Foreword: Scripts are supposed to be written in python and previously written py-scripts can be used as starting point. You can use as well jupyter notebook.

Auto- and Cross-correlation with `python`

1 Auto-correlation

1. Create a list called `u`, such as

   ```
   u = np.array([0.,0.,0.,0.,0.25,0.5,0.75,1.0,0.75,
                 0.5,0.25,0.,0.]).astype(float)
   ```

   Don't forget to import the appropriate modules for the following operations:

   ```
   from scipy import signal
   import matplotlib.pyplot as plt
   import numpy as np
   ```

2. Plot the signal on a figure.

3. Use the function `signal.correlate` to compute the auto-correlation and store the result in `cuu`. Plot the result in a new graph, although it's worth working with `subplot` function.

4. Answer to the following questions:

   - What's the size of `u`?
   - What's the size of `cuu`?
   - What's is explanation (in clear words) for such a result?

2 Cross-correlation

1. Create two lists called `u` and `v`. These data sets are two triangles shifted by 3 points.

   ```
   u = np.array([0.,0.,0.,0.,0.25,0.5,0.75,1.0,0.75,
                 0.5,0.25,0.,0.]).astype(float)
   v = np.array([0.,0.25,0.5,0.75,1.0,0.75,0.5,0.25,
                 0.,0.,0.,0.,0.]).astype(float)
   ```

2. Plot the two signals on the same figure.

3. Use the function `signal.correlate` to compute the cross-correlation and store the result in `ccuv`. Test the different options of the `mode` argument and plot the result `ccuv` on another graph. It's possible to save the figure with the command `plt.savefig("xcorr.png")`.

4. Analyze the results and test the switch between `u` and `v` in the argument order of function `signal.correlate`.

5. Copy the following lines and explore various possibilities by changing `dx` or by multiplying `v` by a given factor (2, 0.1, −0.5 for instance).

```
from scipy import signal
import matplotlib.pyplot as plt
import numpy as np

n=13 ; dx=1.0
x1=0.0 ; x2=n*dx
x = np.arange(x1,x2,dx)
u = np.array([0.,0.,0.,0.,0.25,0.5,0.75,1.0,0.75,
              0.5,0.25,0.,0.]).astype(float)
v = np.array([0.,0.25,0.5,0.75,1.0,0.75,0.5,0.25,
              0.,0.,0.,0.,0.]).astype(float)

ccuv=signal.correlate(u,v,mode='full')

plt.figure(num=1) ; plt.subplot(311) ; plt.xlabel('x')
plt.subplots_adjust(top=0.95, bottom=0.1, left=0.12,
                    right=0.95, hspace=0.25, wspace=0.4)

plt.plot(x,u,'ro-',label='u') ; plt.plot(x,v,'bo-',label='v')
plt.legend()

x1=-(n-1)*dx ; x2=n*dx
print(x1,x2)
x = np.arange(x1,x2,dx)
plt.subplot(312) ; plt.xlabel('lag') ; plt.ylabel('ccuv')
plt.plot(x,ccuv,'o-')
plt.savefig("xcorr.png") ; plt.show()
```

6. Using the script written in section 1, insert in this code the computation of the normalized cross-correlation.

3 Sine function

1. Start a new python script and use the function `np.arange` to create a time axis, called `t`. The time range is between 0 and 5 s with a sampling rate of 100 Hz.

2. Create the list called `s` such as

```
s = np.sin(2 * np.pi * t)
```

3. Compute the auto-correlation of a sine function using the function `signal.correlate`, plot the result as a function of delay time (s) and analyze it.

4. Explore what is the auto-correlation of `s2 = 2.+np.sin(2 * np.pi * t)`

5. Compute the cross-correlation of `s` and `u = np.sin(np.pi * t)`, and analyze the result.