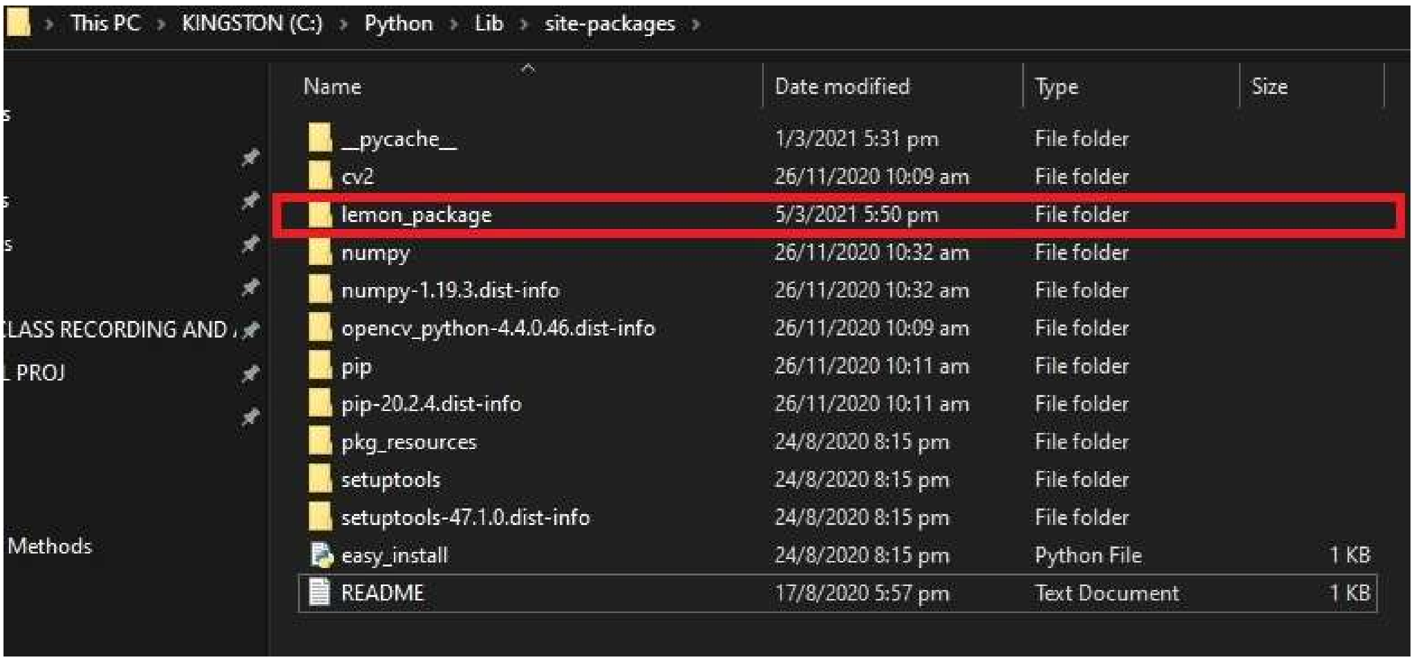
Installing Lemon Package

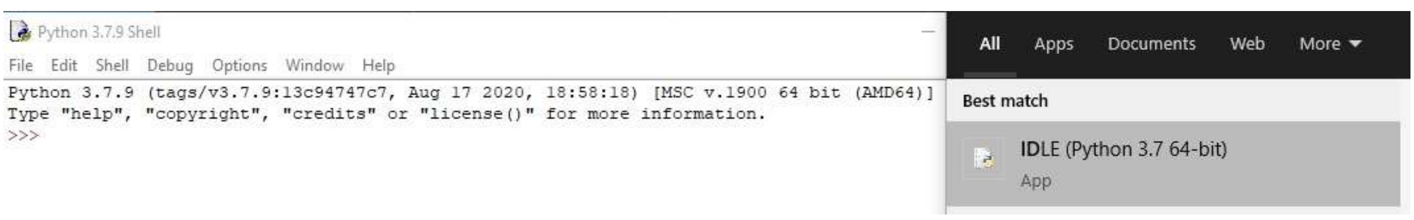
1.) Download the lemon package in the GitHub Repository provided. ***Lemon Package***

2.) Install the lemon package in the directory, Python\Lib\site-packages



3.) Once the lemon package is installed, the lemon modules can now be used.

