Digital Technology

Python – Functions, Pandas, MatPlotLib

POLITECNICO
MILANO 1863

Edoardo Ramalli

edoardo.ramalli@polimi.it

Functions

A function is a reusable piece of code.

The code inside a function is executed only when the function is called.

A function can accept some data to work with, called parameters.

A function can also return an expression to the callee function.

When the execution of a function is terminated the control flow continues, as usual, after the call of the function.

Function – Definition

To define a function in Python we use the keyword def.
Then follows the name of the function.
After that you can define as many parameters you prefer (0 or more).

```
def function_name (parameter1, parameter2, …)  # Remember the : at the end
        # Function Body
        # parameter1, parameter2, … are like variables in the body of the function
        return <Expression>
```

A function define a scope variable, i.e. the variables that are defined inside the function are not visible outside the function itself.

Function – Call

To call, to invoke a function:

```
# the result of the function is stored inside a variable
result_var = function_name(parameter1, parameter2, ⋯)

# the result of the function is not stored
function_name(parameter1, parameter2, ⋯)
```

Function – Default parameters

Sometimes a function can accept one or more parameters but some of them could be optional. In case like this we can define default arguments with =

```
def add (num1, num2):
        return num1 + num2

add (1, 5) → 6

def add (num1, num2=5):
        return num1 + num2


add (1, 6)    → 7

add (1)       → 6
```

If you provide the optional parameter(s), the default ones are overwritten.
The default parameters can be defined only at the end of the parameter list.

Function – Multiple return

A function can returns 0, 1 or more expression. To return an expression we use the keyword return followed by a variable number of expressions separated by ',' .
Actually a function always returns a single data type when we want to return multiple expressions, that is a tuple.
The tuple can store inside itself multiple expressions.

```
def my_function (…):
        a = 5
        b = 10
        …
        …
        return a, b
```

first_result, second_result = my_function (…) → first_result=5, second_result=10

print(type(my_function (…))) → print <class 'tuple'>

Built-In

- print → print to console a variable(s)
- input → Collect a user input from console and returns a string data type
- len → Return the length of an iterable (list, set, dict, tuple)
- type → Return the type of a variable
- max → Return the biggest element from an iterable
- min → Return the smallest element from an iterable
- sum → Return the sum of elements inside an iterable
- round → Return the round of a number
- bool → Casting to bool
- int → Casting to int
- float → Casting to float
- dict → Casting to dict
- list → Casting to list
- set → Casting to set
- sorted → Return a new iterable that is sorted

Module

You can write many functions in a file (called module) and use them in another script by importing them. If you have a function called ‘average’ in a file ‘myfunctions.py’, you can use it in another script with:

```
import myfunctions
print(myfunctions.average([1,2,3]))
```

Module

Python comes with a standard library, offering a wide variety of facilities:
https://docs.python.org/3/library/

For instance, we have the math module for the sqrt function:

```
import math
print(math.sqrt(9))
```

For statistics stuff (avg, median, stdev, etc..) instead we have:

```
import statistics
print(statistics.stdev([1,2,3,4]))
```

We will add other modules to python with pip and https://pypi.org/.

Install others moduls

We have seen how to import a module from the Python standard library:

```
import math
from random import uniform
```

However, there exists many modules which are not included in the standard library (numpy, pandas, scipy, scikit-learn, etc...)
You can install, upgrade, and remove packages using a program called pip:

```
pip install pandas
pip uninstall pandas
pip freeze
```

Pandas

Pandas is a data analysis and manipulation library.
Pandas allows you to open tabular data (like csv, spreadsheet and databases) and obtain Python objects called DataFrame.

https://pandas.pydata.org/docs/getting_started/index.html

Pandas – DataFrame

Let's manually create a DataFrame:

```
# Instead of using every time 'pandas' you can define an abbreviation with the keyword 'as'

import pandas as pd
my_df = pd.DataFrame(
        {"Name": ["Alice", "Bob", "Charlie", "Tom"],
        "Age": [25, 30, 35, 27] })
print(my_df)
```

Each column is a pandas Series.
You can select a column using square brackets:

```
print(my_df['Age'])
```

Pandas – DataFrame filtering

Let's keep only the rows of dataframe table when the age >= 30

```
print(my_df[my_df["Age"] >= 30])
```

Now we would like to add a column in the dataframe.
This column reports the age of the people present in thedataframe, in 50 years.

```
my_df["Age_50"] = my_df["Age"] + 50        # The new column is Age_50
```

The new column is built in this way: for each row, we take the corresponding value under the column Age, and we add 50 to the value and store the result under the column Age_50 in the corresponding row.

