$ Supplement Note 1: Introduction to Python

Python is a general purpose computer programming language which has been widely used in various fields of application such as engineering, scientific research, AI, machine learning, etc. It is considered as one of the easiest programming language for beginners. Tips on how to install python packages as well as tutorial lecture videos on python are available in the web such as **YouTube**.

Python is not a requirement in this class, but will frequently be used in the supplement notes to aid the course.In addition to python, you should also at least install the following three sub-packages of python:

1. **numpy**(*numeric python*): basic data structures and tools for numerical computation. It is wrapped around with codes written in C or FORTRAN for faster computation.

2. **scipy**(*scientific python*): scientific packages based on codes written in C, C++ and FORTRAN.

3. **matplotlib**: plotting packages.

Example1.py is our first example of python code. It may be executed by typing *phthon3 Example.py* in the sub-directory of the computer which contains the code: Example.py. Note that each line in the codes starts from the left and all the contents after the sign '#' are helping comments. In the contents of input-output for python, the characters inside a pair of single quotes('...') of double quotes("...") are *strings*, i.e. a series of characters. In order to use the functions in the numpy sub-package we have to *import* the sub-package **numpy** which is referred to in this code as *np*. The function *numpy.arange* produce a *numpy array* of increasing integers. More details and examples may be found in the *Google* web by typing

'numpy.arange'. In **python**, as well as in **C**, the element count of arrays starts from 0, while in *FORTRAN* it starts from 1. The power x^n is expressed in numpy as 'x**n'. The function $numpy.linspace(-1, 3, 10)$ yields a numpy array which contains 10 numbers evenly distributed in the interval $[-1, 3]$ (including the numbers at both ends). One advantage of using numpy arrays is "*broadcasting*". For example, $x2 = x**2$ and $x3 = x**3$ contain the squares and the cubes of each number in the numpy array x. Likewise, $f(x)$ contains the evaluated values of the cubic function $f(t) = 1 + t - 3.1\ t^2 + t^3$ for each number t in the numpy array x.

```
# Example1.py :
import numpy as np
a = np.arange(1,11,2)  # numpy array of increasing integers starting from 1
print('a=', a)         # to an integer less than 11, with stepping 2
print('a[0]=', a[0])   # the first element of the numpy array a
print('a[-1]=', a[-1]) # the last element of the numpy array a
b = a[1:-1]
print("a[1:-1]=", b)
c = a[1:]
print('a[1:]=', c)
d = a[1:-1]
print('a[1:-1]=', d)
suma = np.sum(a)
print('sum of a =', suma)
aa = a*a
print('a*a=', aa)
a2 = a**2
print('a**2=',a**2)
x = np.linspace(-1,3,10)
print('x=\n',x)
x2, x3 = x**2, x**3
f = 1+ x- 3.1*x2 + x3
print('f(x)=\n', f)
```