```python
#! python3.7
# -*- coding: utf-8 -*-
from numpy import zeros, linspace, exp
from matplotlib.pyplot import style, figure, axes
from celluloid import Camera

# Набор команд, за счёт которых анимация строится в отдельном окне
from IPython import get_ipython
get_ipython().run_line_magic('matplotlib', 'qt')

# Определение функции, задающей начальное условие
def u_init(x) :
    u_init = -x + 1.
    return u_init

# Определение функции, задающей левое граничное условие
def u_left(t) :
    u_left = exp(-t)
    return u_left

# Определение входных данных задачи
a = 0.; b = 1.
t_0 = 0.; T = 0.3

# Определение числа интервалов пространственно-временной сетки,
# на которой будет искаться приближённое решение
N = 200; M = 300

# Определение сетки по пространству
h = (b - a)/N; x = linspace(a,b,N+1)
# Определение сетки по времени
tau = (T - t_0)/M; t = linspace(t_0,T,M+1)

# Выделение памяти под массив сеточных значений решения УЧП
# В строке с номером m этого массива будут храниться сеточные значения решения,
# соответствующие моменту времени t_m
u = zeros((M + 1,N + 1))

# Задание начального условия (на начальном временном слое)
for n in range(N + 1) :
    u[0,n] = u_init(x[n])

# Задание граниченого условия
for m in range(1,M + 1) :
    u[m,0] = u_left(t[m])

# Реализация схемы бегущего счёта
for m in range(M) :
    # Вычисление решения на новом временном слое t_{m+1}
    for n in range(1,N + 1):
        # Поиск u(x_n,t_{m+1}) методом Ньютона
        z = u[m + 1,n - 1]
        for s in range(7) :
            z = z - ((z - u[m,n])/tau + z*(z - u[m+1,n-1])/h - exp(z**2))\
                    /(1/tau + (2*z-u[m+1,n-1])/h-2*z*exp(z**2))
        u[m + 1,n] = z

# Анимация отрисовки решения
style.use('dark_background')
fig = figure()
camera = Camera(fig)
ax = axes(xlim=(a,b), ylim=(0.,3.))
ax.set_xlabel('x'); ax.set_ylabel('u')
for m in range(M + 1) :
    # Отрисовка решения в момент времени t_m
    ax.plot(x,u[m], color='y', ls='-', lw=2)
    camera.snap()
animation = camera.animate(interval=15, repeat=False, blit=True)

```

```
70  # Листинг программы, реализущей решение нелинейного уравнения переноса
71  # по схеме бегущего счёта
```