311 – Numerical Computations
Lab 6: Newton-Raphson Method

Python Libraries (of interest in this course):

- **SciPy: provides a lot of Numerical Algorithms (Examples: optimize.bisect (See Lab 5) and optimize.newton (See Current Lab**)).

- **NumPy: Fast array and Matrix manipulation**

- **SymPy: Symbolic Computations (Which is the converse of Numerical Computations).**

- **In Our course, we will study/use the first two libraries, the third will be covered only in the current lab (just as general knowledge).**

- **Finally there is also the library Matplotlib: Creates interactive charts/figures/visualizations.**

A) Global variables in Python

```
x=5
def f(w):
    return w+x

print(f(10))                    #prints 15
```

```
x=5
def f(w):
    x=x+3
    return w+x

print(f(10))

➔   Error:  local variable 'x' referenced before assignment
```

Solution:

```
x=5
def f(w):
    global x
    x=x+3
    return w+x

print(f(10))      #prints 18
```

<u>The rule:</u> You can only access global variables but you must use the keyword: global [variable Name] If you need to modify it.

B) Newton Method in SciPy:

```
from scipy import optimize
def f(x):
    return x**2-4

print(optimize.newton(f,1))
                                  #prints 2.0000000000000004
```

Exercise: Use SciPy's Newton method to solve:

$$(x^2 - 7x + 9)e^{-0.2x} = 0$$

(with x_0=6)

C) Newton Method Implementation

Based on Newton-Raphson Method, write the function:

```
def  nth_root (x,  n,   e=0.05, maxit=50, init=1)
```

where:

- x: the number that its root is to be computed.
- n: the degree of the root (n >=2), n=2: square root, n=3: cubic root and so on.
- e: epsilon.
- maxit: maximum number of iterations.
- init: initial guess, taken by default to be 1.

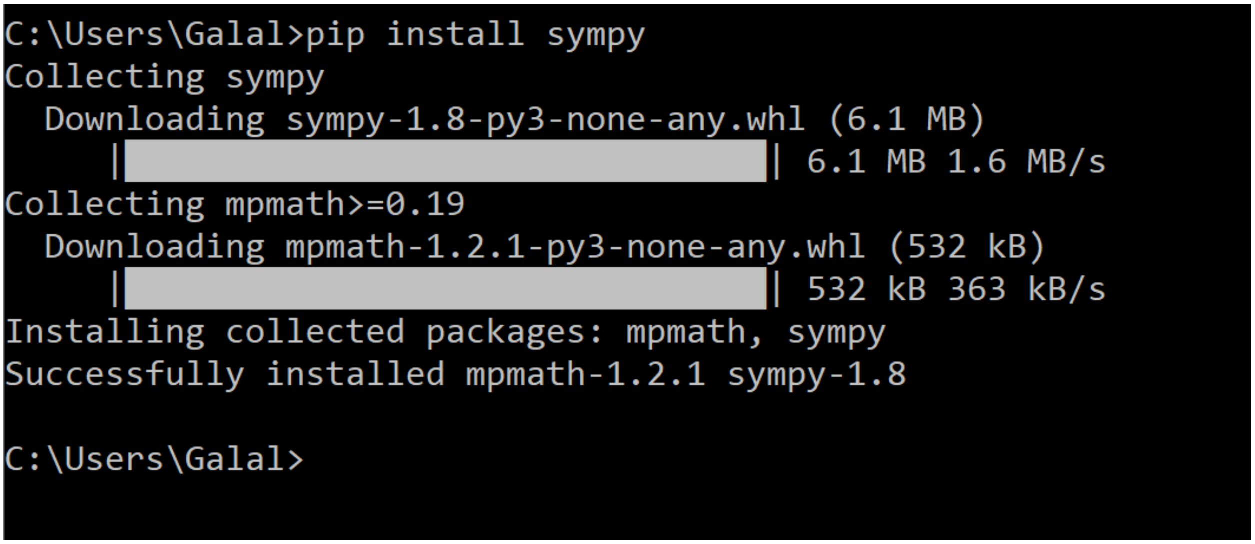
The good point is that newton's method for roots finding converges for any initial guess.

But we can find a better guess:

Better Initial guess: 1 + (x-1)/n

D) What about if derivative is not easy to compute?

install “sympy”!!



```
C:\Users\Galal>pip install sympy
Collecting sympy
  Downloading sympy-1.8-py3-none-any.whl (6.1 MB)
     |████████████████████████████████| 6.1 MB 1.6 MB/s
Collecting mpmath>=0.19
  Downloading mpmath-1.2.1-py3-none-any.whl (532 kB)
     |████████████████████████████████| 532 kB 363 kB/s
Installing collected packages: mpmath, sympy
Successfully installed mpmath-1.2.1 sympy-1.8

C:\Users\Galal>
```

Example (as General Knowledge):

```
from sympy import *
x = Symbol('x')      # create a "symbol" called x
f = sin(x)**2-4*x
fprime = f.diff(x)      #Calculae Derivative
print(fprime)
ff = lambdify(x, f, 'mpmath')
ffprime= lambdify(x, fprime, 'mpmath')
print(ff(0))
print(ffprime(0))
```