Python Programming

A Complete Beginner's Guide

Learn Python from Scratch with Practical Examples

Dr. Noor Ahmed
PhD in Artificial Intelligence

Edition 2026

About the Author

Dr. Noor Ahmed holds a PhD in Artificial Intelligence and has extensive experience in teaching programming and data science to students from diverse backgrounds. With a passion for making complex concepts accessible to beginners, Dr. Ahmed has dedicated his career to developing educational materials that empower learners to achieve their full potential in technology.

Dr. Ahmed's expertise spans across artificial intelligence, machine learning, and software development. He believes that everyone should have the opportunity to learn programming, regardless of their background, and has designed this book to be a comprehensive yet approachable introduction to Python programming.

Through clear explanations, practical examples, and hands-on exercises, this book reflects Dr. Ahmed's commitment to quality education and his understanding of how beginners learn best.

Table of Contents

Chapter 1: Introduction to Python

1.1 What is Python?

Python is a high-level, interpreted programming language created by Guido van Rossum and first released in 1991. It emphasizes code readability and simplicity, making it an excellent choice for beginners and experienced programmers alike.

Why Learn Python?

• **Easy to learn and read** - Python's syntax is clear and intuitive

• **Versatile** - Used in web development, data science, AI, automation, and more

• **Large community** - Extensive libraries and helpful resources available

• **High demand** - Python developers are sought after in the job market

• **Free and open-source** - No cost to get started

1.2 Installing Python

Before you can write Python programs, you need to install Python on your computer. We'll also set up a code editor to make writing Python easier.

Step 1: Download Python

1. Visit the official Python website: python.org

2. Click on 'Downloads' and select your operating system (Windows, macOS, or Linux)

3. Download Python 3.12 or later (avoid Python 2, which is outdated)

4. Run the installer and check 'Add Python to PATH' during installation

Step 2: Choose a Code Editor

A code editor makes writing Python much easier. Here are two popular options for beginners:

Option 1: Visual Studio Code (VS Code)

• Free, lightweight, and powerful

• Download from: code.visualstudio.com

• Install the Python extension after opening VS Code

Option 2: Google Colab

• Free, runs in your web browser

• No installation required

• Visit: colab.research.google.com

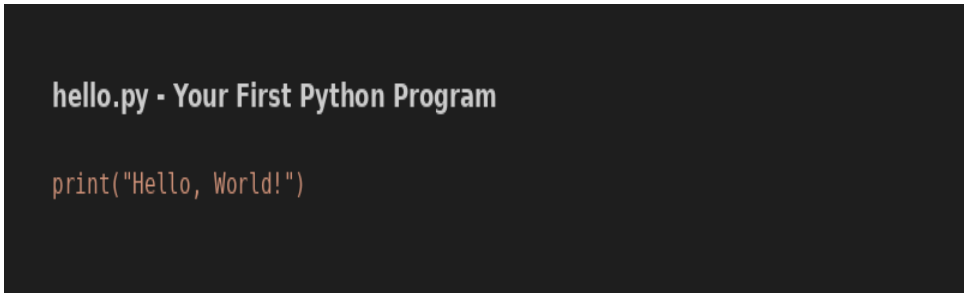
• Requires a Google account

Chapter 2: Your First Python Program

2.1 Hello, World!

The traditional first program in any programming language is 'Hello, World!' Let's write it in Python.

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



That's it! This single line of code will display 'Hello, World!' on your screen. Let's break it down:

• `print()` is a built-in Python function that displays text

• The text inside quotes is called a **string**

• Strings must be enclosed in quotes (either single ' or double ")

2.2 Running Your Program

In VS Code:

1. Create a new file and save it as hello.py

2. Type your code

3. Press F5 or click the run button (■) in the top right corner

In Google Colab:

1. Type your code in a code cell

2. Press Shift+Enter or click the play button (■) next to the cell

2.3 Exercise 1: Customize Your Greeting

Task: Modify the program to print your own name.

