Introduction to Python

Fermín Huarte Larrañaga Ramon Crehuet Simon

June 2, 2019

1 Language Elements (1)

The purpose of these first exercises is, rather proposing the development of involved algorithms, that you develop practice in the basic features of Python. Some of the exercises are meant to be typed directly in the **ipython3** interactive shell or the Jupyter Notebook environment, you may however use another shell or directly the python3 interpreter. Some other exercises are best solved by writing them in a plain text editor as an independent piece of code. Consider developing them in a Jupyter notebook, specially if you are a beginner.

1.1. Write your ”Hello World!” Python program!!!

1.2. As explained, you do not need to declare variables before assigning them to an object. Instead, beware using unassigned variables! Copy this code, try it out and understand the error:

```python
""" Calculate the molar volume of a substance """
vol = 5.2 # litre
mass = 4.5 # kg
mol = mass / molecular_weight
density = vol / mol
print (" d = {:3g} kg/L".format(density))
```

1.3. Open a **ipython3** shell, type (one by one) the following commands in the ipython shell and understand the outcome of each instruction:

```python
0x = 0
x0 = start
a = 2
b = A + 4
x = 3
n = 3.0
x == n
x is n
ans = True
python ="I love learning Python!"
"love" in python
"love" in python is ans
("love" in python) is ans
```

1.4. Open a **ipython3** shell, type (one by one) the following commands in the ipython shell and understand the outcome of each instruction:

```
7/3
7//3
x = 7
type(x)
import math
math.sqrt(x)
math.sqrt(-x)
type(y)
y = 3 + 1j
type(y)
import cmath #this is the complex math module
cmath.sqrt(y)
cmath.sqrt(x)
cmath.sqrt(-x)
```

1.5. Alghough it is not highly recommendable if you are new in programming, you may often find pieces of code that use **compact notation**. This is just an alternative (and compact) way of performing arithmetic operation. Here you have some examples. Open a **ipython3** shell, type (one by one) the following commands in the ipython shell and understand the outcome of each instruction:

```
i = 1
i += 1
print(i)
a = 10
a *= 2
print(a)
a /= 5 # can you predict the value of a? (DO NOT PRESS RETURN!)
```

1.6. Compute the number of seconds in one day. *A: 86400 s*

1.7. I have 3 bills of €50, 3 of €20, 7 of €10, and 1 of €5. How much money do I have?

1.8. Let p be a bank's interest rate in percent per year. An initial amount A has after n years grown to:

$$A\left(1+\frac{p}{100}\right)^n$$

Write a program for computing how much money €1000 have grown to after three years with 5% interest rate.

1.9. Work out a program that calculates prints the volume and surface area of a cube with sides of length 1.5. Use `format` to print out the result with 1 decimal figure.

1.10. Work out a program that calculates the volume and surface area of a sphere with radius 3. Use `import math` to obtain the value of π. Use `format` to print out the result with 2 decimal figures.

1.11. Check numerically that, for a given volume (of your choice), a sphere offers less surface area than a cube.

1.12. Given a triangle with two adjacent sides with length 11 and 8, and angle of 37° between them, find out the length of the third side of the triangle. *Hint: Use the cosine formula:* $c^2 = a^2 + b^2 - 2\cos(\gamma)$. *A: 6.66*

1.13. The Gaussian function is one of the most widely used functions in science:

$$\psi(x) = \frac{1}{\sqrt{2\pi}\delta} \exp\left(\frac{1}{2}\left(\frac{x - x_0}{\delta}\right)^2\right),$$

where x_0 is the point at which the function reaches a maximum and actually the center of the function (it is symmetric) and δ is related to the width of the function. Find out the value of a Gaussian function with $x_0 = 0.4$ and $\delta = 0.33$ at $x = 0.8$. *A: 0.5799*

1.14. *Polar Coordinates (I):* A common way to represent a point in a bidimensional space is employing cartesian (x, y) coordinates. Alternatively, one may use a polar coordinate system in which a radius (ρ) and a *polar angle* (θ) corresponding to the distance and orientation with respect to the origin, respectively.

