```
# ========================================================================== #
# RESOLUTION D'UNE EQUATION DIFFERENTIELLE DU PREMIER ORDRE : y' = f(y,x). #
# METHODE DE RUNGE-KUTTA (ORDRE 4).                                        #
# Version du 07/12/2014.                                                   #
# ========================================================================== #

# Importation des bibliothèques/modules requis.
# =============================================
import numpy as np
import matplotlib.pyplot as plt

# La fonction donnant la dérivée y' en fonction de y et de la variable de référence
x.
#
=============================================================================================
def f(y,x):
    return x + y

# Le coeur de l'algorithme : le schéma numérique proprement dit.
# ==============================================================
def RK4(y0,a,b,n,s):
    # Le pas
    h = (b-a)/n
    # Initialisation du tableau des abscisses grâce à la fonction linspace de numpy.
    x = np.linspace(a,b,n+1)
    # Initialisation du tableau des ordonnées (n+1 valeurs car il y a n pas ...).
    # Ce tableau numpy est initialisés avec des 0.
    # Remarque : on pouvait également travailler avec des listes Python "classiques".
    y = np.zeros(n+1)
    y[0] = y0
    # Le schéma numérique.
    for i in range(n):
        k1 = f(y[i],x[i])
        y_mid = y[i] + h * k1 / 2
        x_mid = x[i] + h / 2
        k2 = f(y_mid,x_mid)
        y_mid2 = y[i] + h * k2 / 2
        k3 = f(y_mid2,x_mid)
        k4 = f(y[i] + h * k3,x[i])
        y[i+1] = y[i] + h * (k1 + 2 * k2 + 2 * k3 + k4) / 6
    # Génération des points pour l'affichage graphique.
    plt.plot(x,y,s,label=str(n)+' pas')
    # A l'issue du calcul des n points, on affiche y[n], c'est à dire la valeur
approchée de y(b).
    print('Approximation de y(' + str(b) + ') par la méthode de Runge-Kutta (ordre 4)
(' + str(n) + ' pas) :',y[n])

# Programme principal.
# ====================

# Initialisations
y0, x0, b = 1, 0, 8

print('Résolution de y\'(x)=x+y(x) avec la condition initiale : y(' + str(x0) + ')='
+ str(y0))

# On efface le graphique précédent ...
plt.clf()

# Génération des nuages de points avec différents nombres de pas
RK4(y0,x0,b,10,'ro')  # points rouges
RK4(y0,x0,b,100,'go') # points verts
RK4(y0,x0,b,400,'yo') # points jaunes

# Solution exacte
print('Valeur exacte de y(' + str(b) + ') : ' + str(-1-b+2*np.exp(b)))
x = np.linspace(x0,b,201)
```

```
plt.plot(x,-1 -x + 2 * np.exp(x),label='Solution exacte')

# Titre
plt.title('Méthode de Runge-Kutta (ordre 4)')
plt.legend(loc = 'upper left')

# Affichage graphique
plt.show()

# Fin du programme principal.
# ===========================
```