**Module 1 -** Computers and Programs

**Learning Objectives**

* Understand the fundamental role of computers in various daily and industrial tasks.
* Describe the basic function of a computer as an information processing system.
* Explain the significance of programmable instructions in computer versatility.
* Identify different forms and capabilities of computers across various devices.
* Recognize the universal potential of computers through appropriate programming.

Contents

[**1.** **Computers** 1](#_Toc202184851)

[**2.** **Computer Programs** 2](#_Toc202184852)

[**3.** **Programming Languages** 3](#_Toc202184853)

[**4.** **Discovering Python** 4](#_Toc202184854)

[**Benevolent Director For Life** 4](#_Toc202184855)

[**Why Python Stands Out** 5](#_Toc202184856)

[**5.** **Setting Up Python: Installation Guide for Windows and macOS** 7](#_Toc202184857)

[**5.1.** **Installing Python on Windows** 7](#_Toc202184858)

[**5.2.** **Installing Python on macOS** 7](#_Toc202184859)

[**5.3.** **Verifying Python Installation** 8](#_Toc202184860)

[**5.3.1.** **On Windows:** 8](#_Toc202184861)

[**5.3.2.** **On macOS:** 8](#_Toc202184862)

[**5.4.** **How to Use Python Shell** 8](#_Toc202184863)

[**5.4.1.** **On Windows:** 8](#_Toc202184864)

[**5.4.2.** **On macOS:** 9](#_Toc202184865)

[**5.5.** **Installation of Visual Studio Code** 9](#_Toc202184866)

[**5.5.1. Installation on Windows 10**](#_Toc202184867)

[**5.5.2.** **On macOS:** 10](#_Toc202184868)

[**5.6.** **Setting Up Python in Visual Studio Code** 10](#_Toc202184869)

[**5.7.** **Running Python Code Using Command Prompt (Windows)** 10](#_Toc202184870)

[**5.8.** **Running Python Code Using Terminal (macOS)** 11](#_Toc202184871)

[**5.9.** **Running Python Code in Visual Studio Code** 11](#_Toc202184872)

# **Computers**

In our digital era, computers have become essential to everyday life. Whether used for gaming, writing academic papers, shopping online, listening to music, or staying connected on social media, computers significantly enhance our experiences and capabilities. Beyond personal use, these adaptable devices play crucial roles in weather forecasting, airplane design, filmmaking, business operations, financial management, and industrial automation. This widespread presence raises a fundamental question: What exactly is a computer, and how does it manage such a wide range of tasks?

A computer can be briefly described as a device that processes and manipulates data according to a programmable set of instructions. This definition relies on two key points. First, computers are essentially information processing systems. They take in input data, transform it into something meaningful or useful, and then output the results for us to use or interpret. While other devices like calculators or gas pumps process information for specific tasks, such as calculating sums or measuring fuel dispensed, they lack the broad versatility of computers.

The distinguishing feature of computers is their ability to operate based on changeable programs. A program is like a recipe for the computer, a precise sequence of instructions that dictate its actions. By changing the program, we can alter the computer's behavior, allowing it to switch seamlessly from drafting documents to managing finances or entertaining us with games. This flexibility is what makes a computer a truly universal tool, capable of performing any task outlined in a program.

Computers come in various forms, including desktops, laptops, tablets, and smartphones, as well as specialized computers embedded in countless devices. A significant insight of computer science is the realization that, fundamentally, all these computers share the same capabilities. With the appropriate programming, any computer can perform the tasks of any other. This concept transforms the personal computer from a simple electronic device to a universal machine, limited only by the tasks we can envision and program. Essentially, the power of a computer lies in its ability to bring our ideas to life, making it an indispensable tool in the modern world.

# **Computer Programs**

In the digital age, the true potential of computers is unlocked not by the hardware but by the software they run. Software, comprising the programs that guide the hardware, is the essence of any computer system. Without it, even the most advanced machines are mere intricate sculptures. This essential truth highlights the importance of programming, the process of creating [software](https://guides.library.uq.edu.au/). This book is dedicated to programming, a skill that, while challenging, is accessible to anyone with determination and a willingness to learn.

Programming is not just for aspiring computer scientists or tech industry professionals. In a world where computers are integral to daily life, understanding programming demystifies these powerful tools, providing deeper insights into their capabilities and limitations. It transforms users from passive operators to architects of their digital experiences. Whether you want to enhance your computer skills or create your own applications, programming opens up a realm of possibilities.

Beyond practical uses, programming fosters creativity and problem-solving. It can be both enjoyable and fulfilling, offering a way to create solutions that are functional and sometimes artistically compelling. Many find joy in programming as a hobby, while others use these skills to tackle complex problems by breaking them down into manageable parts. The ability to think like a programmer, to understand both the system as a whole and its intricate details, is valuable across many fields.