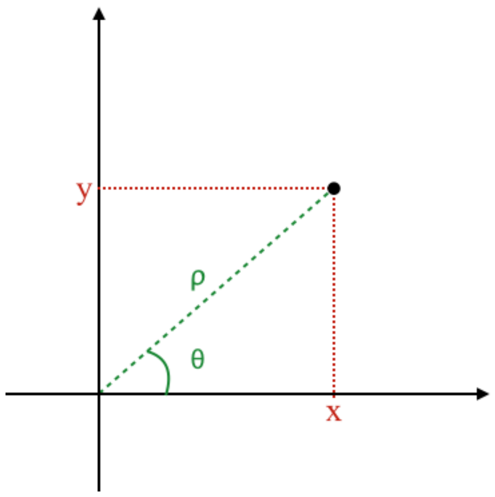

Figure 1: Polar and cartesian coordinates.

Conversion between polar and cartesian coordinates can be carried out according to:

$$\rho = \sqrt{x^2 + y^2}$$
$$\cos(\theta) = \frac{x}{\rho}$$

Write a program that calculates the polar coordinates corresponding tor a pair of cartesian coordinates. (*Check:* $(x = 0.5, y = 0.75) \rightarrow (\rho = 0.90, \theta = 0.98)$.

1.15. *Polar Coordinates (II):* The previous program will not work properly for cartesian coordinate values with negative y value. Check that for $(x = 0.5, y = -0.75)$ you obtain $(\rho = 0.90, \theta = 0.98)$. Check the IPython help on the `math.atan2()` function and use it to improve the code. i (*Check:* $(x = 0.5, y = 0.75) \rightarrow (\rho = 0.90, \theta = -0.98)$.

1.16. Ok, you made it through the first session, relax and type this in the Python3 console: `import this`

2 Language Elements (2)

2.1. Given the two following genetic sequences:

`agcgccttgaattcggcaccaggcaaatctcaaggagaagttccggggagaaggtgaaga`

and

`cggggagtggggagttgagtcgcaagatgagcgagcggatgtccactatgagcgataata`

- assign the two sequences to two different variables, concatenate them in a single variable named `seq`.
- print out the nucleotides of the first and last codon
- Calculate and print the percentage of each base in the global sequence.

2.2. Use ipython3. Let us have the sequence (copy it literally, with line break!):

```
dna = """tgaattctatgaatggactgtccccaaagaagtaggacccactaatgcagatcctgga
tccctagctaagatgtattattctgctgtgaattcgatcccactaaagat"""
```

- Count the occurrences of 'GAATTC' (corresponds to EcoRI)
- Type: `dna.find?`
- Locate the codons expressing EcoRI
- Count the occurrences of 'GGATCC' (corresponds to BamHI)
- Type: `dna`. See the line break?
- Type: `dna.replace?`
- Repair the dna sequence (get rid of the line break), count the occurrences of 'GGATCC' (corresponds to BamHI), and locate the codons expressing it.

2.3. Lists behave as pointers. This behaviour can be very confusing at first. Open a **ipython3** shell, type (one by one) the following commands in the ipython shell and understand the outcome of each instruction:

```
x = 3
y = x
x = 5
print(y) #variable y retains the original value (object)
x = [1, 2, 3]
y = x
x[1] = 5
print(y) #y is actually a pointer to x, so, when x changes, so does y!
```

2.4. Write a python3 program that generates the 3×3 identity matrix and prints it matrix form. We will see that Numpy makes this much easier!

2.5. Repeat Exercise 2.4. for a 10×10 matrix, use list comprehensions and the `range()` function. We will see that Numpy makes this much easier!

2.6. More about copying and references. Open a **ipython3** shell, type (one by one) the following commands in the ipython shell; try to predict what will be the content of b and c, after each operation.

```
a=[1,2,3,[4,5]]
b=a.copy()
c=a
```

```
a[0]=2
print(c)
a[3][0]=0
print(b)
print(c)
```

Fortunately there is the copy module that solves this problems. Use it and repeat the same previous assignments and operations.

```
import copy
b = copy.deepcopy(a)
```

2.7. Write a new version of code in Exercise 1.15.. In this new version, the computer should prompt the user to introduce the value of the cartesian coordinates x, y. The user should introduce both cartesian values in one single line.