4.) To find the roots using the lemon modules, open the Python IDLE first.



Using Brute Force (Iterative method)

1.) Import the necessary packages such as Numpy and lemon_package. Next is to create an equation using a function that the roots will be solved.

2.) Then declare the number of roots to be found and the number of iteration(epochs) and their starting values(h and s).

```
>>> import numpy as np
>>> from lemon_package import BruteForceIteration as bfi
>>> from lemon_package import BruteForceIntermsX as bfx
>>> sample3 = lambda x: np.log(x**2+1)
>>> no_roots = 4
>>> epochs = 100000
>>> h = -10
>>> s = 10
```

3.) Lastly, call the function and assign it to two variables. The function will return two values, the roots and the epoch where the root was last found.

```
>>> roots,epochs = bfi.b_force(sample3,no_roots,epochs,h,s)
>>> print("The roots are",roots,"found at",epochs)
The roots are [-0.] found at 99999
```

Using Brute Force (Interms of X)

1.) Repeat step 1 in the Brute Force (Iterative method)

2.) Transform the equation depending on the degree of the polynomial. Isolate each X to create transformed expressions of the equation.

3.) Append the transformed equations into a single list

```
>>> sample1 = lambda x: 2*x**4 + 3*x**3 - 11*x**2 - 9*x + 15
>>> f1 = lambda x: (2*x**4 + 3*x**3 - 11*x**2 + 15)/9
>>> f2 = lambda x: ((2*x**4 + 3*x**3 - 9*x + 15)/11)**(1/2)
>>> f3 = lambda x: ((-2*x**4 + 11*x**2 + 9*x - 15)/3)**(1/3)
>>> f4 = lambda x: ((-3*x**3 + 11*x**2 + 9*x - 15)/2)**(1/4)
>>> funcs2 = [f1,f2,f3,f4]
```

4.) Declare the number of roots to be found and the number of iteration (epochs)

5.) Assign the function to two variables and then call the function. The function will return the roots and the epoch where the root was last found.

```
>>> no_roots = len(funcs2)
>>> epochs = 100
>>> roots,epochs = bfx.b_forcex(funcs2,no_roots,epochs)
>>> print("The roots are",roots,"found at epoch",epochs)
The roots are [1.   +0.j 1.732+0.j] found at epoch 16
```

Using the Newton-Rhapson Method

1.) Import the method into the IDLE by typing "from lemon_package import Newton-Rhapson"

```
Python 3.7.9 Shell
File  Edit  Shell  Debug  Options  Window  Help
Python 3.7.9 (tags/v3.7.9:13c94747c7, Aug 17 2020, 18:58:18) [MSC v.1900 64 bit
(AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> from lemon_package import Newton
>>> |
```

2.) Provide a function where the root will be found and provide the necessary parameters of the newton-rhapson module. Lastly, call the newton-rhapson method to get the roots of the function provided.

```
Python 3.7.9 Shell
File  Edit  Shell  Debug  Options  Window  Help
Python 3.7.9 (tags/v3.7.9:13c94747c7, Aug 17 2020, 18:58:18) [MSC v.1900 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> from lemon_package import Newton as nw
>>> import numpy as np
>>> func = lambda x:  np.sin(2*x)-np.cos(2*x)
>>> roots,epoch = nw.newton(func,n_roots = 5,epochs = 100, tol = 1.0e-05,inits = np.arange(0,6.28))
>>> print("The roots are {}, found at epoch: {}".format(roots,epoch))
The roots are [0.393 1.963 3.534 5.105 8.247], found at epoch: 2
>>> |
```

