Python programming
Lab. Work n° 1 : First steps

Preliminaries : Verify that you have a Python 3.x.x installed on your machine, **do not use any obsolete version as 2.x.x.** Better it to use an IDE such as `PyCharm` (best) or `Visual Studio Code`. If you are not very confident with development environments, choose one IDE and use it all along the class.
Please remind that teachers can be called to help you on any problem you get. Don't get stuck on an issue for too long.

Exercise n°1 : In the beginning...

At the very beginning, any programmer tests its development environment by writing the famous "Hello world!" program, *i.e.* the one that prints that exact message on the console.

1. In your environment (IDE) create and edit a python script named `hello.py` to insert the appropriate `print` instruction. Run that script. Observe the result on the console. If it doesn't work, ask for help!
2. Run the interactive mode of Python (also known as REPL mode - Read Eval Print Loop) and type `print("Hello world!")`. Then get off REPL by typing `quit()`.
3. In REPL mode, compute $\frac{13}{4}$ and $\frac{4}{2}$ using operators `/` and then `//`. What can you say about the results?
4. In REPL mode, compute the value of $\sqrt{3} + \frac{56}{9} \times |-\frac{1}{4}| + 63^2$ using a single instruction. You may need to import the `math` module and get its basic usage with the command `help("math")`.

Exercise n°2 : Input

1. Modify the script `hello.py` so that when run the user will be able to type its name just after an appropriate message and a welcome message will appear in return. Something like (red is the input) :

```
What's your name? Smith
Hello Smith, welcome to the programming school.
```

2. Create and edit a new script named `circle.py` which asks for a diameter and prints the area of a corresponding circle. Something like :

```
Diameter: 5
Area: 19.634954084936208
```

Exercise n°3 : If...

1. Write a program (a script) named `oddeven.py` which asks for a value and prints if it is odd or even. Like :

```
Enter a value: 45
45 is odd.
```

2. Write a program named `lessorgreater` which generate an (allegedly) hidden random value in $[1, 1000]$, asks user for a value and prints if that value is less, equals to, or greater than the hidden value. (*Hint : there exists a module named* `random` *which you can use to call* `random.randint(a,b)` *to get a random value in* $[a, b[$) :

```
Hidden value is: 738
Give me a value: 45
Sorry 45 is less than my hidden value
          ...another run...
Hidden value: 822
Give me a value: 999
999 is greater than the hidden value
          ...another run...
Hidden value: 148
Give me a value: 148
148 is the hidden value
```

Exercise n°4 : For

1. Create a script named `piapprox.py` that let approximate π by Gregory-Leibnitz serie :

$$\pi = 4 \sum_{n=0}^{\infty} \frac{(-1)^n}{2n+1}$$

As we can not compute to infinity, the user will be able to choose when to stop ($\pi \approx 4 \sum_{n=0}^{N} \frac{(-1)^n}{2n+1}$), like :

```
Sum to? 1000000
pi≈3.1415936535887745
```

2. Create a script named `pibyrandom.py` that computes an approximation of π by Monte-Carlo's method, *i.e.* by choosing random points in the unit square, and counting the proportion of them that lies in the unit circle (distance from origin $\leqslant 1$). The user should be able to choose how many random points will be generated :

```
Sum to? 10000000
pi≈3.1413552
```

Observe how this converges very badly...

Exercise n°5 : While

1. Create and edit a new script name `testwhile.py` that let the user input integer values, marking the end of the input by typing 0 and calculate the sum, like :

```
value? 1
value? 2
value? 100
value? 0
Sum is 103
```

2. Create and edit a script name `binlen.py` that let the user input a strictly positive integer n and prints the length of its binary representation in base 2 (no leading 0). *Hint : use successive divisions by 2.* Like :

```
Value? 0
0 is not strictly positive. Value? 52361735623561
blen(52361735623561)=46
```

