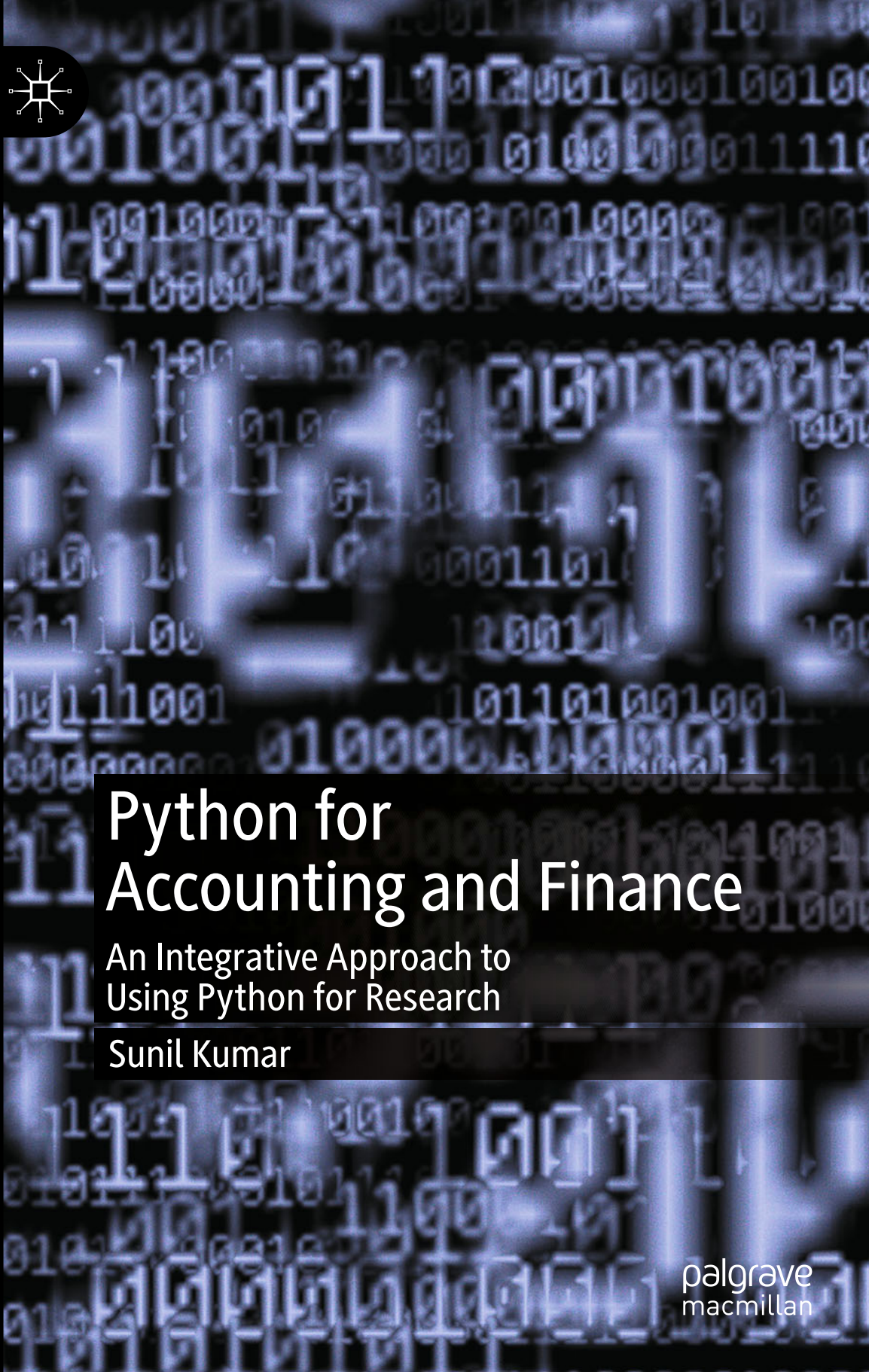

Python for Accounting and Finance

An Integrative Approach to Using Python for Research

Sunil Kumar

palgrave
macmillan

Python for Accounting and Finance

Sunil Kumar

Python for Accounting and Finance

An Integrative Approach to Using Python for Research

Power lies not in the answers but in the tools, we use to find them

palgrave
macmillan

Sunil Kumar
Bristol, RI, USA

ISBN 978-3-031-54679-2 ISBN 978-3-031-54680-8 (eBook)
https://doi.org/10.1007/978-3-031-54680-8