Documentation of the Lemon Modules

To know more about the uses and syntax of the brute force and newton-rhapson module. Check its documentation in the link provided. ***Python Package Manual***

Activity 2.1

```
import numpy as np
```

```
import matplotlib.pyplot as plt
```

$$x^3 - 6x^2 + 11x - 6$$

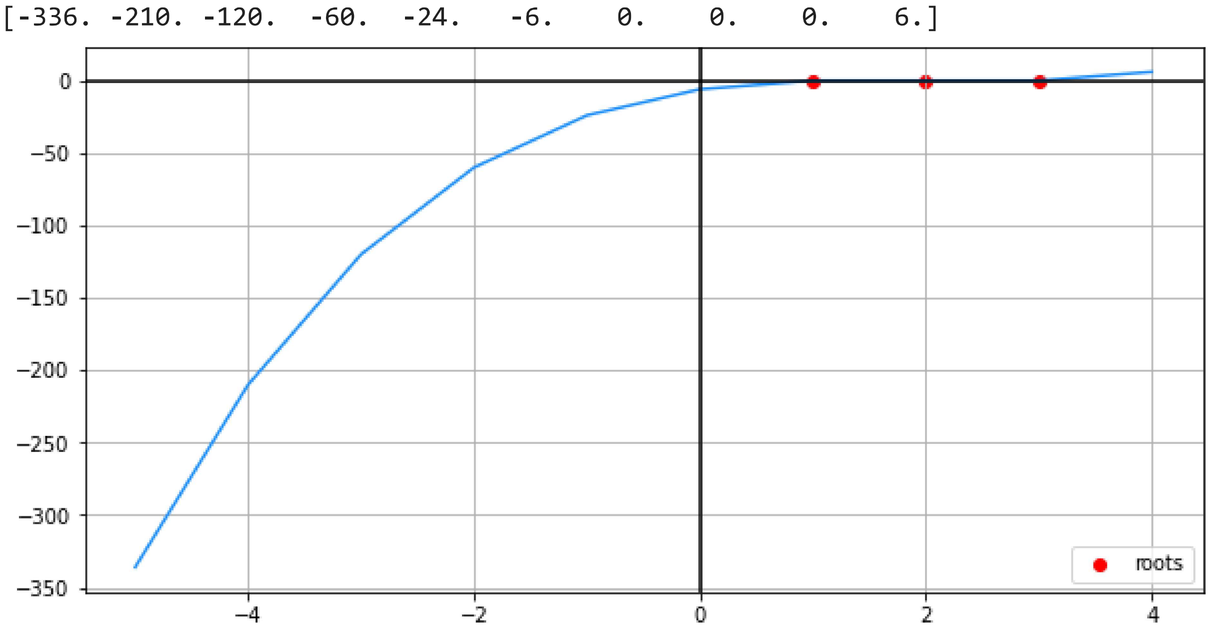
(Polynomial Degree of 3)

```
def f(x): return x**3-6*x**2+11*x-6
x0, x1, x2 = 1, 2, 3
X = np.arange(-5,5,1,dtype=float)
print(X[0:10])
Y = f(X)
print(Y[0:10])

### Now let's plot the images against the pre-images
plt.figure(figsize=(10,5))
plt.plot(X,Y,color='dodgerblue')
### Let's show the x and y axes of the graph
plt.axhline(color='black')
plt.axvline(color='black')
plt.grid()
### Now let's plot the roots of the equation
plt.scatter([x0,x1,x2],[0,0,0], c='red', label='roots')

plt.legend()
plt.show()
```

```
[-5. -4. -3. -2. -1.  0.  1.  2.  3.  4.]
[-336. -210. -120.  -60.  -24.   -6.    0.    0.    0.    6.]
```



