IT: Introduction to Python

Nicholas Sedlmayr*

Institute of Physics, Maria Curie-Skłodowska University,

Plac Marii Skłodowskiej-Curie 1, PL-20031 Lublin, Poland

(Dated: January 24, 2022)

CONTENTS

I. PYTHON

A. The Basics

Python is often already installed on Macs and PCs, to find out which version is installed type

```
python --version
```

in the command line/terminal. Python can be downloaded and installed for free from the website https://www.python.org. To run a python file simply type into the command line

```
python nameoffile.py
```

Note the file extension for the python file! The file "nameoffile.py" can be edited in any text editor, however a dedicated text editor such as Atom or Emacs will help the readability of the file. The traditional file to start with just contains

```
print("Hello, World!")
```

To execute very short command we do not need a file, we can use the command line. Simply type

```
python
```

Then one can type any python command directly into the command line. To exit type

* e-mail: sedlmayr@umcs.pl

```
exit()
```

A Python file has the extension *".py"* and can be edited by any text editor. Indentation is important in Python as it is used to indicate a block of code. Using incorrect indentation will result in an error message. The number of spaces used for each indentation is not important, but must be consistent, and the tab key is probably the most convenient to use (and is easily readable). As an example let us create a file called *example.py*:

```
x=1
if x < 2:
     print("x is less than two.")
elif x>2:
     print("x is greater than two")
else:
     print("x is equal to two")
```

Here we also see the structure of an if statement. To run this we type

```
python example.py
```

into the command line/terminal. Trying this with different indenting will result in error messages. We can define a function called test which does this:

```
# Here we define a function called test
def test(x):
     if x < 2:
          print("x is less than two.")
     elif x>2:
          print("x is greater than two")
     else:
          print("x is equal to two")
# Now we can call this function
test(1)
```

Note the multiple indenting! Comments have also been added. The program will ignore everything after #. Including comments explaining what your code is doing is good practice.

To include stings into a print command (and other things) we can use

```
x=1
y=2
print("x is equal to %s and y is equal to %s" % (x,y))
```

Each %s is replaced by the string listed at the end in order.

To sort a list:

```
guests = ['Mary', 'John', 'Peter', 'Michael', 'Jennifer']
guests.sort()

print(guests)
```

Importing files:

```
import csv
parameter=1
with open("filename%s.csv" % (parameter)) as test1:
    data = np.loadtxt(test1, delimiter=",")
```

This will import a csv file called filename1.csv and create an array called data with the entries from the file.

Plotting can be done using pyplot:

```
from matplotlib import pyplot
import csv
import numpy as np

# First let us import some data
with open("filename1.csv") as test1:
    data = np.loadtxt(test1, delimiter=",")
with open("filename2.csv") as test1:
    data2 = np.loadtxt(test1, delimiter=",")

# Our x axis we can define by hand.
# The follwing will create an array with entries
# 0, 0.01, 0.02, ... 5.
x = np.arange(0, 5, 0.01)

# Now to make the plot
pyplot.plot(x,data,'ro',label='Data 1')
pyplot.plot(x,data2,'r-',label='Data 2')
pyplot.legend()
pyplot.xlabel('t [s]')
pyplot.ylabel('h [m]')
pyplot.title(r'Plot Title')
pyplot.savefig('Figure.pdf')
pyplot.show()
```

Here we plot two sets of data called data and data2 against x. Therefore x, data, and data2 are all arrays. We also add labels and modify the plot markers.

We also need to know how to find a fit:

```
import csv
import numpy as np
from scipy.optimize import curve_fit

# First let us import some data to be fitted
with open("filename1.csv") as test1:
    data = np.loadtxt(test1, delimiter=",")

# Our x axis we can define by hand.
# The follwing will create an array with entries
# 0, 0.01, 0.02, ... 5.
x = np.arange(0, 5, 0.01)

# We define a function with variable t and parameters aa,bb,cc
def func(t, aa, bb, cc):
     return aa * t**2 + bb * t + cc
```

```
# Now we can fit
popt, pcov = curve_fit(func, x, data)
data2 = func(x, *popt)
```