Newtons Raphson method

It involves the use of Initial guess x0, tolerance, and maximum iterations are provided as inputs.The method recursively looks for roots .Moreover, the function needs the derivative, which is represented by the symbol df (x).

CODE:

```
%%timeit -r 4 -n 10000
#newton rhapshon
import numpy as np

f = lambda x: x**4 - 2
f_prime = lambda x: 4*x**3

def newton(f, df, x0, tol):
    # output is an estimation of the root of f using the Newton Raphson method
    if abs(f(x0)) < tol:
        return x0
    else:
        #recursive call
        return newton(f, df, x0 - f(x0)/df(x0), tol)
```

Bisection Method

Given a tolerance and maximum number of iterations, this function employs the bisection method to locate the function's root. The root's location is in the range $[a, b]$. The tolerance and the maximum number of iterations are specified in the function. We start by defining a function $f(x)$ and its derivative $f'(x)$, as well as an initial guess for the root:

Bisection method

```
import numpy as np

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

    # check if a and b have a root
    if np.sign(f(a)) == np.sign(f(b)):
        return "The scalars a and b do not bound a root"
    # get midpoint
    m = (a + b)/2

    if np.abs(f(m)) < tol:
        #  print m as root
        print (m)
```

```
    elif np.sign(f(a)) == np.sign(f(m)):
        # Make recursive call with a = m
        return bisection(f, m, b, tol)
    elif np.sign(f(b)) == np.sign(f(m)):
        # Make recursive call with b = m
        return bisection(f, a, m, tol)

f = lambda x: x**4 - 2
#tolerance
tol=1e-6
#call function set bounds within which function will operate
bisection(f,1,10,tol)
```

Conclusion,Both newton raphson method and bisection method can be used to find roots but the newton raphson method is faster than the bisection method with a speed of