$$x^4 + 0.4x^3 - 6.49x^2 + 7.244x - 2.112$$

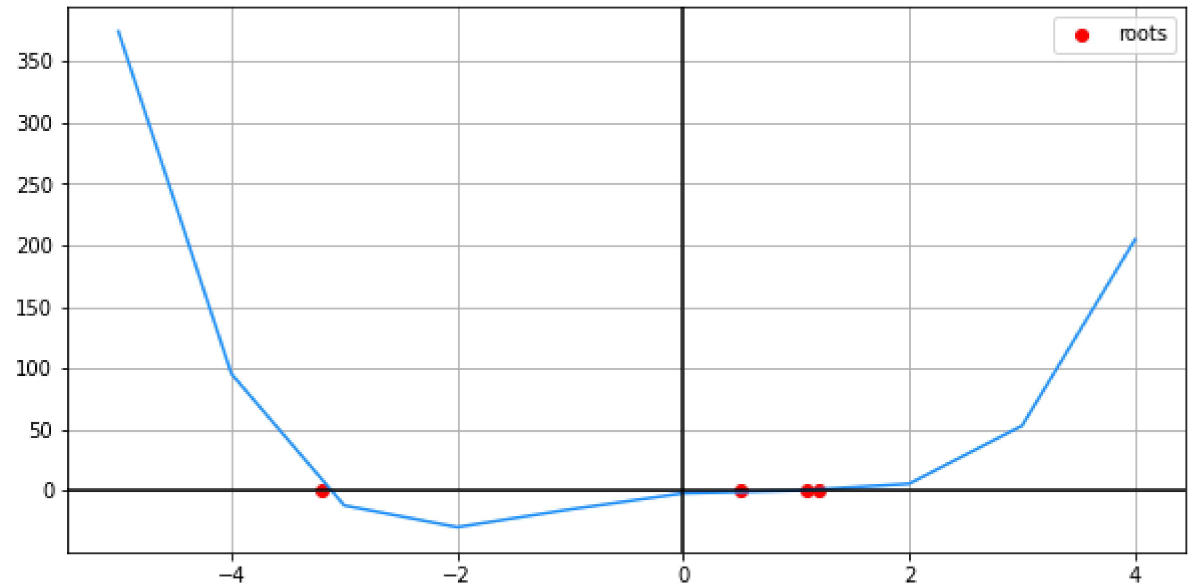
(Polynomial Degree of 4)

```
def f(x): return x**4+0.4*x**3-6.49*x**2+7.244*x-2.112
x0, x1, x2, x3 = -3.2, 1.2, 0.5, 1.1
X = np.arange(-5,5,1,dtype=float)
print(X)
Y = f(X)
print(Y)
### Now let's plot the images against the pre-images
plt.figure(figsize=(10,5))
plt.plot(X,Y,color='dodgerblue')
### Let's show the x and y axes of the graph
plt.axhline(color='black')
plt.axvline(color='black')
plt.grid()
### Now let's plot the roots of the equation
plt.scatter([x0,x1,x2,x3],[0,0,0,0], c='red', label='roots')

plt.legend()
```

```
plt.legend()
plt.show()
```

```
[-5. -4. -3. -2. -1.  0.  1.  2.  3.  4.]
[ 3.74418e+02  9.54720e+01 -1.20540e+01 -2.97600e+01 -1.52460e+01
 -2.11200e+00  4.20000e-02  5.61600e+00  5.30100e+01  2.04624e+02]
```



