Problem 2

We have to fidn the roots for following : a) ( ) = tan( ) and b) ( ) = tanh( ).

a) Bisection method

To find the root for ( ) = tan( ), we will use Bisection method first. The examples and codes are taken from Bisection. I have also used the codes exapmles shown in class.

The **" bisection method "** is a simple method to find the root of a function. We start with guessing a boundary (upper limit and lower limit). The steps are given as follows:

> Choose a starting interval $ [a_0, b_0] $ such that $ f(a_0)*f(b_0) < 0 $. Compute f(m) where m = $\frac{a_0+b_0}{2} $ is the midpoint. We Determine the next subinterval $ [a_1,b_1] $ such that :

```
> If  $ f(a_0)*f(m) < 0 $, for  then next interval , $ a_1=a_0,$ and $ b_1 = m_0 .$ \
> If  $ f(b_0)*f(m) < 0 $, for  then next interval , $ a_1=m_0,$ and $ b_1 = b_0 .$ \
```

> Repeat the steps above untill interval $[a_n, b_n]$ reaches to an limit of accuracy (given). When accuracy limit is reached : Return the midpoint value m = $\frac{a_n+b_n}{2} $.

Lets begin with necessary imports.

```
import numpy as np
import math
import matplotlib.pyplot as plt
# %load_ext pycodestyle_magic
# %pycodestyle_on
```

We will define our **"Bisection"** method below :

```
def bisection(f, lower_bound, upper_bound, max_iterations=100, error=1e-10):
    # we count our steps here
    steps= 1
    # Making sure our steps does not exceed the number of maximum iterations
    while steps < max_iterations:
        m = (lower_bound + upper_bound) / 2.0

        if abs(upper_bound-lower_bound) < error: # we will stop when we reach at certain ac
            return m, steps
        if f(m) > 0:
            upper_bound = m
        else:
            lower_bound = m
```

```
        steps += 1

    final_result = (lower_bound + upper_bound) / 2.0
    return final_result, steps
```

Lets define our function function f , and get an approximation for its root using above method. As its trig function, we provide limits from -2π to 2π and steps 10.

```
f = lambda x: math.tan(x)
root, steps = bisection(f,-2*np.pi,2*np.pi)
accuracy = abs(f(root))
print ("root is:", root)
print ("steps taken:", steps)
print ("accuracy is:", accuracy)
```

```
root is: 3.1415926536355094
steps taken: 38
accuracy is: 4.571619715412493e-11
```

The result we got is equal to 2π which is a correct root. But this function has many repeating roots . Because of our method and the boundary values This is the only possible outcome. If we choose any other boundaries, we might get another root values. But the error in this method is also very very low.

For f = tanh(x)

We use the same nethod again

```
f = lambda x: math.tanh(x)
root, steps = bisection(f,-2*np.pi,2*np.pi)
accuracy = abs(f(root))
print ("root is:", root)
print ("steps taken:", steps)
print ("accuracy is:", accuracy)
```

```
root is: 4.571618997709874e-11
steps taken: 38
accuracy is: 4.571618997709874e-11
```

This code does not give the expected result which is 0 but its very very close . Our error in this case is little bigger than the privious one.