Example:

```
print("Hello, Sarah!")
print("Welcome to Python programming!")
```

Expected Output:

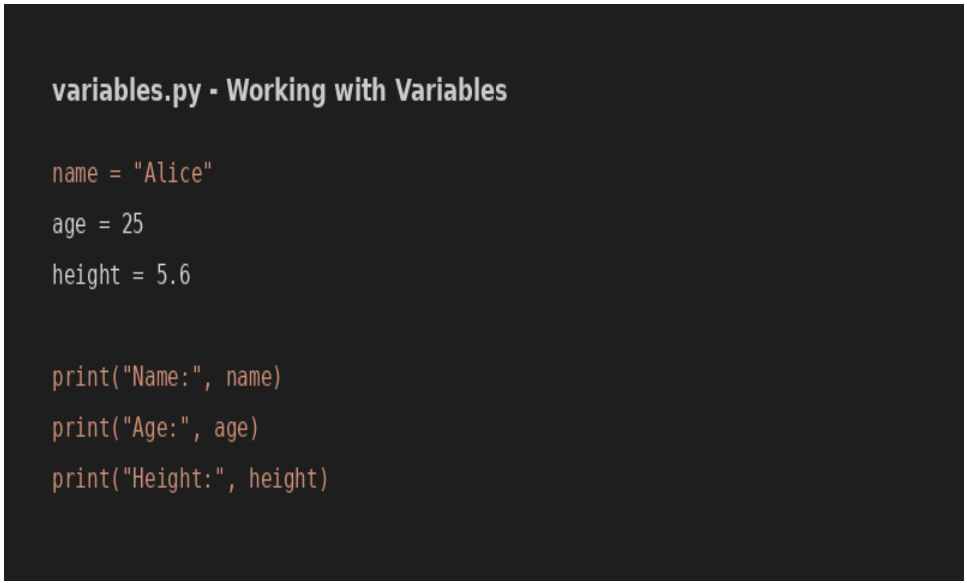
```
Hello, Sarah!
Welcome to Python programming!
```

Chapter 3: Variables and Data Types

3.1 What are Variables?

Variables are containers that store data. Think of them as labeled boxes where you can put information and retrieve it later. In Python, creating a variable is simple:

```
name = "Alice"
age = 25
height = 5.6
```



Variable Naming Rules

• Must start with a letter or underscore (_)

• Can contain letters, numbers, and underscores

• Cannot contain spaces (use underscores instead: my_variable)

• Case-sensitive (age and Age are different variables)

• Cannot use Python keywords (like print, if, for, etc.)

3.2 Basic Data Types

Python has several built-in data types. Here are the most common ones:

1. Strings (str) - Text

```
greeting = "Hello"
favorite_color = 'blue' # Single or double quotes work
```

2. Integers (int) - Whole Numbers

```
age = 25
year = 2026
temperature = -10
```

3. Floats (float) - Decimal Numbers

```
price = 19.99
pi = 3.14159
height = 5.8
```

4. Booleans (bool) - True or False

```
is_student = True
has_license = False
```

3.3 Checking Data Types

You can check the type of any variable using the `type()` function:

```
name = "Alice"
print(type(name)) # Output: <class 'str'>

age = 25
print(type(age)) # Output: <class 'int'>
```

3.4 Exercise 2: Working with Variables

Task: Create variables for yourself and print them.

```
# Create these variables:
full_name = "Your Name"
age = 20 # Your age
height = 5.7 # Your height in feet
is_learning_python = True

# Print them
print("Name:", full_name)
print("Age:", age)
print("Height:", height)
print("Learning Python?", is_learning_python)
```

Chapter 4: Basic Operations

4.1 Arithmetic Operators

Python can perform mathematical calculations just like a calculator:

```
# Addition
result = 10 + 5 # 15

# Subtraction
result = 10 - 5 # 5

# Multiplication
result = 10 * 5 # 50

# Division
result = 10 / 5 # 2.0 (always returns float)

# Floor Division (rounds down)
result = 10 // 3 # 3

# Modulus (remainder)
result = 10 % 3 # 1

# Exponentiation (power)
result = 2 ** 3 # 8
```

calculator.py - Simple Calculator

```
# Simple Calculator
num1 = float(input("Enter first number: "))
num2 = float(input("Enter second number: "))

sum_result = num1 + num2
difference = num1 - num2
product = num1 * num2

print(f"Sum: {sum_result}")
print(f"Difference: {difference}")
print(f"Product: {product}")
```

4.2 String Operations

```
# Concatenation (joining strings)
first_name = "John"
last_name = "Doe"
full_name = first_name + " " + last_name

# Repetition
laugh = "Ha" * 3 # "HaHaHa"

# String formatting (f-strings)
name = "Alice"
age = 25
message = f"My name is {name} and I am {age} years old."
print(message)
```

4.3 User Input

You can ask users to provide information using the `input()` function:

```
# Get user's name
name = input("What is your name? ")
print(f"Hello, {name}!")

# Get user's age (convert to integer)
age_text = input("What is your age? ")
age = int(age_text)
print(f"You are {age} years old.")
```

Important: The `input()` function always returns a string. Use `int()` or `float()` to convert to numbers.