The demand for skilled programmers has never been higher, providing a path to rewarding careers for people from all educational backgrounds. Even those from unrelated fields have discovered that programming skills can lead to exciting opportunities. In the business world, where computers are central, programming knowledge can give you a competitive edge. With these skills, you might even create the next groundbreaking application.

As we explore programming, remember that this journey is about more than just learning to code. It's about gaining the tools to navigate and shape the digital world, turning ideas into reality, and unlocking the full potential of technology. Welcome to the transformative world of programming, where your imagination is the only limit.

# **Programming Languages**

[Programming languages](https://open.umn.edu/opentextbooks/textbooks/a-byte-of-python) serve as the connection between human ideas and computer actions. Ideally, we could communicate with computers using everyday language, similar to scenes in science fiction movies. Technologies like Siri, Google Now, and Cortana have made progress towards this vision, but fully comprehending human language remains a complex challenge for computers. Natural languages, rich with ambiguity and nuances, aren't precisely suited for defining the strict logic required by algorithms.

To address this, computer scientists have created programming languages, specialized notations designed to express computations clearly and without ambiguity. These languages, such as Python, C++, and Java, have strict syntax (rules of structure) and semantics (meaning), enabling programmers to write instructions that computers can execute. Coding involves using these languages to formulate algorithms in a way that computers can understand.

High-level programming languages, including the ones mentioned, are designed for human readability and ease of use, while computers operate on a much simpler, binary language known as machine language. For example, a simple addition in Python, written as c = a + b, is easy for us to understand but needs translation into the binary operations that a computer's CPU can perform.

This translation from high-level to machine language is achieved through two main processes: compiling and interpreting. A compiler converts the entire program from a high-level language (source code) into machine language (executable code) all at once, allowing the computer's hardware to execute it directly. An interpreter, on the other hand, processes the program line by line, translating and executing each instruction on the fly without producing a separate machine language program.

The key difference between these two approaches is efficiency and flexibility. Compiled programs, once translated, run quickly since they are already in the computer's native language. Interpreted languages, however, offer more flexibility during development, allowing for immediate execution and testing of code, which can speed up the programming process.

High-level languages also provide portability. Unlike machine-specific binary code, a program written in a high-level language can run on various types of computers with the appropriate compiler or interpreter. This means the same Python program could run on both a laptop and a tablet, despite differences in their hardware, because both devices have Python interpreters.

In summary, programming languages are essential tools for communicating tasks to computers, crafted to minimize ambiguity and enhance clarity. The choice between compiling and interpreting depends on the program's needs and the development process, balancing speed against flexibility and ease of debugging.

# **Discovering Python**

In the digital age, the pursuit of efficiency and innovation often leads to the automation of mundane or complex tasks. Whether it's managing large datasets, organizing photos, developing games, or creating specialized applications, the need for a powerful and versatile programming tool is clear. This is where Python comes in, a language inspired by the whimsical world of "Monty Python's Flying Circus," not the reptile.

[Python](https://www.python.org/) is more than just another programming language; it's designed to simplify your life. Imagine needing to automate file management, text manipulation, or create a custom database or user-friendly GUI application. While traditional scripting or programming languages can accomplish these tasks, they often do so at the expense of time and simplicity. Python, with its elegant syntax and powerful features, makes these tasks more accessible and enjoyable.

For professionals used to the rigorous development process of C/C++/Java, which involves writing, compiling, testing, and debugging, Python offers a refreshing alternative. Its interpreted nature allows for immediate testing of ideas and building functionalities without the long wait times of compilation. Whether refining a test suite, extending an application, or experimenting with new algorithms, Python enhances productivity and fosters creativity.

# **Benevolent Director For Life**

[Guido van Rossum](https://en.wikipedia.org/wiki/Guido_van_Rossum), a Dutch programmer, is widely recognized as the creator of the Python programming language. He began working on Python in 1989 as a personal “hobby” project, with the goal of designing a language that emphasized code readability and simplicity. The first official release of Python came in 1991, followed by Python 2 in 2000 and the major overhaul, Python 3, in 2008. Interestingly, the name “Python” was inspired not by the snake but by the British comedy series *Monty Python’s Flying Circus*, reflecting Guido’s preference for a language that was fun and engaging to use.

In the broader Python community, Guido van Rossum holds the honorary title of “Benevolent Dictator For Life” (BDFL). This role signified his continued authority and involvement in guiding the direction of Python, including making final decisions on controversial issues during its development. Although he has since stepped down from the BDFL role, at the time referenced in this slide, he was employed at Google, where he spent a portion of his time contributing to Python’s ongoing evolution. His leadership and vision have played a pivotal role in making Python one of the most popular and influential programming languages globally today.

