Python and Bioinformatics

Pierre Parutto

September 17, 2016

Contents

1 Conditional And Repeated Executions **2**
1.1 Python file mode 2
1.2 Conditional Execution - `if`, `elif`, `else` keywords 3
1.2.1 The `if` Keyword 3
1.2.2 The `elif` Keyword 4
1.2.3 The `else` Keyword 6
1.2.4 The `if` Construction 7
1.3 Repeated Execution - `while` Keyword 8
1.3.1 Classical `while` Structure 9

Chapter 1

Conditional And Repeated Executions

1.1 Python file mode

The interpreted mode is nice for typing short commands and getting answers directly, but when constructing more complicated programs one will write codes into files instead. Reading a code file is more complicated than reading a single instruction in interpreted mode, we will first study how Python reads files.

Definition 1 *An instruction, also called statement, is a line of code in a Python code file.*

Definition 2 *The instruction flow is the way Python reads you code files. By default, the flow follows the standard occidental rules for reading: line by line from top to bottom, each line is read from left to right.*

With the default flow, each line is only executed once, starting with the topmost line. In the following sections, we will see two constructions that allows to modify the instruction flow for a group of instructions. Python define groups of instructions based on their indentation level.

Definition 3 *The indentation level of an instruction is the number of tabulation character(s) at the beginning of the line.*

The topmost line starts with 0 tabulation on the left and thus is at the indentation level 0.

Definition 4 *A group of lines, or code block, is an ensemble of successive lines with* **the same level of indentation or higher.**

Example

On the right is the instruction flow associated with some Python instructions:

Through these definitions, we can see that a code file is just a list of instructions that you want Python to perform. With the default instruction flow this list is sequential: every line will be executed in order and if you want to give ten time the same instruction to Python you will need to write it ten times (as you did in lab 1). In the following we are going to see how to modify the instruction flow to execute some instruction group only `if` (resp. `while`) a certain condition is true.

1.2 Conditional Execution - `if`, `elif`, `else` keywords

The conditional execution allows to execute groups of instructions depending of the truth values of some conditions.

1.2.1 The `if` Keyword

The `if` keyword allows you to execute a certain group of instruction **only if** a given condition is true. The Python syntax is the following:

```
if CONDITION:
    GROUP OF
    INSTRUCTIONS
    TO EXECUTE
```

Where `CONDITION` is any expression of Boolean type (that evaluates to `True` or `False`) and `GROUP OF INSTRUCTIONS TO EXECUTE` represents the instruction group that will be executed only if `CONDITION` evaluates to `True`.

Definition 5 *`GROUP OF INSTRUCTIONS TO EXECUTE` is called the body of the `if`.*

Warning

About the body of the `if`:

- The indentation level of the body is +1 compared to the indentation level of the `if` instruction.
- The body of the `if` ends at the first instruction with the same indentation level as the `if` instruction.
- The body of a `if` must contain **at least** one instruction.

Warning

In the following classes I may say "condition is true" this is a shortcut to "condition evaluates to true". You must understand that conditions can be a complicated expressions that may need to be evaluated by Python.

Definition 6 *We call a case a condition test with its associated instruction group.*

Example

On the right is the instruction flow associated with an `if` case:

```
if CONDITION:
    INSTRUCTION      → default
    GROUP            → CONDITION is True
INSTRUCTION          → CONDITION is False
INSTRUCTION
```

1.2.2 The `elif` Keyword

The `elif` keyword can be put after a `if` case at the same level of indentation and allows you to test another condition. The syntax is the following:

```
if CONDITION1:
    INSTRUCTION
    GROUP 1
elif CONDITION2:
    INSTRUCTION
    GROUP 2
elif CONDITION3:
    INSTRUCTION
    GROUP 3
...
elif CONDITIONn:
    INSTRUCTION
    GROUP n
```

Warning

There can be 0 or multiple (as much as you need) `elif` cases.

The conditions of each `elif` cases are evaluated in the order in which they are encountered. Once some condition evaluates to true, its instruction group is executed and the next cases are skipped.

Warning

The order in which you put the different `elif` cases matters, for example:

```
isodd = False
ismult3 = False

a = 9
if a % 2 == 1:
    isodd = True
elif a % 3 == 0:
    ismult3 = True
```

After this code, `isodd` is `True` and `ismult3` is `False`. As the first condition evaluates to `True`, its associated instruction group is evaluated and the following case is skipped. Now if we change the order of the cases:

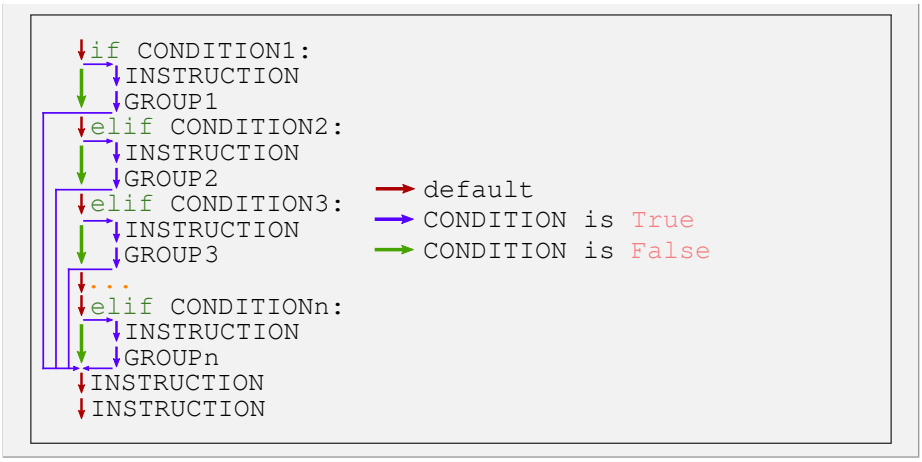
```
isodd = False
ismult3 = False

a = 9
if a % 3 == 0:
    ismult3 = True
elif a % 2 == 1:
    isodd = True
```

the results are `isodd` is `False` and `ismult3` is `True`. As in the first example, the first condition is `True`, hence its associated instruction group is executed and the other case is skipped.