2.8. A triangular number counts objects arranged in an equilateral triangle:

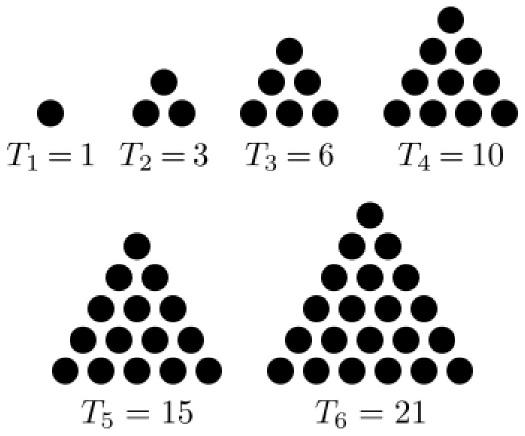

Figure 2: First six triangular numbers. Image from Wikipedia

Such triangular numbers are defined according to:

$$T_i = \frac{i(i+1)}{2}$$

where $i = 1, 2, 3, \ldots$ indicates how many elements (circles in the figure) are found on each side and T_i the total number of elements (circles) in the figure.

Write a program that computes and writes on the screen the sequence of triangular numbers from T_1 to T_n, where n is given by the user.

2.9. The code below has been written to calculate and print the result of the following sum:

$$\sum_{k=1}^{5} \frac{1}{k}$$

The code does not work correctly. Identify and correct the **three errors** in the code:

```
k = 1
M = 5
```

```
while k < M:
     s = s + 1/k
print("La suma val: {}".format(s))
```

(*Check:* The value of the sum is 2.28333.)

2.10. The sum of triangular (Exercise 2.8.) from 1 to a value n yields a tetrahedral number:

$$D_n = \sum_{i=1}^{n} T_i$$

Tetrahedral number count the number of spheres that can be arranged to form a tetrahedron. Write a program that computes the tetrahedral number corresponding to a value n given by the user. (*Check:* For $n = 5$, $D_5 = 35$.)

2.11. Write a code that calculates and prints the sequence of the first N primer numbers, where N is entered by the user. Make sure you have a look at the `list.append()` function.

2.12. Write a code that calculates and prints the sequence of the first N perfect numbers, where N is entered by the user.

2.13. Write a code that calculates the volume of $SO_{2(g)}$ using the Van der Waals equation of state for a given set of (n, p, T), given by the user. Tabulate the volume as a function of temperature for a given pressure.

2.14. The number π can be calculated according to the series:

$$\sum_{j=1}^{\infty} \frac{1}{j^2} = \frac{\pi^2}{6}$$

Write a code that computes the number pi according to this series, truncating the sum to a desired precision, given by the user.

2.15. List indexing. Using a Notebook cell or the IPython shell, define the following list:

```
world = [['hola', 'ciao', 'hello', 'privet'],['adios', 'arrivederci', 'bye',
    'pakah'],['gracias', 'grazie', 'thanks', 'spasibo']]
```

Next, write one by one, the instructions that will make the compute write:

- the word 'ciao'
- the list ['adios', 'arrivederci', 'bye', 'pakah']
- the word 'gracias'

What do you expect that the computer will print out upon the instruction `world[-1][-2]`? Check your guess.

2.16. Series can be efficiently computed by storing all the elements in a list and carrying out the sum with the `sum()` function. Write a second version of the program in Exercise 2.10. where the triangular numbers are first stored in a list and summed afterwards to produce the tetrahedral numbers.

2.17. Write a piece of code that asks the user to introduce (separately) two lists of elements. Next, the code should build a third list containing only the elements present in both lists, and print such list.

2.18. Build a dictionary containing 10 chemical elements and their atomic number.

2.19. Write a code where a dictionary is declared, where the keys are the symbols of chemical elements and the values are the corresponding atomic masses. For practical reasons, you may limit the dictionary to the following table:

Element	mass (g mol^{-1})
H	1.008
C	12. 011
N	14.007
O	15.999
S	32.065
Cl	35.453

Table 1: Atomic masses

Next, the code should prompt the user to type the symbol of a chemical element and, according to the user input, write the corresponding atomic mass.

