Middle Technical University

Technical Engineering College of Artificial Intelligence

Department of Cybersecurity Technology Engineering

Third Year

PYTHON PROGRAMMING

Lecture 1: Introduction

By: Ass. Lec. Abbas Aqeel Kareem

Lecture 1: Introduction

This Lecture prepares you to learn how to program with Python. Preparation includes a brief introduction to computers and computing, programming, and programming languages, as well as the installation of Python and Python interactive programming and integrated development environments (IDEs)

Learning Objectives

After completing this chapter, you should be able to

- Discuss computer programming languages.
- Describe Python and discuss its development, features, and advantages.
- Install Python and required Python IDEs on the computer.
- Get Python and Python IDEs running for the learning activities included in this textbook.

1. Computer programs

A **computer** is an electronic device that stores and processes information. Examples of computers include smartphones, tablets, laptops, desktops, and servers. Technically, a **program** is a sequence of instructions that a computer can run. Programs help people accomplish everyday tasks, create new technology, and have fun.

One of the key principles of modern computers is using stored programs, which are sequences of instructions telling computers what to do. The task of writing programs for computers is called programming.

During the first generation of computers, **machine languages** and assembly languages were used to write programs. Machine languages use sequences of 0 and 1 to code instructions for computers. Programs in a machine language can be directly understood by the CPU of the target computer. The following is an example of machine language for the Intel 8088 CPU:

```
00000011 00000100

10001000 11000011
```

As you can see, although machine language is friendly to computers, it is not friendly to people, because the instruction itself does not tell us what it does in a human- readable language. To solve this problem, **assembly language** was invented. The two instructions above are written in assembly language as follows:

ADD AX, [SI]

MOV BL, AL

The instruction in assembly language is much more friendly to people because one can more easily tell what each instruction does. However, assembly language is not as friendly to computers. For computers to understand programs in assembly language, an **assembler** is needed to translate the programs into code in machine language (Figure 1).

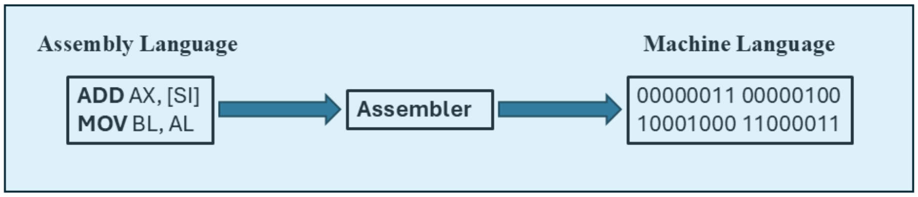
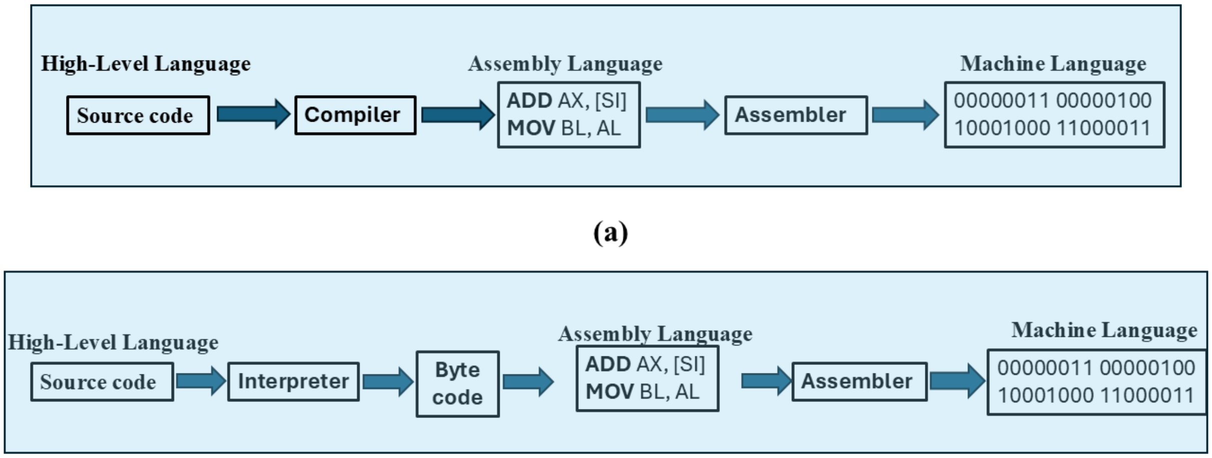


Figure 1: Evolution from machine language to assembly

With assembly language, programming is still a difficult task because instructions in an assembly language are mostly direct mapping of their machine language equivalent and only describe fine and tiny steps of CPU operations. It is difficult to program in assembly language to solve real- world problems.

High- level programming languages are friendly to people but not friendly to computers. For computers to understand and execute programs written in a high- level programming language, they must be translated into machine language. Based on how programs in high- level languages are translated into target machine language, high- level programming languages are either **compiled** or **interpreted** (see Figure 2). A program written in a compiled programming language such as C or C++ needs to be **compiled** into its target machine's codes and linked with copies of code in static libraries, or linked with references to code in dynamic or shared libraries, so that the resulting object can then be executed by the target computer.



(b)

Figure 2: (a) Compilation process for compiled languages; (b) Interpretation process for interpreted languages.

Programs written in an interpreted programming language can be executed directly by the **interpreter** of the programming language, in view of programmers and users. Behind the scenes, however, to speed up the execution of programs written in an interpreted programming language, the programs are often translated first into some intermediate codes, often called **bytecodes**. The bytecodes are then executed within a so called **virtual machine** built into the interpreter of the programming language.

2. Python Programming Language

2.1.The Development and Implementation of Python

Python is an interpreted, high- level, general- purpose programming language. It is one of the most popular programming languages in the world as shown in figure 3. Python was originally conceived and developed in the Netherlands by Guido van Rossum in the late 1980s. Its first release was Python 0.9.0 in 1991. Python 2.0 was released in 2000, nine years after the release of Python 0.9.0. Most of the development of Python was accomplished between 2005 and 2013, when Guido van Rossum was working at Google, where he spent half of his time on the development of the programming language. The latest Python 2 release is Python 2.7.16, whereas the latest release of Python 3 is Python 3.9.0, at the time of writing. Python is an interpreted programming language rather than a compiled programming language. Programs written in Python don’t need to be com piled into target machine code. Instead, they only need to be translated into bytecodes, then executed by a Python Virtual Machine (PVM), which is built into the Python interpreter.

