Lab 1: Introduction to Python[1] (Week 1.1)

1 – Objetives

With this work the student should be able to start using (or refresh their knowledge) of the Python language. In this lab some examples of python programming will be presented.

2 – The development Environment

In the CI4IoT Labs, we will be using the Python language due to the versality, easy learning curve and availability of many libraries in the field of Data Science.

You are free to use any IDE you are familiar with. I advise using PyCharm, Eclipse or Jupyter Notebook.

The Jupyter Notebook has a different interface than a classical IDE but allows for easy experimentation with small code and is very popular for programmers in this area.

The use of libraries in Python allows the acceleration of development in projects. They can be used as long it is not explicitly said to implement that functionality.

3 – Python tutorials

There are many Python tutorials, we leave here some examples:

- Python website – https://docs.python.org/3/tutorial/
- Google for Education – https://developers.google.com/edu/python/
- Numpy – https://docs.scipy.org/doc/numpy/user/quickstart.html

In case you are unfamiliar with Python programming, please follow at least the first tutorial until next class.

4 – Execution

In order to get acquainted with Python, run the next code examples (using the command line interpreter).

Remember that Python uses indentation to delimit blocks of code.

[1] Rui Neves, João Paulo Carvalho

4.1 – If Else

"If-else" is the more common control flow statement. Notice that there is a ":" at the end of every statement. Example:

```
# If elif else example
x = 10
if x < 0:
    x = 0
   print('Negative changed to zero')
elif x == 0:
   print('It is Zero')
elif x == 1:
   print('It is One')
else:
   print('It is more than One')
```

4.2 – For and Range

The "for" statement in Python differs a bit from what you may be used to in other languages such as C or Pascal. Python's "for" statement iterates over the items of any sequence (a list or a string), in the order that they appear in the sequence.

```
# Count the letters in some strings:
words = ['car', 'skate', 'bicycle']
for w in words:
    print(w, len(w))
```

The "range" function is commonly used in python to iterate over a sequence of numbers. It has 3 parameters, first point, last point (not included), and step of the sequence.

```
#for range example
for i in range(1, 10, 2):
    print(i)
```

4.3 – Functions

Functions start with the "def" keyword. For example, we can create a function that writes the Fibonacci series to an arbitrary boundary:

```
def fib(n):    # write Fibonacci series up to n
    """Print a Fibonacci series up to n."""
    a, b = 0, 1
    while a < n:
        print(a, end=' ')
        a, b = b, a+b
    print()
```

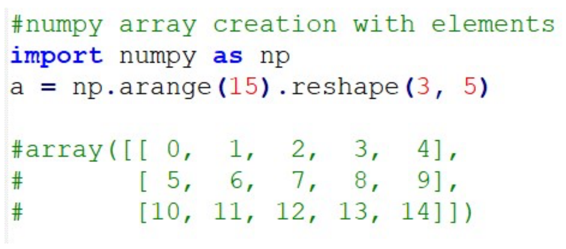
4.4 – Dictionaries

A dictionary consists of a collection of key-value pairs. It is useful to store rapidly some values with a key and a respective value. Some code examples are presented next to create a dictionary with several elements from the beginning, to create a dictionary starting empty and adding several elements, and for iterating in a dictionary.

```
## Create a new Dictionary
thisdict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
print(thisdict)

 ## Can build up a dict by starting with the the empty dict {}
 ## and storing key/value pairs into the dict like this:
 ## dict[key] = value-for-that-key
 dict = {}
 dict['a'] = 'alpha'
 dict['g'] = 'gamma'
 dict['o'] = 'omega'

 print dict  ## {'a': 'alpha', 'o': 'omega', 'g': 'gamma'}

 print dict['a']     ## Simple lookup, returns 'alpha'
 dict['a'] = 6       ## Put new key/value into dict
 'a' in dict         ## True
 ## print dict['z']                ## Throws KeyError
 if 'z' in dict: print dict['z']   ## Avoid KeyError
 print dict.get('z')  ## None (instead of KeyError)
## By default, iterating over a dict iterates over its keys.
## Note that the keys are in a random order.
for key in dict: print key
## prints a g o

## Exactly the same as above
for key in dict.keys(): print key

## Get the .keys() list:
print dict.keys()  ## ['a', 'o', 'g']

## Likewise, there's a .values() list of values
print dict.values()  ## ['alpha', 'omega', 'gamma']

## Common case -- loop over the keys in sorted order,
## accessing each key/value
for key in sorted(dict.keys()):
  print key, dict[key]

## .items() is the dict expressed as (key, value) tuples
print dict.items()  ##  [('a', 'alpha'), ('o', 'omega'), ('g', 'gamma')]

## This loop syntax accesses the whole dict by looping
## over the .items() tuple list, accessing one (key, value)
## pair on each iteration.
for k, v in dict.items(): print k, '>', v
## a > alpha    o > omega     g > gamma
```

4.5 – Numpy arrays

Python has a library to create arrays called numpy arrays. First you need to import the library and give it a name (np in this case). After that you can use the functions from the library. For more examples see the tutorial. Find out what the following achieves: s = a[: , 1:3]



```
#numpy array creation with elements
import numpy as np
a = np.arange(15).reshape(3, 5)

#array([[ 0,  1,  2,  3,  4],
#       [ 5,  6,  7,  8,  9],
#       [10, 11, 12, 13, 14]])
```

4.5 – Strings and Lists

If you still have time you can learn many more things in the tutorials, start by trying Strings and Lists...