Cover illustration: imagenavi

This Palgrave Macmillan imprint is published by the registered company Springer Nature Switzerland AG
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

Paper in this product is recyclable.

To my beloved wife Kavita, and our precious gems Nikshika and Anishka, who constantly inspire me to explore, learn, and share my knowledge. You are the true assets of my life.

Acknowledgements

Completing this book was made possible through the collective support and wisdom of many outstanding individuals. I would like to express my heartfelt gratitude to the University of Massachusetts Boston, its Accounting and Finance Department, and its PhD program, all of which provided an enriching environment that greatly contributed to my research and writing.

Special thanks are due to Professor Atreya Chakraborty, whose guidance and insights have been invaluable. I am also profoundly grateful to the faculty of the Accounting and Finance department at UMass Boston especially, Arindam Bandopadhyaya, Robert Kim, Sangwan Kim, Surit Tinaikar, and Lucia Silva Gao for their unwavering support and encouragement throughout this journey.

I also owe a deep sense of appreciation to Dr. Robert Taggart, Professor of Finance (Retired) at Boston College, whose teachings and mentorship have left an indelible mark on my professional life and academic pursuits. I also extend a special thanks to my friend Ankush Mohekar, who was always there when I needed him throughout this journey.

To all of you who supported me directly or indirectly during the writing of this book, thank you. Your collective wisdom has not only enlightened my path but has also enriched the pages of this work.

Prologue

In the swiftly evolving domains of business, accounting, and finance, harnessing the power of technology to enhance and expand upon traditional research methodologies has become increasingly vital. The advent of robust programming languages like Python has revolutionized the field of data analysis, enabling more sophisticated, nuanced, and efficient examination of complex data.

"Python for Accounting and Finance" is a comprehensive and illuminating exploration into the application of Python within the realm of accounting and finance research, as well as other business disciplines. Its contents are designed to serve as an indispensable guide for a diverse range of individuals engaged in these fields—PhD scholars, research faculty, industry professionals, and business researchers—who are keen to leverage the expansive capabilities of Python to elevate their research.

This book is predicated on the premise that Python has emerged as a programming language of choice in both academic research and applied research. Its open-source nature, combined with an extensive collection of libraries, gives it a distinct edge over many conventional, often proprietary, software programs. The book guides the reader through the comprehensive offerings of Python, from handling an array of data formats, including structured and unstructured data, to employing its advanced machine learning and artificial intelligence capabilities for predictive analytics.

The initial section of this book offers a solid foundation in Python, covering its fundamentals and key libraries, and regular expressions. The subsequent sections progressively delve into more specialized applications, beginning with data acquisition and cleaning, before moving on to exploratory data analysis and visualization, natural language processing, machine learning, and predictive analytics. Each section is meticulously crafted, presenting a judicious blend of theoretical knowledge and practical applications.

One of the standout features of "Python for Accounting and Finance" is its focus on real-world case studies and practical examples, supplemented with ready-to-use codes for most of the activities involved for research in these disciplines. This approach enables readers to contextualize and apply their learning immediately. The book is also replete with exercises that provide hands-on experience, reinforcing the concepts and techniques presented.

To derive maximum benefit from this book, it is imperative to implement the codes yourself and modify them as per your requirements. The learning journey through Python is one of active engagement and personal experimentation—getting your hands dirty, so to speak, is indeed the key to mastering this tool.

By the end of this comprehensive guide, readers will have developed a firm understanding of Python programming within the context of accounting, finance, and broader business research. They will be equipped with the skills to tackle real-world analytical problems in their professional pursuits. The journey through this book is not merely about learning a programming language; it is about embracing a powerful tool that unlocks a deeper understanding of research in these disciplines.

