```python
# import libraries
import numpy as np


# define routines
def bisection(f,a,b,tol,Nmax):
    '''
    Inputs:
      f,a,b       - function and endpoints of initial interval
      tol, Nmax   - bisection stops when interval length < tol
                  - or if Nmax iterations have occured
    Returns:
      astar - approximation of root
      ier   - error message
            - ier = 1 => cannot tell if there is a root in the interval
            - ier = 0 == success
            - ier = 2 => ran out of iterations
            - ier = 3 => other error ==== You can explain
    '''

    '''     first verify there is a root we can find in the interval '''
    fa = f(a); fb = f(b);
    if (fa*fb>0):
       ier = 1
       astar = a
       return [astar, ier]

    ''' verify end point is not a root '''
    if (fa == 0):
      astar = a
      ier =0
      return [astar, ier]

    if (fb ==0):
      astar = b
      ier = 0
      return [astar, ier]

    count = 0
    while (count < Nmax):
      c = 0.5*(a+b)
      fc = f(c)

      if (fc ==0):
        astar = c
        ier = 0
        return [astar, ier, count]

      if (fa*fc<0):
         b = c
      elif (fb*fc<0):
        a = c
        fa = fc
      else:
        astar = c
        ier = 3
        return [astar, ier, count]
```

```
      if (abs(b-a)<tol):
        astar = a
        ier =0
        return [astar, ier, count]

      count = count +1

    astar = a
    ier = 2
    return [astar,ier, count]

# use routines
f = lambda x: 2*x - 1 - np.sin(x)
a = 0
b = np.pi

Nmax = 100
tol = 1e-3

[astar,ier, count] = bisection(f,a,b,tol,Nmax)
print('the approximate root is',astar)
print('the error message reads:',ier)
print('The number of iterations are', count)
```