```
import numpy as np
import pandas as pd
```

1- Write the program for Heun's Method.

```
def heun_method(f, t_initial, t_final, y_initial, h):
    num_steps = int((t_final - t_initial) / h)
    t_values = [t_initial]
    y_values = [y_initial]
    print('\n-------SOLUTION-------')
    print('----------------------')
    print('#\ttn\tyn')
    print('----------------------')
    for i in range(num_steps):
        k1 = f(t_values[i], y_values[i])
        k2 = f(t_values[i] + h, y_values[i] + h * k1)
        y_new = y_values[i] + (h/2.0) * (k1 + k2)
        t_new = t_values[i] + h
        y_values.append(y_new)
        t_values.append(t_new)
        print('%d\t%.2f\t%.4f'% (i+1,t_new,y_new) )
        print('----------------------')
    return t_values, y_values
```

```
def f(t,y):
  return np.exp(t)-y

t_h,y_h=heun_method(f, t_initial=0, t_final=1, y_initial=1,h= 0.1)
```

```
-------SOLUTION-------
----------------------
#       tn      yn
----------------------
1       0.10    1.0053
----------------------
2       0.20    1.0206
----------------------
3       0.30    1.0461
----------------------
4       0.40    1.0820
----------------------
5       0.50    1.1288
----------------------
6       0.60    1.1869
----------------------
7       0.70    1.2568
----------------------
8       0.80    1.3393
----------------------
9       0.90    1.4352
----------------------
10      1.00    1.5454
----------------------
```

2- Write the code to make a table to compare error of Heun's, Euler's, RK4 by using analytical solution.

```
def euler(f, y0, t0, tf, h):

    t = np.arange(t0, tf + h, h)
    y = np.zeros_like(t)
    y[0] = y0
    for i in range(1, len(t)):
        y[i] = y[i-1] + h*f(t[i-1], y[i-1])
    return t, y

def heun(f, y0, t0, tf, h):
    t = np.arange(t0, tf + h, h)
    y = np.zeros_like(t)
    y[0] = y0
    for i in range(1, len(t)):
        k1 = f(t[i-1], y[i-1])
        k2 = f(t[i], y[i-1] + h*k1)
```

```
        y[i] = y[i-1] + h*0.5*(k1 + k2)
    return t, y

def rk4(f, y0, t0, tf, h):
    t = np.arange(t0, tf + h, h)
    y = np.zeros_like(t)
    y[0] = y0
    for i in range(1, len(t)):
        k1 = h*f(t[i-1], y[i-1])
        k2 = h*f(t[i-1] + 0.5*h, y[i-1] + 0.5*k1)
        k3 = h*f(t[i-1] + 0.5*h, y[i-1] + 0.5*k2)
        k4 = h*f(t[i-1] + h, y[i-1] + k3)
        y[i] = y[i-1] + (1.0/6.0)*(k1 + 2*k2 + 2*k3 + k4)
    return t, y


def my_func(f, y0, t0, tf, h, analytical_solution):
    t, y_euler = euler(f, y0, t0, tf, h)
    t, y_heun = heun(f, y0, t0, tf, h)
    t, y_rk4 = rk4(f, y0, t0, tf, h)
    y_true = analytical_solution(t)

    table = pd.DataFrame({'t': t, 'Euler': y_euler, 'Heun': y_heun, 'RK4': y_rk4, 'True val': y_true})
    # true value error = modulus(true val - approx)
    table['Euler-Error'] = np.abs(table['Euler'] - table['True val'])
    table['Heun-Error'] = np.abs(table['Heun'] - table['True val'])
    table['RK4-Error'] = np.abs(table['RK4'] - table['True val'])
    return table
```

Ex 5.4 Q2,10,14 (a part) page number: 291-292

```
def f(t, y):
    return np.exp(t - y)

def analytical_solution(t):
    return np.log(np.exp(t) + np.exp(1) - 1)

my_func(f, 1, 0, 1, 0.5, analytical_solution)
```

	t	Euler	Heun	RK4	True val	Euler-Error	Heun-Error	RK4-Error
0	0.0	1.000000	1.000000	1.000000	1.000000	0.000000	0.000000	0.000000
1	0.5	1.183940	1.218126	1.214041	1.214023	0.030083	0.004103	0.000018
2	1.0	1.436252	1.497555	1.489921	1.489880	0.053628	0.007674	0.000041

Ex 5.4 Q1,9,13 (b part) page number: 291-292

```
def analytical_solution(t):
    return t + 1/(1-t)

def f(t, y):
    return 1 + (t-y)**2

my_func(f, 1, 2, 3, 0.5, analytical_solution)
```

	t	Euler	Heun	RK4	True val	Euler-Error	Heun-Error	RK4-Error
0	2.0	1.000	1.000000	1.000000	1.000000	0.000000	0.000000	0.000000
1	2.5	2.000	1.812500	1.833323	1.833333	0.166667	0.020833	0.000010
2	3.0	2.625	2.481553	2.499971	2.500000	0.125000	0.018447	0.000029

Example 1 on page 269

```
def f(t, y):
    return y -t**2 + 1

def analytical_solution(t):
```

```
    return (t+1)**2 - 0.5 * np.exp(t)

my_func(f, 0, 0, 1, 0.2, analytical_solution)
```

	t	Euler	Heun	RK4	True val	Euler-Error	Heun-Error	RK4-Error
0	0.0	0.000000	0.000000	0.000000	0.500000	0.500000	0.500000	0.500000
1	0.2	0.200000	0.216000	0.218593	0.829299	0.629299	0.613299	0.610705
2	0.4	0.432000	0.462720	0.468167	1.214088	0.782088	0.751368	0.745920
3	0.6	0.686400	0.729318	0.737869	1.648941	0.962541	0.919622	0.911072
4	0.8	0.951680	1.002568	1.014442	2.127230	1.175550	1.124661	1.112787
5	1.0	1.214016	1.266334	1.281697	2.640859	1.426843	1.374526	1.359162

Ex 5.4 Q1,9,13 (a part) page number: 291-292

```
import numpy as np

def f(t, y):
    return t * np.exp(3*t) - 2*y

def y_analytical(t):
    return (1/5)*t*np.exp(3*t) - (1/25)*np.exp(3*t) + (1/25)*np.exp(-2*t)


my_func(f, 0, 0, 1, 0.5, y_analytical)
```

	t	Euler	Heun	RK4	True val	Euler-Error	Heun-Error	RK4-Error
0	0.0	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
1	0.5	0.000000	0.560211	0.296997	0.283617	0.283617	0.276595	0.013381
2	1.0	1.120422	5.301490	3.314312	3.219099	2.098677	2.082390	0.095212

✓ 0s completed at 5:28 PM