```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
#question 3
#determine the real root of f(x) = x^3 - 13x - 12

#part A
def f(x):
    return x**3 - 13 * x - 12

x = np.arange(-5, 5, .1)
y = f(x)


plt.figure()
plt.plot(x,y)
plt.grid("on")
plt.title("f(x) = x^3 - 13x - 12")
plt.xlabel("x")
plt.ylabel("y")
plt.show()


#Zeroes appear to be at -3, -1, and 4

#part b
a = 2
b = 7
tolerance = .001
possibleRoots = []
for i in range(0, 5):
    #if there isnt a root in the range, break
    if f(a)*f(b) > 0:
        break
    #possible root is the average of two guesses
    possibleRoot = (a+b)/2
    possibleRoots.append(possibleRoot)

    #redefining a - b interval
    if f(possibleRoot) * f(a) < 0:
        # root is to the left of possibleRoot
        b = possibleRoot
    elif f(possibleRoot) * f(b) < 0:
        # root is to the right of possibleRoot
        a = possibleRoot

def error(val1, val2):
    return np.abs(100 * (val1 - val2)/val2)

approximateErrors = [np.nan]
for i in range(1, len(possibleRoots)):
    approximateErrors.append( error(possibleRoots[i-1], possibleRoots[i]))

data = {
    ("approximate root") : possibleRoots,
    ("approximate error %") : approximateErrors
}

df = pd.DataFrame(data)
df.index.name = "Iteration count"
print(df)

plt.figure()
plt.plot(range(0, 5), possibleRoots, label="Root guesses")
plt.grid("On")
plt.xlabel("Number of terms")
plt.ylabel("Root value estimate")
plt.show()

#Müller's method approaches zero much more quickly because it is a higher order approximation than the bisection metho
# The bisection method is a first order solution whereas Müller's method is a second order solution, which means Mülle
# converges much faster.
```