Welcome to a transformative journey into the world of Python for Accounting, Finance, and Business Research. Let the exploration begin!

Contents

Part I

Introduction and Fundamentals

1

Introduction to Python for Accounting and Finance Research

The disciplines of accounting and finance are inherently driven by data and heavily rely on analytical methodologies. As technological progress continues, the availability of data for analysis expands accordingly. To maintain competitiveness and ensure informed decision-making, researchers and practitioners in these fields must possess the ability to efficiently analyze large datasets. Python, a versatile programming language, has garnered significant attention in the realm of accounting and finance research. Its user-friendly syntax and robust libraries make it an optimal tool for performing tasks such as data analysis, machine learning, and data visualization. Acquiring proficiency in Python can yield substantial advantages in professional endeavors for doctoral candidates, researchers, and accounting professionals alike.

This chapter aims to introduce Python and elucidate its merits specifically within the context of accounting research. It commences with an overview of the language, encompassing its historical evolution, distinctive features, and wide-ranging applications. The discussion subsequently transitions to a step-by-step guide for installing and configuring a Python environment on the reader's personal computer, enabling a prompt initiation of coding activities. By the conclusion of this chapter, readers will have acquired a foundational understanding of Python and be adequately prepared to explore its extensive capabilities within the domains of accounting and finance research.

S. Kumar, *Python for Accounting and Finance*,
https://doi.org/10.1007/978-3-031-54680-8_1

Benefits of Python in Accounting and Finance Research

Python has gained increasing popularity in accounting and finance research due to its numerous advantages. One notable benefit is its open-source nature, which extends to the majority of supporting libraries and tools. These open-source resources, accompanied by flexible and open licenses, not only make Python a cost-effective choice but also foster a collaborative and open community where code and ideas can be shared.

Moreover, Python's interpreted nature allows for runtime translation of code into executable byte code, resulting in fast and efficient execution. This characteristic makes Python particularly suitable for researchers working with large datasets. Python's multiparadigm nature further enhances its appeal, as it supports various programming and implementation paradigms, including object orientation, imperative, functional, and procedural programming. This flexibility empowers researchers to choose the most appropriate paradigm for their research, free from language constraints.

Python's versatility extends to its multipurpose nature, enabling its utilization for rapid interactive code development, building large applications, and performing low-level system operations as well as high-level analytics tasks. This adaptability enables researchers to employ Python across diverse research domains, ranging from data analysis to intricate modeling.

An additional advantage of Python is its dynamic typing feature, where types are inferred during runtime rather than statically declared as in compiled languages. This simplifies programming and alleviates distractions from technical details, allowing researchers to focus on their research objectives. Python's indentation awareness, which employs indentation for marking code blocks, enhances code readability and comprehension.

Python's cross-platform compatibility ensures its availability across major operating systems, including Windows, Linux, and Mac OS. It is applicable for desktop and web applications and can be employed on various hardware, ranging from powerful servers to smaller devices like the Raspberry Pi. This cross-platform capability ensures researchers can work with Python regardless of their hardware or operating system preferences.

The advantages of Python in accounting and finance research extend beyond its language features. Python bridges the realms of economics and data science by providing researchers with a language that abstracts technical programming aspects and is easily learnable, even for individuals without a technical background. Its concise and English-like syntax makes it an ideal tool for interdisciplinary research. Python's support for prototyping and rapid

iterative development is facilitated by its interactive interpreter tools, allowing researchers to write and execute code line by line and immediately observe results.

Python's capability to handle both structured and unstructured data sets it apart from traditional tools like SAS and STATA, which primarily handle structured data. With Python, researchers can work with diverse data formats, including text, images, audio, and numerical data, facilitating comprehensive analyses and deeper insights. This empowers Python to be a powerful tool in accounting and finance research, for instance, enabling the analysis of text data in financial statements to uncover meaningful insights or measuring market sentiment for investment decision-making. Python also allows researchers to gauge management sentiment while drafting financial statements, aiding in identifying potential red flags. Moreover, Python facilitates research utilizing additional databases not available in structured form. Its capacity for creating self-learning models that adapt to new events enhances the accuracy of predictions and forecasts.

