**Translating MATLAB to PYTHON- CHEAT SHEET**

*The Python scripts in COMP0043 utilize the following Libraries:*

|  |  |
| --- | --- |
| Library/Package Name | Description |
| import numpy as np | Numpy has majority of MATLAB’s functionalities such as: meshgrid, linspace, exp, trapz, pi, etc. |
| import pandas as pd | Pandas is used to convert arrays/matrices into dataframes and leverage functions that numpy lacks in. Eg, ability to derive a mean per time step. |
| from scipy.stats import \* | Scipy.stats has all of the statistic functions, such as log.norm, norm, ncx2 and expon. Whenever \* is imported, that means that all modules in that library are simultaneously imported (ie no need to import one at a time). |
| from pytictoc import TicToc | Pytictoc replicates the use of tic and toc to measure the computational time for a cell to run in a Jupyter Notebook. |
| import matplotlib.pyplot as plt  from Mpl\_toolkits.mplot3d import Axes3D  from Matplotlib import cm | All Matplotlib packages were used to chart 2D and 3D figures.  Cm module is used to determine the color scheme for 3D charts. |
| import Quantlib as ql  from QuantLib import \* | Quantlib provides different pricing engines that can be used as an analytical solution to pricing European Options. |
| from scipy.fft import fft,ifft,fftshift | Scipy.fft replicates all of the functions used to derive the Fourier Transform Method. |

*Basic rules where MATLAB and Python differ in:*

|  |  |  |
| --- | --- | --- |
| Objective | MATLAB | PYTHON |
| Define parameters | mu=0.4;  sigma=0.2;  or  mu=0.4; sigma=0.2; | mu=0.4  sigma=0.2  Or  0.4,0.2=mu, sigma |
| Specifying 1st and last value in an array  (All lists in Python start as position 0) | Array1(1);  Array1(end);  for a range:  Array1(1:100) | Array1[0]  Array1[-1]  for a range:  Array1[0:99] |
| Specifying an array range based on a calculation  (Python requires integer () to be used as a calculation always derives a float) | Array1(10/2:end) | Array1[int(10/2):] |
| Indicating a value is to the power of another | 10^2 | 10\*\*2  Or  Np.power(10,2) |
| Multiplying matrices | A.\*B | Np.dot(A,B) |
| Imaginary unit | z=1+2i | z=1+2j |
| Create a comment row | %This is a comment | #This is a comment |
| Referencing a dynamic value in a static text string | sprintf(‘Parameters include mean=%.2f and sigma=%.2f’,mu,sigma) | print(‘Parameters include mean={} and sigma={}’.format(mu,sigma)) |
| Creating a grid | linspace(-1,1,100)  Or  Start:timestep:end | np.linspace(-1,1,100)  Or  np.arange(start,end,timestep) |
| Taking a mean across each timestep | mean(array) | array=pd.concat([pd.Series(x) for x in array],axis=1)  list(array.apply(lambda row: np.mean(row),axis=1)) |

*Functions imported to match with MATLAB:*

|  |  |  |
| --- | --- | --- |
| MATLAB | PYTHON | Library used |
| Pdf(‘Normal’) | Norm.pdf(array,location=mean,scale=sigma) | Scipy.stats |
| pdf(‘ncx2’) | Nxc2.pdf(array,location=mean,scale=sigma) | Scipy.stats |
| pdf(‘Exponential’) | expon.pdf(array,location=0,scale=mean) | Scipy.stats |
| Pdf(‘lognorm’) | Lognorm.pdf(array,s=sigma,scale=np.exp(mean)) | Scipy.stats |
| [0;cumsum(array)] | array=np.insert(array,0,0,axis=1)  array=np.cumsum(array,axis=1) | Numpy |
| zeros(nsteps,npaths) | Np.zeros(steps\*npaths).reshape(npaths,nsteps) | Numpy |
| Figure(1)  Plot(x,f)  Legends(‘Line’) | Plt.Figure(1)  Plt.plot(x,f,label=’Line’)  Plt.legends()  Plt.show()  Or  Plt.plot(x,f) plt.legends([‘Line’]) plt.show() | Matplotlib.pyplot |
| tic  toc | timing.tic()  timing.toc()  timing.tocvalues() (store as variables) | pytictoc |
| rand()  poissrnd() | Np.random.rand()  Np.random.poission() | Numpy |
| loglog(x,y) | Plt.plot(x,y)  Plt.yscale(log)  Plt.xscale(log) | Matplotlib.pyplot |
| hist(array1,x)/(nsample\*timestep) | Plt.hist(array,bins=x,density=True)  (in-built normalizing function= density=True) | Matplotlib.pyplot |
| Xcorr(array2,’unbiased’) | Np.correlate(array2,array2,mode=2) where it matches with xcorr(array2,’normalized’). To match with unbiased use:  (1/nsteps+1)\* np.correlate(array2,array2,mode=2) | Numpy |
| Blsprice(S0,K,r,T,sigma,q) | ql.Date(),ql.Option.Call(),ql.EuropeanExercise(),ql.Europeanpricingengine()  Need to use various QuantLib modules to replicate blsprice pricing engine | QuantLib |
| real(fftshift(fft(iffshift()) | Np.real(np.fft.fftshift(np.fft.fft(np.fft.ifftshift() | Numpy |

*Creating Loops:*

|  |  |
| --- | --- |
| MATLAB | PYTHON |
| Create an array of values to feed into an existing array  combo=zeros(10);  for i=1:length(combo)  combo[i]=i+1; | Create an empty list and append it. Length is represented by len() which is used in an range. Zip is used to apply for a loop of 2 or more dataframes  combo=[]  for i,j in zip(len(combo),range(10)):  comb=i+1  combo.append(comb)  Create an array of values to feed into an existing array  combo=np.zeros(10)  for i in range(len(combo)):  combo[i]=i+1 |

*Object-Oriented Programming:*

|  |  |
| --- | --- |
| MATLAB | PYTHON |
| Defining a function  Function[mu,sigma,alpha]=  calibration\_ml(array,timestep) | Defining a function  def calibration\_ml(array,timestep):  return mu,sigma,alpha |
| Utilize this function on another script  [muS,sigmaS,alphaS]=calibration\_ml(array,t) | Utilize this function on another script  From calibration\_ml\_file import calibration\_ml  muS,sigmaS,alphaS=calibration\_ml(array,t)  Where calibration\_ml\_file is the .py file (needs to be that format) that we are referencing and calibration\_ml is the function we are importing |