Python:

Notes from CS Dojo.

Vocabularies:

Type of data: string, number, integer, boolean, list, dictionary, list of dicts, datetime

A **string** : characters put between “ “ or ‘ ‘.

Ex1: the script: print(“1”) will give the output 1 and write the exemple Ex1 as:

In [1]: print(“1”)

Output (after clicking the Run button): 1

**Variable** and **value**:

The script: a = 1 may be interpreted as” the value 1 is assigned to the variable a”. More precisely, a variable is like a name tag: “the variable a refers to the value 1”.

Ex2: The two-line code

[3 lines]: a = 1

print(a)

Output: 1

Ex3:

[2 lines]: a = “hello”

print(a)

Output: hello

Ex4: a variable can be assigned to another one.

Functions:

A function is a collection of instructions

To create a function, use the command def and follow the rule as follows:

[3 lines]: def func12():

print(aa)

print(aaa)

The instructions which constitute the function func12 have to be aligned.

The function will not be executed/run unless it is called as follows

[3 lines]: def func12():

print(aa)

print(aaa)

func12()

Output (in 2 lines) : aa

aaa

NB: We do not need to use the command print.

A mapping or function with argument or input

Ex6:

[2 l]: def fun(x):

return 2\*x

x is called input or argument;

A variable a which calls the function fun with a given value of the argument (e.g. 3) will be assigned the corresponding return value (2\*3 in the example).

Ex7:

[4 l]: def fun(x):

return 2\*x

a = fun(3)

print(a)

Output: 6

Ex8:

[3 l]: def fun(x):

return 2\*x

b = fun()

Output: Error

Ex9: function with multiple arguments

[4 l]: def funny(x, y):

return 2\*x + y

a = fun(3, 1)

print(a)

Output: 7

Ex10: combining instructions and mapping

[6 l]: def fu(x):

print(x)

print(“hello”)

return 2\*x

a = fu(3)

Output (in 2 lines): 6

hello

Ex10: combining instructions and mapping

[6 l]: def fu(x):

print(x)

print(“hello”)

return 2\*x

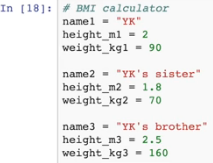
a = fu(3)

[1 l]: print(a)

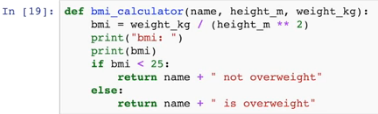
Output: 6

Ex11: Construction of a Body Mass Index calculator

11.1/ Introduce the variables and values associated to the parameters :



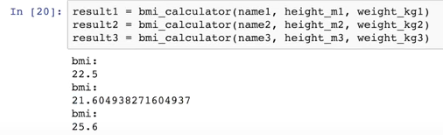
11.2/ define the function:



11.3/ construct the code for a particular task or result-variable:



11.4/ run the result-code in order to obtain the value of the result-variable:

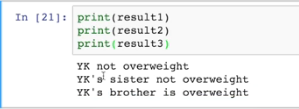


11.5/ construct the code for the qualitative-output associated to the value of

the result-variable:



11.6/ run it in order to display the qualitative-output.



List:

It is used to store a list of things (similar to array in Java)

Exl1 : The list is assigned to the variable a and we write:

[1 line]: a = [3, 10, -1]

Run the above code will define the list, and printing it gives:

[1 line]: print(a)

Output: [3, 10, -1]

Exl2 : To add a new item (e.g. ‘hello’) to the list, we use the predefined function ‘**append**’ with argument the new item, and write:

[2 lines]: a.append(1)

print(a)

Output: [3, 10, -1, hello]

Other predefined functions exist the list datatype, like the function pop() without argument, which delete the last item in the list.

[1 line]: a.pop()

print(a)

Output: [3, 10, -1]

To retrieve an item from a list: use the constructor [] with argument the index of the item to be retrieved. NB: the first item is indexed 0.

[2 lines]: a[1]

print(a)

Output: 10

To change the value of an item: use the constructor [] with argument the index of the item to be changed, and assign to the variable a[] its new value

[2 lineq]: a[0] = 222

print(a)

Output: [222, 10, -1]

Dictionary.

A table with keys and their associated values (also key-value pairs)

Start with an empty dict

[1 line]: d= {}

Then create the key-value pair

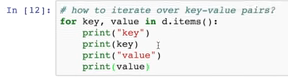
[3 lines]: d[“Tom”] = 24

d[“Georges”] = 18

d[“Paul”] = 19

Keys can be strings or numbers

To iterate over key-value pairs, use the loop command for:



One can append a number, a string, or another list on an existing list.

