Hello world!

Get into programming! In this very first activity, you will learn to manipulate numbers, variables and code your first loop with `Python`.

Lesson 1 (Numbers with `Python`).
Check that `Python` works correctly, by typing the following commands in the `Python` console:

```
>>> 2+2
>>> "Hello world!"
```

Here are some instructions to try.

- **Addition.** `5+7`.
- **Multiplication.** `6*7`; with brackets `3*(12+5)`; with decimal numbers `3*1.5`.
- **Power.** `3**2` for $3^2 = 9$; negative power `10**-3` for $10^{-3} = 0.001$.
- **Real division.** `14/4` is equal to `3.5`; `1/3` is equal to `0.3333333333333333`.
- **Integer division and modulo.**
  - `14//4` returns `3`: it is the quotient of the Euclidean division of 14 by 4, note the double slash;
  - `14%4` returns `2`: it is the remainder of the Euclidean division of 14 by 4, we also say "14 modulo 4".

Note. Inside the computer, decimals numbers are encoded as "floating point numbers".

Activity 1 (First steps).

Goal: code your first calculations with `Python`.

1. How many seconds are there in a century? (Do not take leap years into account.)
2. How far do you have to complete the dotted formula to obtain a number greater than one billion?
$$(1+2)\times(3+4)\times(5+6)\times(7+8)\times\cdots$$
3. What are the last three digits of
$$123456789\times123456789\times123456789\times123456789\times123456789\times123456789\times123456789 \quad ?$$
4. 7 is the first integer such that its inverse has a repeating decimal representation of period 6:
$$\frac{1}{7} = 0.\underbrace{142857}\underbrace{142857}\underbrace{142857}\ldots$$
Find the first integer whose inverse has a repeating decimal representation of period 7:
$$\frac{1}{???} = 0.00\underbrace{abcdefg}\underbrace{abcdefg}\ldots$$
Hint. The integer is bigger than 230!

5. Find the unique integer:
   - which gives a quotient of 107 when you divide it by 11 (with integer division),
   - and which gives a quotient of 90 when you divide it by 13 (with integer division),
   - and which gives a remainder equal to 6 modulo 7.

Lesson 2 (Working with an editor).
From now on, it is better if you work in a text editor dedicated to `Python` rather than with the console. You must then explicitly ask to display the result:

```
print(2+2)
print("Hello world!")
```

In the following you continue to write your code in the editor but we will no longer indicate that you must use `print()` to display the results.

Lesson 3 (Variables).
Variable. A *variable* is a name associated with a memory location. It is like a box that is identified by a label. The command “`a = 3`” means that I have a variable “`a`” associated with the value 3.
Here is a first example:

```
a = 3  # One variable
b = 5  # Another variable

print("The sum is",a+b)   # Display the sum
print("The product",a*b)  # Display the product

c = b**a      # New variable...
print(c)      # ... that is displayed
```

Comments. Any text following the hashtag character “#” is not executed by `Python` but is used to explain the program. It is a good habit to comment extensively on your code.

Names. It is very important to give a clear and precise name to the variables. For example, with the right names you should know what the following code calculates:

```
base = 8
height = 3
area = base * height / 2
print(area)
# print(Area)  # !! Error !!
```

Attention! `Python` is case sensitive. So `myvariable`, `Myvariable` and `MYVARIABLE` are different variables.

Re-assignment. Imagine you want to keep your daily accounts. You start with $S_0 = 1000$, the next day you earn 100, so now $S_1 = S_0 + 100$; the next day you add 200, so $S_2 = S_1 + 200$; then you lose 50, so on the third day $S_3 = S_2 - 50$. With `Python` you can use just one variable `S` for all these operations.

```
S = 1000
S = S + 100
S = S + 200
```

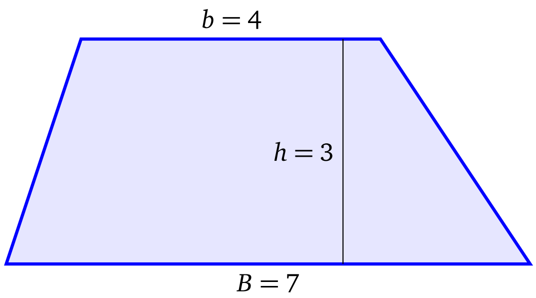
```
S = S - 50
print(S)
```

You have to understand the instruction "`S = S + 100`" like this: "I take the contents of the box `S`, I add 100, I put everything back in the same box".

Activity 2 (Variables).

Goal: use variables!

1. (a) Define variables, then calculate the area of a trapezoid. Your program should display `"The value of the area is ..."` using `print("The value of the area is",area)`.



