**Core Python:-**

**SCALAR TYPES:**

Scalar types in Python are the basic data types that can be used to store single values. They include:

Integers - whole numbers, such as 1, 2, 3, etc.

Floats - numbers with decimal points, such as 1.2, 3.14, etc.

Strings - sequences of characters, such as "Hello, world!"

Booleans - values that can be either true or false

Scalar types can be used to store a wide variety of data, such as numbers, text, and logical values. They are the most basic data types in Python, and they are used in many different ways in Python programs.

**OPERATORS:**

Operators are symbols that are used to perform operations on scalar types. For example, the addition operator (+) is used to add two numbers together.

\* Arithmetic operators: +, -, \*, /, %

\* Logical operators: &&, ||, !

\* Comparison operators: <, >, <=, >=, ==, !=

**CONTROL FLOW:**

Control flow refers to the way that a program's execution is controlled. It is used to determine which statements are executed and in what order.

For example, the if statement is used to execute a statement only if a certain condition is met.

\* If statements: execute a statement only if a certain condition is met

\* While loops: execute a block of statements repeatedly while a certain condition is met

\* For loops: execute a block of statements a specified number of times

**STRINGS:**

Strings are sequences of characters. They are enclosed in single quotes (') or double quotes (").

**COLLECTORS:**

Collectors are data structures that can store a collection of values.

They are used to store and manipulate data in a more efficient way than using individual variables. There are many different types of collectors in Python, such as lists, dictionaries, and sets.

Iteration is the process of looping over a collection of values. It is used to process each value in the collection one at a time.

There are many different ways to iterate over a collection in Python, such as using for loops, while loops, and generators.

**MODULARITY:**

Modularity is a software design principle that breaks down a large program into smaller, self-contained modules. This makes the program easier to understand, maintain, and test. Modularity is also a key concept in object-oriented programming, which is a popular programming paradigm in Python.

There are many benefits to using modularity in Python. Some of these benefits include:

Increased readability and maintainability: Modularity makes code easier to understand and maintain because each module is focused on a specific task. This makes it easier to find the code that you need to change or fix.

Improved testability: Modularity makes code easier to test because each module can be tested independently. This helps to ensure that the code is working correctly and that changes to one module do not break other modules.

Reduced coupling: Modularity reduces coupling between modules, which means that modules are less dependent on each other. This makes the code more flexible and easier to change.

Increased reusability: Modularity makes code more reusable because modules can be easily reused in other projects. This saves time and effort, and it also helps to ensure that the code is consistent across projects.

There are many different ways to implement modularity in Python. Some of the most common methods include:

Using modules: Modules are the most common way to implement modularity in Python. A module is a self-contained file that contains a set of related functions, classes, and variables. Modules can be imported into other modules, which allows you to reuse code and avoid duplicating code.

Using packages: Packages are a way to organize modules into a hierarchy. Packages can be used to group related modules together and to make it easier to find and import modules.

Using functions: Functions are a way to encapsulate a block of code into a single unit. Functions can be used to break down large tasks into smaller, more manageable pieces.

Using classes: Classes are a way to create reusable objects. Classes can be used to encapsulate data and behavior, which makes it easier to create complex applications.

Modularity is an important concept in Python programming. By using modularity, you can write code that is easier to understand, maintain, and test. You can also reuse code and avoid duplicating code, which saves time and effort