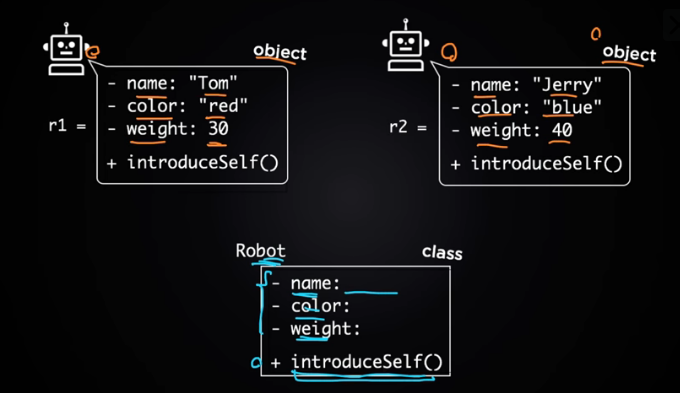
Classes and objects:

A class is a collection of attributes or Instance variables and a given function.

A class can deals with objects.

An object is a collection of variables with given values, and a function call.

An object can be put/assigned to a variable.



Ex14: Create a class called Robot:

14.1/ the created class has its first letter capitalized;

14.2/ the function inside the class is called **method** of the class

14.3/ the word ‘self’ (the one followed by .name) is associated to objects related to the class

(it is like the ‘this’ in Java). It is also called the object.

This ‘self’ would be the argument of the method (actually, of any method in the class) and we have:



14.4/ when the above code is runned, the class Robot will be created.

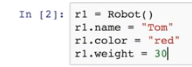
Ex15: create an object associated to the class Robot

15.1/ Use a variable which refers to the object:



Here the variable r1 refers to the object Robot() using the default Python constructor ‘()’ for the class robot

15.2/ add the attributes on the object:



Until step 15.3, we will only need the attribute name;



Run the code in order to create the object

15.2/ We can call the method on the created object by the code:



15.3/ running the above code will give:



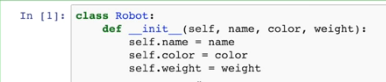
**Ex16**: improve the class code by adding a **custom constructor** instead of the default Python constructor.

A constructor is written with its name surrounded by double underscore: Ex: \_\_init\_\_

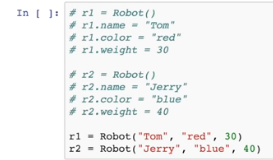


A constructor is a function whose arguments we want to set for the attributes of particular objects we want to create. The attributes are introduced by ‘self.’.

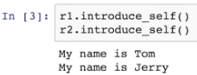
One can identify the argument’s name of the constructor with the name of the associated attribute.



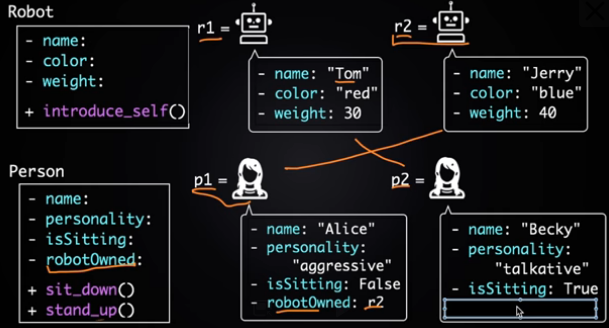
Now, the object assigned to r1 can use the constructor defined in the class, and we can rewrite the attribution 15.2/ as:



We can call the function introduce\_self() on the created objects and run it to obtain:



**Another example**: Two classes in interaction



Class : Robot

Objects related to Robot: r1 and r2

with qualitative (string-like) and numerical attributes

and introduce\_self() method

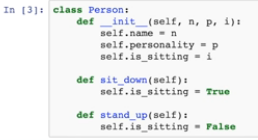


Class: Person

Objects related to Person: p1 and p2

with qualitative and quantitative (number, boolean) attributes

and two methods: sit\_down() and stand\_up().



An object is an image of the class for a particular value of the instance variable, e.g.



The attributes act on the objects from ahead as:



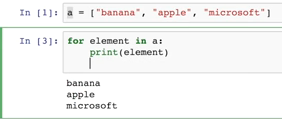
(here, the attribute ‘robot\_owned’ acts on the object p1 and the associated image is r2).

