A logo of the university of scotland

Description automatically generated

School of Computing, Engineering and Physical Sciences

MSc Information Technology

**COMP11124**

**Object Oriented Programming**

**Course Work**

**Submitted By:**

|  |  |
| --- | --- |
| **Banner ID** | **Name** |
| Sambat Adhikari | B01804202 |
| Mahesh Regmi | B01805066 |
| Suraj Shrestha | B01811462 |
| Jasleen | B01804328 |
| Muhammad Fahad | B01813165 |

**17 July 2025**

Table of Contents

[1. INTRODUCTION 3](#_Toc202136643)

[2. Week 1 – Introduction to Python and Development Environment 4](#_Toc202136644)

[2.1 Python Overview 4](#_Toc202136645)

[2.2 Setting up Visual Studio Code (VS Code) 6](#_Toc202136646)

[2.3 Outcome 9](#_Toc202136647)

# INTRODUCTION

The aim of this course work report is to introduce and build upon the programming skills using Python language, with main focus on object-oriented programming (OOP) paradigm. The following report contains a collection of programs and exercises completed as a group to demonstrate the skills and knowledge obtained during the study of Object-Oriented Programming module (COMP11124).

The report consists of multiple concepts learned and practiced from week 2 through to week 8. Every week focuses on a specific number of programming concepts, starting from basic fundamental aspects of python programming to more complex and object-oriented concepts such as class and objects.

This coursework is undertaken by five members, all of whom have worked together to contribute to the completion of tasks, explaining the logic and techniques to apply theoretical knowledge into practical implementation. The report also includes relevant code snippets, screenshots and explanations. The structure of the report follows a week-by-week breakdown ensuring clarity and relevance to module’s learning outcomes.

# Week 1 – Introduction to Python and Development Environment

In Week 1, the goal was to gain a brief insight into the development environment and develop a basic foundation of the Python programming language. Before initiating the learning process, it was crucial to set up the proper tools required for writing and executing python code.

### 2.1 Python Overview

Python is a high-level, interpreted programming language known for its simple syntax and readability. It was created by Guido van Rossum and released in 1991. The most recent major version of Python is Python 3, which is used for the preparation of this report.

Python has similarities to the English language with influence from mathematics. It is a widely popular programming languages in the computer world that can be used for a variety of applications like building web services, software development, handling big data or performing complex mathematics.