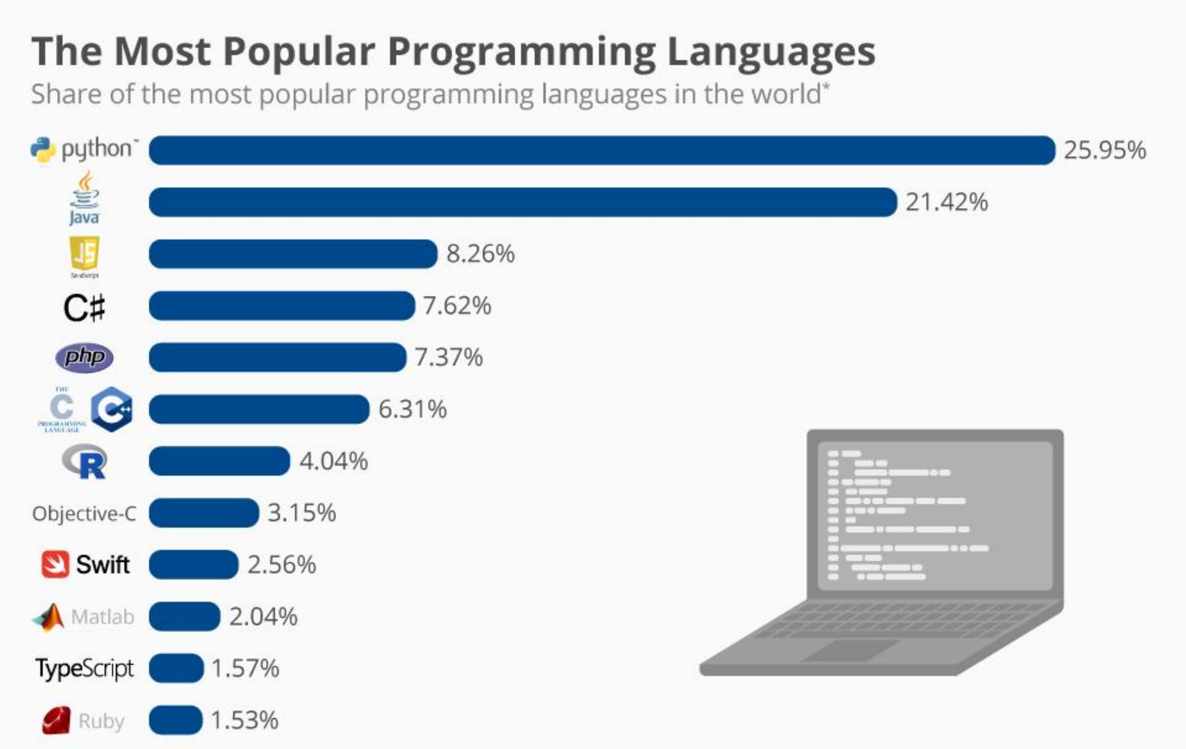


Figure 3: The Most Popular Programming Languages

Python language has different implementations with different names. CPython, written in C and known simply as Python, is what we will be using, and it is the implementation you will get from Python standard distribution at python .org by default. In addition to CPython, the following are some alterna tive implementations of Python:

1. JPython or Jython, and implementation for running on Java Virtual Machine (JVM)
2. IronPython, an implementation for running on .NET
3. PyPy, an implementation for working with a just- in- time (JIT) compiler
4. Stackless Python, an implementation of a branch of CPython supporting microthreads
5. MicroPython, an implementation for running on microcontrollers

These implementations may be of interest only for some special applications.

2.2. Applications of Python Programming

Python is used in many application domains. Here's a sampling.

1. **Web and Internet Development** : Python offers many choices for web development:
   - Frameworks such as Django and Pyramid.
   - Micro-frameworks such as Flask and Bottle.
   - Advanced content management systems such as Plone and django CMS.
2. **Scientific and Numeric :** Python is widely used in scientific and numeric computing:
   - SciPy is a collection of packages for mathematics, science, and engineering.
   - Pandas is a data analysis and modeling library.
   - IPython is a powerful interactive shell that features easy editing and recording of a work session, and supports visualizations and parallel computing.
   - The Software Carpentry Course teaches basic skills for scientific computing, running bootcamps and providing open-access teaching materials.
3. **Education:** Python is a superb language for teaching programming, both at the introductory level and in more advanced courses.
   - Books such as How to Think Like a Computer Scientist, Python Programming: An Introduction to Computer Science, and Practical Programming.
   - The Education Special Interest Group is a good place to discuss teaching issues.
4. **Desktop GUIs:** The Tk GUI library is included with most binary distributions of Python. Some toolkits that are usable on several platforms are available separately:
   - wxWidgets
   - Kivy, for writing multitouch applications.
   - Qt via pyqt or pyside
5. **Software Development:** Python is often used as a support language for software developers, for build control and management, testing, and in many other ways.
   - SCons for build control.
   - Buildbot and Apache Gump for automated continuous compilation and testing.

    - Roundup or Trac for bug tracking and project management.

6. **Business Applications:** Python is also used to build ERP and e-commerce systems:
    - Odoo is an all-in-one management software that offers a range of business applications that form a complete suite of enterprise management applications.
    - Tryton is a three-tier high-level general purpose application platform.

2.3. Popular Applications Made With Python

1. Instagram: A major social media platform that utilizes Python for its backend, notably leveraging the Django framework.
2. Spotify: The music streaming giant uses Python extensively for data analysis, recommendation systems, and various backend services.
3. Dropbox: This popular cloud storage service relies on Python for its server-side logic and backend operations.
4. Reddit: The social news aggregation and discussion platform has a significant portion of its codebase in Python.
5. Pinterest: Another prominent social media platform that uses Python for its backend.
6. Uber: The ride-sharing and logistics company utilizes Python for various functionalities, including its backend services.
7. Google: Python is a core language at Google, used in various services like YouTube, Google App Engine, and for internal tools and infrastructure.

2.4. Advantages and limitations

- **Designed for everyone**: Clean, readable, and beginner-friendly syntax.
- **General-purpose and powerful**: Suitable for a wide range of applications.
- **Low overhead**: Simpler than languages like Java (e.g., “Hello World!” in one line).
- **Supports multiple paradigms**: Structured, imperative, object-oriented, functional, and procedural programming.
- **Extensive libraries**: Thousands of third-party packages plus large standard libraries. Examples: NumPy, SciPy, Pandas, Matplotlib, TensorFlow.
- **Flexibility**: Used in fields like data analytics, machine learning, web apps, and automation

While Python offers substantial benefits, it is not without limitations. As an interpreted language, its execution speed is typically slower than that of compiled languages such as C or C++, making it less suitable for applications where real-time performance is critical.