2.20. Modify program in Exercise 2.19. so that, if the user introduces a chemical element not present in the dictionary, the computer will prompt the user to introduce the corresponding mass and integrate it in the dictionary. As a check, print the complete dictionary and make sure that any new mass value is actually a numerical value.

2.21. Following Exercise 2.19. write a computer code that calculates the molar mass of a compound according to its molecular formula. The program should ask the user for the number of atoms of each element in the molecular formula. (*Check:* Clenbuterol $C_{12}H_{18}Cl_2N_2O$: 277.195 g mol^{-1}.)

2.22. The keys in a dictionary are not restricted to be strings. In fact, any Python object whose contents cannot be changed can be used as key. A dictionary may be used, for instance, to build systematically polynomials, such as:

$$pol(x) = -1 + x^3 + 5x^6$$

The data associated with this polynomial can be viewed as a set of power-coefficient pairs, in this case the coefficient ?1 belongs to power 0, the coefficient 1 belongs to power 3, and the coefficient 5 belongs to power 6. A dictionary can be used to map each power to a coefficient:

```
pol = {0: -1, 3: 1, 6: 5}
```

The main advantage with respect to a list is that we need to store only the non-zero coefficients. The polynomial may later be evaluated as:

```
x = float(input("x value: ")
func = 0.0
for power in pol:
    fun += pol[power]*x**power
```

Or we can use the `sum` built-in function:

```
x = float(input("x value: ")
func = 0.0
fun = sum([pol[power]*x**power for power in pol])
```

2.23. Write a code snippet that uses both a list and a dictionary to represent the polynomial $\frac{1}{3} + 7x^{99}$. Print the list and the dictionary, and use both to evaluate the polynomial for x = 1.05. Compare the performance using the `%timeit` ipython3 command.

2.24. Write a code that, given a polynomial as a dictionary, differentiates it and returns the dictionary representation of the derivative.

2.25. Chemical elements: Let us build a dictionary to translate the symbol of chemical elements into the correponding names. Type the following instructions in `ipython3` and understand the result of the sentences:

```
PerTab = dict(H = 'hydrogen', Na = 'sodium', Li = 'lithium', K = 'potasium', \
S = 'sulphur')
PerTab
PerTab.keys()
PerTab.values()
PerTab['Li']
PerTab.get('Li')
```

2.26. Build one phonebook using a Python dictionary. Suggestion: use names of people as `keys` and their respective phone numbers as corresponding `values`.

2.27. Telephone book: Build a dictionary containing names of people as `keys` and their respective phone numbers as corresponding `values`. Check that the order in which the dictionary is printed is absolutely arbitrary. Now, add the international area code (+34) to all the numbers in your phone book.

2.28. Chemical elements (II): Given two dictionaries translating the symbol of chemical elements into the correponding names:

```
PerTab1 = dict(H = 'hydrogen', Na = 'sodium', Li = 'lithium', K = 'potasium', \
S = 'sulphur')
PerTab2 = dict(He = 'helium', Na = 'sodium', C = 'carbon', O = 'oxygen', \
S = 'sulphur')
```

Merge the two dictionaries. Suggestion: use `sets`.

2.29. *A card game:* Let us develop a very simple Python code for a two player card game. Each player has a deck of 10 cards with numerical values from 0 to 9. In each round, the two players pick a card from their respective decks and compare the values. The player with the highest valued card wins the round and earns 1 point. The cards played may not be played again. The game is over when the players run out of cards (the deck is empty) and the consists in showing one of the cards and the player with most points wins the game. Player 1 will be the user and Player 2 will be the CPU. Player 1 will actually choose the card to play (from those he/she has not played before!) while Player 2 (CPU) will choose the card randomly.

Suggestion: To simulate the computer play, you may use the `random` Python module. This module contains a set of functions for random number generation. In particular, the `random.choice()` function picks randomly an element from a list (given as an argument of the function). See an example:

```
import random
a = ["alfa", "beta", "gamma"]
```

```python
b = random.choice(a)
print("The value {} was picked!".format(b))
```