**Installation of Python**

Download and install Python from <https://www.python.org/downloads/>



Figure 1: Python Download

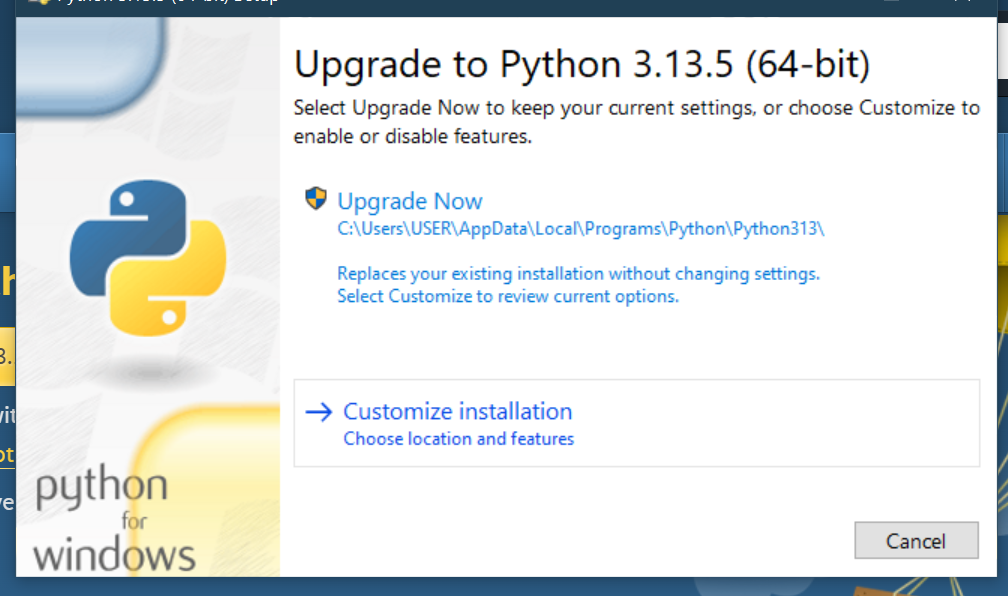


Figure 2: Python Installation

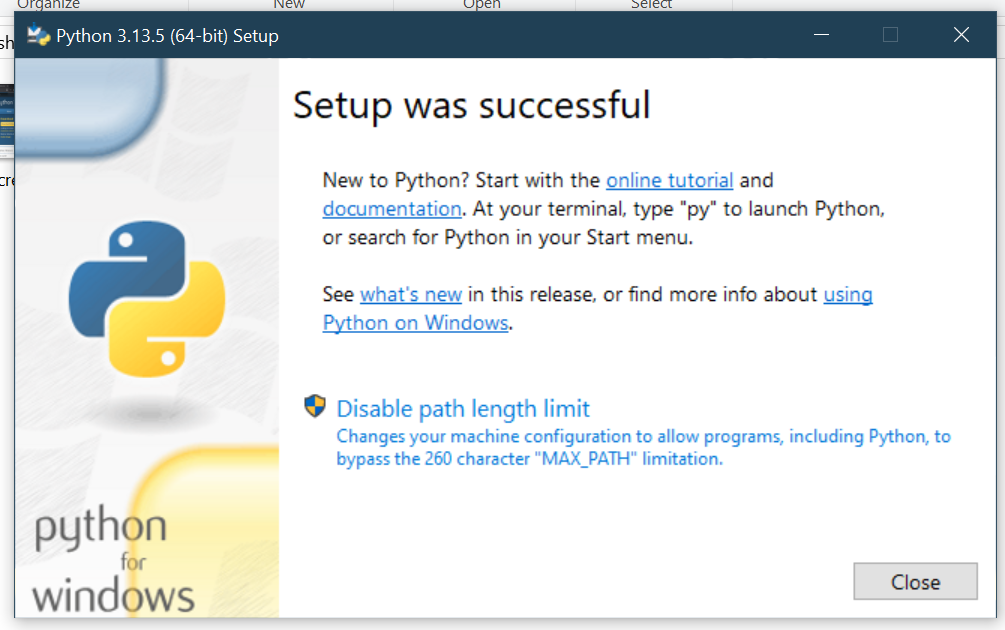


Figure 3 : Python Setup Completed

### 2.2 Setting up Visual Studio Code (VS Code)

Visual Studio Code was chosen as the integrated development environment (IDE) for this coursework. VS Code is a lightweight and opensource IDE developed by Microsoft offering feature such as:

1. Support for Python through the Python extension.
2. Built-in debugger to improve development process.
3. Provides fully featured integrated terminal to run shell commands.
4. User-friendly interface, syntax highlighting and intelligent code completion.

**Installation of VS Code**

Download and install Visual Studio Code from <https://code.visualstudio.com/download>