2.5. Install Python

Windows doesn’t typically come with a system Python. Fortunately, installation involves little more than downloading and running the Python installer from the Python.org website.

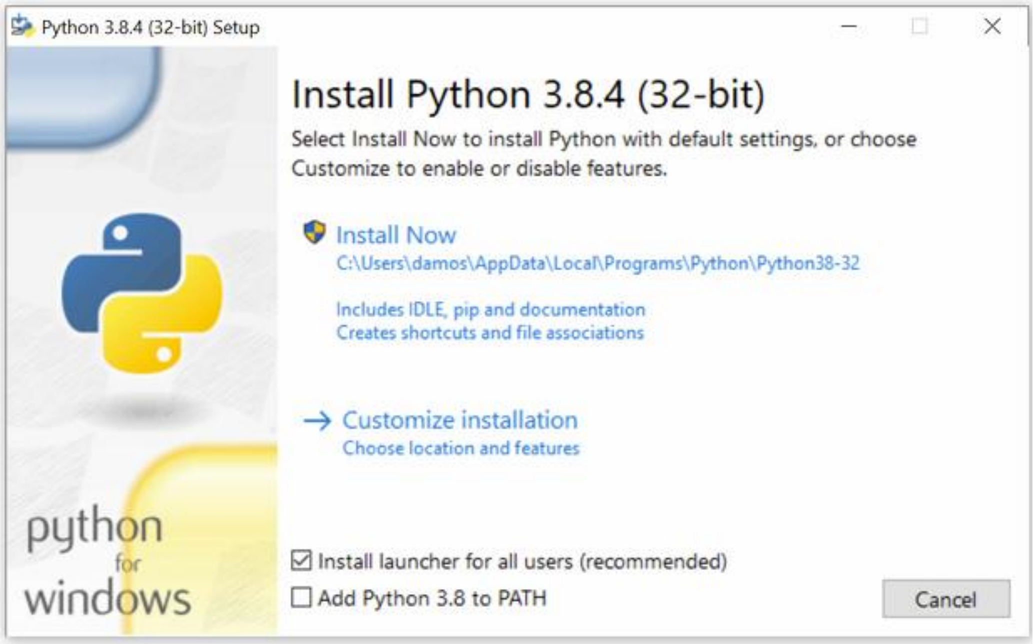
Step 1: Download the Python3 Installer

Open a web browser and navigate to the following URL: https://www.python.org/downloads/windows/

Click Latest Python 3 Release- Python 3.x.x located beneath the “Python Releases for Windows” heading near the top of the page. As of this writing, the latest version was Python 3.11. Then scroll to the bottom and click Windows x86-64 executable in staller to start the download.

Step 2: Run the Installer

Open your Downloads folder in Windows Explorer and double-click the file to run the installer. A dialog that looks like the following one will appear:



Python 3.8.4 (32-bit) Setup

Install Python 3.8.4 (32-bit)

Select Install Now to install Python with default settings, or choose Customize to enable or disable features.

Install Now
C:\Users\damos\AppData\Local\Programs\Python\Python38-32

Includes IDLE, pip and documentation
Creates shortcuts and file associations

→ Customize installation
Choose location and features

☑ Install launcher for all users (recommended)
□ Add Python 3.8 to PATH

It’s okay if the Python version you see is greater than 3.9.0 as long as the version is not less than 3. Make sure you select the box that says Add Python3.x to PATH. If you install Python without selecting this box, then you can run the installer again and select it.

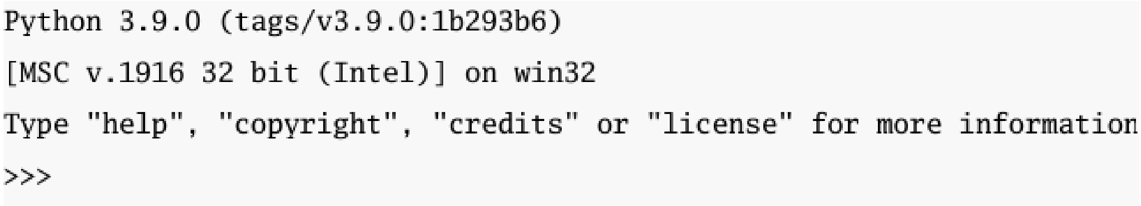
Click Install Now to install Python 3. Wait for the installation to finish, then continue to open IDLE.

You can open IDLE in two steps:

1. Click the Start menu and locate the Python 3.9 folder.

2. Open the folder and select IDLE (Python 3.9).

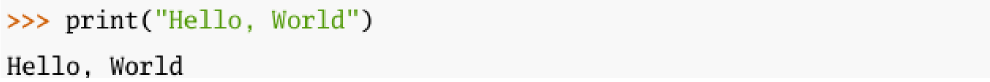
IDLE's interactive window contains a Python shell, which is a textual user interface used to interact with the Python language. You can type a bit of Python code into the interactive window and press Enter to immediately see the results. Hence the name interactive window. The interactive window opens automatically when you start IDLE. You'll see the following text, with some minor differences depending on your setup, displayed at the top of the window:



```
Python 3.9.0 (tags/v3.9.0:1b293b6)
[MSC v.1916 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license" for more information
>>>
```

This text shows the version of Python that IDLE is running. You can also see information about your operating system and some commands you can use to get help and view information about Python.

The >>> symbol in the last line is called the prompt. This is where you'll type in your code. Let's try something a little more interesting than adding numbers. A rite of passage for every programmer is writing a program that prints the phrase "Hello, World" on the screen. At the prompt in the interactive window, type the word print followed by a set of parentheses with the text "Hello, World" inside:



