# Variables Declaration

- **What is variable?**
  - It is a reference to a value stored in a computer's memory. It can be stored in a variety of categories or data types, like **numbers (int, float, etc.), boolean values (true or false), and sequences (strings, lists, etc.).**

- **Fundamentals of Python Programming**
  - **Syntax** - Python has a clear and readable syntax, which makes it easy for beginners to learn and write code. Indentation (whitespace) is used to define blocks of code, eliminating the need for braces {} as in some other languages.
  - **Variables and Data Types** - Variables are used to store data. Python is dynamically typed, meaning you don't need to declare the data type explicitly. **Common data types** include **integers (int)**, **floating-point numbers (float), strings (str), lists (list), tuples (tuple), dictionaries (dict), and more.**
  - **Control Flow** - Python supports conditional statements (if, elif, else) for decision-making. Looping constructs include for loops and while loops.
  - **Functions** - Functions in Python are defined using the def keyword. Functions can have parameters and return values.

- **Rules in Declaring a variable in Python**
  - A variable name must start with a letter or the underscore character
  - A variable name cannot start with a number
  - A variable name can only contain alpha-numeric characters and underscores (A-z, 0-9, and _ )
  - Variable names are case-sensitive (age, Age and AGE are three different variables)
  - A variable name cannot be any of the Python Keywords.
  - The rest of the variable name can contain letters, numbers, and underscores.
  - Variable names are case-sensitive (**myVar** is different from **myvar**).
  - Avoid using reserved words as variable names (keywords like **if, else, while**, etc.).

- **Keywords in Python**
  - Keywords are reserved words that have special meanings in Python and cannot be used as variable names
  - if, else, while, for, break, continue, def, class, True, False, None, and, or, not, import, from, as, return, global, pass, try, except, finally, with, yield, lambda, is, in, assert, del, elif.

- **Rules for local and global variables in Python**
  - In **Local**, a variable is assigned a value anywhere within the function's body, it's assumed to be a local unless explicitly declared as global.
    * Defined inside a function.
    * Limited to the scope of that function.
    * Not accessible outside the function.

  - In **Global,** was required for all global references, you'd be using global all the time. You'd have to declare as global every reference to a built-in function or to a component of an imported module. This clutter would defeat the usefulness of the global declaration for identifying side-effects.
    * Defined outside of any function or block.
    * Accessible throughout the entire program.

- **Operators**
  - are symbols used to carry out specific functions/computations.

| Operator | Description |
|---|---|
| ** | Exponentiation (raise to the power) |
| ~ + - | Ccomplement, unary plus and minus (method names for the last two are +@ and -@) |
| * / % // | Multiply, divide, modulo and floor division |
| + - | Addition and subtraction |
| >> << | Right and left bitwise shift |
| & | Bitwise 'AND' |
| ^ \| | Bitwise exclusive `OR' and regular `OR' |
| <= < > >= | Comparison operators |
| <> == != | Equality operators |
| = %= /= //= -= += *= **= | Assignment operators |
| is is not | Identity operators |
| in not in | Membership operators |
| not or and | Logical operators |