Overview of Python Programming Language

Python is a high-level, interpreted programming language that has become increasingly popular among accounting and finance researchers in recent years. The language was created in 1991 by Guido van Rossum with a focus on code readability and simplicity, making it an ideal choice for researchers with little to no programming experience.

Python's syntax is designed to be easy to read and write, with a structure that emphasizes code readability over traditional syntax elements such as semicolons or braces. This makes Python a popular choice for beginners and experts alike in the accounting and finance research fields. In addition, Python supports multiple programming paradigms, including object-oriented, imperative, and functional programming, providing researchers with flexibility in how they approach their research.

Python has a large standard library that provides a wide range of functionalities, including string manipulation, file I/O, web scraping, and database management. This extensive library of tools has made Python a popular choice for researchers who want to work with large datasets or perform complex data analysis tasks. In addition, Python has a large and active community that contributes to a wide range of open-source libraries and frameworks, making it easy to build complex applications in accounting and finance research.

Python's popularity in data science and machine learning is largely due to the availability of popular libraries such as NumPy, Pandas, Matplotlib, and Scikit-learn. These libraries provide tools for data manipulation, visualization, and statistical analysis, making Python a popular language for scientific computing. In the field of accounting and finance research, Python can be used to analyze financial data, perform sentiment analysis on financial statements, and build predictive models for forecasting financial trends.

Python is also known for its ease of use and versatility, making it a popular choice for rapid application development and prototyping. Python code is usually one-third to one-fifth the size of equivalent C++ or Java code, leading to less coding time. Additionally, Python's simple, easy-to-learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python is completely free to use and distribute, making it an affordable choice for researchers.

Python's dynamic semantics and automatic garbage collection make it a powerful tool for accounting and finance research. The language provides high-level dynamic data types and supports dynamic type checking, allowing researchers to work with a variety of data formats, including text, images, and audio, in addition to numerical data. Python is also ideal for prototyping and rapid, iterative development, as its interactive interpreter tools provide environments where researchers can write and execute each line of code in isolation and see the results immediately.

Python is an interpreted, object-oriented, high-level programming language that is cross-platform and can be used on a wide range of operating systems, including Windows, Linux, and Mac OS. Python can be used as a scripting language or can be compiled to byte code for building large applications. Python is easy to learn because of its simplicity and frequent usage of English keywords in the code, making it an ideal choice for interdisciplinary research in accounting and finance.

Python also provides easy integration with other programming languages such as C, C++, COM, ActiveX, CORBA, and Java. This allows researchers to use Python in conjunction with other programming languages to create hybrid solutions that leverage the strengths of each language. Additionally, Python's popularity in the field of data science and machine learning can be attributed to its powerful libraries such as TensorFlow, PyTorch, and Keras. These libraries provide advanced tools for building and training neural networks, deep learning models, and natural language processing applications. Python's high-level syntax makes it easy to represent complex mathematical concepts and algorithms, which is essential for creating machine learning models.

In addition to its strong community support, Python also offers a wealth of resources for learning and development. There are numerous online courses, tutorials, and documentation available for free that can help users get started with Python and become proficient in its use. Many universities and academic institutions also teach Python as a part of their curriculum, reflecting the growing importance of the language in various fields.

Python is widely used in the financial industry, where it is used to develop tools for risk management, pricing models, and algorithmic trading. Its versatility makes it an ideal tool for financial analysis, and it is often used to analyze large datasets, create visualizations, and build predictive models. Many financial institutions also use Python for backtesting trading strategies and building automated trading systems.

Python is also popular in the field of web development, where it is used to create dynamic and interactive web applications. Python's simplicity and ease of use make it a popular choice for web developers, and it is often used with popular web frameworks such as Django and Flask.