   (b) Define variables to calculate the volume of a box (a rectangular parallelepiped) whose dimensions are 10, 8, 3.
   (c) Define a variable `PI` equals to 3.14. Define a radius $R = 10$. Write the formula for the area of a disc of radius R.
2. Put the lines back in order so that, at the end, x has the value 46.

```
(1)  y = y - 1
(2)  y = 2*x
(3)  x = x + 3*y
(4)  x = 7
```

3. You place the sum of 1000 dollars in a savings account. Each year the interest on the money invested brings in 10% (the capital is multiplied by 1.10). Write the code to calculate the capital for the first three years.
4. I define two variables by `a = 9` and `b = 11`. I would like to exchange the content of `a` and `b`. Which instructions should I use so that at the end `a` equals 11 and `b` equals 9?

```
a = b
b = a
```

```
c = b
a = b
b = c
```

```
c = a
a = b
b = c
```

```
c = a
a = c
c = b
b = c
```

Lesson 4 (Use functions).

- **Use** `Python` **functions.**
  You already know the `print()` function that displays a string of characters (or numbers). It can be used by `print("Hi there.")` or through a value:

```
string = "Hi there."
print(string)
```

There are many other functions. For example, the function `abs()` calculates the absolute value of a number: for example `abs(-3)` returns 3, `abs(5)` returns 5.

- **The module `math`.**
  Not all functions are directly accessible. They are often grouped into ***modules***. For example, the `math` module contains mathematical functions. For instance, you will find the square root function `sqrt()`. Here's how to use it:

```
from math import *

x = sqrt(2)
print(x)
print(x**2)
```

  The first line imports all the functions of the module named `math`, the next lines calculate $x = \sqrt{2}$ (as an approximate value) and then display x and x^2.

- **Sine and cosine.**
  The `math` module contains the trigonometric functions sine and cosine and even the constant `pi` which is an approximate value of π. Be careful, the angles are expressed in radians.
  Here is the calculation of $\sin(\frac{\pi}{2})$.

```
angle = pi/2
print(angle)
print(sin(angle))
```

- **Decimal to integer.**
  In the `math` module there are also functions to round a decimal number:
  - `round()` rounds to the nearest integer: `round(5.6)` returns 6, `round(1.5)` returns 2.
  - `floor()` returns the integer less than or equal to: `floor(5.6)` returns 5.
  - `ceil()` returns the integer greater than or equal to: `ceil(5.6)` returns 6.

Activity 3 (Use functions).

Goal: use functions from `Python` *and the* `math` *module.*

1. The `Python` function for gcd is `gcd(a,b)` (for greatest common divisor). Calculate the gcd of $a = 10\,403$ and $b = 10\,506$. Deduce the lcm from a and b. The function lcm does not exist, you must use the formula:
$$\mathrm{lcm}(a,b) = \frac{a \times b}{\gcd(a,b)}.$$
2. By trial and error, find a real number x that checks all the following conditions (several solutions are possible):
   - `abs(x**2 - 15)` is less than `0.5`

- `round(2*x)` returns 8
- `floor(3*x)` returns 11
- `ceil(4*x)` returns 16

Hint. `abs()` refers to the absolute value function.

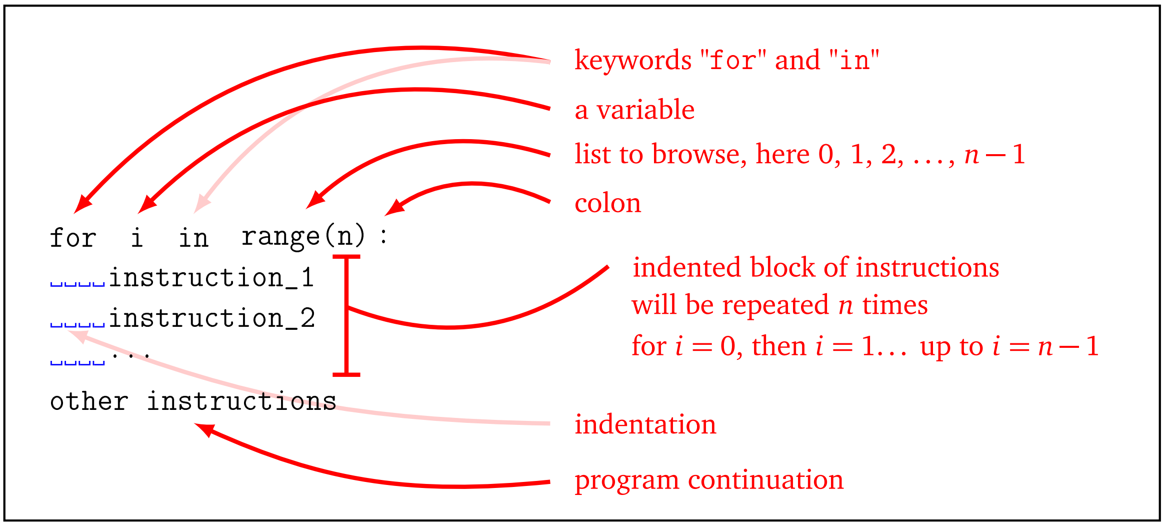
3. You know the trigonometric formula

$$\cos^2\theta + \sin^2\theta = 1.$$

Check that for $\theta = \frac{\pi}{7}$ (or other values) this formula is numerically true (this is not a proof of the formula, because `Python` only makes approximate computations of the sine and cosine).

