Example 133. Python Let us redo Example 132 by implementing the trapezoidal rule.

```
>>> def trapezoidal_rule(f, a, b, n):
        h = (b - a) / n
        integral = f(a) + f(b)
        for i in range(1,n):
            integral += 2*f(a + i*h)
        return h/2*integral

>>> def f(x):
        return 1/x
```

Comment. Writing `x += y` is a useful and common short alternative to `x = x + y`.

Choosing `n` to be 2 and 4 is equivalent to $h=\frac{3-1}{2}=1$ and $h=\frac{3-1}{4}=\frac{1}{2}$ and so we get the same values as in Example 132:

```
>>> trapezoidal_rule(f, 1, 3, 2)

   1.1666666666666665

>>> trapezoidal_rule(f, 1, 3, 4)

   1.1166666666666667
```

As expected, further increasing n produces better approximations:

```
>>> [trapezoidal_rule(f, 1, 3, 10**n) for n in range(1,4)]

   [1.1015623265623264, 1.0986419169811203, 1.0986125849642736]
```

Indeed, the following convincingly illustrates that the error in the trapezoidal rule is $O(h^2)$.

```
>>> from math import log

>>> [trapezoidal_rule(f, 1, 3, 10**n) - log(3) for n in range(1,6)]

   [0.0029500378942166616, 2.9628313010565677e-05, 2.962961638264261e-07,
   2.9629636522088276e-09, 2.962430301067798e-11]
```

However, note that our computer had to work pretty hard to get the final approximation, because that entailed computing about 10^5 values. We clearly should use a higher order method in order to compute to higher accuracy. One option is to do what we did in the last part of Example 132.

Simpson's rule

Let us spell out what happens in general when we proceed as in the last part of Example 132. We start with $T(h)$, the trapezoidal rule applied with step size h, which is given by

$$T(h)=\frac{h}{2}[f(x_0)+2f(x_1)+\cdots+2f(x_{n-1})+f(x_n)].$$

Then, since $T(h)$ is an approximation of order $N=2$, the Richardson extrapolation

$$S(h):=\frac{2^N T(h)-T(2h)}{2^N-1}=\frac{4}{3}T(h)-\frac{1}{3}T(2h)$$

is an approximation of higher order, known as **Simpson's rule**. It takes the following form:

(Simpson's rule) Suppose n is even. The following is an approximation of order 4:

$$\int_a^b f(x)\mathrm{d}x \approx \frac{h}{3}[f(x_0)+4f(x_1)+2f(x_2)+4f(x_3)+\cdots+2f(x_{n-2})+4f(x_{n-1})+f(x_n)]$$

Proof. To see this, note that the trapezoidal approximations $T(h)$ and $T(2h)$ are given by

$$\begin{aligned} T(h) &= \frac{h}{2}[f(x_0)+2f(x_1)+\cdots+2f(x_{n-1})+f(x_n)], \\ T(2h) &= h[f(x_0)+2f(x_2)+\cdots+2f(x_{n-2})+f(x_n)]. \end{aligned}$$

Here, we need n to be even so that $T(2h)$ uses the points x_0, x_2, ..., x_{n-2}, x_n. Therefore, the Richardson extrapolation is

$$\begin{aligned} S(h)=\frac{4}{3}T(h)-\frac{1}{3}T(2h) &= \frac{h}{3}[2f(x_0)+4f(x_1)+\cdots+4f(x_{n-1})+2f(x_n)] \\ &\quad -\frac{h}{3}[f(x_0)+2f(x_2)+\cdots+2f(x_{n-2})+f(x_n)] \\ &= \frac{h}{3}[2f(x_0)+4f(x_1)+2f(x_2)+\cdots+2f(x_{n-2})+4f(x_{n-1})+2f(x_n)], \end{aligned}$$

where the right-hand side is Simpson's rule. That this is an approximation of order 4 follows because the error of $T(h)$ is an even function in h, so that the order of $S(h)$ increases to 4 (rather than the otherwise expected 3). □

