A brief introduction to Python

E. Busato

1 Before you start writing your code

Before you start writing your code, import the packages you'll need. The packages we'll need in the exercices are imported like this:

```
%matplotlib inline
import math
import numpy as np
import scipy.stats as stats
import matplotlib.pyplot as plt
import pandas as pd
```

2 Create a variable or a list of variables

- Variable creation:

  ```
  x = 3
  ```

  This creates a variable `x` and sets its value to 3.

- List creation:

  ```
  x = [1, 2, 3]
  ```

  This creates a list `x` and sets its components to 1, 2 and 3.

- Remarks:

  – Lists are often used to store samples of random variables.

  – The number of components of a list can be obtained using the `len` function. For example, to get the length of the vector `x` created above, use:

  ```
  n = len(x)
  ```

  – In the previous examples, we considered a variable and a vector with numeric values. One can also define variables and vectors of strings and booleans. For example :

```
x = ["this", "that"]
x = [True, False, True, True]
```

3 Printing values

- In order to print values of variables or lists, you can use the `print()` function:

```
print(x)
```

- For a more powerful and flexible printing, one can use the `print()` function using the string formatting syntax. For example, supposing you have two variables `x` and `y`, you can print them as follows:

```
print("x=%.2f; y=%.2f"% (x, y))
```

4 Drawing histograms

- Suppose you have a sample stored in a vector `x`. In order to draw the histogram of `x` you can do the following:

```
plt.hist(x, bins=100, range=[0,100])
```

5 Drawing 2D datasets

- Suppose you have a 2-dimensional dataset stored in 2 python lists `x` and `y`. In order to draw `y` as a function of `x`, do:

```
plt.plot(x, y)
```

6 Drawing a function

- For example, to draw the normal distribution, do

```
pts = np.linspace(xmin, xmax, n)
plt.plot(pts, stats.norm.pdf(pts, loc=mean, scale=sigma))
```

The first line generates a list of `n` values from `xmin` to `xmax`. These values are then used in the second line as the abscissa values. For the ordinate values, one uses the `stats.norm.pdf` function which returns the p.d.f. value.

7 Reading csv file

- In order to read csv files, you can do the following: `python df = pd.read_csv('file.csv')`

  This line returns a so-called dataframe. dataframes are advanced structures used in python to work with datasets. In our exercices we'll stick to the simple python lists or numpy arrays as datasets holders. In order to get the list of values for the column named 'x' in the csv file, just do

```
x = df['x'].tolist()
```

8 Generate random variables

- In order to generate an array with 10 uniform random variables between 0 and 1, do:

```
x = np.random.uniform(0,1,10)
```

  This will automatically create an array with 10 components whose values are numbers drawn from a uniform distribution.

- The same syntax can be used to generate random variables distributed according to other common distribution by replacing `uniform` by the appropriate function. For example, to generate a normal random variable, do:

```
x =  np.random.normal(size=1000, loc=mean, scale=sigma)
```

- Note that `x` is a numpy array and not a list.

9 For loops

- In order to perform a for loop, you can use the following syntax. For example:

```
n=10
x = np.random.normal(loc=0, scale=1, size=n)
print("sample:", x)
for i in range(0, n):
    print("x[%i] = %f" % (i, x[i]))
```