In the world of scientific computing, Python is used extensively for numerical computing, simulation, and visualization. Its powerful libraries such as NumPy, SciPy, and Matplotlib provide advanced tools for numerical computing and data visualization. Python is also used in the field of computational biology, where it is used for data analysis, genome sequencing, and gene expression analysis.

Python's versatility and ease of use have made it an increasingly popular choice for educators and students in the field of computer science. Its simple syntax and powerful libraries make it an ideal language for teaching programming concepts and techniques. Python is often used in introductory computer science courses, where students learn programming fundamentals and gain practical experience with coding.

Python is a versatile, powerful, and widely used programming language that has become increasingly popular in recent years. Its ease of use, flexibility, and strong community support have made it a popular choice for a wide range of applications, from web development and data science to scientific computing and machine learning. With its powerful libraries and tools, Python has become an essential tool for researchers, developers, and educators in various fields, making it an exciting language to learn and use.

Installing and Setting up Python Environment

The process of installing and setting up Python for use in accounting and finance research may appear daunting to newcomers, but it can be accomplished with relative ease. Initially, the first step entails downloading and installing Python itself, which can be obtained from the official Python website. Once downloaded, executing the installation file and following the prompts completes the installation process.

Alternatively, individuals may opt for Anaconda, a widely used Python distribution that comes pre-packaged with numerous popular libraries and tools for scientific computing and data analysis. Anaconda includes Jupyter Notebook, an interactive development environment (IDE) for Python. To install Anaconda, individuals can visit the Anaconda website, download the appropriate version suitable for their operating system, and then proceed with the installation process by executing the downloaded file and adhering to the prompts.

Following the installation of Anaconda, users can access the Anaconda Navigator, a graphical user interface that facilitates application and environment management within Anaconda. Launching the Navigator involves opening the Anaconda Prompt or Terminal and entering "anaconda-navigator" followed by the Enter key.

Once the Navigator interface is accessible, users can create a new environment specifically tailored to their accounting and finance research. This can be accomplished by navigating to the "Environments" tab and selecting the "Create" option. A suitable name for the environment should be assigned, and the desired version of Python should be chosen, with the recommendation being the latest version.

Upon the creation of the environment, users can proceed to install additional packages and libraries necessary for their research. This can be done by navigating to the "Home" tab and selecting the "Install" option. Users can then search for packages by name or explore available packages categorized accordingly. Once the desired package is located, a simple click followed by selecting "Apply" will initiate the installation into the designated environment.

To commence using Python for accounting and finance research, users can launch Jupyter Notebook from the Navigator interface. Jupyter Notebook provides an interactive web-based environment where users can develop and share documents containing live code, equations, visualizations, and narrative text. Launching Jupyter Notebook involves selecting the "Home" tab and clicking "Launch" under the Jupyter Notebook section.

After launching Jupyter Notebook, users can create a new notebook by selecting "New" and then "Python 3". This action will create a notebook where Python code can be written and executed by pressing Shift + Enter.

While Jupyter Notebook serves as a commendable environment for Python code development and execution, alternative Integrated Development Environments (IDEs) are available that users may prefer. Prominent IDEs for Python include PyCharm, Spyder, and Visual Studio Code. These IDEs offer additional features such as code highlighting, code completion, and debugging capabilities, which can enhance the Python development experience. Users are encouraged to explore these options and select the IDE that aligns best with their specific requirements.

Having successfully installed and configured Python along with the necessary tools, users can now embark on exploring the immense potential this powerful language holds for their accounting and finance research endeavors.

2

Introduction to Python Language

This chapter serves as an integral foundation for the application of Python in accounting and finance research, illuminating the rudimentary aspects of Python programming. This chapter delves into the fundamental components that constitute the Python programming language, thus establishing a robust groundwork for the intricate applications of Python in subsequent chapters. This exploration of Python's basics will equip readers with the necessary skills to manipulate data effectively and to create functional and efficient code for their research endeavors.