Alternative approach. We can also proceed like for the trapezoidal rule, except that we use quadratic instead of linear interpolations on each segment. More precisely, starting with equally spaced nodes $x_0, x_1, ..., x_n$ with n even, we can interpolate $f(x)$ on each segment $[x_{i-1}, x_{i+1}]$, with i odd, by a quadratic function $p(x)$. The integral on that segment works out to be

$$\int_{x_{i-1}}^{x_{i+1}} f(x)\mathrm{d}x \approx \int_{x_{i-1}}^{x_{i+1}} p(x)\mathrm{d}x \overset{\text{work}}{=} \frac{h}{3}[f(x_{i-1})+4f(x_i)+f(x_{i+1})],$$

where we wrote $h=(b-a)/n$ for the distance between nodes. Adding together the integrals over all segments, each node x_i, with i odd, will show up once with the above factor of $4h/3$ while x_i, with i even, (except x_0 and x_n) will show up twice with a factor of $h/3$. We thus again get Simpson's integration rule.

Comment. The above rule is often called **Simpson's 1/3 rule**. There is also **Simpson's 3/8 rule** which is derived similarly but is based on a cubic (instead of a quadratic) interpolation; it thus requires an additional node (the resulting error term is of the same order but about half).

Similar to Theorem 131, one can show that the error in Simpson's rule is as follows:

Theorem 134. (Simpson's rule with error term) If f is C^4 smooth, then

$$\int_a^b f(x)\mathrm{d}x = \frac{h}{3}[f(x_0)+4f(x_1)+2f(x_2)+\cdots+4f(x_{n-1})+f(x_n)] - \frac{(b-a)}{180}f^{(4)}(\xi)h^4$$

for some $\xi \in [a,b]$.

Comment. In the alternative approach above, we constructed Simpson's rule so that quadratic polynomials would be integrated without error; the error term tells us that, in fact, cubic polynomials are also integrated without error. In other words, the error term is a pleasant surprise!

Example 135. Use Simpson's rule with $h=\frac{1}{2}$ to approximate $\int_1^3 \frac{1}{x}\,\mathrm{d}x = \log(3) \approx 1.09861$.

Solution.

$$\int_1^3 f(x)\,\mathrm{d}x \approx \frac{h}{3}\left[f(1)+4f\left(\frac{3}{2}\right)+2f(2)+4f\left(\frac{5}{2}\right)+f(3)\right] = \frac{1}{6}\left[1+4\cdot\frac{2}{3}+2\cdot\frac{1}{2}+4\cdot\frac{2}{5}+\frac{1}{3}\right] = \frac{11}{10} = 1.1$$

Comment. Note that $\frac{11}{10}$ is exactly what we obtained in the last part of Example 132.
Indeed, recall that Simpson's rule with $h=\frac{1}{2}$ ($n=4$) is equivalent to applying Richardson extrapolation to the trapezoidal approximations with $h=\frac{1}{2}$ ($n=4$) and $h=1$ ($n=2$).

Example 136. Python Let us compute $\int_1^3 \frac{1}{x}\,\mathrm{d}x = \log(3) \approx 1.09861$ by implementing Simpson's rule. The following code assumes that `n` is even.

```
>>> def simpson_rule(f, a, b, n):
        h = (b - a) / n
        integral = f(a) + f(b)
        for i in range(1,n):
            if i%2 == 1:
                integral += 4*f(a + i*h)
            else:
                integral += 2*f(a + i*h)
        return h/3*integral

>>> def f(x):
        return 1/x
```

With `n` set to 4, we obtain $\frac{11}{10}$ as in Examples 132 and 135:

```
>>> simpson_rule(f, 1, 3, 4)

   1.0999999999999999
```

Our approximations (here, $n=10$ and 100) quickly approach the true value:

```
>>> [simpson_rule(f, 1, 3, 10**n) for n in range(1,3)]

   [1.0986605986605984, 1.0986122939305363]
```

Indeed, the following convincingly illustrates that the error in Simpson's rule is $O(h^4)$.

```
>>> from math import log

>>> [simpson_rule(f, 1, 3, 10**n) - log(3) for n in range(1,6)]

   [4.830999248861545e-05, 5.262426494567762e-09, 5.282441151166495e-13, -
   1.5543122344752192e-15, -7.105427357601002e-15]
```