Day 1

Functions of Python:

- General-purpose language supporting multiple programming paradigms.
- Emphasizes code readability and ease of maintenance.
- Offers a vast standard library for common programming tasks and functionalities.

Interactive Mode of the Python Interpreter:

- Real-time execution of Python statements.
- Useful for quick testing, experimentation, and learning.
- Accessed by running the Python interpreter without providing a script file.

Comments in Python:

- Single-line comments start with the '#' symbol.
- Multi-line comments use triple quotes (''' or """).
- Essential for explaining code functionality and improving readability.

Applications Used in Python:

1. Web Development:
   - Django and Flask are popular web development frameworks.
2. Data Analysis:
   - Used for data manipulation, analysis, and visualization.
3. Scientific Computing:
   - Supports complex mathematical operations and scientific computations.
4. Machine Learning and AI:
   - Widely used for building machine learning models and AI applications.
5. Automation and Scripting:
   - Simplifies complex processes through automation and scripting.
6. Desktop GUI Applications:

- Used for developing desktop applications with user-friendly interfaces.

7. Game Development:

- Enables the creation of 2D and simple 3D games.

Day 2

Two Different Data Types in Python:

1. *Numeric Data Type:*

- Integers, representing whole numbers.
- Floats, representing numbers with decimal points.
- Complex numbers, in the form a + bj.

2. *Non-Numeric Data Type:*

- Strings, representing sequences of characters.
- Lists, ordered and mutable collections of items.
- Tuples, ordered and immutable collections of items.
- Sets, unordered collections of unique items.
- Dictionaries, unordered collections of key-value pairs.

Conversion Between Data Types:

- Use built-in functions like `int()`, `float()`, `str()`, `list()`, `tuple()`, `set()`, or `dict()` for conversion between data types.

Naming Conventions in Python:

- Variables: Lowercase with words separated by underscores (e.g., `my_variable`).
- Functions: Lowercase with words separated by underscores (e.g., `my_function`).
- Classes: CamelCase naming convention, starting with an uppercase letter for each word (e.g., `MyClass`).
- Constants: Uppercase with words separated by underscores (e.g., `MY_CONSTANT`).

Casting:

- *Casting* is the conversion of one data type to another.
- Use built-in functions like `int()`, `float()`, and `str()` for casting in Python.

Example:

```python
 Casting to integer
x = int(2.8)  # x will be 2

Casting to float
y = float("3.5")  # y will be 3.5

Casting to string
z = str(5)  # z will be the string '5'
```

Day 3

Default Argument Values:

- Provides default values for function parameters.
- Used when corresponding arguments are not provided during function calls.

Keyword Arguments:

- Allows passing arguments to functions using parameter names.
- Order of arguments does not matter if parameter names are specified.

Arbitrary Argument Lists:

- Supports the use of arbitrary argument lists for functions.

- Enables functions to accept an arbitrary number of arguments, accessed as a tuple inside the function.

Unpacking Argument Lists:

- Supports unpacking argument lists using the `*` operator.
- Allows passing elements of a list or tuple as arguments to a function directly.

Lambda Expressions:

- Anonymous functions in Python with any number of arguments but only one expression.
- Used for short, simple functions, and where a named function is not required.

Conventions for Documentation Strings:

- *Docstrings* describe the purpose and usage of functions, modules, classes, and methods in Python.
- Writing conventions include:
  - Describing function purpose and return values.
  - Providing information about parameters and their types.
  - Mentioning any exceptions the function might raise.
  - Including relevant usage examples.

Day 4

Patterns and Flags:

- Regular expressions (RegEx) are patterns for matching character combinations in strings.
- Flags modify RegEx behavior, allowing options like case-insensitive and multiline matching.

Usage:

- RegEx is used for pattern matching, string parsing, and text manipulation.

- Provides a flexible way to search, replace, and extract information from strings.

Flags:

- Modify RegEx behavior, like case-insensitive and multiline matching.
- Include options such as `re.IGNORECASE`, `re.MULTILINE`, and `re.DOTALL`.

Methods of RegExp and String:

- `re` module in Python used for working with RegEx.
- Methods include `match()`, `search()`, `findall()`, and `sub()`.

Character Classes:

- Represent a set of characters that can be matched.
- Include predefined classes like `\d`, `\w`, and `\s`.

Word Boundary \b:

- Matches the empty string at the beginning or end of a word.
- Useful for finding whole words in text.

Inverse Classes:

- Denoted by using a capital letter, e.g., `\W`, `\D`, and `\S`.
- Match any non-word, non-digit, and non-whitespace characters, respectively.

Spaces are Regular Characters:

- Spaces are treated as regular characters in RegEx.
- To match spaces, use the space character itself.

A Dot is Any Character:

- The dot `.` in RegEx matches any character except a newline.
- Useful for matching any character in a string.

The Dotall "s" Flag:

- The `re.DOTALL` flag allows the dot `.` to match any character, including a newline.

Escaping, Special Characters:

- Special characters in RegEx have specific meanings and functions.
- Use the backslash `\` to escape special characters and match them literally.