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

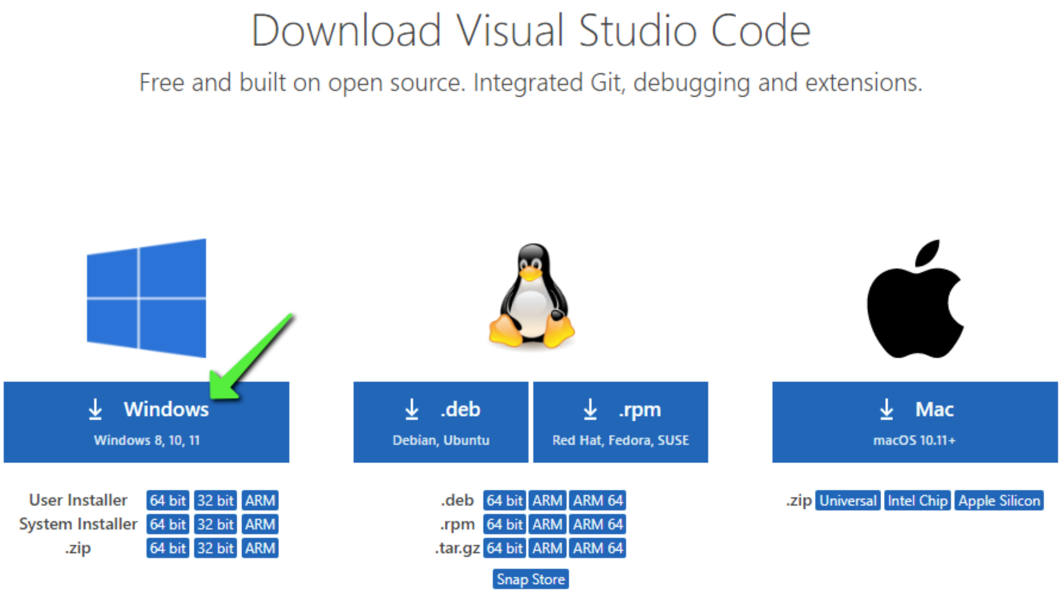
A function is code that performs some task and can be invoked by a name. The above code invokes, or calls, the print() function with the text "Hello, World" as input.

2.6. Program with VS Code IDE

Interactive programming environments like IDLE are good for testing Python statements, code blocks, and some real programming tasks such as data analytics, where interactive programming is more suitable. However, writing programs that contain thousands of lines of code in a Python interactive shell or Jupyter Notebook is inconvenient. As previously mentioned, we will be using VS Code as our IDE. **Installing Visual Studio Code on Windows:**

Step 1: Downloading Visual Studio Code

Navigate to the following link: https://code.visualstudio.com/download Click on the blue download button that says Windows under the Windows Logo:

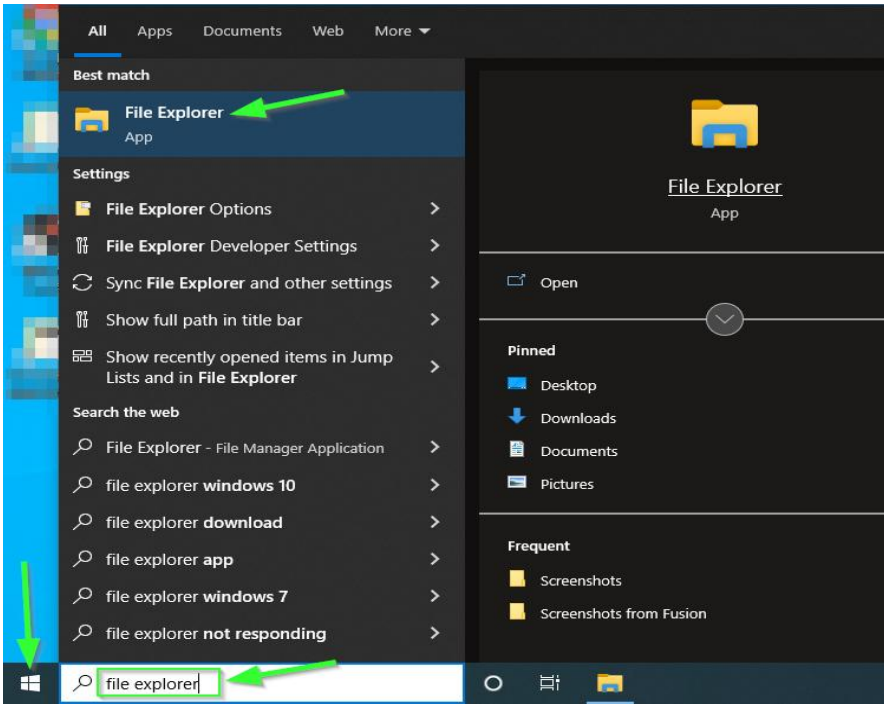


Download Visual Studio Code

Free and built on open source. Integrated Git, debugging and extensions.

Step 2: Launching the Visual Studio Code Installation Package

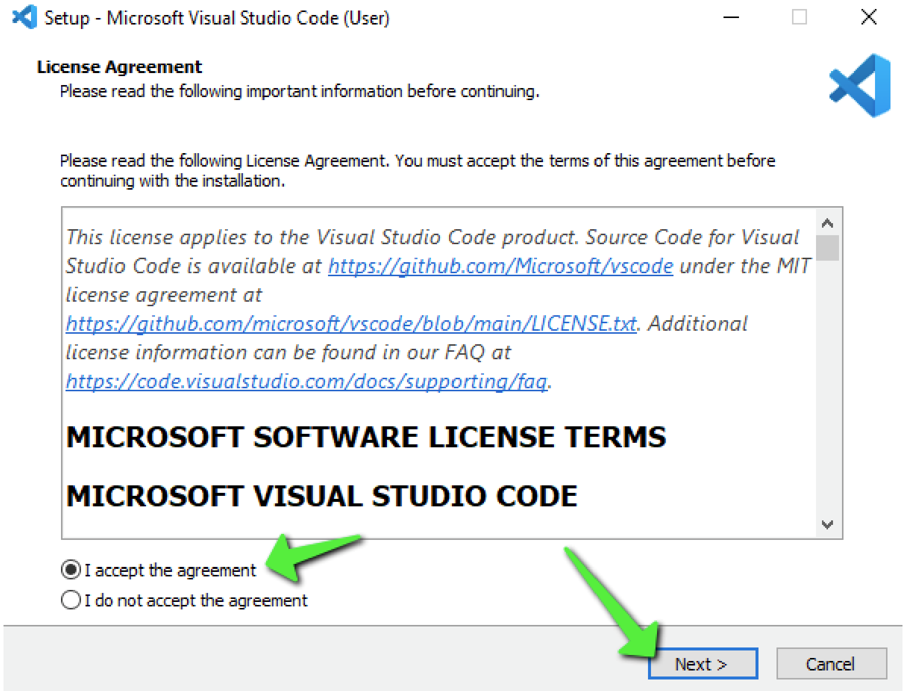
Once the download completes, locate it by clicking on the Start Button and type ‘File Explorer’. Launch File Explorer:



On the left side of the File Explorer window, locate and click on Downloads. Double-click on the downloaded 'VSCodeUserSetup-x64-1.x.x.exe file.

Step 3: Installing

Once the installer launches, step through the installation process. First, accept the License Agreement, then click Next .



