**Day 1**

**Inheritance**

Inheritance is a mechanism that allows a new class (subclass or derived class) to inherit the properties and behaviours of an existing class (base class or parent class). This promotes code reusability and the creation of a hierarchy of classes.

**More on Inheritance**

Inheritance helps in creating a hierarchy of classes, allowing for a more organized and efficient code structure. Multiple inheritance in Python occurs when a class is derived from more than one base classes. This means that the derived class inherits attributes and methods from multiple parent classes.

**Explaining Inheritance**

Inheritance is a mechanism that allows a new class (subclass or derived class) to inherit the properties and behaviours of an existing class (base class or parent class).  
Multiple inheritance in Python occurs when a class is derived from more than one base classes. This means that the derived class inherits attributes and methods from multiple parent classes. This promotes code reusability and the creation of a hierarchy of classes.

**Private Variables**

“Private” instance variables that cannot be accessed except from inside an object don’t exist in Python. However, there is a convention that is followed by most Python code: a name prefixed with an underscore (e.g., \_spam) should be treated as a non-public part of the API (whether it is a function, a method or a data member). It should be considered an implementation detail and subject to change without notice.

**More on Private Variables**

 In Python, variables can have different levels of visibility or access control. In protected variables, Variables and methods are accessible within the class and its subclasses (using a single leading underscore as a convention or setters and getters). In private variables, variables and methods are accessible only within the class (using a double leading underscore).

**Private Variables cont.**

In Python, we can use a single leading underscore (\_) as a convention to indicate that a variable is intended to be private. However, it's important to note that this is just a convention, and it does not enforce strict privacy like some other programming languages. Python does not have true "private" variables in the sense that they are inaccessible from outside the class.

**Day 2**

Iterators and generators are features in Python that allow for more efficient and memory-friendly iteration over sequences of data. They are particularly useful when working with large datasets or when you want to generate values on-the-fly.

**Generator Expressions**

Generator expressions are a concise way to create generators in Python. They have a syntax like list comprehensions but use parentheses () instead of square brackets []. Like generators created using the yield keyword in a function, generator expressions generate values on-the-fly and are more memory-efficient than creating a list.

**Explaining Generators**

Some simple generators can be coded succinctly as expressions using a syntax similar to list comprehensions but with parentheses instead of square brackets. These expressions are designed for situations where the generator is used right away by an enclosing function. Generator expressions are more compact but less versatile than full generator definitions and tend to be more memory friendly than equivalent list comprehensions.

**Error Output Redirection and Program Termination**

The sys module also has attributes for stdin, stdout, and stderr. The latter is useful for emitting warnings and error messages to make them visible even when stdout has been redirected. The sys module in Python provides access to the interpreter's variables and functions that interact with the Python runtime environment. It includes attributes for stdin, stdout, and stderr for handling input and output streams.

**How to Match Any Pattern of Text**

Matching any pattern of text in Python is typically done using regular expressions, which are supported by the re module. Regular expressions (regex or regexp) are powerful tools for pattern matching and text manipulation.

**Regular Expressions**

Regular expressions (regex or regexp) are sequences of characters that define a search pattern. They are used for string manipulation and searching within text. Regular expressions provide a concise and flexible means for matching strings of text, character patterns, or even complex patterns within larger bodies of text.

**Day 3**

**Dates and Times, Data Compression, Performance Measurement, and Quality Control**

In Python, the datetime module provides classes for working with dates and times. The datetime class allows us to represent and manipulate dates and times. Python provides modules for various compression algorithms, including gzip, zipfile, and bz2. These modules allow us to compress and decompress data. The time module can be used for basic performance measurement in Python. The doctest module provides a tool for scanning a module and validating tests embedded in a program’s docstrings. Test construction is as simple as cutting-and-pasting a typical call along with its results into the docstring.

For basic performance measurement in Python, we can use the time module. The timeit module is also useful for more accurate measurement and benchmarking.

**Performance Measurement with the time it module**

The timeit module provides a simple way to measure the execution time of small bits of Python code. It has both a simple interface for string-based timing and a more flexible interface for specifying setup code and the number of loops.

**Output Formatting**

The reprlib module in Python provides a version of repr() that is customized for abbreviated displays of large or deeply nested containers. This module is particularly useful when dealing with large data structures, preventing the console or output from being flooded with an excessive amount of information.

**Output Formatting cont.**

The pprint (pretty-print) module provides a way to pretty-print arbitrary Python data structures in a more readable and visually appealing format. The textwrap module is commonly used to format text for display in a specific width or to create neatly formatted paragraphs. The locale module provides a way to handle and query locale-specific information, such as formatting of numbers, dates, and currency symbols. The reprlib module provides a version of repr() that is customized for abbreviated displays of large or deeply nested containers.

**Day 4**

**Logging**

Logging in Python refers to the process of recording log messages from a program for the purpose of monitoring, debugging, and understanding its behaviour. The logging module in Python provides a flexible and powerful logging framework that allows developers to incorporate logging into their applications.

**Virtual Environments**

A virtual environment in Python is a self-contained directory that contains its own Python interpreter and allows us to install and manage Python packages for a specific project without interfering with the system-wide Python installation. Virtual environments are essential for managing dependencies and ensuring that our project runs with the correct versions of libraries.

**Virtual Environments cont.**

Virtual environments are invaluable tools for web developers, providing several benefits that make them crucial in the development workflow. We use them because of dependency isolation, version consistency, ease of collaboration, etc.

**Managing packages with pip**

pip is the package installer for Python, and it is a powerful tool for managing Python packages. It simplifies the process of installing, upgrading, and removing Python packages, making it an essential tool for Python developers.

**Floating point Arithmetic**

Floating-point arithmetic in computers is a way of representing and performing mathematical operations on real numbers with a fractional component. While floating-point arithmetic is a powerful tool, it comes with certain limitations and considerations due to the finite precision of computer hardware. Understanding these limitations is important for avoiding unexpected behaviour in numerical computations.

**Applying New Knowledge**

Python can be used for handling and processing data for our current project. Libraries like Pandas and NumPy are powerful for data manipulation and analysis.