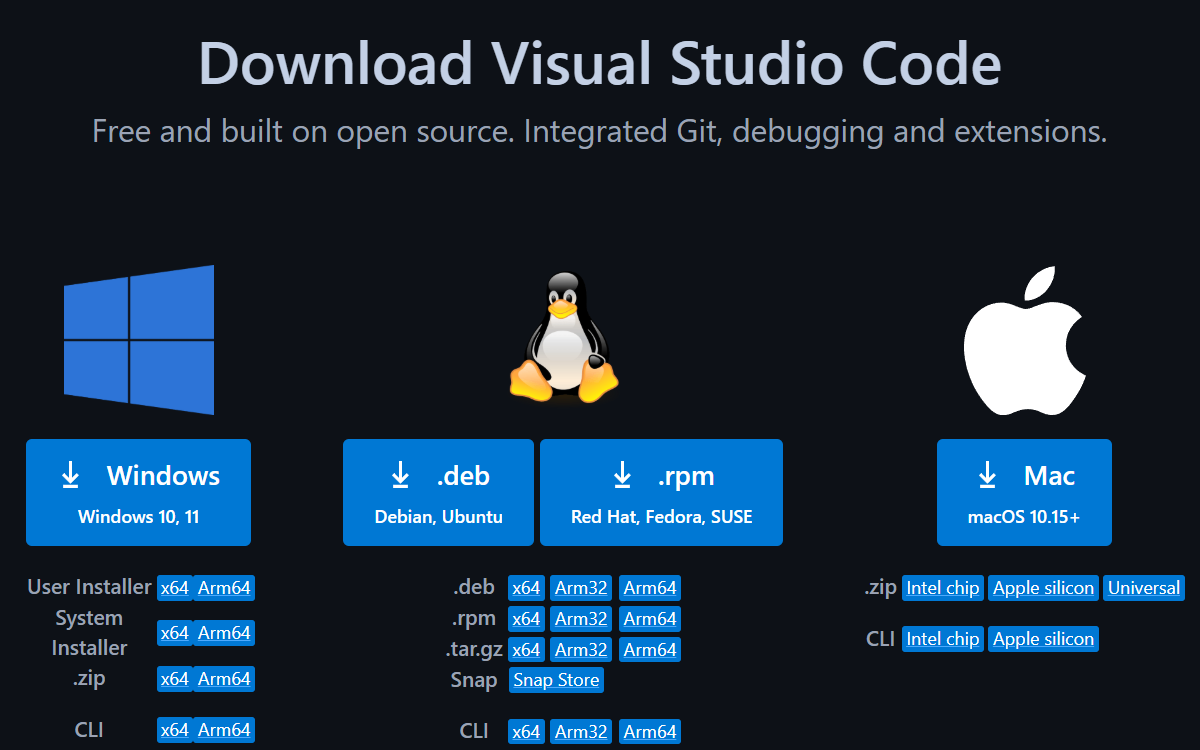


Figure 4:Visual Studio Code Download

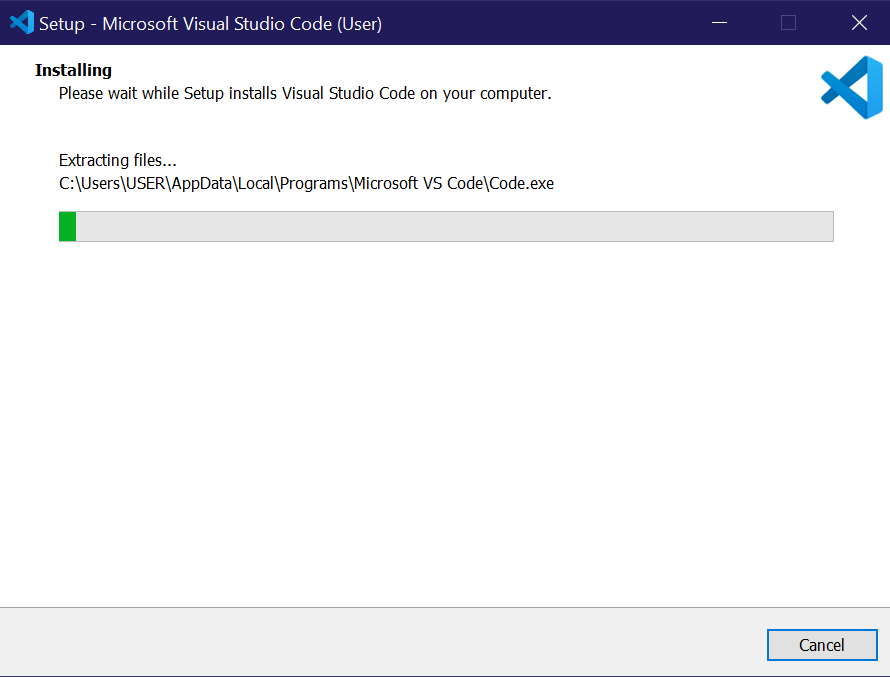


