Question 4

The smallest nonzero root is **4.49340945791** (same results from both method to such accuracy)

Bisection Method

Initial range : [4,5)
Total iteration : 47

```
import numpy as np
import math
#iteration counter: to count how many steps to reach the threshold value
i=0

#Define the target funciton
def f(x):
    return math.tan(x)-x

## NOTE: Inital generator, range:[4,5), half open interval, choosing a
number uniformally randomly
a = np.random.random()+4;
b = np.random.random()+4;

## Here guarantees f(b)<= 0 <= f(a)
while f(a)<0:
    b = a;
    a = np.random.random()+4;

while f(b)>0:
    b = np.random.random()+4;


#to print the initial range
print abs(a-b)

### Setting the threshold epsilon
epsilon_ = math.pow(10,-15)

### Begin the bisection iterations. # NOTE:  Return r as root.
```

```
if f(a)*f(b) != 0:
    while abs(a-b) > epsilon_ :
        c = (a+b)/2
        if f(c) > 0:
            a = c;
        elif f(c) < 0:
            b = c;
        else:
            a = b;
        i = i+1;
    r = (a+b)/2;
elif  f(a) == 0:
    r = a;
else:
    r = b;

print r # print the root = 4.49340945791

print i# print the number of iteratoins to get the root 47
```

Newton Method

Range : [4.2, 4.7)
Iterations : 7

```
import numpy as np
import math
#iteration counter: to count how many steps to reach the threshold value
i=0

#Define the target funciton
#def f(x):
#    return math.tan(x)-x;

# here iter(x) is x - f(x)/f'(x)
def iter(x):
    tanx = math.tan(x);
    tanx2 = math.pow(tanx,2);
```

```
    return x - (tanx - x)/tanx2

## NOTE: Inital generator, range:[4.2,4.7). Since we get smallest root from
bisection around 1.57.
x_1 = 0

x_2 = np.random.random()*.5+4.2;


### Setting the threshold epsilon
epsilon_ = math.pow(10,-15)

while abs(x_1-x_2) > epsilon_:
    x_1 = x_2;
    x_2 = iter(x_1);
    i = i+1;

# x_2= 4.49340945791
print x_2

# i = 7
print i
```