We begin our exploration with an examination of Python's Data Types, Variables, and Operators, providing a clear understanding of the building blocks used to structure and manipulate data. Following this, we delve into Control Flow Statements, paving the way for more complex programming logic. A detailed study of Functions, Modules, and Data Structures in Python then allows us to understand the abstraction and organization of code and data, respectively. With a firm grasp of these, we transition into File Handling in Python, an essential skill for any researcher dealing with data. The subsequent discussion of the **os** module provides an overview of how Python interacts with the operating system, a skill vital for automating and streamlining tasks. Finally, we introduce the paradigm of Object-Oriented Programming in Python, a powerful tool for encapsulating data and functionality into reusable and modular code. This comprehensive exploration of Python's basics thus forms a solid foundation for the advanced applications of Python in the realm of accounting and finance research.

S. Kumar, *Python for Accounting and Finance*,
https://doi.org/10.1007/978-3-031-54680-8_2

Data Types, Variables, and Operators

Python is an object-oriented programming language, which means that all data is stored in objects. In Python, there are several built-in data types that can be used to store different kinds of data. The most commonly used data types in Python include:

- Integer: used to store whole numbers (e.g., 1, 2, 3)
- Float: used to store decimal numbers (e.g., 1.0, 2.5, 3.14)
- Boolean: used to store True or False values
- String: used to store text (e.g., "hello world")
- List: used to store multiple items in a single variable (e.g., [1, 2, 3])
- Tuple: used to store multiple items in a single variable (e.g., (1, 2, 3))
- Dictionary: used to store key-value pairs (e.g., {"name": "John", "age": 25})

To create a variable in Python, you simply need to assign a value to a name using the equals sign (=). For example:

```
x = 1
```

In this case, we have created a variable named *x* and assigned it the value of 1. Once a variable has been created, it can be used in expressions and calculations.

Python also provides several operators that can be used to perform calculations and manipulate data. These include:

- Arithmetic operators: used to perform basic arithmetic operations (e.g., + for addition, − for subtraction, * for multiplication, / for division)
- Comparison operators: used to compare two values and return a boolean value (e.g., == for equality, < for less than, > for greater than)
- Logical operators: used to combine two or more boolean values (e.g., and for logical AND, or for logical OR, not for logical NOT)
- Assignment operators: used to assign a value to a variable (e.g., = for simple assignment, += for addition assignment, −= for subtraction assignment)

Here's an example of how you can use variables and operators in Python:

```
x = 10
y = 5
z = x + y
print(z)
```

In this example, we have created two variables named x and y and assigned them the values of 10 and 5, respectively. We have then created a third variable named z and assigned it the value of $x + y$, which is 15. Finally, we have printed the value of z to the console using the print() function.

Control Flow Statements

Control flow statements in Python allow you to control the order in which statements are executed in a program. They are used to specify the sequence of execution of statements based on certain conditions.

a. If, Else, and Elif Statements

If, else, and elif statements are conditional statements that are used to execute certain code based on specific conditions. These statements allow the program to make decisions based on whether a certain condition is true or false.

- If Statements

The if statement is used to execute code if a certain condition is true. The syntax of an if statement is as follows:

```
if condition:
    # code to execute if condition is true
```

Here's an example:

```
x = 10
if x > 5:
    print("x is greater than 5")
```

In this example, the condition is x > 5. Since x is equal to 10, which is greater than 5, the code inside the if statement will be executed, and the output will be x is greater than 5.

- Else Statements

The else statement is used to execute code if the condition in the if statement is false. The syntax of an else statement is as follows:

```
if condition:
    # code to execute if condition is true
else:
    # code to execute if condition is false
```

Here's an example:

```
x = 2
if x > 5:
    print("x is greater than 5")
else:
    print("x is less than or equal to 5")
```

In this example, the condition is **x > 5**. Since **x** is equal to **2**, which is less than **5**, the code inside the else statement will be executed, and the output will be **x is less than or equal to 5**.