Figure 5: Visual Studio Code Installation

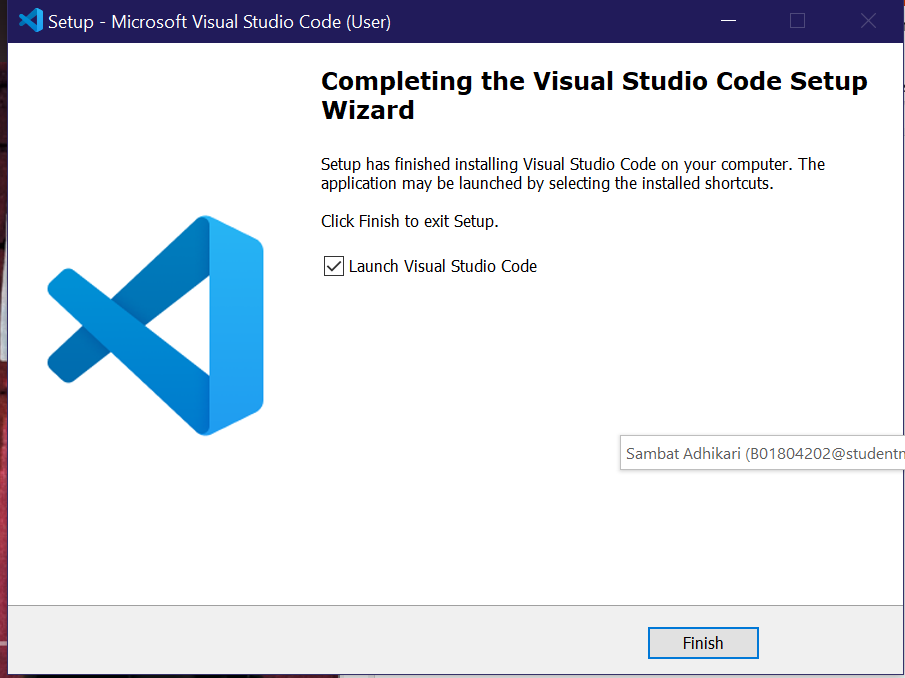


Figure 6: Visual Studio Code Setup Completed

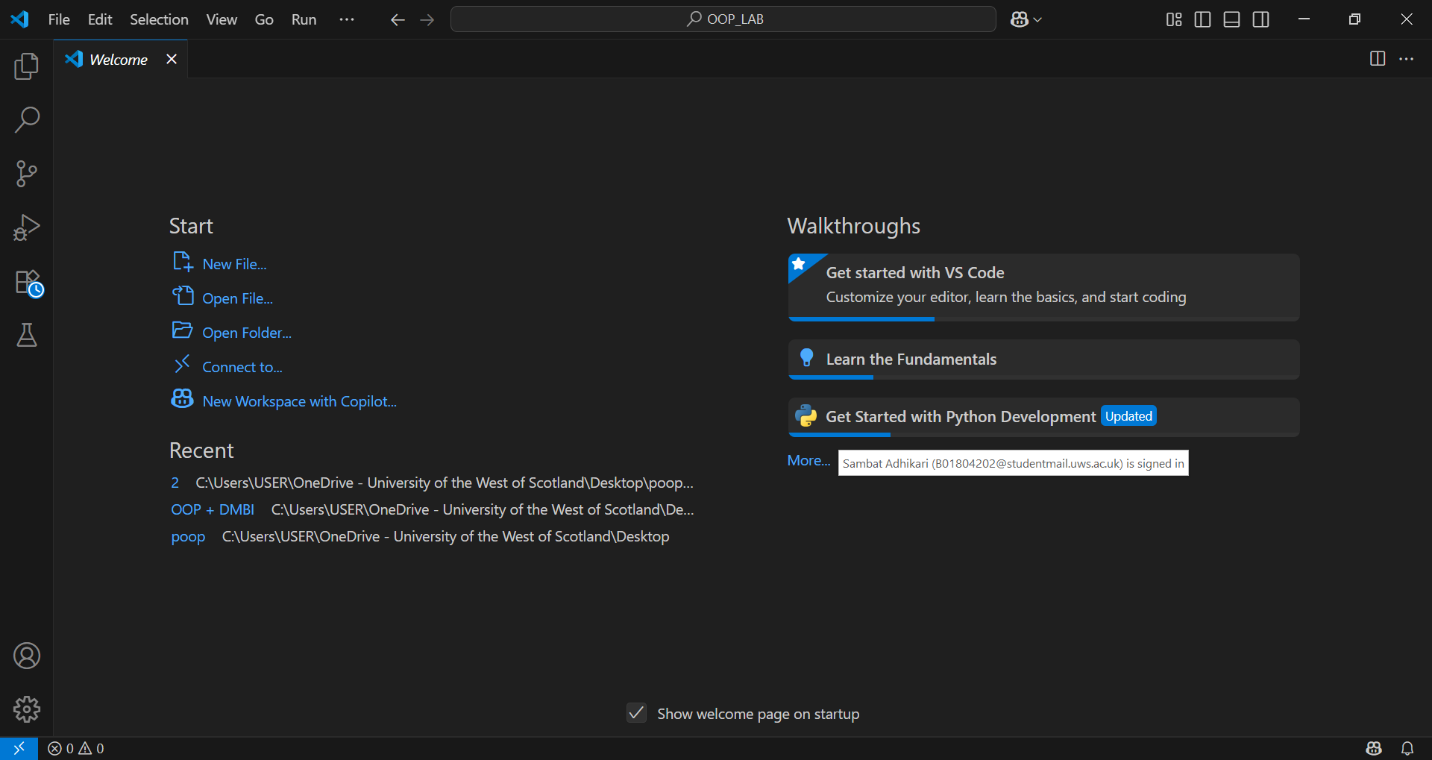


Figure 7: Visual Studio Code

After installation of Visual Studio Code, we also install Python extension in VS Code.

