311 – Numerical Computations
Lab 5: Installing Scipy and Numpy, The bisection Method

A-Matrices in Python/ Matrices in Numpy :

In Python, they are basically: List of lists

```
a = [[7, 1, 3, 2], [7, 3], [6,  1, 2]]
for i in range(len(a)):
    for j in range(len(a[i])):
        print(a[i][j], end=' ')
    print( )
```

Output:

7 1 3 2
7 3
6 1 2

Same Program:

```
a = [[7, 1, 3, 2], [7, 3], [6, 1,2]]
for i in a:
    for j in i:
        print(j, end=' ')
    print( )
```

However, we will later shift to “Matrices” in the special library: <u>Numpy</u>.

B- Default arguments in Python:

```
def f (a=1, b=2, c=3):            # 3 optional arguments

        return a+2*b-c

print(f())

print(f(5))

print(f(2,3))

print(f(c=8))
```

```
def f (a, b=2, c=3):            #2 optional arguments
        return a+2*b-c


print(f(5))
print(f(2,3))
print(f(4,c=8))
print(f(c=8))
print(f())
```

However, the following function is Not valid (Syntax Error):

```
def f (a=4, b, c=3):

        return a+2*b-c
```

Because non-default argument follows default argument

C- Installing NumPy and SciPy:

1-Install PIP:

PIP is a <u>package management system</u> used to install and manage software packages in Python.

Start On the command prompt by:

```
C:\Users\Galal>python -V
Python 3.9.1
C:\Users\Galal>pip –V                (or pip --version)
pip 21.0.1   …..
```

If pip is not installed, you have to install it first. Now you can proceed:

```
pip install numpy
```

```
C:\Users\Galal\ >pip install numpy
Collecting numpy
  Downloading numpy-1.20.2-cp39-cp39-win_amd64.whl (13.7 MB)
     |                                | 13.7 MB 30 kB/s
Installing collected packages: numpy
Successfully installed numpy-1.20.2
```

```
pip install scipy
```

- Now the following program should work:

```
from scipy import optimize
def f(x):
    return (x**2 - 1)


print(optimize.bisect(f, 0, 2))
print(optimize.bisect(f, -2, 0))
```

So How to set the desired tolerance, for example????

The secret is in the **Default Arguments (See Section B)**,,

(See also Next Page)!!

D- Numpy and Scipy Documentation:

Example: bisect function documentation:

SciPy.org | Docs | SciPy v1.6.2 Reference Guide | Optimization and root finding (scipy.optimize)

scipy.optimize.bisect

scipy.optimize.**bisect**(*f, a, b, args=(), xtol=2e-12, rtol=8.881784197001252e-16, maxiter=100, full_output=False, disp=True*) [source]

Find root of a function within an interval using bisection.

Basic bisection routine to find a zero of the function *f* between the arguments *a* and *b*. *f(a)* and *f(b)* cannot have the same signs. Slow but sure.

Parameters:

f : *function*
: Python function returning a number. *f* must be continuous, and f(a) and f(b) must have opposite signs.

a : *scalar*
: One end of the bracketing interval [a,b].

b : *scalar*
: The other end of the bracketing interval [a,b].

xtol : *number, optional*
: The computed root `x0` will satisfy `np.allclose(x, x0, atol=xtol, rtol=rtol)`, where `x` is the exact root. The parameter must be nonnegative.

rtol : *number, optional*
: The computed root `x0` will satisfy `np.allclose(x, x0, atol=xtol, rtol=rtol)`, where `x` is the exact root. The parameter cannot be smaller than its default value of `4*np.finfo(float).eps`.

maxiter : *int, optional*
: If convergence is not achieved in *maxiter* iterations, an error is raised. Must be >= 0.

args : *tuple, optional*
: Containing extra arguments for the function *f*. *f* is called by `apply(f, (x)+args)`.

full_output : *bool, optional*
: If *full_output* is False, the root is returned. If *full_output* is True, the return value is `(x, r)`, where x is the root, and r is a RootResults object.

disp : *bool, optional*
: If True, raise RuntimeError if the algorithm didn't converge. Otherwise, the convergence status is recorded in a RootResults return object.

Returns:

x0 : *float*
: Zero of *f* between *a* and *b*.

The documentation homepage:

https://docs.scipy.org/doc/

E- Example: Solving the Cubic Equation

a- The algebraic method!!! **(Don't try it !!!)**

$$x = \sqrt[3]{\left(\frac{-b^3}{27a^3}+\frac{bc}{6a^2}-\frac{d}{2a}\right)+\sqrt{\left(\frac{-b^3}{27a^3}+\frac{bc}{6a^2}-\frac{d}{2a}\right)^2+\left(\frac{c}{3a}-\frac{b^2}{9a^2}\right)^3}}$$

$$+ \sqrt[3]{\left(\frac{-b^3}{27a^3}+\frac{bc}{6a^2}-\frac{d}{2a}\right)-\sqrt{\left(\frac{-b^3}{27a^3}+\frac{bc}{6a^2}-\frac{d}{2a}\right)^2+\left(\frac{c}{3a}-\frac{b^2}{9a^2}\right)^3}} - \frac{b}{3a}.$$

Abel–Ruffini theorem: There is no algebraic solution to general polynomial equations of degree five or higher with arbitrary coefficients.

b- Write your own (non-recursive) version of the bisection method:

def bisection(f,a,b,e):

- For simplicity, assume initially that: f(a)f(b) <0.
- Stop and return the value x, where f(x)<=e.
- Print the sequence of solutions (helpful if debugging is needed).

c- Use your function to solve the cubic equation:

$$0.4\ x^3 - 9\ x^2 - 12 = 0$$

(Use the two ends: 20 and 25 and e=0.005)