**Development 1: A Starting guide:**

State: this means storage of information => Fundamental unit of information: Binding a name to a value => **CAUTION:** the label (variable name) name must be unique

And empty state, storing nothing, is a state that often occurs => a empty state is returned when a empty variable or non-existing variable is looked up

It is possible to give a variable the same state as another variable, **BUT** changing the state of the original variable will not change the state of the other variable.

A state of a variable can be altered and removed if no longer needed.

A state can contain multiple states => list, dictionary’s

Statements: a set of instructions given to a program

A combination of statements are called a program.

Statements contain 2 parts:

1. **Syntax:** structured text with constant parts => keywords => keywords are reserved words and **CANNOT** be used as a variable name, function name or any other identifier.
2. **Semantics:** describes how the statement transforms itself and a state into a new statement and a new state => gives a statement meaning

There are statements that combine two statements: **AND**

Variable: used to bind and retrieve values in the state => variable assignment embeds the binding operation => these become bindings in the state

\*programming languages cannot be more powerful than the machine

\*\*certain states can be reached with certain language and some can’t. This makes languages more expressive than others.

Data type: allows the developer to define data with a certain attribute that tells the compiler or interpreter how the developer intends to use this data.

A data type can be thought of as a set:

1. Contains all elements that make up the data type
2. Has important connections:
   1. These connections are structured. This structure determines a network of ties between the elements which are all and the only paths that can be followed to get to a certain point (value)
   2. These connections are also called operators
   3. When using a operator they must be given in a specific order this can be seen like following an arrow

**Expressions:** a specification to determine a value via computation => Expressions combined with statements form programs

**Evaluating expression:** following the defined structure, one piece at a time. Reduce the expression to simpler and simpler form until a value within the data type is reached

**Example:**

3 + 2 = 5 **=>** this means that we van **GO** from 3 + 2 to 5 and **NOT** 3 + 2 = 5

\*\*Computations move from a complex specification to a simpler answer but **NOT** the other wat around

**Computing:** any activity that uses computers to manage, process and communicate information => based on following arrows => data type

**Control flow:**

Any program can be written using basic **control structures.** A control structure (or flow of control) is a block of code that analyses variables and chooses a direction based on the given parameters (expressions).

Simple: a control structure is just a decision that the computer makes

Flow of control through any given program is implemented with three basic types of control structures: **Sequential, Selection and Iteration**.

**Sequential:** Straight path, Sequential execution is when statements are executed one after another in order. You don't need to do anything more for this to happen. => Example: normal statements

**Selection:** Selection used for decisions, branching - choosing between 2 or more alternative paths.Selects a path based on the outcome of an expression (Boolean) => if.. else..

**Iteration:** Repeating a piece of code multiple times in a row => while loop

A close up of a clock

Description automatically generated

**Operators:** these are special symbols that carry out arithmetic or logical computation => the value that a operator operates on is called a operand

**Arithmetic Operators:** are used to perform mathematical operations like addition, subtraction, multiplication etc.

A screenshot of a social media post

Description automatically generated

**Comparison operators:** are used to compare values. It either returns a **True** or **False** (Boolean value) according to the condition.

A screenshot of a social media post

Description automatically generated

**Logical operators:** these are the **and**, **or** and **not** operators

A screenshot of a cell phone

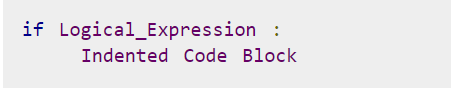
Description automatically generated

**Assignment operators:** used to assign values to variables.A picture containing light

Description automatically generated

**If statement:** A bare Python if statement evaluates whether an expression is True or False. It executes the underlying code only if the result is True.

**Syntax:**



**Flowchart:**

**A close up of a map

Description automatically generated**

**If else statement:** If the result is True, then the code block following the expression would run. Otherwise, the code indented under the else clause would execute.

**Syntax:**

**A close up of a logo

Description automatically generated**

**If else flowchart:**

**A close up of a map

Description automatically generated**

**If elif else statement:** The first three if-else constructs can only address two outcomes, i.e., True or False. However, the expression next to “if” can also evaluate to a different value. It means to have more conditions, not just a single “else” block.

**Syntax:**

**A screenshot of a cell phone

Description automatically generated**

**If elif else flowchart:**

**A close up of a map

Description automatically generated**

**While loop:** A **while** loop statement programming language repeatedly executes a target statement as long as a given condition is true.

**Syntax:**

**A close up of a logo

Description automatically generated**

**While flowchart:**

**A close up of a logo

Description automatically generated**