$$e^x + x - 4$$

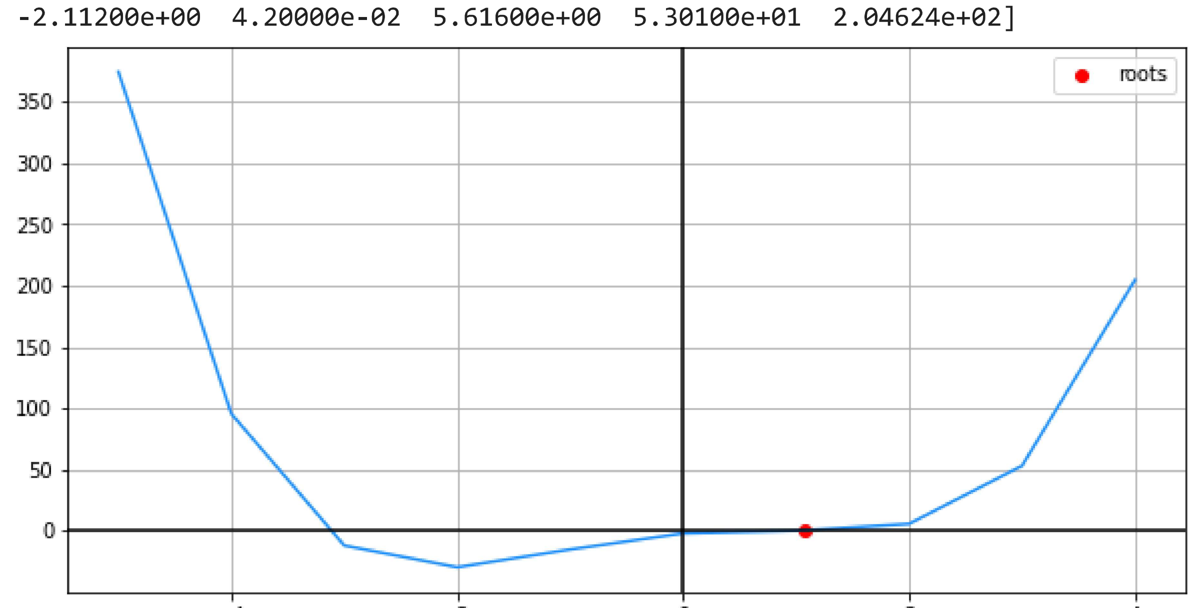
(Transcendental function)

```
func = lambda x: x +np.exp(x) - 4
x0 = 1.074
X = np.arange(-5,5,1,dtype=float)
print(X)
Y = f(X)
print(Y)

### Now let's plot the images against the pre-images
plt.figure(figsize=(10,5))
plt.plot(X,Y,color='dodgerblue')
### Let's show the x and y axes of the graph
plt.axhline(color='black')
plt.axvline(color='black')
plt.grid()
### Now let's plot the roots of the equation
plt.scatter([x0],[0], c='red', label='roots')

plt.legend()
plt.show()
```

```
[-5. -4. -3. -2. -1.  0.  1.  2.  3.  4.]
[ 3.74418e+02  9.54720e+01 -1.20540e+01 -2.97600e+01 -1.52460e+01
 -2.11200e+00  4.20000e-02  5.61600e+00  5.30100e+01  2.04624e+02]
```



$$\frac{sin(x)}{cos(x)}$$

(Transcendental function)

```
def f(x): return  np.sin(x)/np.cos(x)
x0, x1, x2 = 0, 3.142, 6.283
X = np.arange(-10,10,1,dtype=float)
print(X[0:10])
Y = f(X)
print(Y[0:10])

### Now let's plot the images against the pre-images
plt.figure(figsize=(10,5))
plt.plot(X,Y,color='dodgerblue')
### Let's show the x and y axes of the graph
plt.axhline(color='black')
plt.axvline(color='black')
plt.grid()
### Now let's plot the roots of the equation
plt.scatter([x0,x1,x2],[0,0,0], c='red', label='roots')

plt.legend()
plt.show()
```

```
[-10.  -9.  -8.  -7.  -6.  -5.  -4.  -3.  -2.  -1.]
[-0.64836083  0.45231566  6.79971146 -0.87144798  0.29100619  3.38051501
 -1.15782128  0.14254654  2.18503986 -1.55740772]
```