Setup - Microsoft Visual Studio Code (User)

License Agreement
Please read the following important information before continuing.

Please read the following License Agreement. You must accept the terms of this agreement before continuing with the installation.

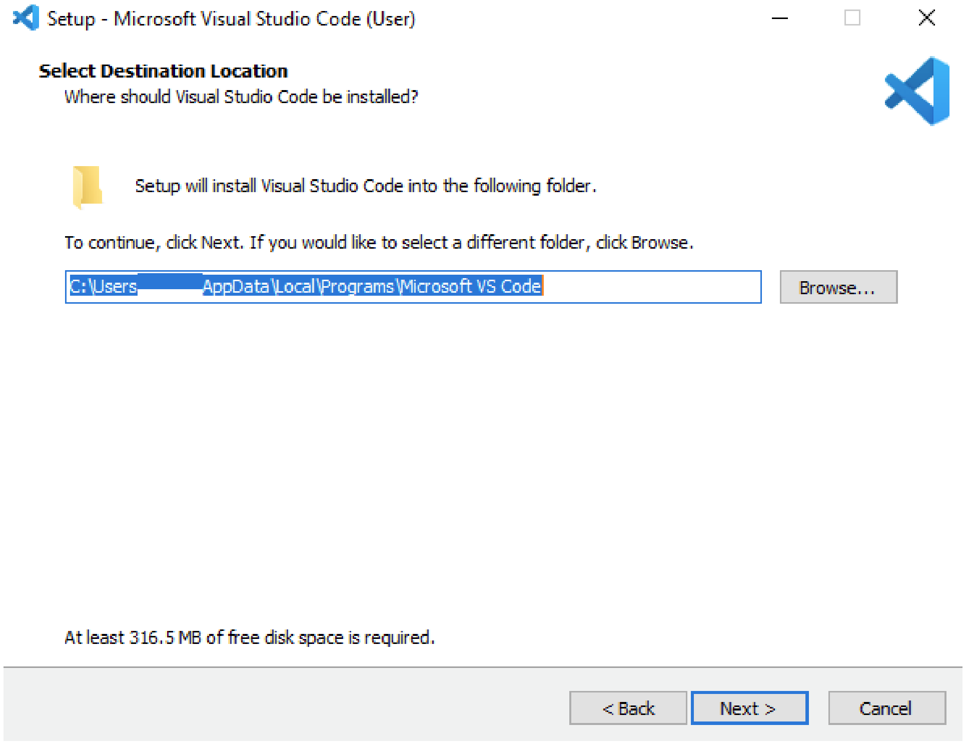
This license applies to the Visual Studio Code product. Source Code for Visual Studio Code is available at https://github.com/Microsoft/vscode under the MIT license agreement at https://github.com/microsoft/vscode/blob/main/LICENSE.txt. Additional license information can be found in our FAQ at https://code.visualstudio.com/docs/supporting/faq.

MICROSOFT SOFTWARE LICENSE TERMS

MICROSOFT VISUAL STUDIO CODE

● I accept the agreement
○ I do not accept the agreement

Accept the default location for installation by clicking Next >.



Setup - Microsoft Visual Studio Code (User)

Select Destination Location
Where should Visual Studio Code be installed?

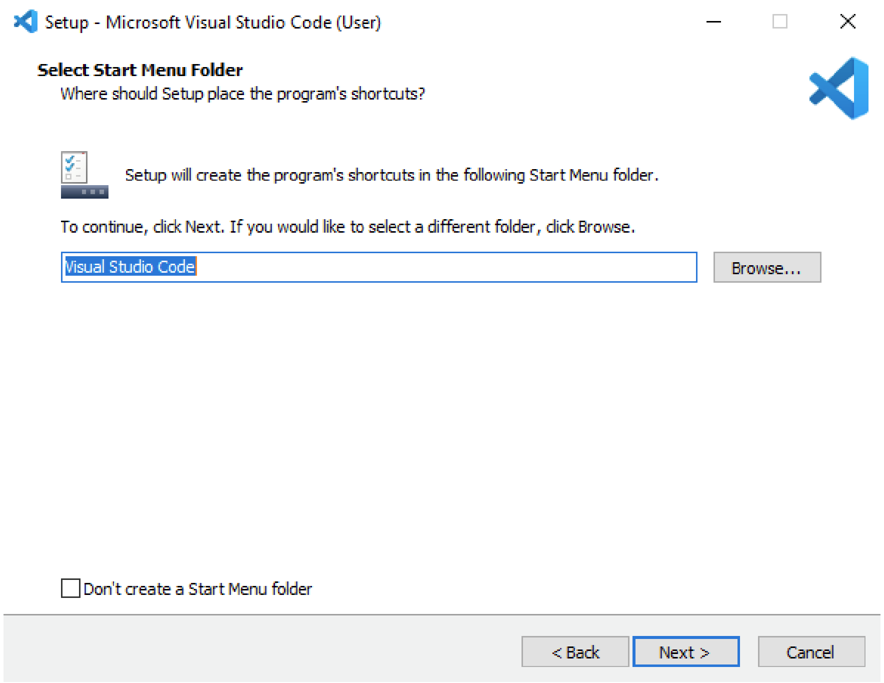
Setup will install Visual Studio Code into the following folder.

To continue, click Next. If you would like to select a different folder, click Browse.

C:\Users[redacted]AppData\Local\Programs\Microsoft VS Code

At least 316.5 MB of free disk space is required.

Accept the default Start Menu folder name by clicking Next >.



Setup - Microsoft Visual Studio Code (User)

Select Start Menu Folder

Where should Setup place the program's shortcuts?