In similar way we can perform operations between the column of the dataframe.

```
print(my_df["Age_50"] - my_df["Age"])
```

Pandas – Import/Export a Dataframe

You can save a DataFrame to any of the format supported by Pandas:

```
import pandas as pd
df = pd.read_csv("apple.csv")
df.to_excel("./apple.xslsx")
```

We will see more on pandas in the next lessons

MatPlotLib

Matplotlib is a comprehensive library for creating static, animated,
and interactive visualizations in Python.
The most used module of Matplotlib is PyPlot, which makes Matplotlib work like MATLAB. Each pyplot function makes some change to a figure,
and various states are preserved across function calls, so that it keeps track of things like the current figure and plotting area. (=> Not a RESTful API)

To install MatPlotLib:

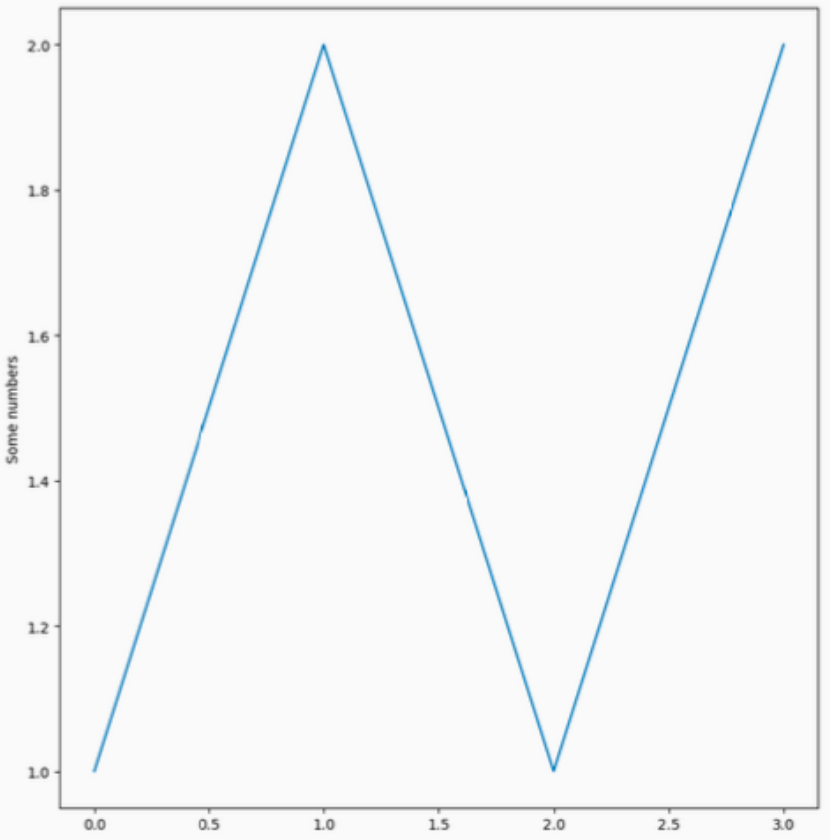
```
pip install matplotlib
```

https://matplotlib.org/stable/users/installing.html#

https://matplotlib.org/stable/gallery/index.html

MatPlotLib

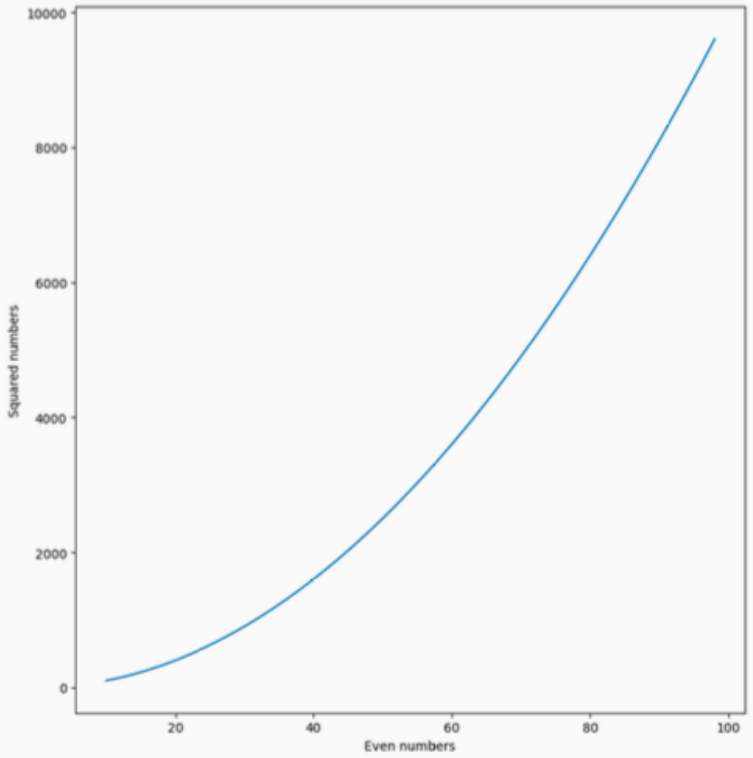
```
import matplotlib.pyplot as plt
plt.plot([1,2,1,2])
plt.ylabel("Some numbers")
plt.show()
```



If you provide a single list or array to the plot() command, matplotlib assumes it is a sequence of y values, and automatically generates the x values for you.

MatPlotLib

```
import matplotlib.pyplot as plt
xs = range(10,100,2)
ys = [x ** 2 for x in xs]
plt.plot(xs, ys)
plt.xlabel("Even numbers")
plt.ylabel("Squared numbers")
plt.show()
```



If you give 2 arrays you can plot x versus y (scatter plot).

Homework

Write a function such that accepts a dataframe of a stock with columns:
Date, Open, High, Low, Close, Volume, Dividends, Stock Splits.

Each row of the dataframe represents the opening price, the closing price, etc., of a specific stock in a specific date reported under the column 'Date'.

The function should returns the biggest rise date an the biggest drop date, in percentage variation, between all the stock history available.

The csv containing the data information is available on Beep in the Python Class 3 folder.

Percentage variation = $\frac{new-old}{old} * 100$

It is not mandatory. It is not graded.