**Data Types** : is a classification that specifies which type of value a variable has and what type of mathematical, relational or logical operations can be applied to it without causing an error. (Source from outside course).

**Variabels** : Names that wr give to cerrtain values in our programs (Number, stting, float, or other). Variables are important in programming because they let you perform operations on data that may change.

Restriction in Naming a Variabel :

1. you shouldn't use as variable names any of the key words or functions that Python reserves for its own, like print.
2. Don’t use spaces
3. Must start with a letter or an underscore (\_)
4. Must be made up of letters, numbers and underscores.

contoh :

rio\_bastian = valid variabel name

rio\_bastian09 = valid variabel name

09\_rio\_bastian = invalid variabel name

rio\_&\_bastian = invalid variabel, cause uses the special character

**Implicit Conversion** = The interpreter automatically converts one data type into another, without having to explicitly tell it to do so (int and float).

**Explicit Conversion** = Manually convert from one data type to another by calling the relevant function for the data type we want to convert to (str(8)).

* expression - a combination of numbers, symbols, or other values that produce a result when evaluated
* data types - classes of data (e.g., string, int, float, Boolean, etc.), which include the properties and behaviors of instances of the data type (variables)
* variable - an instance of a data type class, represented by a unique name within the code, that stores changeable values of the specific data type
* implicit conversion - when the Python interpreter automatically converts one data type to another
* explicit conversion - when code is written to manually convert one data type to another using a data type conversion function:
  + str() - converts a value (often numeric) to a string data type
  + int() - converts a value (usually a float) to an integer data type
  + float() - converts a value (usually an integer) to a float data type

**Functions** : A function is a block of organized, reusable code that is used to perform a single, related action. What about getting values out of a function? This is where the concept of **return values** comes into play. The work that functions do can produce new results.

But there was no return statement in the function. So the value of results is none. **None** is a very special **data type** in Python used to indicate that things are empty or that they return nothing.

**Self-documenting code** : Written in a way that’s readable and doesn’t conceal it’s intent.

**refactoring** is use to re-write code to more self-documenting. The names of the variables and the function reflect their purpose, which helps the reader understand the code more quickly.

Using **comments(#)** lets you explain why a function does something a certain way. It also allows you to leave notes to your future self or other programmers to remind you of what needs to be improved and why. Obviously, it's much easier to read your own code than someone else's.

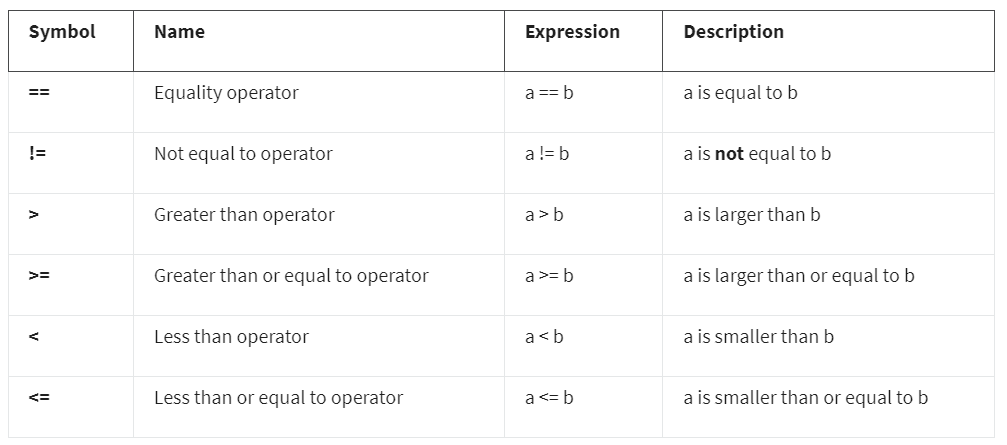
**Terms**

* return value - the value or variable returned as the end result of a function
* parameter (argument) - a value passed into a function for use within the function
* refactoring code - a process to restructure code without changing functionality

**Knowledge**

* The purpose of the def() keyword is to define a new function.
* Best practices for writing code that is readable and reusable:
  + Create a reusable function - Replace duplicate code with one reusable function to make the code easier to read and repurpose.
  + Refactor code - Update code so that it is self-documenting and the intent of the code is clear.
  + Add comments - Adding comments is part of creating self-documenting code. Using comments allows you to leave notes to yourself and/or other programmers to make the purpose of the code clear.

**Comparison Operators with Equations**



Comparison Operators with Strings

