

### Basics

Spaces and indentation (tabs) are relevant parts of the code

Instructions in a *block of code* will have the same indentation

A *block of code* contains one or more lines of code inside it. The contained lines will be indented one more level than the container one.

A colon : opens a new *block of code* in the following line.

# at the beginning of a line marks this line as non-executable. For example

```
# This is a single line comment
```

To create multiple line comments, use three apostrophes in a row

```
'''
```

```
This is a
```

```
Multiline comment
```

```
'''
```

### Variables

There's no declaration of variables. When you use a new name in an assign sentence, that becomes a variable of the type of the value assigned to it

**a = 10** creates a variable called *a* that stores an *integer* value

The decimal separator is the point

**a = 3.14** creates a variable called *a* that stores a decimal (*float*) value

To transform a numerical value into a string you *cast* it using the function **str()**

**str(10)** will create the string value **"10"**

To transform a string value into an integer you *cast* it using the function **int()**

**int("10")** will create the integer value **10**

A variable contains a literal value of a certain type (integer, alphanumeric, decimal, boolean, etc) and it can be used to perform different operations or construct logical expressions

### Basic input / output

**print(a)** Prints the content of the variable called *a* and opens a new line

```
print("Helo world")
Hello word
and opens a new line
```

**print(a,end=",")** Prints the content of the variable called *a* and then a comma without opening a new line

```
print(name,end=",")
My name,
```

**input()** Reads a **string** value form the keyboard

**a = input()** will store in a variable called *a* the value entered by the user as a **string**

**int(input())** Reads a **string** value form the keyboard, and *casts* it into an **integer**

**a = int(input())** will store in a variable called *a* the value entered by the user casted as an **integer**

### Flow control (more to be added along the course)

**if (<expr>):** Forks the execution stream according to the logical value of the expression *<expr>*

```
if (a == b):
<do something>
```

**else:** As part of an *if .. else* block, starts the block code to be executed if the *expression* was **False**

```
else:
<do something esle>
```

**elif (<expr2>):** Compound an *else:* statement with a new *if* statement

```
elif (a<b):
<and another>
```

**while (<expr>):** Generates a *loop* that will run as long as the expression *<expr>* is **True**

```
while (a<10):
<do something>
<update a>
```



### Arithmetic operators

+	add	12 + 5 returns 17
-	subtract	12 - 5 returns 7
*	product	12 * 5 returns 60
/	decimal division	12 / 5 returns 2.4
//	division (whole numbers)	12 // 5 returns 2
%	remainder of the division	12 % 5 returns 2
**	exponentiation	12 ** 5 returns 248832

### Comparison operators (logical)

<	less than	12 < 5 evaluates as <b>False</b>
<=	less than or equal to	12 <= 5 evaluates as <b>False</b>
==	equal to	12 == 5 evaluates as <b>False</b>
>=	greater than or equal to	12 >= 5 evaluates as <b>True</b>
>	greater than	12 > 5 evaluates as <b>True</b>
!=	not equal to	12 != 5 evaluates as <b>True</b>

In a comparison, the sign of equality (=) can never be alone as it would be confused with the assignment of values ( $a = 10$ ). This is why the logical equality operator is a double sign of equality.

Therefore **a=10** means *assign the value 10 to the variable a* and **a==10** means *is the content of the variable a a number 10?*

### Lists

To be added next course

### Maths

<b>abs(arg)</b>	receives an <b>integer</b> number as an argument and returns the integer absolute value	<b>abs(-12)</b> returns 12
<b>math.fabs(arg)</b>	receives a <b>float</b> as an argument and returns the <b>float</b> absolute value	<b>math.fabs(-1-2.34)</b> returns 12.34 <b>math.fabs(-12)</b> returns 12.0
<b>math.floor(arg)</b>	receives a <b>float</b> as an argument and rounds it down to the nearest <b>integer</b>	<b>math.floor(2.5)</b> returns 2 <b>math.floor(-3.4)</b> returns -4
<b>math.ceil(arg)</b>	receives a <b>float</b> as an argument and rounds it up to the nearest <b>integer</b>	<b>math.ceil(2.5)</b> returns 3 <b>math.ceil(-3.4)</b> returns -3
<b>math.pi</b>	returns the value of Pi	<b>math.pi</b> returns 3.1415926-53589793

You will need to **import math**

### Logical operators

<b>&lt;expr1&gt; and &lt;expr2&gt;</b>	<b>True</b> if and only if the two expressions are <b>True</b>	(a<0) <b>and</b> (a<5)
<b>&lt;expr1&gt; or &lt;expr2&gt;</b>	<b>True</b> if and only if at least one of the two expressions is <b>True</b>	(a<0) <b>or</b> (a>=5)
<b>not &lt;expr&gt;</b>	<b>True</b> if and only if <expression> is <b>False</b>	<b>not</b> (a==0)

By *logical* we understand an expression or operation that can only take two different values: **True** or **False**

