Lab 2

Conditions And Loops

Sup'Biotech 3

Python

Pierre Parutto

October 6, 2016

Preamble

Document Property

Authors	Pierre Parutto
Version	1.0
Number of pages	11

Contact

Contact the assistant team at: supbiotech-bioinfo-bt3@googlegroups.com

Copyright

Contents

1 Introduction

In this second lab, we will manipulate `if` and `while` constructions. This lab is done using the Python interpreter file mode: you will write a Python file containing lines of codes that will then be interpreted by Python. In the codes you will write, there will be **two types of special variables**:

- **input variables(s)** that are used to pass specific values to your code.
- **output variable(s)** that will be read to obtain the result of your code.

There may be **none, one or multiple** input and output variable(s).

1.1 How to test your code

Input variables must not appear in you code as they are specified by the user. If you want to test your code you must use the interpreter. Let us take for example a file containing the following code:

```
res = a + b
```

where `a`, `b` are two input variables and `res` is the output variable.

To test this code, we are going to use the values `a = 6`, `b = 8`, proceed as follows:

1. Start by entering these variables in the python interpreter:

   ```
   >>> a = 6
   >>> b = 8
   ```

2. Then press the small green triangle to execute the code from the file, this produces the following code in the interpreter:

   ```
   >>> runfile(someFileName, wdir=someDirName)
   ```

   Note:

   - This line is directly entered by Spyder, you don't have to do anything.
   - `someFileName` and `someDirName` depend on your computer and the directory from which you run Python;
   - **if there are error(s) in your code file** it is after the `runfile` command that Python will tell you.

3. Finally, if there were no errors during the evaluation of the file, you can ask Python to give you the value(s) of the output variable(s):

   ```
   >>> res
   14
   ```

If you want to test with other values of the input variable(s) you will need to go through all these three steps again.

2 Warm-up

2.1 Scalar Product

The scalar product between two vectors $X = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$ and $Y = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix}$ is given by:

$$X.Y = x_1 \times x_2 + y_1 \times y_2$$

Write a block of code that given the input variables `x1`, `x2`, `y1`, `y2`, compute the scalar product and put the result in the output variable `scalar`.

Variables:

- Input:
  - `x1` an integer.
  - `x2` an integer.
  - `y1` an integer.
  - `y2` an integer.
- Output:
  - `scalar` an integer.

Example:

- In `x1 = 1`
  In `x2 = 2`
  In `y1 = 3`
  In `y2 = 4`
  Out `scalar = 14`
- In `x1 = 5`
  In `x2 = 3`
  In `y1 = 2`
  In `y2 = 6`
  Out `scalar = 27`

Correction:

```
scalar = x_1 * x_2 + y_1 * y_2
```

3 if

3.1 Odd Or Even

Write a block of code that outputs in the variable `parity` `True` if the input variable `n` is odd and `False` otherwise.

Variables:

- Input:
  - `n` an integer.
- Output:
  - `parity` a Boolean.

Example:

- In `n = 1`
  Out `True`
- In `n = 2`
  Out `False`

Correction:

- A first version with an `if`, `else` construction:

```
if n % 2 == 0:
    parity = True
else:
    parity = False
```

- A second version directly using a comparison operator:

```
parity = n % 2 == 1
```

3.2 Is Multiple Of

Write a block of code that outputs in the variable `multiple` `True` if the input variable `a` is a multiple of the input variable `b` and `False` otherwise.

Variables:

- Input:
  - `a` an integer.
  - `b` an integer.
- Output:
  - `multiple` a Boolean.

Example

- In `a = 4, b = 2`
  Out `True`
- In `a = 13, b = 8`
  Out `False`

Correction:

- A fist version with an `if`, `else` construction:

```
if a % b == 0:
    multiple = True
else:
    multiple = False
```

- A second version, using a comparison operator:

```
multiple = a % b == 0
```

3.3 To Lower Case

Write a block of code that outputs in the variable `small` the lower case corresponding to the letter given in the input variable `inchar`. If `inchar` is already a small letter then `inchar == small`.

To complete this question, you will need to use the following functions:

- `ord`: given a character return an integer representing the position of the character in the Unicode table.
- `chr`: given an integer representing the position of a character in the Unicode table, return the associated character.