Example

On the right is the instruction flow associated with `if`, `elif` cases:



1.2.3 The `else` Keyword

Finally, the else keyword allows to provide an instruction group to be executed if all the previous conditions evaluated to false. Its syntax is the following:

```
if CONDITION1:
    INSTRUCTION
    GROUP 1
[elif CONDITION2:
    INSTRUCTION
    GROUP 2
...
elif CONDITIONn:
    INSTRUCTION
    GROUP n]
else:
    INSTRUCTION
    GROUP
```

Remark

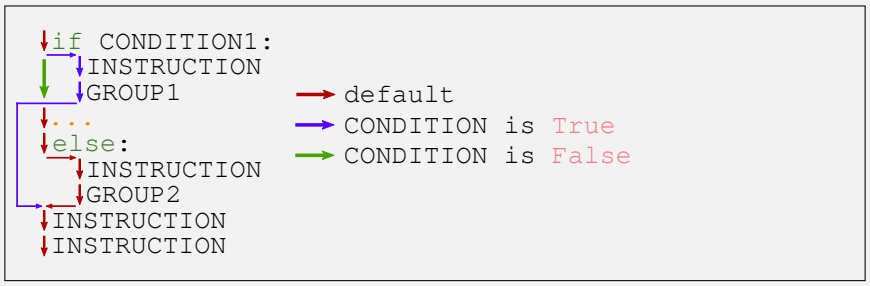
- The brackets "[" and "]" represent the fact that there may be 0 or multiple `elif` cases before an `else`.
- The `else` case do not have any associated condition.

Warning

The can be either 0 or 1 `else` case.

Example

On the right is the instruction flow associated with `if`, `elif`, `else` cases:



where `...` represents any number of `elif` cases.

1.2.4 The `if` Construction

These paragraphs present multiple remarks on the use of the `if`, `elif` and `else` keywords.

Definition 7 *An if construction is composed of a `if` case followed by 0 or multiple `elif` cases and 0 or 1 `else` case.*

In a `if` construction there can be **only 0 or 1** instruction group executed no matter the complexity of the construction. Hence the different instruction groups inside a construction are not independent.

Warning

In a `if` construction, the first keyword is always a `if`.

Here is a table of all the possible `if` constructions:

```
if CONDITION1:
    INSTRUCTION
    GROUP 1
```

```
if CONDITION1:
    INSTRUCTION
    GROUP 1
elif CONDITION2:
    INSTRUCTION
    GROUP 2
```

```
if CONDITION1:
    INSTRUCTION
    GROUP 1
else:
    INSTRUCTION
    GROUP 2
```

```
if CONDITION1:
    INSTRUCTION
    GROUP 1
elif CONDITION2:
    INSTRUCTION
    GROUP 2
...
elif CONDITIONn:
    INSTRUCTION
    GROUP n
```

```
if CONDITION1:
    INSTRUCTION
    GROUP 1
elif CONDITION2:
    INSTRUCTION
    GROUP 2
...
elif CONDITIONn:
    INSTRUCTION
    GROUP n
else:
    INSTRUCTION
    GROUP
```

Where the `...` represent any number of `elif` cases.

Warning

Be careful with the following code:

```
if CONDITION1:
    INSTRUCTION
    GROUP 1
if CONDITION2:
    INSTRUCTION
    GROUP 2
```

It presents **two** `if` constructions, as the two instruction groups are independent of each other.

1.3 Repeated Execution - `while` Keyword

The repeated execution allows to execute the same instruction group multiple times, while some given condition evaluates to true. Its syntax is very similar to the `if` keyword and is the following:

```
while CONDITION:
    INSTRUCTION
    GROUP TO
    EXECUTE
```

Where `CONDITION` is any expression of Boolean type (that evaluates to `True` or `False`) and `GROUP OF INSTRUCTIONS TO EXECUTE` represents the instruction group that will be executed while `CONDITION` evaluates to `True`.

As for the `if` keyword, the following definitions and important points are also true:

Definition 8 *`GROUP OF INSTRUCTIONS TO EXECUTE` is called the body of the `while`.*

Warning

About the body of the `while`:

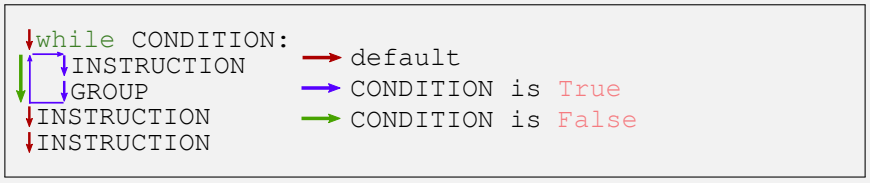
- The indentation level of the body is +1 compared to the indentation level of the `while` instruction.
- The body of the `while` ends at the first instruction with the same indentation level as the `while` instruction.
- The body of a `while` must contain **at least** one instruction.

Warning

`CONDITION` must become true at some point otherwise you obtain an **infinite loop**.

Example

On the right is the instruction flow associated with a `while` construction:



1.3.1 Classical `while` Structure

A classical use of `while` structures is to repeat some instruction group a certain number of time. To do that, one uses a counter variable that counts the number

of turns done. The following code performs `n` repeats using a counter variable `i`:

```
i = 0
while i < n:
    SOME
    INSTRUCTIONS
    i = i + 1
```