Chapter 5: Conditional Statements

5.1 The if Statement

Conditional statements allow your program to make decisions:

```
age = 18

if age >= 18:
print("You are an adult")
print("You can vote!")
```

age_classifier.py - Conditional Statements

```
age = int(input("Enter your age: "))


if age < 13:
    print("You are a child")
elif age < 20:
    print("You are a teenager")
elif age < 60:
    print("You are an adult")
else:
    print("You are a senior citizen")
```

Important Notes:

• The colon ： at the end is required

• Indentation is crucial in Python

• Code inside the if block must be indented (usually 4 spaces)

5.2 Comparison Operators

```
== # Equal to
!= # Not equal to
> # Greater than
< # Less than
>= # Greater than or equal to
<= # Less than or equal to
```

5.3 if-elif-else Statements

```
score = 85

if score >= 90:
print("Grade: A")
elif score >= 80:
print("Grade: B")
elif score >= 70:
print("Grade: C")
elif score >= 60:
print("Grade: D")
else:
print("Grade: F")
```

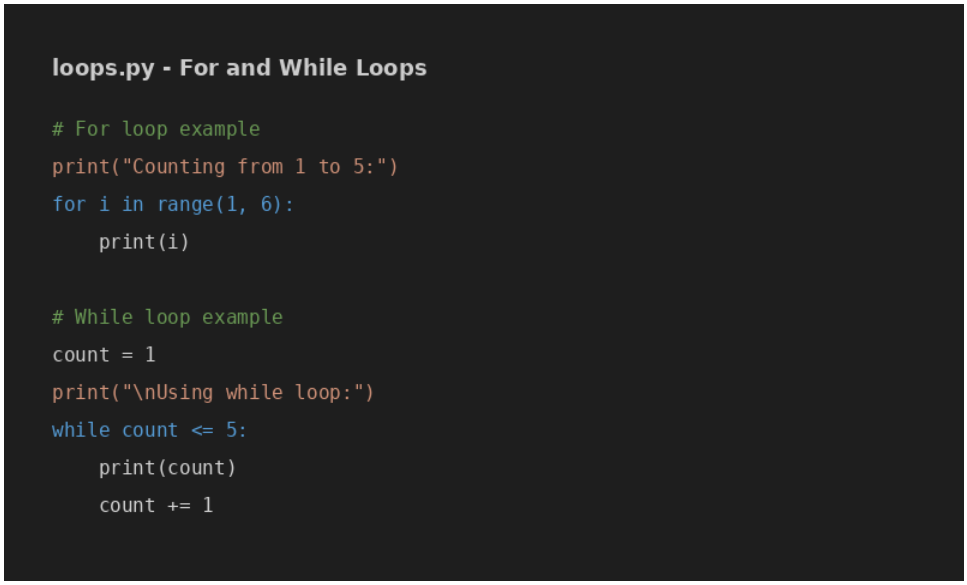
Chapter 6: Loops

6.1 The while Loop

```
count = 1

while count <= 5:
print(count)
count += 1

print("Done!")
```



6.2 The for Loop

```
# Print numbers 1 to 5
for i in range(1, 6):
print(i)

# Using range()
for i in range(5): # 0, 1, 2, 3, 4
print(i)
```

6.3 Exercise: Multiplication Table

```
num = int(input("Enter a number: "))

print(f"Multiplication table for {num}:")
for i in range(1, 11):
```

```
result = num * i
print(f"{num} x {i} = {result}")
```

Chapter 7: Lists and Collections

7.1 Creating Lists

```
# Create lists
numbers = [1, 2, 3, 4, 5]
fruits = ["apple", "banana", "orange"]
mixed = [1, "hello", 3.14, True]
empty_list = []
```

shopping_list.py - Working with Lists

```
# Create a shopping list
shopping_list = ["milk", "eggs", "bread"]

print("Shopping List:")
for i, item in enumerate(shopping_list, 1):
    print(f"{i}. {item}")

# Add new item
shopping_list.append("cheese")

print("\nUpdated List:")
for i, item in enumerate(shopping_list, 1):
    print(f"{i}. {item}")
```

7.2 Accessing and Modifying Lists

```
fruits = ["apple", "banana", "orange"]

# Access
print(fruits[0]) # apple
print(fruits[-1]) # orange (last item)

# Modify
fruits[1] = "mango"
fruits.append("grape")
fruits.remove("orange")
```