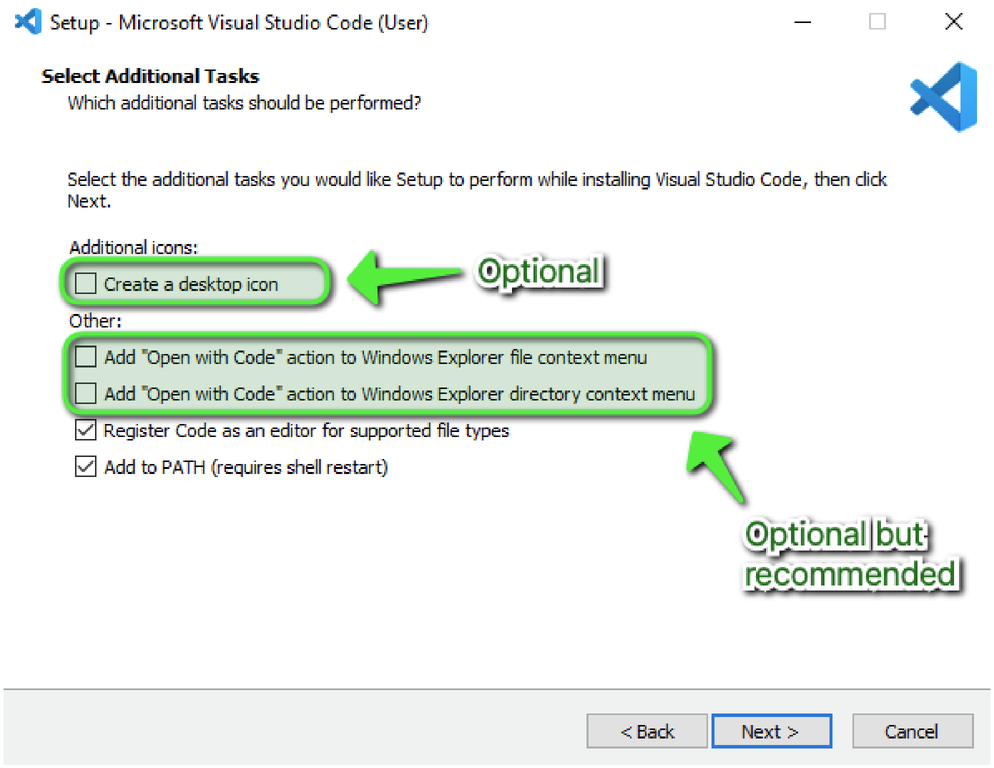
Setup will create the program's shortcuts in the following Start Menu folder.

To continue, click Next. If you would like to select a different folder, click Browse.

Visual Studio Code

□ Don't create a Start Menu folder

Optionally check the boxes for 'Creating a desktop icon', and adding VS Code to the Right-Click menu functionality of Windows File Explorer, then click Next >.



Setup - Microsoft Visual Studio Code (User)

Select Additional Tasks

Which additional tasks should be performed?

Select the additional tasks you would like Setup to perform while installing Visual Studio Code, then click Next.

Additional icons:

□ Create a desktop icon

Other:

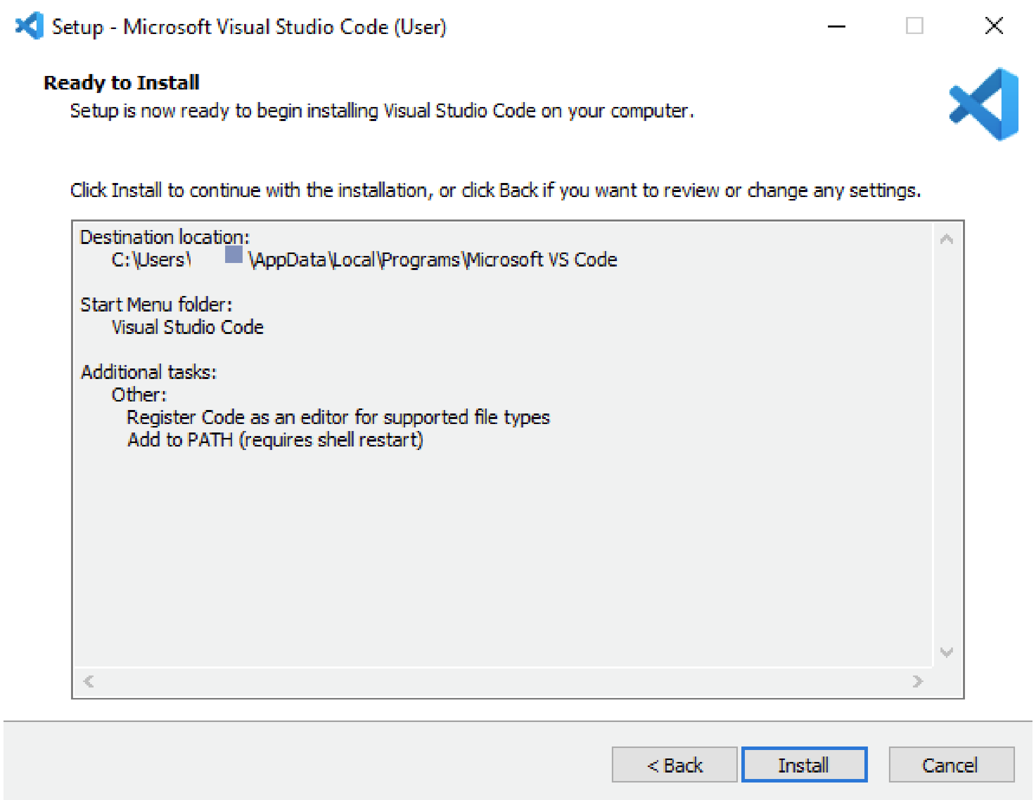
□ Add "Open with Code" action to Windows Explorer file context menu

□ Add "Open with Code" action to Windows Explorer directory context menu

☑ Register Code as an editor for supported file types

☑ Add to PATH (requires shell restart)

Confirm the installation options, then click Install.


Setup - Microsoft Visual Studio Code (User)

Ready to Install

Setup is now ready to begin installing Visual Studio Code on your computer.

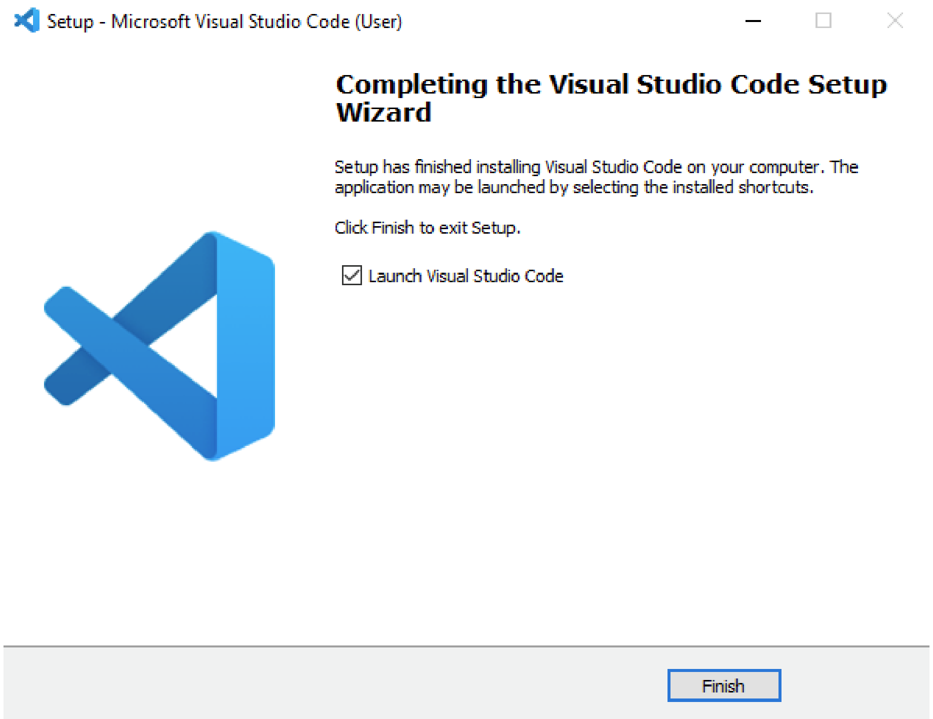
Click Install to continue with the installation, or click Back if you want to review or change any settings.

