For question 3,

Part a) Primitive I didn't use the adaptive method to find the error and accumulate the integral.

Instead, I use EQN 5.20 to approximate the error

And I defined f(x) and fprime(x)

But fprime turns out to be $$5\sin\left(20\sqrt{x}\right)/\sqrt{x}$$

Thus, to avoid divide by 0 I use Taylor expansion to the big O(x^6) precision

$$\begin{aligned} 5\sin(20\sqrt{x})/\sqrt{x} &= \frac{5}{\sqrt{x}}\left(20\sqrt{x} - \frac{(20\sqrt{x})^3}{3!} + \frac{(20\sqrt{x})^5}{5!} - \frac{(20\sqrt{x})^7}{7!} + \frac{(20\sqrt{x})^9}{9!} + O(x^{11/2})\right) \\ &= 5\left(20 - \frac{20^3 x}{3!} + \frac{20^5 x^2}{5!} - \frac{20^7 x^3}{7!} + \frac{20^9 x^4}{9!} - \frac{20^{11} x^5}{11!} + O(x^5)\right) \end{aligned}$$

(wolfram alpha) to approximate the fprime

then I just iterate and accumulate use this formula below:

$$I \approx h\left(\frac{1}{2}f(a) + \frac{1}{2}f(b) + \sum_{k=1}^{N} f(a+kh)\right)$$

and use a trap method (if the error is below 1e-6 break to end) to give the final result.

Part a) pretty much similar to the method in the primitive, I use the question asked method (adaptive one) with formula 5.34 and 5.30

This requires a method to determine if the number is a odd or not.

And I found I save 7 steps to generate a answer with same level of precision.(adaptive method works!)

Part b) use the answer in question a as the first column and use simple arithmetic to generate the triangle(use two value to generate the next

one),[eqn5.51]. But how to print out the triangle takes me huge time to figure out a way and logic.
At first, I just return z in the main method and get a matrix, but the 0s are annoying.
So I try use a defined function to change 0 to nan and then use numpy's Boolean logic to get rid of the nan, like this

```
z[z==0 ] = nan
z = z[logical_not(isnan(z))]
return z
```

result is sad as its actually move the space of 0 too .

```
0.455832417821 7 5
[ 0.14797948  0.32523191  0.38431605  0.51228285  0.57463317  0.58732097
  0.40299745  0.36656898  0.35269804  0.34897386  0.43010337  0.43913868
  0.44397666  0.44542552  0.44580376  0.44841467  0.45451843  0.45554375
  0.45572735  0.45576775  0.45577749  0.45391293  0.45574569  0.4558275
  0.45583201  0.45583242]
```

But I finally worked out as I wrote in the code. I user matlab's array printing method , it turns out to be same in python.
And 7 5 just means at column 5 row 7 the integration has error required.