Unit 7. Control Flow

Héctor D. Menéndez

Endless Science

endsci.net

Index

Index

Control Flow

Computers execute the program code sequentially.

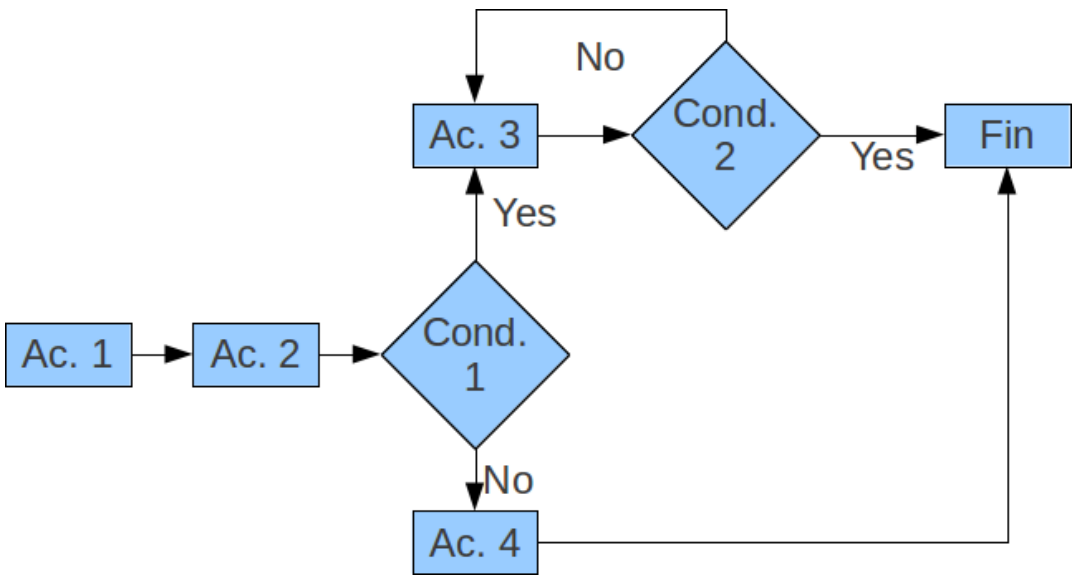
Usually the programmer wants to either repeat a specific part of the code several times or, depending on some conditions, she might want to execute different things.

Control flow statements allow the programmer to organize the code execution depending on the circumstances.

The control statements in Python are if-else, for and while statements.

Control Flow Representation

The control flow is normally represented with arrow diagrams where the normal statements execute sequentially, and the conditions decide whether the program moves on or repeats code:



Control Flow: blocks

When a control statement is used the code related to it is set inside of a block. In Python, the way to define a block is:

1. Create the control flow statements and write ":" to start the block.
2. Increase the indentation of the code using spaces of tabulars (normally 2 spaces or 1 tabular).
3. The new block ends when the indentation goes back to the previous block.

Control Flow: Blocks

```
This is the block one.
This is a condition statement:
    This is the block 2 #The indentation is two spaces different
    Another condition:
        This is block 3.
    This is block 2 again #Block 3 ended when the indentation ended
This is block 1.
```

Listing 1: Block's declaration

Index

Control Flow: if-else

The `if-else` statements are used when the execution of a code block depends on the result of a logical sentence.

The structure of an `if-else` sentence is the following:

```
1 if (condition): #This is the first condition
2   ... #This represents the block that would be executed
3   ...
4 elif (condition): #This is an extra condition
5   ...
6 else: #This runs if the previous conditions are False.
7   ...
```

Listing 2: if-else declaration

Control Flow: if-else

The statement `if` is always needed. It starts the condition.

The statement `elif` is optional. You can put several of them. They are only executed if the `if`'s condition is false and every precedent `elif`'s condition is also false.

The statement `else` is also optional and it is always the last one. It only runs if all the previous conditions are false.

Control Flow: Simple if Example

```
1 name = "John"
2 height = 1.70
3 age = 18
4 print("Your name is " + name)
5 print("Your height is " + str(height) + " m" )
6 print("You are " + str(age) + " yo" )
7 if (age >= 18):
8   print("And you can vote")
```

Listing 3: if example

Control Flow: Simple `if-else` Example

```
1 age = 18
2 if (age >= 18):
3   print("You can vote")
4 else:
5   print("You can't vote")
```

Listing 4: if-else example

Control Flow: Simple if-elif-else Example

```
1 age = 18
2 drivingLicense=False
3 if (age >= 18 and drivingLicense):
4   print("You can drive")
5 elif (age >= 18):
6   print("You need a driving license")
7 else:
8   print("You can walk")
```

Listing 5: if-elif-else example

Index

Control Flow: Repetitions or Loops

Loops or repetitive statements establish a condition and repeat a piece of code while the condition is satisfied.

There are two main ways to create loops:

1. The classic `while` statements that is constantly checking the condition at the end of the block's execution to decide whether it needs to repeat the code again.
2. The `for` loop, providing a list of elements to iterate and repeating the code for each element in the list.

Control Flow: `while`

The `while` statement needs a condition (logical expression) and the code block that it will repeat until the condition is false.

```
1 while (condition):
2     ...
```

Listing 6: `while` declaration

Control Flow: `while` Example

A program that counts from 1 to 10.

```
1 i=1
2 while (i <= 10):
3   print(i)
4   i=i+1
```

Listing 7: `while` counter example

Control Flow: `while` Example

A program that checks a password.

```
users={"user1":"password1","user2":"password2","user3":
    "password3"}
rightPassword = False
while (not rightPassword):
  username = input("Give me your username: ")
  password = input("Give me your password: ")
  if(username in users):
    if (users[username] == password):
      rightPassword = True
    else:
      print("The password is incorrect")
  else:
    print("User not found")
print("Password correct")
```

Listing 8: `while` password example

for

The `for` statement iterates over the elements of a list and runs the repeated code for every element:

```
1 for variable in listVariable:
2     ...
```

Listing 9: `for` declaration

for

A program that counts from 1 to 10:

```
1 for number in range(10):
2   print(number+1)
```

Listing 10: `for` counter example

for

A program that shows all the users in a system:

```
1 users={"user1":"password1","user2":"password2","user3":
      "password3"}
2 for user in users:
3   print(user)
```

Listing 11: `for` users example

Index

Exercise 1: List of numbers

Write a program that asks the user to introduce positive numbers ($>= 0$) constantly. The program ends when the user introduces a negative number. Put every number into a list (but the negative) and, at the end of the program, show the maximum number, the minimum number and the list of numbers introduced by the user.

For example:

```
Introduce a number: 1
Introduce a number: 2000
Introduce a number: 500
Introduce a number: -1
The maximum is 2000, the minimum is 1 and the numbers
    introduced are [1,2000,500]
```

Exercise 2: The Triangle

Create a program that draws a triangle with * symbols. The triangle will have N levels, where N is provided by the user. It follows the pattern:

```
N=2          N=3           N=4
  *            *             *
***          ***           ***
            *****         *****
                         *******
```

Exercise Extra: The Diamond

Extend the previous program to draw a diamond. It follows the pattern:

```
N=2          N=3           N=4
 *            *             *
***          ***           ***
 *          *****         *****
             ***       *******
              *         *****
                         ***
                          *
```