# **Why Python Stands Out**

* **Simplicity and Efficiency**: Python's syntax is designed for clarity and simplicity, making it an excellent choice for beginners and experts alike. Its high-level data types allow for expressive code that can perform complex operations succinctly. Python encourages readable and maintainable code, with indentation replacing the clutter of brackets and no need for variable declarations.
* **Versatile Toolset**: With Python, you're not just learning a language; you're gaining access to an extensive ecosystem of standard modules and packages. These tools can serve as the foundation for virtually any project, from web development and data analysis to automation and scientific computing. Plus, the community-contributed packages further expand Python's capabilities into specialized fields.
* **Interactive Development**: Python's interpreter supports interactive modes, offering an exploratory coding environment. This feature is invaluable for learning, experimenting, and debugging, allowing for real-time feedback and adjustments.
* **Extensibility**: For those who know C, extending Python with new functionalities or connecting it to existing libraries is straightforward. This extensibility makes Python not just a language for application development but a platform for integrating and enhancing other software components.
* **Cross-Platform Compatibility**: Python runs on Windows, macOS, and Unix systems, ensuring your projects can reach the widest possible audience.

As we embark on this journey through Python's landscape, we encourage you to engage with the language actively. The best way to learn Python is by using it: experiment, build, break, and rebuild. Our tutorial modules are designed to introduce Python's fundamentals through hands-on examples and practical exercises, starting from basic expressions and data types to advanced concepts like exceptions and custom classes.

By the end of this exploration, you'll not only have a solid understanding of Python but also the skills to harness its potential for your projects, whether they're personal, professional, or purely exploratory. Welcome to the world of Python programming, where creativity meets capability.

Python is a high-level, general-purpose, and multi-paradigm programming language, meaning it supports various programming styles, including object-oriented, procedural, and functional programming. Its simplicity, readability, and versatility make it suitable for both beginners and professionals in a wide range of domains.

Python is highly portable and runs on a wide variety of platforms, including:

* Windows
* Mac OS X
* Linux/Unix
* AIX
* IBM i (AS/400)
* iOS
* Solaris
* VMS
* HP-UX

Python has extensive libraries and frameworks that support development across numerous domains:

* GUI development
* Web frameworks (e.g., Django, Flask)
* Multimedia applications
* Database programming
* Networking
* Test frameworks
* Automation and scripting
* Documentation generation
* Scientific computing (e.g., NumPy, SciPy)
* Image processing

Due to its clean and intuitive syntax, Python has influenced the design of many modern programming languages, including R, Ruby, and Swift Languages.

Python's wide adoption and influence across platforms and programming communities make it one of the most powerful and flexible languages available today.

# **Setting Up Python: Installation Guide for Windows and macOS**

Embarking on your Python programming journey begins with setting up the environment on your computer. This guide will walk you through installing Python on Windows and macOS, ensuring you are ready to dive into the world of programming with minimal fuss.

# **Installing Python on Windows**

* **Download the Installer:**

Visit the official Python website (python.org) and navigate to the Downloads section. The website should automatically offer you the latest version of Python for Windows. Click on the download link for the Windows installer.

* **Run the Installer:**

Once the installer is downloaded, locate it in your [downloads](https://www.python.org/downloads/) folder and double-click to run it. You will be greeted with the installation window.

**Important**: Ensure you check the box that says, "Add Python X.X to PATH" before clicking "Install Now." This step is crucial as it makes Python accessible from the Command Prompt.

* **Customize Installation (Optional):**

For most users, the default installation options will suffice. However, if you need to customize locations or configurations, you can click on “Customize installation” and adjust the settings accordingly.

* **Complete the Installation:**

Follow the on-screen instructions to complete the installation. Once finished, you will see a screen with a message saying, “Setup was successful.” Click "Close" to finish the setup process.

# **Installing Python on macOS**

* **Download the Installer:**

Head to the official Python website (python.org) and click on the Downloads tab. The site should automatically suggest the latest version of Python for macOS. Download the macOS installer.

* **Run the Installer:**

Open the downloaded package and follow the installation wizard. The wizard will guide you through the necessary steps, including agreeing to the license agreement and selecting an install location.

* **Install Python:**

Continue through the installer's steps and click "Install" to begin the installation process. You may need to enter your administrator password to authorize the installation.

* **Complete the Installation:**

Once the installation is complete, you will see a confirmation screen. Click "Close" to exit the installer.

# **Verifying Python Installation**

After installation, it is crucial to verify that Python is correctly installed and accessible from your system's command line or terminal.

## **On Windows:**

* + Open Command Prompt by typing ***cmd*** in the search bar and pressing Enter.
  + In the Command Prompt, type ***python --version*** and press Enter. If Python is correctly installed, you should see the version number displayed.