Destination location:
C:\Users\ [redacted] \AppData\Local\Programs\Microsoft VS Code

Start Menu folder:
Visual Studio Code

Additional tasks:
Other:
Register Code as an editor for supported file types
Add to PATH (requires shell restart)

The installation will proceed.


Setup - Microsoft Visual Studio Code (User)

Completing the Visual Studio Code Setup Wizard

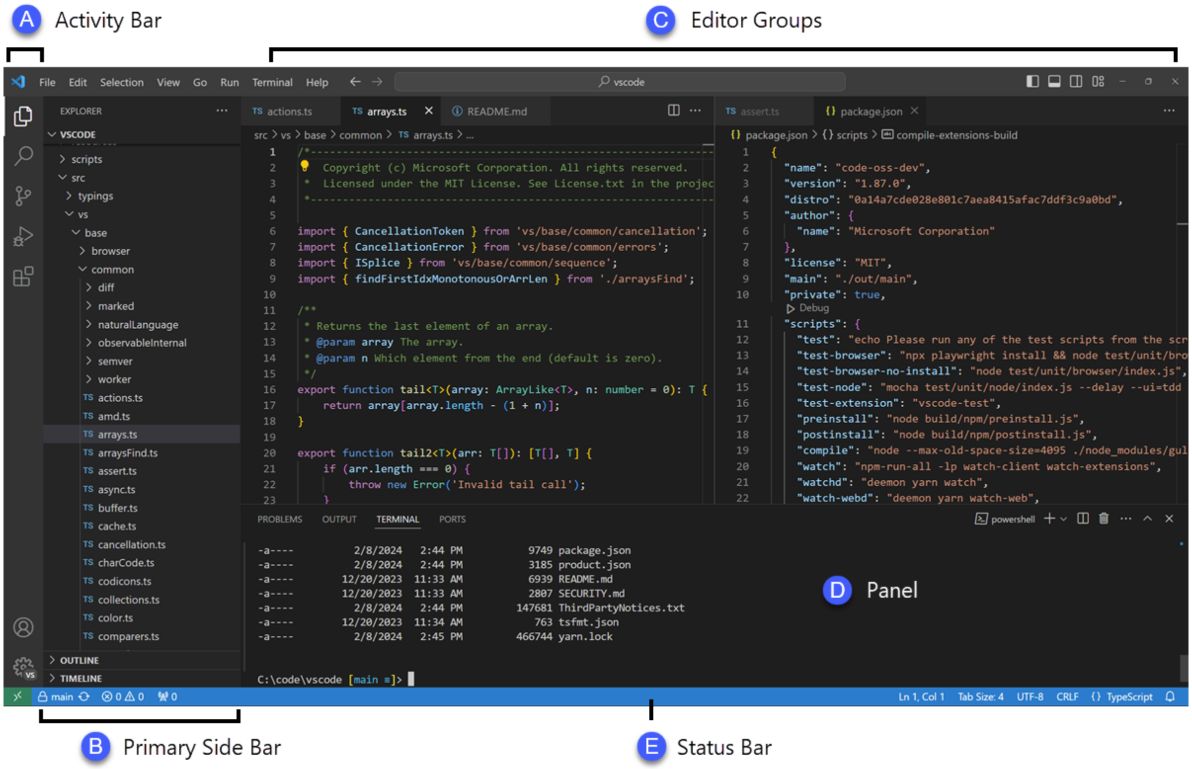
Setup has finished installing Visual Studio Code on your computer. The application may be launched by selecting the installed shortcuts.

Click Finish to exit Setup.

☑ Launch Visual Studio Code

Click Finish to exit the installation and (by default) launch Visual Studio Code:

The Visual Studio Code installer will create an icon in the Start Menu. To locate it, click on the Start Menu and search for 'Code'.



```
/*---------------------------------------------------------------
 * Copyright (c) Microsoft Corporation. All rights reserved.
 *  Licensed under the MIT License. See License.txt in the projec
 *---------------------------------------------------------------

import { CancellationToken } from 'vs/base/common/cancellation';
import { CancellationError } from 'vs/base/common/errors';
import { ISplice } from 'vs/base/common/sequence';
import { findFirstIdxMonotonousOrArrLen } from './arraysFind';

/**
 * Returns the last element of an array.
 * @param array The array.
 * @param n Which element from the end (default is zero).
 */
export function tail<T>(array: ArrayLike<T>, n: number = 0): T {
    return array[array.length - (1 + n)];
}

export function tail2<T>(arr: T[]): [T[], T] {
    if (arr.length === 0) {
        throw new Error('Invalid tail call');
    }
```

```
{
  "name": "code-oss-dev",
  "version": "1.87.0",
  "distro": "0a14a7cde028e801c7aea8415afac7ddf3c9a0bd",
  "author": {
    "name": "Microsoft Corporation"
  },
  "license": "MIT",
  "main": "./out/main",
  "private": true,
  "scripts": {
    "test": "echo Please run any of the test scripts from the scr
    "test-browser": "npx playwright install && node test/unit/bro
    "test-browser-no-install": "node test/unit/browser/index.js",
    "test-node": "mocha test/unit/node/index.js --delay --ui=tdd
    "test-extension": "vscode-test",
    "preinstall": "node build/npm/preinstall.js",
    "postinstall": "node build/npm/postinstall.js",
    "compile": "node --max-old-space-size=4095 ./node_modules/gul
    "watch": "npm-run-all -lp watch-client watch-extensions",
    "watchd": "deemon yarn watch",
    "watch-webd": "deemon yarn watch-web",
```

