```python
from ODES.Cauchy_problem import Cauchy_problem
from ODES.Temporal_schemes import Euler, Inverse_Euler, Crank_Nicolson, Embedded_RK
from Physics.Orbits import Kepler
import matplotlib.pyplot as plt
from numpy import array


from numpy import linspace


def Simulation(tf, N, U0):

    """
    This is a Simulation code to integrate Cauchy problems with some numerical scheme.

    The objective of this Milestone is to create different functions or abstractions
    by means of functional programming or function composition.
    Namely, the following modules are created to allow this functional composition:
            1) ODES.Cauhy_problem
            2) ODES.Temporal_schemes
            3) Physics.Orbits

    The idea is to create a Cauchy problem abstraction to integrate different physical
    problems with different temporal schemes.

    Different abstractions :
                    1) dU/dt = F(U, t)
                    2) Temporal scheme to integrate one step
                    3) Cauchy problem to perform different steps

     """


    t = linspace(0, tf, N)
    schemes = [  (Euler, None, None ),  (Embedded_RK, 2, 1e-1), (Embedded_RK, 8, 1e-1)
    for (method, order, eps)  in schemes:

       if order != None:
          U =  Cauchy_problem( Kepler, t, U0, method, q=order, Tolerance=eps )
       else:
          U =  Cauchy_problem( Kepler, t, U0, method )

       plt.axes().set_aspect('equal')
       plt.plot( U[:,0] , U[:,1], ".")
       plt.show()

if __name__ == "__main__":

  Simulation(100, 100, array( [ 1., 0., 0., 1. ] ) )
```

Importamos de otros códigos los esquemas temporales, la función de estado que define las ecuaciones de movimiento del sistema y el problema de Cauchy que resolverá numericamente

Es la función principal que corre la simulación
tf es el tiempo final de la simulación
N el numero de pasos de tiempo
Uo es la condición inicial de nuestro sistema

Se discretion e el tiempo, creamos un array de tiempos t que va desde o hasta tf dividido en N pasos

Aquí definimos los sistemas de integración que se usarán
Euler (sin orden ni tolerancia)
RK embebido con órdenes 2 y 8 y una tolerancia de 10^{-1}

Método · orden · Tolerancia error

Iteramos sobre los esquemas temporales

Cada solución U es grafía cada,mostrando la trayectoria del cuerpo del espacio (orbita)

Aquí se ejecuta la simulación con:
Tiempo final de 100 unidades
100pasos de tiempo
Estado inicial Uo= 1,0,0,1

Este código realiza una simulación numérica de un problema de orbitas usando diferentes esquemas de integración temporal