## **On macOS:**

* + Open Terminal by going to Applications > Utilities > Terminal.
  + Type ***python3 --version*** in the Terminal and press Enter. You should see the Python version number if the installation was successful.

# **How to Use Python Shell**

## **On Windows:**

* + **Start the Python Shell:** In the Command Prompt, type python and press Enter. You should see something like this:

A black screen with white text

Description automatically generated

* + **Using the Shell:** The ***>>>*** prompt indicates that you are now in the Python shell. You can type Python commands directly here. For example:

A black screen with a black background

Description automatically generated

* + To exit the Python shell, you can type ***exit()*** or press Ctrl + Z and then Enter.

## **On macOS:**

* + **Start the Python Shell:** In Terminal, type ***python3*** and press Enter. You should see something like this:

**A black screen with white text

Description automatically generated**

* + **Using the Shell:** The ***>>>*** prompt indicates that you are now in the Python shell. You can type Python commands directly here. For example:

**A black screen with a black background

Description automatically generated**

* + To exit the Python shell, you can type ***exit()*** or press ***Ctrl + D***.

# **Installation of Visual Studio Code**

After installing Python, an editor is needed to write and run Python codes.

## **Installation on Windows:**

* + Download Visual Studio Code
    - Go to the [Visual Studio Code website](https://code.visualstudio.com/).
    - Click on the "Download" button for Windows.
  + Install Visual Studio Code
    - Once the download is complete, open the installer file.
    - Follow the setup instructions: accept the agreement, choose the installation location, and select additional tasks (e.g., creating a desktop icon).
    - Click "Install" and wait for the installation to complete.
    - Once installed, launch Visual Studio Code.

## **On macOS:**

* + Download Visual Studio Code
    - Go to the [Visual Studio Code website](https://code.visualstudio.com/).
    - Click on the "Download" button for macOS.
  + Install Visual Studio Code
    - Once the download is complete, open the .dmg file.
    - Drag the Visual Studio Code app to the Applications folder.
    - Launch Visual Studio Code from the Applications folder.

# **Setting Up Python in Visual Studio Code**

After installing Visual Studio Code, Python Extension for Visual Studio Code is needed.

* Open Visual Studio Code.
* Go to the Extensions view by clicking the Extensions icon in the Activity Bar on the side of the window or by pressing Ctrl+Shift+X (Windows) or Cmd+Shift+X (macOS).
* Search for "Python" and click "Install" on the extension provided by Microsoft.

# **Running Python Code Using Command Prompt (Windows)**

* Open Command Prompt
  + Press ***Win+R***, type ***cmd***, and press Enter.
* Navigate to the Python Script Directory
  + Use the cd command to navigate to the directory where your Python script is located. For example:

*cd path\to\your\script*

* Run the Python Script
  + Execute your Python script by typing:

*python your\_script.py*

# **Running Python Code Using Terminal (macOS)**

* Open Terminal
  + Press ***Cmd+Space*** to open Spotlight Search, type ***Terminal***, and press Enter.
* Navigate to the Python Script Directory
  + Use the cd command to navigate to the directory where your Python script is located. For example:

*cd path\to\your\script*

* Run the Python Script
  + Execute your Python script by typing:

*python3 your\_script.py*

# **Running Python Code in Visual Studio Code**

* Open Visual Studio Code
  + Launch Visual Studio Code.
* Open Your Python Script
  + Open the file containing your Python script by navigating to ***File > Open File*** and selecting your script.
* Run the Python Script
  + Open the integrated terminal by selecting ***View > Terminal***.
  + Ensure the terminal is in the correct directory or navigate using the ***cd*** command.
  + Type the following command to run your script:

*python your\_script.py*

* + Alternatively, you can run the script by clicking the "Run" button provided by the Python extension.

*This OER module incorporates and adapts content from CSCI1301 (****Introduction to Computer Programming, Application Development, Computer Science I & II****) Open Course**materials by Myungjae Kwak, Jonathan Jenkins, Joobum Kim, and Kevin Floyd from Middle Georgia State University, available on the ALG Manifold platform. Updates have been made to align the material with our course objectives, instructional approach, and current technological practices.*

## **Reference and Additional Materials:**

<https://alg.manifoldapp.org/projects/itec2260_csci1301-mga>

<https://en.wikipedia.org/wiki/Guido_van_Rossum>

<https://docs.python.org/3/library/index.html>

<https://www.python.org/downloads/operating-systems/>

<https://docs.python.org/3/using/index.html>

<https://guides.library.uq.edu.au/>

<https://open.umn.edu/opentextbooks/textbooks/a-byte-of-python>

<https://www.python.org/downloads/>