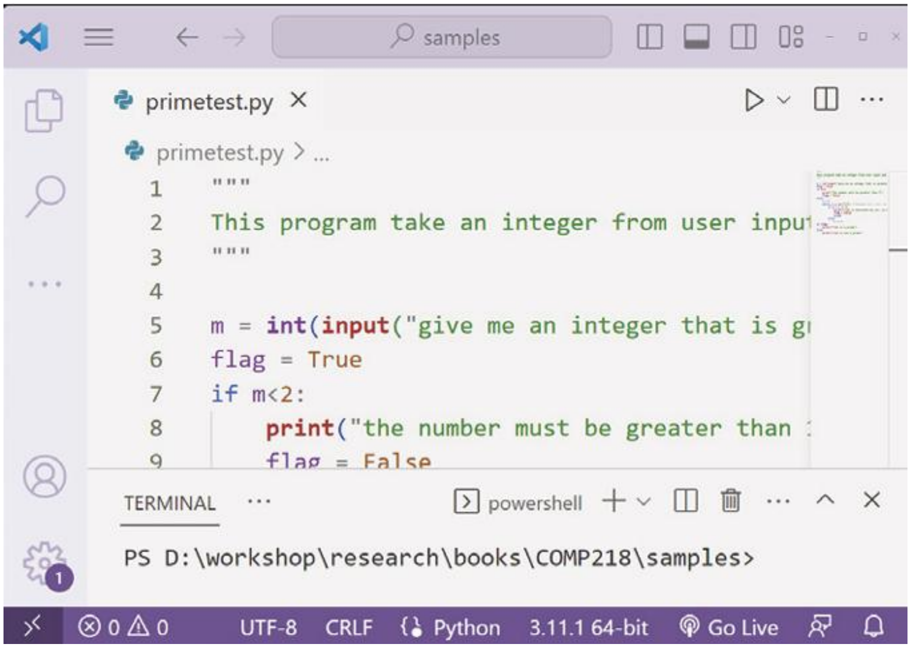
```
-a----          2/8/2024   2:44 PM           9749 package.json
-a----          2/8/2024   2:44 PM           3185 product.json
-a----        12/20/2023  11:33 AM           6939 README.md
-a----        12/20/2023  11:33 AM           2807 SECURITY.md
-a----          2/8/2024   2:44 PM         147681 ThirdPartyNotices.txt
-a----        12/20/2023  11:34 AM            763 tsfmt.json
-a----          2/8/2024   2:45 PM         466744 yarn.lock

C:\code\vscode [main ≡]>
```

VS Code comes with a simple and intuitive layout that maximizes the space provided for the editor, while leaving ample room to browse and access the full context of your folder or project. The user interface is divided into six main areas:

- Editor - The main area to edit your files. You can open as many editors as you like side by side vertically and horizontally.
- Primary Side Bar - Contains different views like the Explorer to assist you while working on your project.
- Secondary Side Bar - Opposite the Primary Side Bar. By default, contains the Chat view. Drag and drop views from the Primary Side Bar to the Secondary Side Bar to move them.
- Status Bar - Information about the opened project and the files you edit.
- Activity Bar - Located on the far left-hand side. Lets you switch between views and gives you additional context-specific indicators, like the number of outgoing changes when Git is enabled. You can change the position of the Activity Bar.
- Panel - An additional space for views below the editor region. By default, it contains output, debug information, errors and warnings, and an integrated terminal. The Panel can also be moved to the left or right for more vertical space.

Our first small project is to write a program that takes an integer from the user input and tells whether it is a prime number or not. To create a Python program file within VS Code, choose "New File" from File, and then save the file as xyz .py, where xyz is your preferred name for the small project. Here, we use "primetest .py." VS Code will ask you to choose a folder for the program, and you may create one and then select it to open after the file is created.



```
"""
This program take an integer from user input
"""


m = int(input("give me an integer that is gr
flag = True
if m<2:
    print("the number must be greater than :
    flag = False
```

```
PS D:\workshop\research\books\COMP218\samples>
```

At this time, you are not required to fully understand the program, but if you are interested, you may type the code line by line into your VS Code and run it by clicking the play button at the top- right corner of the VS Code window, just to get a taste of programming in VS Code with Python. Please note that if you have multiple Python program files open in VS Code and want to run a particular one within the IDE, you will need to click the file to make it active; then, within the editing area, click the right button of the mouse to pop up a menu and select run for your particular program. This is even more important if you have multiple editing tabs open for different programs.

3. Basics of Python

3.1. Basic Output

The print() function displays output to the user. Output is the information or result produced by a program. The sep and end options can be used to customize the output. Table 1 shows examples of sep and end. Multiple values, separated by commas, can be printed in the same statement.

Example 1: uses of print()

Code	Output

`print("Today is Monday.")`
`print("I like string beans.")`
	`Today is Monday.`
	`I like string beans.`

`print("Today", "is", "Monday")`
`print("Today", "is", "Monday", sep="...")`
	`Today is Monday`
	`Today...is...Monday`

`print("Today is Monday, ", end="")`
`print("I like string beans.")`
	`Today is Monday, I like string beans.`

`print("Today", "is", "Monday", sep="? ", end="!!")`
`print("I like string beans.")`
	`Today? is? Monday!!I like string beans.`

By default, each value is separated by a space character in the output. The sep option can be used to change this behavior. By default, the print() function adds a newline character at the end of the output. A newline character tells

3.2. Basic input

Computer programs often receive input from the user. Input is what a user enters into a program. An input statement, variable = input("prompt"), has three parts:

1. A variable refers to a value stored in memory. In the statement above, variable can be replaced with any name the programmer chooses.

2. The input() function reads one line of input from the user. A function is a named, reusable block of code that performs a task when called. The input is stored in the computer's memory and can be accessed later using the variable.

3. A prompt is a short message that indicates the program is waiting for input. In the statement above, "prompt" can be omitted or replaced with any message.

Example 2: uses of input()

Code	Output

`variable = input("prompt: ")`
`print("the variable = ",variable)`
	`prompt: 5`
	`the variable = 5`

3.3. Variables

In Python, variables are names that can be assigned a value and then used to refer to that value throughout your code. Values are assigned to variable names using a special symbol called the assignment operator (=). The = operator takes the value to the right of the operator and assigns it to the name on the left. Ex: `age = 6` or `birth = "May 15"`. The left side of the assignment statement is a variable, and the right side is the value the variable is assigned.

Python has reserved words, known as keywords, which have special functions and cannot be used as names for variables (or other objects).

Table 1: Keywords

False	await	else	import	pass
None	break	except	in	raise
True	class	finally	is	return
and	continue	for	lambda	try
as	def	from	nonlocal	while
assert	elif	asynch	del	global
with	yield	if	not	or