**Data Types**

Python supports various data types, including integers, booleans, floating-point numbers, complex numbers, and strings.

Literals are constant values and an alternative to using variables.

Examples of literals include strings, escape sequences (e.g., "\t" for a tab), and numeric literals.

Literal values, such as numbers, can be used directly in expressions.

Integers:

Python provides four primary mathematical operators: addition (+), subtraction (-), multiplication (\*), and division (/).

The plus operator (+) can be used for adding numbers and for string concatenation.

User input can be collected using the input() function.

Floating-Point Numbers:

Floating-point numbers are used to represent real numbers, including decimal fractions.

They are essential for tasks involving numerical accuracy and precision.

Python supports standard arithmetic operations for floating-point numbers.\

Booleans:

Booleans represent binary values: True and False.

True is equivalent to the integer 1, and False is equivalent to 0.

Python uses logical operators (and, or, not) to perform Boolean operations.

Logical operators are used to evaluate conditions and return True or False.

Unpacking Argument Lists:

Unpacking argument lists in Python allows you to pass multiple arguments from an iterable (e.g., a list or tuple) to a function as separate arguments.

It is denoted by the asterisk (\*) symbol in function definitions and calls.

This feature enables flexibility when working with variable-length argument lists.

**Strings:**

Strings are sequences of Unicode characters represented by the str data type in Python.

The str() function is used to create strings, and it returns an empty string when no argument is supplied.

You can use str() to convert other data types to strings.

The += operator can be used to concatenate strings or add values to an existing string.

Lambda Expressions:

Lambda expressions, also known as lambda functions, allow you to create small, anonymous functions using the lambda keyword.

They are typically used for simple, one-line functions.

Lambda functions can be used wherever function objects are required, such as in sorting functions.

They are semantically equivalent to normal function definitions.

Conventions about Documentation Strings:

Documentation strings (docstrings) are used to provide documentation for Python code.

The first line of a docstring should be a concise summary of the object's purpose.

If there are more lines in the docstring, the second line should be blank, visually separating the summary.

The Python parser does not strip indentation from docstrings, so tools may need to handle indentation.

Function annotations are optional metadata that provide information about the types used in user-defined functions.

Coding Style (PEP 8):

PEP 8 is the Python Enhancement Proposal that defines the style guide for Python code.

It includes recommendations for code formatting, indentation, line length, blank lines, commenting, and naming conventions.

Using 4-space indentation is encouraged, while tabs are best avoided.

Code should be wrapped to avoid lines exceeding 79 characters for readability.

Consistent naming conventions, like using UpperCamelCase for classes and lowercase\_with\_underscores for functions, are recommended.