# **Design of “EasyPython” Language**

## Basic Idea:

Our idea is to implement a new high-level language that could be considered as an intermediate between Python and C languages. Our new language will give us easy to code ability like Python (e.g. no main function required for code to execute) with increase speed advantage like C because of its static nature. Coding in it can also give a learning advantage to programmers with C/C++ background to understand python environment before actually moving to it.

# **Main Features and Properties:**

* It will be a sequential programming language built on Turing machine concept and will exhibit Turing Complete property. So, we will have ability to store our variables and also conditional and looping blocks.
* It will be a case sensitive language.
* Can be used for general purpose tasks.
* The programming paradigm which it implements will be procedural and will be of imperative nature.
* Static type checking will be used in it to for better performance.
* It will be a typed language.
* Our language will be cross platform i.e. Linux and Windows (specifically windows 10).
* Our language will basically be interpreted and the interpreter actually will have the structure of a classic compiler i.e. when we will run our code, our raw source code is scanned for tokens. These tokens will be parsed into a tree representing the logical structure of the program, which is finally transformed into byte code. Lastly, this byte code is executed by the machine.
* Our language intends to do better in following ways:
  + - To make C language easy to code like Python
    - To improve performance of Python language
    - Reduce run-time errors by removing Duck-Typing like in Python
* It will have a multi-pass compiler.
* Functionalities like print () to print output to screen, import statement to import files.
* Our language will support classes data types for object-oriented programming.

## **Semantics Overview:**

* Our variables will be mutable since we will by using static type checking.
* Types will be defined before compilation i.e. no dynamic type checking will occur. The possible types will only be from available data types which are integer, float, bool, char, string, tuples, list, struct, class.
* We will handle and evaluate literal and variables like in C language.
* The function block of our programming structure will be reusable just like conventional languages and the way to reuse them will be by just calling their names with required arguments if present.
* The following arithmetic operators will be implemented: +, -, /, %, \*\*, \*.
* The assignment operator will be “=” and conditional assignment operators will be “<=, >=, !=” while syntactic sugar used will be “+=, -=, \*=, /=”.
* The scoping rules will be:
  + - If a variable is defined in a block, it scope will include only that block.
    - If a variable is defined globally, its scope will include all the program.
    - When a name is used in a code block, it is resolved using the nearest enclosing scope.
* Our language now will not contain pointers so we will implement value semantics. However, we will give ability to mutate already defined variables.

