FUNCTIONAL PROGRAMMING

Functional programming is a programming paradigm in which we try to bind everything in pure mathematical functions style. It is a declarative type of programming style. Its main focus is on “what to solve” in contrast to an imperative style where the main focus is “how to solve“. It uses expressions instead of statements. An expression is evaluated to produce a value whereas a statement is executed to assign variables.

Concepts of Functional Programming

Any Functional programming language is expected to follow these concepts.
Pure Functions: These functions have two main properties. First, they always produce the same output for the same arguments irrespective of anything else. Secondly, they have no side-effects i.e. they do modify any argument or global variables or output something.
Recursion: There are no “for” or “while” loop in functional languages. Iteration in functional languages is implemented through recursion.
Functions are First-Class and can be Higher-Order: First-class functions are treated as first-class variable. The first-class variables can be passed to functions as a parameter, can be returned from functions or stored in data structures.
Variables are Immutable: In functional programming, we can’t modify a variable after it’s been initialized. We can create new variables – but we can’t modify existing variables.

Functional Programming in Python

Python too supports Functional Programming paradigms without the support of any special features or libraries.

Pure Functions

As Discussed above, pure functions have two properties.
It always produces the same output for the same arguments. For example, 3+7 will always be 10 no matter what.
It does not change or modifies the input variable.
The second property is also known as immutability. The only result of the Pure Function is the value it returns. They are deterministic. Programs done using functional programming are easy to debug because pure functions have no side effects or hidden I/O. Pure functions also make it easier to write parallel/concurrent applications. When the code is written in this style, a smart compiler can do many things – it can parallelize the instructions, wait to evaluate results when needing them, and memorize the results since the results never change as long as the input doesn’t change.

Recursion

During functional programming, there is no concept of loop or loop, instead recursion is used. Recursion is a process in which a function calls itself directly or indirectly. In the recursive program, the solution to the base case is provided and the solution to the bigger problem is expressed in terms of smaller problems. A question may arise what is base case? The base case can be considered as a condition that tells the compiler or interpreter to exit from the function.

Immutability

Immutability is a functional programming paradigm can be used for debugging as it will throw an error where the variable is being changed not where the value is changed. Python too supports some immutable data types like string, tuple, numeric, etc.