```
import numpy as np
```

Brute Force Iteration

```
no_roots = 4
epochs = 100000
h = -10
s = 10
sample3 = lambda x: np.log(x**2+1)
```

```
def b_force(func,no_roots,epochs,x0,x1,tol = 1e-5):
  roots = []
  end_epoch = 0
  for epoch in range(epochs):
    if np.allclose(0,func(x0),tol):
      roots.append(x0)
      end_epoch = epoch
      if len(roots) == no_roots:
        break
    x0+=1e-4
  for epoch in range(epochs):
    if np.allclose(0,func(x1),tol):
      roots.append(x1)
      final_roots = np.unique(np.around(roots,3))
      end_epoch = epoch
      if len(roots) == no_roots:
        break
    x1-=1e-4
  if len(roots) != 0:
    return final_roots, end_epoch
  else:
    final_roots = "Roots cannot be found"
    end_epoch = -1
    return final_roots,end_epoch
```

```
roots,epochs = b_force(sample3,no_roots,epochs,h,s)
print("The roots are", roots, "found at",epochs)
```

```
The roots are [-0.] found at 99999
```

Brute Force interms of X

```
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]
no_roots = len(funcs2)
epochs1 = 100
```

```
def b_forcex(funcs,no_roots,epochs,x0 = 0):
  roots = []
  for func in funcs:
    x0=0
    for epoch in range(epochs):
      x_prime = func(x0)
      if np.allclose(x0, x_prime):
```

```
        roots.append(x0)
        final_roots = np.unique(np.around(roots,3))
        end_epoch = epoch
        break
      x0 = x_prime
  if len(final_roots) != 0:
    return final_roots, end_epoch
  else:
    final_roots = "Roots cannot be found"
    end_epoch = -1
    return final_roots, end_epoch
```

```
roots,epoch = b_forcex(funcs2,no_roots,epochs1)
print("The roots are",roots,"found at",epoch)
```

```
The roots are [1.   +0.j 1.732+0.j] found at 16
```

Newton-Rahpson Method

```
def derivative(f,x,dx = 1e-6):
  diff = f(x+dx)-f(x-dx)
  return diff/(2*dx)

def newton(func,n_roots,epochs, tol = 1.0e-05,inits = np.arange(-5,5)):
  x_roots = []
  for init in inits:
    x=init
    for epoch in range(epochs):
      f_prime = derivative(func,x)
      x_new = x - (func(x)/f_prime)
      if np.allclose(x, x_new, tol):
        x_roots.append(x)
        final_roots = np.unique(np.around(x_roots,3))
        final_roots = final_roots[:n_roots]
        break
      x = x_new
  return final_roots, epoch
```

```
func = lambda x: np.sin(2*x)-np.cos(2*x)
roots,epoch = newton(func,n_roots = 5,epochs = 100, tol = 1.0e-05,inits = np.arange(-5,5))
print("The roots are {}, found at epoch: {}".format(roots,epoch))
```

```
The roots are [-9.032 -4.32  -2.749 -1.178  0.393], found at epoch: 3
```