Lesson 5 ("for" loop).

The "for" loop is the easiest way to repeat instructions.



Note that what delimits the block of instructions to be repeated is ***indentation***, i.e. the spaces at the beginning of each line that shift the lines to the right. All lines in a block must have exactly the same indentation. In this book, we choose an indentation of 4 spaces.
Don't forget the colon "`:`" at the end of the line of the `for` declaration!

- **Example of a "for" loop.**
  Here is a loop that displays the squares of the first integers.

```
for i in range(10):
    print(i*i)
```

  The second line is shifted and constitutes the block to be repeated. The variable `i` takes the value 0 and the instruction displays 0^2; then `i` takes the value 1, and the instruction displays 1^2; then 2^2, 3^2...
  In the end this program displays:

$$0, 1, 4, 9, 16, 25, 36, 49, 64, 81.$$

  Warning: the last value taken by `i` is 9 (and not 10).

- **Browse any list.**
  The "for" loop allows you to browse any list. Here is a loop that displays the cube of the first prime numbers.

```
for p in [2,3,5,7,11,13]:
    print(p**3)
```

- **Sum all.**
  Here is a program that calculates
  $$0+1+2+3+\cdots+18+19.$$

```
mysum = 0
for i in range(20):
    mysum = mysum + i
print(mysum)
```

Understand this code well: a variable `mysum` is initialized at 0. We will add 0, then 1, then 2... This loop can be better understood by filling in a table:

Initialisation: `mysum`= 0

i	mysum
0	0
1	1
2	3
3	6
4	10
...	...
18	171
19	190

Display: 190

- `range()`.
  - With `range(n)` we run the entire range from 0 to $n-1$. For example `range(10)` corresponds to the list `[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]`.
    Attention! the list stops at $n-1$ and not at n. What to remember is that the list contains n items (because it starts at 0).
  - If you want to display the list of items browsed, you must use the command:
    `list(range(10))`
  - With `range(a,b)` we go through the elements from a to $b-1$. For example `range(10,20)` corresponds to the list `[10, 11, 12, 13, 14, 15, 16, 17, 18, 19]`.
  - With `range(a,b,step)` you can browse the items a, $a+\text{step}$, $a+2\text{step}$... For example `range(10,20,2)` corresponds to the list `[10, 12, 14, 16, 18]`.

- **Nested loops.**
  It is possible to nest loops, i.e. use a loop inside the block of another loop.

```
for x in [10,20,30,40,50]:
    for y in [3,7]:
        print(x+y)
```

In this small program x is first equal to 10, y takes the value 3 then the value 7 (so the program displays 13, then 17). Then $x = 20$, and y again equals 3, then 7 again (so the program displays 23, then 27). Finally the program displays:

$$13, 17, 23, 27, 33, 37, 43, 47, 53, 57.$$

Activity 4 ("for" loop).

Goal: build simple loops.

1. (a) Display the cubes of integers from 0 to 100.
   (b) Display the fourth powers of integers from 10 to 20.
   (c) Display the square roots of integers 0, 5, 10, 15,... up to 100.
2. Display the powers of 2, from 2^1 to 2^{10}, and memorize the results!
3. Experimentally search for a value close to the minimum of the function
$$f(x) = x^3 - x^2 - \frac{1}{4}x + 1$$
on the interval $[0, 1]$.
Hints.
   - Build a loop in which a variable i scans integers from 0 to 100.
   - Defined $x = \frac{i}{100}$. So $x = 0.00$, then $x = 0.01$, $x = 0.02$...
   - Calculate $y = x^3 - x^2 - \frac{1}{4}x + 1$.
   - Display the values using `print("x =",x,"y =",y)`.
   - Search by hand for which value of x you get the smallest possible y.
   - Feel free to modify your program to increase accuracy.
4. Seek an approximate value that must have the radius R of a ball so that its volume is 100.
Hints.
   - Use a scanning method as in the previous question.
   - The formula for the volume of a ball is $V = \frac{4}{3}\pi R^3$.
   - Display values using `print("R =",R,"V =",V)`.
   - For π you can take the approximate value 3.14 or the approximate value `pi` of the `math` module.

Activity 5 ("for" loop (continued)).

Goal: build more complicated loops.

1. Define a variable n (for example $n = 20$). Calculate the sum
$$1^2 + 2^2 + 3^2 + \cdots + i^2 + \cdots + n^2.$$
2. Calculate the product:
$$1 \times 3 \times 5 \times \cdots \times 19.$$
Hints. Begin by defining a `myproduct` variable initialized to the value 1. Use `range(a,b,2)` to get every other integer.
3. Display multiplication tables between 1 and 10. Here is an example of a line to display:
```
7 x 9 = 63
```
Use a display command of the style: `print(a,"x",b,"=",a*b)`.