- Elif Statements

The elif statement is used to test multiple conditions and execute different code based on which condition is true. The syntax of an elif statement is as follows:

```
if condition1:
    # code to execute if condition1 is true
elif condition2:
    # code to execute if condition2 is true
else:
    # code to execute if all conditions are false
```

Here's an example:

```
x = 2
if x > 5:
    print("x is greater than 5")
elif x == 5:
    print("x is equal to 5")
else:
    print("x is less than 5")
```

In this example, the first condition (**x > 5**) is false, so the program moves on to the next condition (**x == 5**). Since **x** is not equal to **5**, the program executes the code inside the else statement, and the output is **x is less than 5**.

b. For and While Loops

Another key aspect of programming is the ability to perform iterative tasks. Two ways to do this in Python are with for loops and while loops.

- For Loops

A for loop is used to iterate over a sequence of elements. This sequence can be a list, tuple, string, or any other iterable object. The basic syntax for a for loop is:

```
for variable in sequence:
    # code to execute
```

The variable is assigned to each element in the sequence, and the code inside the loop is executed for each element.

For example, let's say we want to print each number in a list:

```
numbers = [1, 2, 3, 4, 5]

for num in numbers:
    print(num)
```

Output:

```
1
2
3
4
5
```

We can also use a for loop to iterate over a string:

```
my_string = "Hello, World!"

for char in my_string:
    print(char)
```

Output:

```
H
e
l
l
o
,

W
o
r
l
d
!
```

We can also use the range function to create a sequence of numbers to iterate over. The range function takes three arguments: start, stop, and step. The start argument is the first number in the sequence (inclusive), the stop argument is the last number in the sequence (exclusive), and the step argument is the amount by which to increment each number.

```
for i in range(1, 6):
    print(i)
```

Output:

```
1
2
3
4
5
```

- While Loops

A while loop is used to execute a block of code repeatedly as long as a condition is true. The basic syntax for a while loop is:

```
while condition:
    # code to execute
```

The condition is evaluated at the beginning of each iteration. If the condition is true, the code inside the loop is executed. This continues until the condition becomes false.
For example, let's say we want to print the numbers 1 to 5 using a while loop:

```
     i = 1
while i <= 5:
    print(i)
    i += 1
```

Output:

```
1
2
3
4
5
```

We can also use the break and continue statements in for and while loops, just like in if statements. The break statement is used to exit the loop completely, while the continue statement is used to skip the current iteration and move on to the next one.

```
# Example of using break and continue in a for loop
for i in range(1, 6):
    if i == 3:
        break
    elif i == 2:
        continue
    print(i)
```

Output:

```
1
```

```
# Example of using break and continue in a while loop
i = 1

while i <= 5:
    if i == 3:
        break
    elif i == 2:
        i += 1
        continue
    print(i)
    i += 1
```

Output:

```
1
4
5
```

c. Break, Continue, and Pass Statements

Python also provides two other important control flow statements: **break** and **continue**. These statements allow you to alter the flow of a loop based on certain conditions.

- Break Statement

The **break** statement is used to terminate a loop prematurely. When the interpreter encounters a **break** statement within a loop, it immediately exits the loop and resumes execution at the next statement after the loop. This is useful when you need to exit a loop early based on some condition.

Here's an example of using the **break** statement in a **for** loop:

```
for i in range(1, 11):
    if i == 5:
        break
    print(i)
```

In this example, the loop will iterate over the numbers 1 through 10. However, when the loop variable i is equal to 5, the break statement is executed, and the loop is terminated prematurely. As a result, only the numbers 1 through 4 are printed.

- Continue Statement

The continue statement is used to skip the current iteration of a loop and move on to the next iteration. When the interpreter encounters a continue statement within a loop, it immediately skips to the next iteration of the loop without executing any further statements in the current iteration. This is useful when you need to skip over certain iterations of a loop based on some condition.

Here's an example of using the continue statement in a while loop:

```
i = 0
while i < 10:
    i += 1
    if i % 2 == 0:
        continue
    print(i)
```

In this example, the loop will iterate over the numbers 1 through 10. However, when the loop variable **i** is even, the **continue** statement is executed, and the current iteration of the loop is skipped. As a result, only the odd numbers are printed.