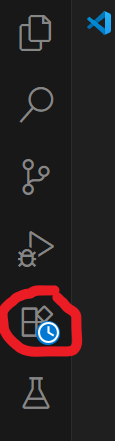


Figure 8: Side Bar containing extension

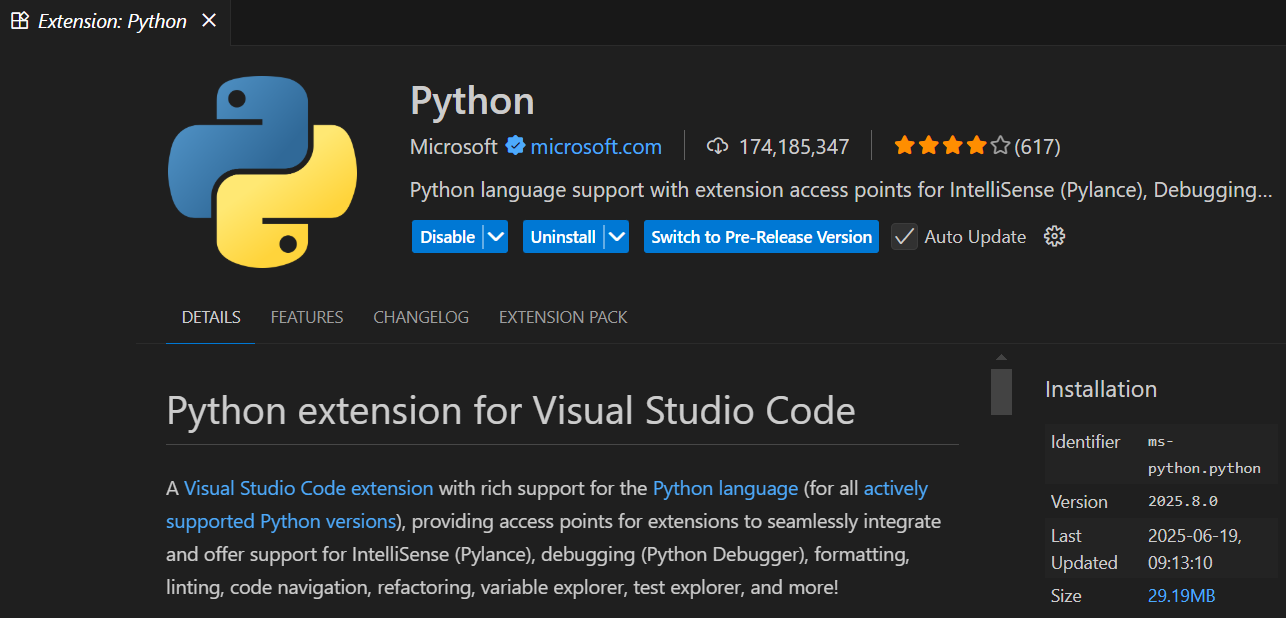


Figure 9: Installation of Python extension

### 2.3 Writing our first Python Program

After the setup of VS Code, we write our first python program “Hello World” to ensure everything is working properly.

