Condition and Branching:

Conditions:
Conditions are specific criteria that must be met for a set of instructions to be executed.For example, in the real world, people dance when there is music.The presence of music is a condition and when it is true, people will dance. Similarly, in Python programming, we use conditions to determine which set of instructions to execute.

Branching:
Branching is the process of using conditions to determine which set of instructions to execute. If a certain condition is met, the python program will execute one set of instructions, and if the condition is not met, it will execute a different set of instructions.
1.”if-else” statements, where a specific block of code is executed if a condition is met and another block of code is executed if the condition is not met.

2.Multiple “if” statements, where multiple conditions are checked, and the corresponding block is executed for the first true condition.

3.”if-elif” statements, where multiple conditions are checked in sequence, and the first true condition’s corresponding block of code is executed

Python Loops:
Looping means repeating something over and over until a particular condition is satisfied.
There are two types of Loops: For loop and while loop to handle looping requirements.

While Loop in Python: A while loop is used to execute a block of statements repeatedly until a given condition is satisfied. When the condition become false, the line immediately after the loop in the program is executed.
Syntax: while expression:

```
            statement(s)
```

For Loop in Python: A for loops are used for sequential traversal. It is used for iterating over a sequence(that is either a list, a tuple, a dictionary, a set, or a string).
Syntax: for iterator_var in sequence:

```
          statement(s)
```

Python Functions:
A python function is a block of statements that return the specific task. Increased code readability and reusability are the benefits of using functions.
There are mainly two types of functions:
1.Built-in library function and 2. User-defined function

Python Class:
A class is a user-defined blueprint or prototype from which objects are created.
Python is an object oriented programming language and class is like an object constructor or a blueprint for creating objects.

Encapsulation:
It is one of the fundamental concepts in OOP. It describes the idea of wrapping data and the methods that work on data within one unit. A class is an example of encapsulation as it encapsulates all the data that is member functions, variables etc.
The members are Private, Protected and Public
Protected are those members of the class that cannot be accessed outside the class but can be accessed from within the class and its subclasses.The convention by prefixing the name of the member by a single underscore"_". Although the protected variable can be accessed out of the class as well as in the derived class.
Private members are similar to protected members, the difference is that the class members declared private should neither be accessed outside the class nor by any base class.In python, there is no existence of private instance variables that cannot be accessed except inside a class. A private member prefix the member name with double underscore"__".

Inheritance:
It is the mechanism that allows you to create a hierarchy of classes that share a set of properties and methods by deriving a class from another class. Inheritance is the capability of one class to derive or inherit the properties from another class.
The benefits of Inheritance are:
-It represents real-world relationship well.
-It provides the reusability of a code.So, we don't have to write the same code again and again. Also, it allows us to add more features to a class without modifying it.
Python inheritance Syntax:

```
class BaseClass:
        {Body}
Class DerivedClass(BaseClass):
        {Body}
```

Abstraction:
Abstraction is used to hide the internal functionality of the function from the users. The users only interact with the basic implementation of the function but inner working is hidden. Users are familiar with 'what function does' but they don't know 'how it does'. A class that consists of one or more abstract methods is called the abstract class. Abstract class can be inherited by the subclass and abstract method gets its definition in the subclass. Abstraction classes are meant to be the blueprint of the other class.
Syntax of Abstraction:

```
From abc import ABC
Class ClassName(ABC):
```