Introduction to Programming in Python

John Karuitha, PhD

February 27, 2025

1 Week 2: Code Documentation and Python Inbuilt Functions

In this lecture, we will explore essential aspects of programming in Python: documenting your code, understanding basic inbuilt functions like `print()`, `input()`, and `range()`, and getting introduced to other Python functions that will help you work efficiently.

---

2 1. Documentation and Commenting Your Code

2.1 Why Document Your Code?

Documentation is a crucial aspect of programming, especially when working in teams or on large projects. Proper documentation ensures that: - Other developers (or your future self) can understand what your code does. - Bugs can be easily identified and fixed. - Code is easier to maintain and update.

In Python, documentation is primarily done through **comments** and **docstrings**.

2.2 Comments

Comments are lines of text in your code that Python ignores during execution. They are used to explain what certain parts of the code do or to provide context for future reference.

In Python, a comment starts with the `#` symbol. Everything after the `#` on the same line is ignored by the Python interpreter.

2.2.1 Example:

```
# This is a comment explaining what the following line does
print("Hello, World!")  # This prints Hello, World! to the console
```

- **Single-Line Comments**: Use the `#` symbol at the beginning of the line.
- **Multi-Line Comments**: While Python does not have a specific syntax for multi-line comments, you can use `#` at the beginning of each line, or use a docstring (more on this below).

2.3 Docstrings

Docstrings (short for documentation strings) are a special type of comment used to describe the purpose of a function, class, or module. They are written inside triple quotes (`""" ... """`) and are typically placed at the beginning of functions or classes.

2.3.1 Example:

```
def greet():
    """
    This function prints a simple greeting message.
    """
    print("Hello, welcome to Python!")
```

The advantage of docstrings is that they can be accessed using Python's built-in help system.

```
help(greet)
```

This will display the docstring for the `greet()` function, making it easier to understand what the function does.

---

3 The `print()` and `input()` Functions

3.1 The `print()` Function

The `print()` function is one of the most basic and commonly used Python functions. It outputs (or "prints") text or other values to the console.

3.1.1 Syntax:

```
print(value, ..., sep=' ', end='\n')
```

- **value**: What you want to print (this could be text, numbers, variables, etc.).
- **sep**: Optional. Specifies how to separate multiple values. By default, it uses a space `' '`.
- **end**: Optional. Specifies what to append at the end of the printed text. By default, it's a newline (`'\n'`), which means the output will move to the next line after printing.

3.1.2 Examples:

```
# Printing a simple message
print("Hello, World!")  # Output: Hello, World!

# Printing multiple values
print("Hello", "World", sep=', ')  # Output: Hello, World!

# Changing the end of the print output
print("Hello", end='!')
print(" World!")  # Output: Hello! World!
```

3.2 The `input()` Function

The `input()` function allows you to get input from the user. It pauses the program and waits for the user to type something, then resumes once they hit the `Enter` key.

3.2.1 Syntax:

```
input(prompt)
```

- **prompt**: The message or question you want to display to the user, prompting them to enter something.

The `input()` function always returns the user's input as a string.

3.2.2 Example:

```
name = input("What is your name? ")
print("Hello, " + name + "!")
```

In this example, the program asks the user for their name, stores it in the `name` variable, and then prints a personalized greeting.

---

4 3. The `range()` Function

The `range()` function generates a sequence of numbers, which is useful for looping through a set of values.

4.0.1 Syntax:

```
range(start, stop, step)
```

- **start**: (Optional) The number to start the range from. By default, it starts at 0.
- **stop**: The number at which the range stops (not inclusive).
- **step**: (Optional) The amount by which the range increments. By default, it steps by 1.

4.0.2 Examples:

```
# Generating numbers from 0 to 4
for i in range(5):
    print(i)  # Output: 0, 1, 2, 3, 4

# Generating numbers from 2 to 8
for i in range(2, 9):
    print(i)  # Output: 2, 3, 4, 5, 6, 7, 8

# Generating even numbers between 0 and 10
for i in range(0, 11, 2):
    print(i)  # Output: 0, 2, 4, 6, 8, 10
```

---

5 4. Other Python Inbuilt Functions

Python comes with many other inbuilt functions that perform useful tasks. Here are a few common ones that you will frequently use:

5.1 `len()`

The `len()` function returns the length of a sequence (such as a string, list, or tuple).

5.1.1 Example:

```
name = "Python"
print(len(name))  # Output: 6
```

In this example, the `len()` function calculates the number of characters in the string `"Python"`.

5.2 `type()`

The `type()` function returns the type of the object passed to it. This is useful when you want to check if a variable is a string, integer, float, etc.

5.2.1 Example:

```
age = 21
print(type(age))  # Output: <class 'int'>
```

5.3 `int()` and `float()`

These functions convert a string or number into an integer (`int()`) or a floating-point number (`float()`).

5.3.1 Examples:

```
# Convert a string to an integer
age = input("Enter your age: ")
age = int(age)
print(age + 5)  # If input was '25', output will be 30

# Convert a string to a float
price = float("19.99")
print(price + 5)  # Output: 24.99
```

5.4 `sum()`

The `sum()` function returns the sum of all items in an iterable (like a list or tuple).

5.4.1 Example:

```
numbers = [1, 2, 3, 4, 5]
print(sum(numbers))  # Output: 15
```

---

6 Conclusion

In this lecture, we've introduced the importance of code documentation, learned how to use the `print()` and `input()` functions to interact with the user, explored how to generate sequences with the `range()` function, and got an overview of other useful Python inbuilt functions like `len()`, `type()`, `int()`, `float()`, and `sum()`. These basic tools are foundational to writing Python programs, and you will continue to build on them as you progress in your learning journey(McKinney 2022; Downey 2024).

References

Downey, Allen B. 2024. *Think Python*. 3rd ed. Green Tea Press.
McKinney, Wes. 2022. *Python for Data Analysis*. " O'Reilly Media, Inc.".