3. Create a script named `randmean.py` that generates random numbers in the interval $[0, 1000]$ until the mean of them is in the interval $[450, 550]$. Like :

```
Random numbers: 297, 899, 915, 118, 602, 550, 788, 365, 743, 441, 539, 653, 705, ▷
    ▷ 585, 627, 472, 396, 792, 837, 183, 646, 196, 287,
I needed 23 random numbers to get the mean 549.3913043478261.
```

4. Refine the preceding program so that the user can choose the fineness of the interval around 500.

```
Radius: 10
Random numbers: 763, 605, 441, 60, 522, 838, 31, 489, 916, 269,
I needed 10 random numbers to get the mean 493.4.
```

5. Refine the preceding program so that the same experience can be repeated n times (n chosen by the user), and so that at the end he will also know how many random number have been generated in the mean, for the n experiences to get the mean in between $[500 - r, 500 + r]$ for a given r :

```
Number of experiences? 3
Radius: 50
Random numbers: 274, 669,
I needed 2 random numbers to get the mean 471.5.
Random numbers: 286, 663,
I needed 2 random numbers to get the mean 474.5.
Random numbers: 369, 323, 311, 484, 876,
I needed 5 random numbers to get the mean 472.6.
I needed 3.0 rand numbers in the mean to get a random sequence whose mean is in ▷
    ▷ [450,550]
```

Exercise n°6 : function

1. Modify the script `randmean.py` to make the part that realizes one experience (computing the number of random numbers such that the mean is in the range) a function (name it `experience`).
2. Modify the script `randmean.py` to make the part that realizes a set of experiences (computing the mean number of random sequences...) a function (name it `mean_for_samples`) and generates all the means for a range of radii from 50 to 1. The results may looks like :

```
Number of experiences? 2000
Mean 9.3335 for range [450,550]
Mean 9.2145 for range [451,549]
Mean 9.677 for range [452,548]
Mean 10.4985 for range [453,547]
```

```
Mean 10.311 for range [454,546]
Mean 10.0115 for range [455,545]
Mean 10.2505 for range [456,544]
Mean 10.9465 for range [457,543]
...
Mean 148.4555 for range [495,505]
Mean 207.6195 for range [496,504]
Mean 363.957 for range [497,503]
Mean 611.961 for range [498,502]
Mean 1526.674 for range [499,501]
```

3. Modify the script `lessorgreater.py` so that it defines a function `test(value,hidden)` which prints an appropriate message if $h < v$, $h = v$ or $h > v$ and returns true if $h = v$, else false if not. Add a script named `game.py` that uses `lessorgreater.py` as a module so that when run an hidden value is chosen and the user is invited to input a value until it equals to the hidden one.
4. Modify the "game" so that, at the end, the user will know the number of attempts he made.

Exercise n°7 : tuples

1. Create a script named `syracuse.py`, which computes for every integer $n \in [1, 100]$ the flying time, the high fly time and the maximum altitude of its Syracuse sequence. Reminder :

   Let f the function defined over positive integers be :

$$f(n) = \begin{cases} \frac{n}{2}, \text{ if } n \text{ is even} \\ 3n+1, \text{ if } n \text{ is odd} \end{cases}$$

   The Syracuse sequence $S(n)$ is defined by :

$$S_0(n) = n$$
$$S_{i+1}(n) = f(S_i(n))$$

   The flying time of $S(n)$, $\text{ft}(n)$, is the smallest i such that $S_i(n) = 1$ (after that point the sequence is obviously degenerated to the trivial cycle 1, 4, 2).

   The maximum altitude of $S(n)$, $\text{ma}(n) = \max\{S_i(n)\}$.

   The high fly time of $S(n)$, $\text{hft}(n)$, is i such that $\forall j \in [0, i], S_j(n) \geqslant n$ and $S_{i+1}(n) < n$.

   The result of a run may looks like :

```
Syracuse of ? 15
S(15): 15, 46, 23, 70, 35, 106, 53, 160, 80, 40, 20, 10, 5, 16, 8, 4, 2, 1
ft(15)=17
ma(15)=160
hft(15)=10
```

2. Modify `syracuse.py` so that it produces the results for all n in $[1, N]$ (N inputed by the user).