8. Recurrence in reverse:

In [16]:

```
from scipy import integrate
import numpy as np
import math

y_new = 0  #to be used as y_n
y_old = 0  #to be used as y_(n+1)
n = 32  #starting from y_32 as we found that taking N = 32 would yield the desir
ed y_20

for i in range(n,20,-1):  #the loop is till i=21 as we want to find y_20
    y_new = (math.exp(1) - y_old)/i  #the reverse recurrence.
    y_old = y_new
print(y_old)
```

```
0.12380383076256993
```

Trying the definite integral using Scipy's inbuilt integrate.quad routine.

In [17]:

```
n = 20  #as we want y_20
f = lambda x: (x**n)*np.exp(x)  #defining our function
integrate.quad(f, 0, 1)
```

Out[17]:

```
(0.12380383076256998, 1.6808102031436923e-11)
```

Therefore, the value from our algorithm and the inbuilt routine are as follows:

(i) Value$_{algo}$ = 0.12380383076256998 (ii) Value$_{scipy}$ = 0.12380383076256993

Relative error = (Value$_{algo}$ - Value$_{scipy}$)/Value$_{scipy}$ = 4.483799159471622e-16

This is what we wanted in our problem, ie, an error in y_{20} of 2^{-52} or 10^{-16}. Thus, starting from N=32 without knowing the value of y_{32} but approximating it to be 0, we have converged to a highly accurate value of y_{20}.