The methods act on the objects from behind as:

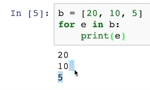


For loops:

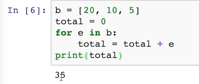
To iterate a list by the for loop:



Or simply (instead of ‘element’, one can use any other word (e.g. ‘e’)



The for loop can compute the total in a list of numbers:

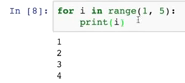


NB: The initialization is put before the loop

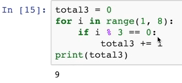
The range function range(, ) create a serie of integers starting from the first argument to the last one minus 1, and when it is put as argument of the list function list(), we obtain a list:



To iterate a range by the for loop:



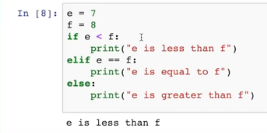
To use if condition:



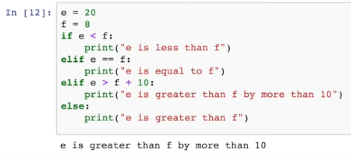
Note that the double equal is a conditional equality not an assignment of value;

The percentage is just the reminder function which gives the reminder of the division of the first argument with rapport to the second one.

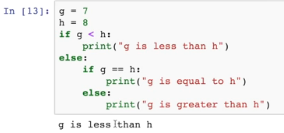
Like the def clause, the if and the else clauses form each of them a block.

The command elif means else if. I.e. another if condition, and the else command is the last condition. The if clause is one condition, else condition is another one, and else is the complementary condition of all above listed conditions.

Or



One can also explicit the complementary condition in the else clause:



Data visualization:

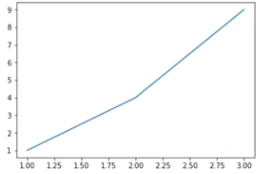
Start by import needed modules already included in the machine and to give them nickname.



Ex: Plot a table of (x,y)-values



The function plt.plot acts on lists or multi-lists and when followed by plt.show(), the execution of the code will show the graph:



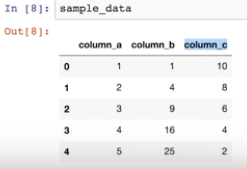
Load an data file (e.g. sample\_data.csv) by the module pandas:.



The loaded data is assigned to the variable on the left-hand.

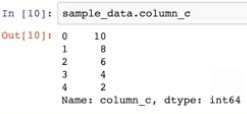
The pre-defined function/command .read\_csv() acts on the module and has as argument the name of the datafile.

One can check the data on the notebook file by running the data variable:



One can check the type of the data variable by using the command type() with data variable as argument. Type of datas: dataframe, series, etc.

Retrieve some info from the data variable:



NB: the attribute ‘column\_c’ is the attribute name used to build this particular column of the data.

More retrieve:

Ex1: retrieve the item of index 0 in the column c:



Ex2: retrieve the data with attribute ‘country’ having the value ‘United States’ from ‘data’::



Ex3: retrieve the ‘year’ and ‘population’ attributes from the data ‘us’:

Modules imported by Corona\_germany\_simple\_model.ipynb:

**Pandas** is a software library written for the Python programming language for data manipulation and analysis. In particular, it offers data structures and operations for manipulating numerical tables and time series. (Wiki)

**“NumPy** is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays.” (Wiki)

**“Matplotlib** is a plotting library for the Python programming language and its numerical mathematics extension NumPy.” (Wiki)

***matplotlib*.*pyplot***: “pyplot is mainly intended for interactive plots and simple cases of programmatic plot generation” (matplotlib.org)

“**SciPy** is a free and open-source Python library used for scientific computing and technical computing. SciPy contains modules for optimization, linear algebra, integration, interpolation, special functions, FFT, signal and image processing, ODE solvers and other tasks common in science and engineering.” (Wiki)

**Scipy.stats**. ”This module contains a large number of probability distributions as well as a growing library of statistical functions.” (Scipy.org)

**Pickle**: “To serialize an object hierarchy, you first create a pickler, then you call the pickler’s [dump()](https://docs.python.org/2/library/pickle.html#pickle.dump) method. To de-serialize a data stream, you first create an unpickler, then you call the unpickler’s [load()](https://docs.python.org/2/library/pickle.html#pickle.load) method.” (docs.python)

Read a comma-separated values (csv) file into DataFrame. **Sep:** str, default ‘,’

Delimiter to use.

**PyMC3** is a Python package for Bayesian statistical modeling and probabilistic machine learning which focuses on advanced Markov chain Monte Carlo and variational fitting algorithms. It is a rewrite from scratch of the previous version of the PyMC software. (Wiki)

**Theano** is a Python library and optimizing compiler for manipulating and evaluating mathematical expressions, especially matrix-valued ones. In Theano, computations are expressed using a NumPy-esque syntax and compiled to run efficiently on either CPU or GPU architectures. (Wiki)