**Q2.1**

What would go wrong if fabs were omitted from this function?-

from math import cos , fabs , pi

def g(x):

n =1; total = term = cos( x) # First term

while fabs ( term ) >(1.0e -7\* fabs ( total ) + 1.0e -13):

n +=2 # Advance to next term

term = cos( n\*x )/(n \*n) total += term # Add term to total

return total

**For reference, this is the function g(x), which is defined by an infinite series:**

g(x) = cos(x)/1^2 + cos(3x)/3^2 + cos(5x)/5^2 + cos(7x)/7^2 +···

**Answer**

If fabs were omitted, the condition for the while loop would always appear to not be met if the variable ‘term’ is a negative number. This is because fabs returns the modulus of a floating point number. The condition for the while loop is to start iterating if fabs(term) is greater than a manipulation of fabs(total). If cos(x) is negative then total and term will also be negative (as they are also set equal to cos(x)). Without fabs, term would always be less than the manipulation of total which means that the condition for the loop would never be met for the while loop (if cos(x) is a negative number). This would mean that only cos(x) would be returned, which is only the first term of the mathematical function g(x). This would make the whole code useless, as the purpose of the code is to get a value for the function g(x) applied to a number x to a certain degree of accuracy. While a value will be found, it will not be obtained at the desired degree of accuracy.

If cos(x) is a positive number (and in turn, term and total would also be positive), then it would keep looping until term is assigned a negative value, then we would end up with the same issue as before. So for a positive cos(x), a value for g(x) would be obtained which would be more accurate than its negative counterparts. However, this value wouldn’t necessarily be as accurate as it would be if fabs were not omitted.

**Q2.2**

Why do you think the number 1.0e−7 is chosen for the stopping condition, at least in comparison to a much larger or much smaller number, say 1e−3, or 1e−26?

**Answer**

A much larger number isn’t used because then the value for g(x) wouldn’t be obtained at a high accuracy. A much smaller number isn’t used because the code would take a longer time to generate to output, but mainly it’s because there isn’t much of a point using a much smaller number in terms of obtaining a more accurate value. This is because as the while loop continues to iterate, each subsequent iteration makes less of a difference on the value. So even though 1e-26 would output a more accurate g(x) value than 1e-7, it wouldn’t be significantly or noticeably more accurate.

**Q2.5**

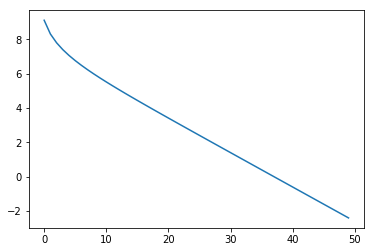
Write a program to plot a graph of log(Rn −R∞) against n.

From your graph explain why when n is large Rn is given approximately by:

Rn = R∞ +Ae−Bn

**Answer**

This is my graph:



You can see that after a certain point (when n is large), the gradient becomes a straight line. So, you can put Rn = R∞ +Ae−Bn in the form y = mx + c. In this case, the gradient becomes -B and ec is A.