These functions are used as follows:

```
>>> ord("A")
65
>>> ord("!")
33
>>> chr(69)
"E"
>>> chr(ord("a"))
"a"
```

You will need to use the facts that:

- The characters a,b,c,...,z are adjacent in the Unicode table:
  `ord("b") = ord("a") + 1` and `ord("z") = ord("a") + 25`;
- The characters A,B,C,...,Z are also adjacent in the Unicode table:
  `ord("B") = ord("A") + 1` and `ord("Z") = ord("A") + 25`.

Variables:

- Input:
  - `inchar` a string.
- Output:
  - `small` a string.

Example

- In `inchar = "A"`
  Out `"a"`
- In `inchar = "c"`
  Out `"c"`

Correction:
If the letter is already small, we do not do anything, otherwise we find the position of the letter in the alphabet and add to it the position of the small "a".

```
if ord(inchar) >= ord("A") or ord(inchar) <= ord("Z"):
    small = chr(ord("a") + (ord(inchar) - ord("A")))
else:
    small = inchar
```

4 while

4.1 Sum

Write a block of code that outputs in the variable `mysum` the sum from 1 to `n` (included), where `n` is an input variable.

Variables:

- Input:
  - `n` an integer.
- Output:
  - `mysum` an integer.

Example

- In `n = 100`
  Out `5050`
- In `n = 87`
  Out `3567`

Correction:
Here we just use the classical structure of a `while` loop.

```
i = 1
mysum = 0
while i <= n:
   mysum = mysum + i
   i = i + 1
```

4.2 Sum Of Multiples

Write a block of code that given the input variables `a` and `n` that outputs in the variable `coucou` the sum from 1 to n of the integers multiples of `a`.

Variables:

- Input:
  - `a` an integer.
  - `n` an integer.
- Output:
  - `coucou` an integer.

Example

- In `a = 2`, `n = 11`
  Out `30`
- In `a = 3`, `n = 11`
  Out `18`

Correction:

```
i = 1
coucou = 0
while i <= n:
   if i % a == 0:
       coucou = coucou + i
   i = i + 1
```

4.3 Integer Sequence

Let:

$$u(n) = \begin{cases} 0 & \text{if } n = 0 \\ 3 * u(n-1) + 1 & \text{otherwise} \end{cases}$$

Write a block of code that given the input variable `n` outputs the value $u(n)$ in the output variable `seq_res`.

Variables:

- Input:
  - `n` an integer.
- Output:
  - `seq_res` an integer.

Example

- In `n = 2`
  Out `4`
- In `n = 6`
  Out `364`

Correction:

```
i = 1
seq_res = 0
while i <= n:
   seq_res = 3 * seq_res + 1
   i = i + 1
```

4.4 Integer Sequence - 2

Let:

$$u(n) = \begin{cases} u0 & \text{if } n = 0 \\ u(n-1)^2 \% 2 + u(n-1) & \text{otherwise} \end{cases}$$

Write a block of code that given the input variables `n` and `u0`, outputs the value $u(n)$ in the output variable `yolo`.

Variables:

- Input:
  - `n` an integer.
  - `u0` an integer.
- Output:
  - `yolo` an integer.

Example

- In `u0 = 0`, `n = 2`
  Out `0`
- In `u0 = 3`, `n = 3`
  Out `4`

Correction:

```
i = 1
seq_res = u0
while i <= n:
   seq_res = seq_res**2 % 2 + seq_res
   i = i + 1
```

4.5 Fibonacci Sequence

The Fibonacci sequence is a sequence of integers defined for $n \in \mathbb{N}$:

$$F(n) = \begin{cases} F(n-1) + F(n-2) & \text{if } n > 2 \\ 1 & \text{if } n = 1 \\ 0 & \text{if } n = 0 \end{cases}$$

Write a block of code that outputs in the variable `fibo` the value of the Fibonacci sequence at the rank `n` an input variable.

Variables:

- Input:
  - `n` an integer.
- Output:
  - `fibo` an integer.

Example

- In `n = 8`
  Out `21`
- In `n = 5`
  Out `5`

Correction:
We need to remember at each `n` the two previous values, to do that, we use the variables `fibo` that will store the value at `n-1` and `fibo_pred` that will store the value at `n-2`. The variable `tmp` is used to swap the values of `fibo` and `fibo_pred`.

```
if n == 0:
    fibo = 0
else:
    fibo_pred = 0
    fibo = 1

    i = 2
    while i <= n:
       tmp = fibo
       fibo = fibo + fibo_pred
```

```
    fibo_pred = tmp
    i = i + 1
```