Coding and entities in Python

Marco Petolicchio

October 21, 2020

A first algorithm

Motivation

- In this script we create a tiny algorithm, for which a number inserted by the user has to been verified for its evenness.
- For doing that, we will use the *modulo* function in Python, which yelds the rest of a division.
- If we divide by 2 that number, and if the rest is equal to zero, we can say that this number is *even*, otherwise *odd*.

First iteration

```
var = input("Enter number here: ")

var = float(var)

if(var%2==0):
     print("EVEN")
else:
     print("ODD")
```

Moving towards a better approach

While we have defined the variable with the name `var` as an input by the user, we used it to riassign as a float of that var (`var = float(var)`).

But we can encapsulate it in a single line, which offers a more readable script. Be aware that the number of the opening parentheses have to be equal to the close.

Second iteration

```
var = float(input("Enter number here: "))

if(var%2==0):
     print("EVEN")
else:
     print("ODD")
```

Remaining problems

We have defined `var` as a variable object of *type* **float**. But when we try to use this algurithm with float numbers, it fails to check the evenness of that number.

WHY?

A (super light) introduction to the syntax of Python

General

- Python was created in 1991.
- Python is a **high-level** *programming language*, *objects*-oriented.
- The major characteristics of the language are the possibility to have not-typized variables, and the indentation of the code.

We use the function `print()` in order to make something appear on screen, as the text “Hello world”:

```
print("Hello world")
```

What is a programming language?

A programming language is in informatics a *formal* language, with its own syntax, for which we can elaborate alghoritms and computations on data.

We write a program in a choosen language, then we use a **compiler** which translates that sets of instructions, variables, and so on, into a machine-ready set of instructions.

High vs. low level

The machine level is the lowest one, while the more we *abstract* from them, the more the language is high level.

- High l. languages: Java, Python, R, LISP, Prolog
- Low l. languages: Assembly, machine-language, C (relative)

Actually, we can say that a language is a higher level respect to another one, and not classify them in categories.

Indentation

Despite from other languages, in Python the formal sequences of instructions are expressed towards the use of indentation (4 block spaces or TAB):

```
if a :
    do X
if b :
    if b1 :
        do Y
    else :
        do Z
```

Libraries

We include the libraries we need with an `include()` statement, and from this we import everythinf (*) or just a subset of the library:

```
include(library1)
include(library2)

from library1 import *
from library2 import subLibrary2
```

Data types

Numbers

- `int` (integers)
- `long` (integers long)
- `float` (decimal)
- `complex` (complex numbers)

Letters

- `char` (character)
- `str` (string)

Truth

- `bool` (boolean: TRUE or FALSE)

Data structures

Array

An array consists of a collection of elements, identified by an index or a key. Usually the counting of the elements into an array start from '0'

```
a = ["John", "Jake", "Ugo", "Mario", "Jiri", "Ekaterina"]

print(a[2]) = "Ugo"
```

List

Dynamic arrays that can include every kind of objects. Mutable.

Tuple

Same as list but immutable.

Dictionary

Couples of object with `key` and `value`. The `key` is an immutable object (as a `str`)

Rethinking the algorithm

Why it doesn't work?

Despite from the fact that we used the `float` type, the *modulo* operator works with the *integers*. So, when whe have for example `var=5.2` it yields for a remaining, then the algurithm prints ODD.

Can we move that algorithm further, in order to calculate either the decimals?

YES, in a lot of manners, and now we will see one of them.

Converting `var` as an array of elements

In this passage, we will create `input` as a type array, which contains all the digits in the input.

```
input = input("Enter number here: ")

arr = [i for i in input]
print(arr)
```

Check evenness

Now, we use that array to extract the last digit of the array, and we move the algurithm to be checked against only this digit, that we define as an integer:

```
var = int(arr[-1])
print(digit)
```

Third Iteration

```
input = input("Enter number here: ")

arr = [i for i in input]

var = int(arr[-1])


if(var%2==0):
    print("EVEN")
else:
    print("ODD")
```

Conclusions

This script has been made just for introducing into the computing world of Python.

Despite from this, it is quite unnecessary and flaky: a decimal number cannot be odd or even :)

By the way, the definitions of data type and structures will be useful when we will start to put our hands on the texts and perform computational analyses.