Computational science is about how to solve problems. Programming is the main thing in computational science. Most of the human made machines are designed to accomplish only one single work. Computers by themselves can’t accomplish any labor; but computers with programs, can do a lot of different things. A computer program is a simple sequence of instructions, sequence that the computer must follow closely; these different instructions are made using a programming language, and python is just one of them. Programming languages are different from natural languages in that are less verbose, less redundant and specially unambiguous, as well as space saving. Most languages today have a set of rules (a syntax) described as Backus – Naur form; they are composed by “expressions”, being an expression an abstract concept that has some value. The expressions can be combined by the construction “<non-terminal> -> replacement”, where the arrow signals an assignment operation. Every expression can be successively replaced until it gets to a terminal value, where it can be no longer replaced. Some other rules come from this construct, like “expression -> expression operator expression”, “expression -> number” or “expression -> (expression)”; notice that all of these keep the same pattern of the original construct “<non-terminal> -> replacement”. A recursive definition requires at least a set of two definitions; otherwise, it would be circular, that being a meaningless definition.

A variable is a name that refers to a value. In python, a variable is assigned using the form “<variable name> = expression”, where “=” corresponds to the assignment symbol. As a convention, the first letter of the variable must be lowercase. Care must be taken to avoid confusing mathematical “equal to” with the assignment operator “=” used in python.

In python, text is represented as a string, which is a sequence of characters (letters, digits and symbols); strings can contain any number of characters. A string is built by surrounding text by single or double quotes. The operator “+”, when used with two strings, concatenates them; also, using the operator “\*” for multiplying a string times an integer, produces a bigger string composed of multiple times the initial string. Strings can be indexed and sliced; in python, the first component the string is indexed as zero. The method for indexation is “string[expression]”, where expression is anything that evaluates to a number. The way to slice is “string[expression1:expression2]”, and the slice includes the characters from “string[expression1]” to “string[expression2 – 1]”. The method “string.find(str)” produces the position where first occurs the substring str in string, or returns -1 if it is not found. The print function sends an output to the console.

To manufacture a multiline string, that includes returns and spaces, triple single quotes or triple double quotes are used.

Functions are procedures that are used frequently used, so instead of declaring them every time it is needed, it is defined in a generical way, and easily called when desired. To define a function the structure is:

def nameFunction (parameters):

procedure

return xxx

And to use the function, is called by the name and passing it the necessary arguments. In this case x = nameFunction(parameters) , it returns xxx, which will be kept in a variable x. Every function has inputs and outputs. To get an output from a function, it must return something. If a variable is passed to a function, it does not get modified as a result of the function; the variables defined inside the function, that is, that are local to the function, are encapsulated and won’t exist outside the function.

In order to make decisions, it is used the if construct, which evaluates a test expression and, according to its value (True or False) performs the instructions of one block or another. The way the if is built is:

if <test expresion> :

<block>

else:

<else block>

The test expression is usually constructed with less than (<), more than (>), is equal to (==), is different to (!=). The end of the blocks is determined by indentation. “or” and “and” can also be used to evaluate the test expression.

To perform a repetitive action, the “while” loop is used; is constructed like this: while <test expression>:

block

The block is executed while the text expression is true, and when it changes, the block execution stops; in order to avoid that a block keeps going forever, in every iteration of the block, something must change. Sometimes, an index variable is used to keep track of the execution, and when it reaches a certain value, it makes the test expression False, so the loop execution stops. The while loop can perform some instruction multiple times, but if can perform it only once. One way to make a while stop is putting a break inside it.

While programming is unavoidable to have mistakes; its very important to develop a programming technique that allows to find easily the mistakes, and a method to find and correct programming mistakes systematically. The methodology to find and correct mistakes is called debugging, and it has some important steps:

Read the error message; in python, the error message gives a type of error and number of line where execution stopped; although the error can be found there, it can also be upstream.

One option is to copy a similar function, and approach to the needed function, changing the function, step by step, one small part every time, evaluating what happens when every small thing is changed.

If one is using third parties code and things are not working, it is necessary to go back to the shared code and evaluate if every part is doing what it is supposed to be doing.

Bugs that don’t stop code execution are harder to find and make programs work erratically; if one is not getting the expected results, it is useful to print intermediate results.

Sometimes, code works partially; if one gets to a point where major changes must be done, one can screw up what is really working, so it is really important to not loose what is working well; it is preferable to comment the original code, save a copy or use version controllers, like git.

Recapitulating the debugging strategy: error messages must be examined, work from example code doing small step modifications, get sure that code from third parties work correctly, print intermediate results, and keep and compare old versions.

Comments are very useful when debugging; single line comments are made with #, and docstrings are made with triple single quotes; docstrings are multiple line comments that the python interpreter retains as function documentation. Comments must be kept updated, while code is modified. The idea is that code must be concise, always consistent with the style chosen to document the code, and must give important information; that is, the obvious must not be commented.

String is data type that is structured; it is a sequence of characters that can be split into smaller pieces (characters). But strings aren’t complex enough, not everything can be done with a string, so python implements ‘bigger’ structured data types. Lists are sequences of anything; in python, they are formed with square brackets, and list elements are separated by commas: <list> = [<expresion>, <expresion>, …]. List elements are indexed in the same way that strings. Lists can nest other lists inside of them.

Lists are mutable, that meaning it may be modified after being created. Strings are not mutable; elements can be appended to strings, but they form a new string, won’t modify the original. This seemingly subtle difference gets more notorious if one introduces a new variable and assigns it to same object: if a list changes with one name, also changes with the other; this is called aliasing: there are two ways to call the same object. Functions can’t change the values of strings or numbers outside themselves, but can mutate list elements.