7.3 Looping Through Lists

```
fruits = ["apple", "banana", "orange"]

for fruit in fruits:
print(fruit)
```

```
for i, item in enumerate(fruits, 1):
print(f"{i}. {item}")
```

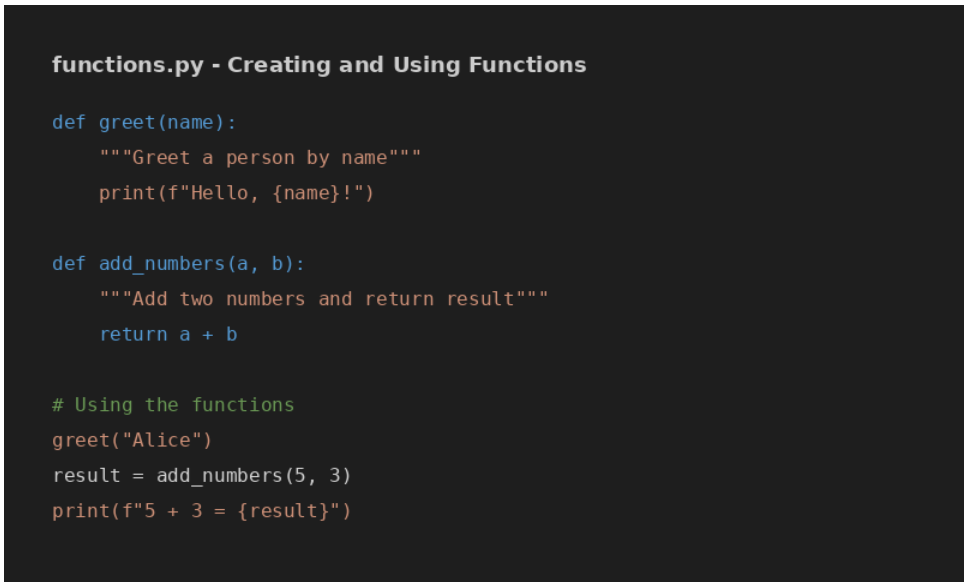
Chapter 8: Functions

8.1 Creating Functions

```
def greet(name):
print(f"Hello, {name}!")

def add_numbers(a, b):
return a + b

# Using functions
greet("Alice")
result = add_numbers(5, 3)
print(f"5 + 3 = {result}")
```



8.2 Default Parameters

```
def greet(name, greeting="Hello"):
print(f"{greeting}, {name}!")

greet("Alice") # Hello, Alice!
greet("Bob", "Hi") # Hi, Bob!
```

Chapter 9: Dictionaries

9.1 Creating Dictionaries

```
person = {
"name": "Alice",
"age": 25,
"city": "New York"
}

# Access values
print(person["name"])
print(person.get("age"))
```

grade_book.py - Working with Dictionaries

```
# Student grade book
grades = {
    "Alice": 95,
    "Bob": 87,
    "Charlie": 92
}

# Display all grades
print("Student Grades:")
for student, grade in grades.items():
    print(f"{student}: {grade}")

# Calculate average
average = sum(grades.values()) / len(grades)
print(f"\nClass Average: {average:.2f}")
```

9.2 Looping Through Dictionaries

```
person = {"name": "Alice", "age": 25}

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

Chapter 10: File Handling

10.1 Reading and Writing Files

```
# Write to file
with open("output.txt", "w") as file:
file.write("Hello, World!")

# Read from file
with open("output.txt", "r") as file:
content = file.read()
print(content)
```

Chapter 11: Next Steps

11.1 What You've Learned

Congratulations! You've completed this Python beginner's guide. You now know:

• How to write and run Python programs

• Variables and data types

• Basic operations and user input

• Conditional statements (if, elif, else)

• Loops (while and for)

• Lists and how to work with them

• Functions and return values

• Dictionaries and key-value pairs

• File handling

11.2 Continue Your Learning

• **Object-Oriented Programming** - Classes and objects

• **Modules and Packages** - Organizing projects

• **Error Handling** - Try-except blocks

• **Regular Expressions** - Pattern matching

• **Working with APIs** - Web services

11.3 Recommended Resources

• Python.org - Official documentation

• Codecademy - Interactive courses

• LeetCode - Coding challenges

• Stack Overflow - Q&A; community

11.4 Conclusion

Learning to program is a journey. The key to becoming proficient is consistent practice and working on real projects. Don't be discouraged by challenges—every expert was once a beginner.

Remember: The best way to learn programming is by doing. Start building, experimenting, and most importantly, have fun!

Good luck on your Python programming journey!

Happy Coding! ■