

Chapter 1: Review of the Fundamentals - Python Programming

Tutorial by Ayes Chinmay

July 2025

Contents

1	Introduction	2
2	Python Quick Start	2
3	Variables and Naming Names	2
4	Combined Assignment Operators	2
5	Summary of Python Arithmetic Operators	3
6	Elementary Data Types: Integer and Floating Point	3
7	Basic Input and Output	3
8	Function Definitions	3
9	The Python “if” Statement	4
10	The Python “while” Statement	4
11	A Couple of Cool Little Apps	4
12	Summary of Python Boolean Operators	4
13	Function Arguments and Return Values	5
14	The Forward Reference Problem	5
15	Python Strings	5
16	Python Lists (and a Cool Sorting App)	5
17	The “for” Statement and Ranges	6
18	Tuples	6
19	Dictionaries	6
20	Sets	6
21	Global and Local Variables	6

1 Introduction

This tutorial covers the fundamentals of Python programming as outlined in Chapter 1. It is designed for beginners and provides detailed explanations, examples, and exercises for each topic.

2 Python Quick Start

Python is a versatile, readable programming language. To start:

1. Download Python from <https://www.python.org/downloads/> (e.g., Python 3.12).
2. Install and verify by running `python -version` in a terminal.
3. Use an editor like IDLE or Visual Studio Code.

Example: Hello, World!

```
1 print("Hello, World!")
```

Run this in IDLE or save as `hello.py` and execute with `python hello.py`.

3 Variables and Naming Names

Variables store data. Use meaningful names following these rules:

- Start with a letter or underscore (e.g., `age`, `_count`).
- Use letters, numbers, or underscores (e.g., `user2`).
- Case-sensitive (e.g., `Age` \neq `age`).

Example:

```
1 name = "Alice"  
2 age = 25
```

4 Combined Assignment Operators

These combine arithmetic and assignment:

- `+=`: Add and assign (e.g., `x += 5` is `x = x + 5`).
- `-=`, `*=`, `/=`, `//=`, `%=`, `**=`.

Example:

```
1 x = 10  
2 x += 5 # x is now 15  
3 x *= 2 # x is now 30  
4 print(x)
```

5 Summary of Python Arithmetic Operators

Python supports:

- `+`: Addition
- `-`: Subtraction
- `*`: Multiplication
- `/`: Division
- `//`: Integer division
- `%`: Modulus
- `**`: Exponentiation

Example:

```
1 a = 10
2 b = 3
3 print(a + b, a / b, a // b, a ** 2) # 13, 3.333..., 3, 100
```

6 Elementary Data Types: Integer and Floating Point

- **Integer**: Whole numbers (e.g., 42, -7).
- **Float**: Decimal numbers (e.g., 3.14, 0.001).

Example:

```
1 integer = 100
2 floating = 2.5
3 print(integer * floating) # 250.0
```

7 Basic Input and Output

Use `print()` for output and `input()` for user input. Example:

```
1 name = input("Enter your name: ")
2 print("Hello, " + name)
```

Output (if user enters "Alice"):

```
Enter your name: Alice
Hello, Alice
```

8 Function Definitions

Functions are reusable code blocks defined with `def`. Example:

```
1 def greet(name):
2     return "Hello, " + name
3
4 print(greet("Bob")) # Hello, Bob
```

9 The Python “if” Statement

The if statement controls flow based on conditions. Example:

```
1 age = 18
2 if age >= 18:
3     print("Adult")
4 else:
5     print("Minor")
```

10 The Python “while” Statement

A while loop repeats while a condition is true. Example:

```
1 count = 1
2 while count <= 5:
3     print(count)
4     count += 1
```

Output: 1 2 3 4 5

11 A Couple of Cool Little Apps

App 1: Simple Calculator

```
1 num1 = float(input("Enter first number: "))
2 num2 = float(input("Enter second number: "))
3 op = input("Enter operator (+, -, *, /): ")
4 if op == "+":
5     print(num1 + num2)
6 elif op == "-":
7     print(num1 - num2)
8 elif op == "*":
9     print(num1 * num2)
10 elif op == "/":
11     print(num1 / num2)
```

App 2: Number Guessing Game

```
1 import random
2 number = random.randint(1, 10)
3 guess = int(input("Guess a number (1-10): "))
4 while guess != number:
5     guess = int(input("Wrong! Try again: "))
6 print("Correct!")
```

12 Summary of Python Boolean Operators

- and, or, not
- Comparison: ==, !=, <, >, <=, >=

Example:

```

1 x = 5
2 y = 10
3 print(x < y and y > 0) # True

```

13 Function Arguments and Return Values

Functions can take multiple arguments and return values. Example:

```

1 def add(a, b):
2     return a + b
3
4 result = add(3, 4)
5 print(result) # 7

```

14 The Forward Reference Problem

Python requires functions to be defined before they are called. Example (will cause an error):

```

1 print(my_func()) # Error: my_func not defined yet
2 def my_func():
3     return "Hello"

```

Fix: Define the function first.

15 Python Strings

Strings are sequences of characters, enclosed in "" or ". Example:

```

1 text = "Python"
2 print(text[0]) # P
3 print(text.upper()) # PYTHON
4 print(len(text)) # 6

```

16 Python Lists (and a Cool Sorting App)

Lists are ordered, mutable collections. Example:

```

1 numbers = [5, 2, 8, 1]
2 numbers.sort()
3 print(numbers) # [1, 2, 5, 8]

```

Sorting App

```

1 numbers = []
2 while True:
3     num = input("Enter a number (or 'done'): ")
4     if num == "done":
5         break
6     numbers.append(float(num))
7 numbers.sort()
8 print("Sorted:", numbers)

```

17 The “for” Statement and Ranges

Use for loops to iterate over sequences. Example:

```
1 for i in range(1, 6):  
2     print(i)    # 1 2 3 4 5
```

18 Tuples

Tuples are immutable sequences. Example:

```
1 point = (3, 4)  
2 print(point[0])    # 3
```

19 Dictionaries

Dictionaries store key-value pairs. Example:

```
1 student = {"name": "Alice", "age": 20}  
2 print(student["name"])    # Alice  
3 student["grade"] = "A"  
4 print(student)
```

20 Sets

Sets store unique, unordered elements. Example:

```
1 numbers = {1, 2, 2, 3}  
2 print(numbers)    # {1, 2, 3}  
3 numbers.add(4)  
4 print(numbers)
```

21 Global and Local Variables

Local variables are defined inside functions; global variables are outside. Example:

```
1 x = 10    # Global  
2 def my_func():  
3     x = 5    # Local  
4     print(x)    # 5  
5 my_func()  
6 print(x)    # 10
```

22 Exercises

1. Write a program that takes two numbers as input and prints their sum, difference, product, and quotient.
2. Create a function that checks if a number is even or odd.

3. Write a program that stores 5 names in a list, sorts them, and prints the sorted list.
4. Create a dictionary to store 3 students' names and grades, then print each student's information.

23 Conclusion

This tutorial covered the core concepts of Python programming, from variables and data types to control structures, functions, and data structures. Practice the exercises and refer to <https://docs.python.org/3/> for further exploration.