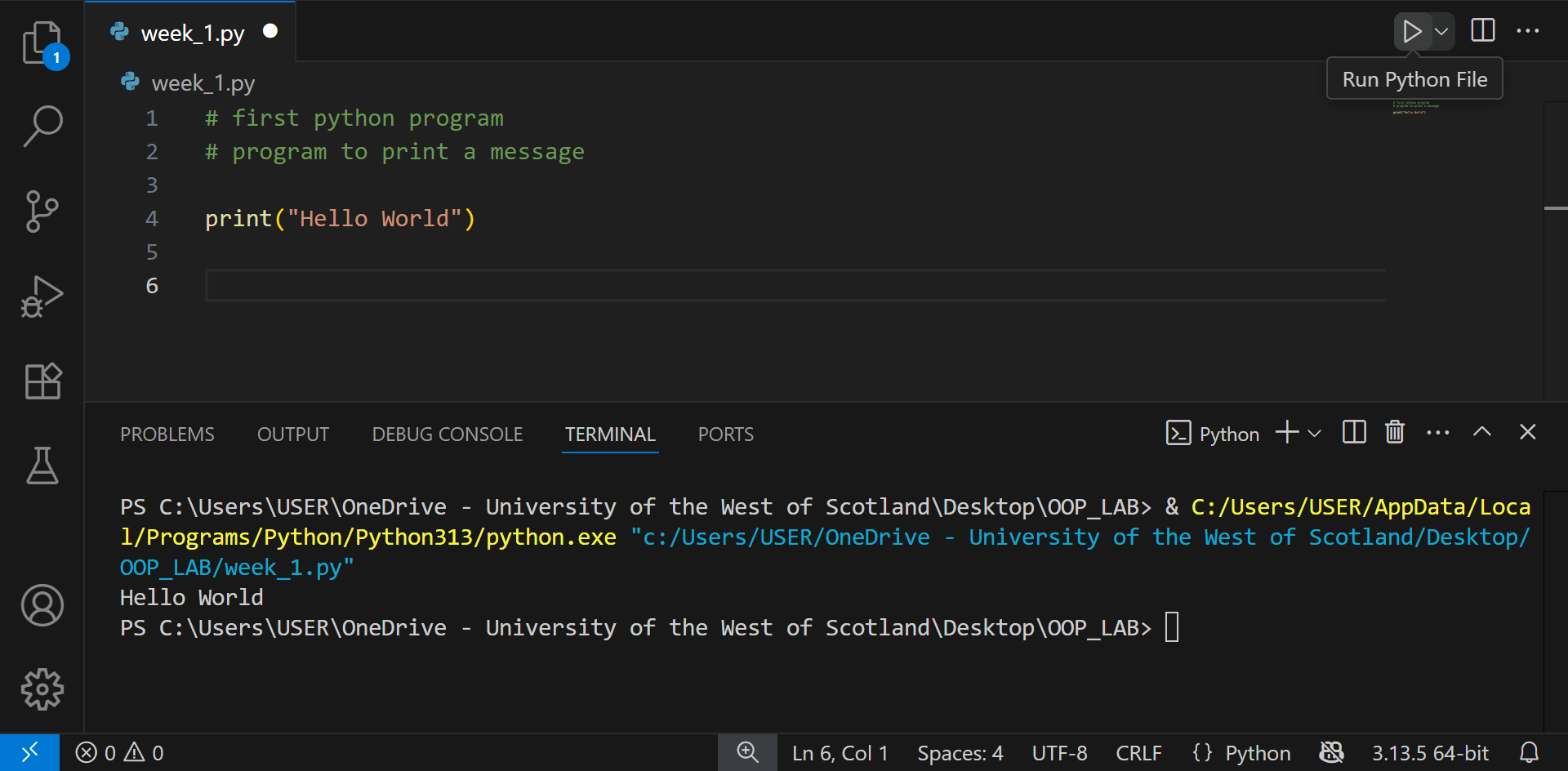


Figure 10: "Hello World" Program

### 2.3 Outcome

By the end of Week 1, all group members had:

1. Successfully installed Python 3 and Visual Studio Code
2. Installed the Python extension within VS Code
3. Executed their first Python program

This ensures a consistent working environment across all group members for future lab exercises.

# 3.Week 2 – Comparisons, Conditionals, Lists, Loops and Input

In Week 2, we dived deeper into the fundamental concepts of Python programming to lay the foundation later required for complex needs. In this time period we learned about concepts that allows us to write basic programs that make decisions, repeat instructions, handle multiple values and interact with user.

### 3.2 Comparison and Logical Operators

**3.2.1. Comparison Operator**

We use comparison operators to compare two or more values. The result of a comparison operator is a Boolean value (either True or False).

The comparison operators used in Python are:

|  |  |  |  |
| --- | --- | --- | --- |
| Operator | Name | Example | Result |
| > | Greater than | 5 > 4 | True |
| 4 > 5 | False |
| < | Less than | 4 < 5 | True |
| 5 < 4 | False |
| == | Equal to | 5 == 5 | True |
| 4 == 5 | False |
| != | Not Equal to | 4 != 5 | True |
| 4 != 4 | False |
| >= | Greater than or equal to | 5 >= 4 | True |
| 4 >= 5 | False |
| <= | Less than or equal to | 4 <= 5 | True |
| 5 <= 4 | False |

**3.2.2. Logical Operator**

We use logical operators to evaluate combine multiple conditions to return a single value. The result of logical operator is also a Boolean value.

