'
else:
    grade = 'F'

print(f"Your grade is {grade}.")
```

Explanation This program assigns a grade based on the marks obtained. Depending on the range in which the marks fall, the corresponding grade is assigned and printed.

2.2.9.3 Nested if-else Statements

Nested if-else statements allow you to include an if-else statement inside another if-else block for handling more complex conditions.

```
# Example of nested if-else statements

number = 25

if number > 0:
    if number % 2 == 0:
        print(f"{number} is a positive even number.")
```

```
    else:
        print(f"{number} is a positive odd number.")
```

Explanation This example checks if a number is positive and then further checks whether it is even or odd using nested if-else statements.

```
# Another example of nested if-else statements

score = 92

if score >= 50:
    if score >= 90:
        print("Excellent!")
    else:
        print("Good job!")
else:
    print("Better luck next time.")
```

Explanation This program checks if a score is at least 50. If true, it further checks if the score is 90 or above, printing “Excellent!” if it is, and “Good job!” if it isn’t. If the score is below 50, it prints “Better luck next time.”

3

Summary

In summary, this book has no content whatsoever.

References

Knuth, Donald E. 1984. “Literate Programming.” *Comput. J.* 27 (2): 97–111. https://doi.org/10.1093/comjnl/27.2.97.