The logical operators used in Python are:

|  |  |  |  |
| --- | --- | --- | --- |
| Operator | Description | Example | Result |
| and | Returns TRUE if all  conditions are TRUE | 5 > 4 and 5 > 2 | True |
| 5 > 4 and 5 > 6 | False |
| or | Returns TRUE if at least  one condition is TRUE | 4 < 5 and 4 < 6 | True |
| 4 < 5 and 4 < 3 | False |
| not | Reverses the result  TRUE if FALSE,  FALSE if TRUE | not (4 > 5) | True |
| not (4 < 5) | False |

### 3.2 Conditional Statement

### 3.2 List

### 3.2 Loop

### 3.2 Input

**1. Introduction**

This report documents the completion of Week 2's practical exercises focusing on Python programming fundamentals. The lab covered:

* Conditional statements and logical operators
* List creation and manipulation
* Loop structures (while and for loops)
* User input handling

All exercises were implemented in a single Python file (lab\_week\_2.py) with extensive testing and verification.

**2. Implementation Details**

**2.1 Conditionals**

**Basic Conditional Statements**

# Age classification

age = 12

age\_group = "child"

if age > 18:

age\_group = "adult"

print(f"The age group is {age\_group}")

# Wind speed check

wind\_speed = 8

if wind\_speed < 10:

print("It is a calm day")

else:

print("It is a windy day")

**Grading System**

Implemented a multi-condition grading system:

grade = 76

if grade < 50:

print("You failed")

elif grade < 60:

print("You passed")

elif grade < 70:

print("You got a good pass")

else:

print("You got an excellent pass")

**2.2 Lists**

**List Operations**

city\_list = ["Glasgow", "London", "Edinburgh"]

print(city\_list[2]) # Access third item

print(city\_list[-2:]) # Slice last two items

# Modification

city\_list.append("Manchester")

city\_list[1] = "Birmingham"

**Color List Task**

colors = ["red","yellow","green"]

colors[0] = "blue"

if "red" in colors:

print("Red is in the list")

selected\_colors = colors[1:3]

**2.3 Loops**

**While Loop**

i = 10

while i < 60:

print(i)

i += 5

**For Loop with Control**

# Break example

for i in range(5):

if i == 3:

break

print(i)

# Continue example

for i in range(5):

if i == 2:

continue

print(i)

**Summary Tasks**

# Even numbers

numbers = list(range(1,11))

for num in numbers:

print(f"{num} is {'even' if num%2==0 else 'odd'}")

# Sum of squares

sum\_sq = sum(i\*\*2 for i in range(1,6))

print(f"Sum of squares: {sum\_sq}")

# Countdown

count = 10

while count > 0:

print(count)

count -= 1

print("Liftoff!")

**2.4 User Input**

**Age Classifier**

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

if age < 18:

print("Minor")

elif age <= 65:

print("Adult")

else:

print("Senior")

**Enhanced Temperature Converter**

temp = float(input("Enter temperature: "))

unit = input("Enter unit (C/F/K): ").upper()

if unit == "C":

f = (temp \* 9/5) + 32

k = temp + 273.15

print(f"{temp}°C = {f}°F, {k}K")

elif unit == "F":

c = (temp - 32) \* 5/9

print(f"{temp}°F = {c}°C, {c+273.15}K")

else:

c = temp - 273.15

print(f"{temp}K = {c}°C, {(c\*9/5)+32}°F")

**3. Testing & Verification**

All exercises were rigorously tested with:

1. Boundary cases (e.g., grade = 49, 50, 69, 70)
2. Invalid input handling (non-numeric age input)
3. Edge cases (empty lists, zero values)

**4. Challenges & Solutions**

1. **List Index Errors**
   * Occurred when accessing out-of-range indices
   * Solution: Added length checks before access
2. **Input Validation**
   * Non-numeric input would crash programs
   * Implemented try-except blocks for robustness
3. **Loop Control**
   * Initially created infinite loops
   * Added debug print statements to monitor variables

**5. Conclusion**

The lab successfully covered fundamental Python concepts through practical implementation. Key achievements:

* Mastered conditional logic and list operations
* Implemented robust user input handling
* Gained proficiency in loop control structures

**Areas for Improvement:**

* Implementing functions for code organization
* Adding graphical interfaces
* Exception handling for edge cases

**6. Appendix**

