

Simulations numériques de marches aléatoires : programmes en Python

Pour une bonne compréhension, ces programmes doivent être étudiés successivement. Il est important d'exécuter le code Python et même de tester des petites modifications.

Génération de nombres aléatoires

01_random.py

```
#!/usr/bin/python
# -*- coding: utf-8 -*-
"""
cf. documentation cf http://docs.python.org/library/random.html
random number generation - génération de nombres aléatoires
functions of interest : choice, randint, seed
"""

from random import *

facepiece = ['pile', 'face']
valeurpiece = [0.01, 0.02, 0.05, 0.1, 0.2, 0.5, 1., 2.]

for i in range(1):
    # choice : random choice of an element from a list
    print(choice(facepiece), choice(valeurpiece))
    # randint : return a random integer number between 2 values
    (including limits)
    print(randint(0,10)) # imprime un nombre aléatoire entre 0 et
10
    print(choice(range(0,11,1))) # same function, using choice and
range to create the list

# seed(ANY_DATA) : seeding of the random number generator with any
(constant) data
# in order to generate reproducible random sequences.
# seed() - without data - uses internal clock value to "randomly"
initiate the generator !

for j in range(3):
    #seed('ma chaîne personnelle') # reproducible initialization
    seed() # to randomly initiate the generator
```

```
for i in range(10):  
    print(randint(1000,9999))  
print(" ")
```

Histogrammes de nombres aléatoires

```
<sxh python; title : 02_random_histogram.py> #!/usr/bin/env python # -*- coding: utf-8 -*-
```

```
from random import * # cf. documentation cf http://docs.python.org/library/random.html import  
numpy as np import matplotlib.pyplot as plt #  
http://matplotlib.sourceforge.net/api/pyplot\_api.html#module-matplotlib.pyplot import  
matplotlib.mlab as mlab #  
http://matplotlib.sourceforge.net/api/mlab\_api.html#module-matplotlib.mlab
```

```
#seed('ma chaîne personnelle') # reproducible initialization seed()
```

```
rval=[] for j in range(10000):
```

```
    rval.append(randint(0,99)) # append to the list a random (integer)  
    number between 0 and 99
```

```
# print rval # uncomment just to see the list of random numbers
```

```
# analysis - histogram - see http://matplotlib.sourceforge.net/examples/api/histogram\_demo.html #  
http://fr.wikipedia.org/wiki/Histogramme xh=np.array(rval) # see  
http://www.scipy.org/Cookbook/BuildingArrays transforme une liste en un tableau numérique de  
Numpy # print xh
```

```
fig = plt.figure() ax = fig.add_subplot(111)
```

```
n, bins, patches = ax.hist(xh, 10, facecolor='green', alpha=0.75) print n # les nombres d'occurrences  
par classe print bins # les classes, de largeur identique
```

```
# modifier le nombre de nombres générés, les nombres de classes-bins,
```

```
plt.show() </sxh>
```

Représenter le déplacement d'un objet

```
<sxh python; title : 03_tkinter_simple_move.py> #!/usr/bin/python # -*- coding: utf-8 -*-
```

```
from Tkinter import * import time
```

```
window = Tk() sizex=400 sizey=100 canvas = Canvas(window, width = sizex, height = sizey)  
canvas.pack() x = 100 # initial left-most edge of first ball y = 30 # initial top-most edge of first ball  
r=20 # ball diameter depx=2 # displacement at each move in x direction depy=0 # displacement at
```

each move in y direction

```
ball=canvas.create_oval(x,y,x+r,y+r,fill="blue")
```

```
#moves no_moves=10 for j in range(no_moves):
```

```
    canvas.move(ball, depx, depy)
    canvas.after(10)          # time delay in milliseconds
    canvas.update()
```

```
time.sleep(5) # on attend quelques secondes window.destroy()
```

```
</sxh>
```

Représenter le déplacement de nombreux points

```
<sxh python; title : 04_tkinter_many_moves.py> #!/usr/bin/python # -*- coding: utf-8 -*-
```

```
from Tkinter import * import time
```

```
window = Tk() sizex=400 sizey=600 canvas = Canvas(window, width = sizex, height = sizey)
canvas.pack() x = 100 # initial left-most edge of first ball y = 30 # initial top-most edge of first ball
r=20 # ball diameter depx=2 # displacement at each move in x direction depy=0 # displacement at
each move in y direction
```

```
# create balls: no_particles= 20 dy = (sizey-2.)/(no_particles+1) # y initial separation between balls
print dy ball_list=[] for i in range(no_particles):
```

```
    ball=canvas.create_oval(x,y,x+r,y+r,fill="blue")
    y = y+dy
    ball_list.append(ball)
```

```
#moves no_moves=100 for j in range(no_moves):
```

```
    for ball in ball_list:
        canvas.move(ball, depx, depy)
    canvas.after(10)
    canvas.update()
```

```
time.sleep(5) # on attend quelques secondes window.destroy() </sxh>
```

Marche aléatoire d'un petit nombre de pas

```
<sxh python; title : 05_tkinter_random_walk_few_steps_1D.py> #!/usr/bin/env python # -*- coding:
utf-8 -*-
```

```
from Tkinter import * from random import choice # http://docs.python.org/library/random.html import
numpy as np import matplotlib.pyplot as plt #
```

```
http://matplotlib.sourceforge.net/api/pyplot\_api.html#module-matplotlib.pyplot import
matplotlib.mlab as mlab #
http://matplotlib.sourceforge.net/api/mlab\_api.html#module-matplotlib.mlab
```

```
window = Tk() sizex=200 sizey=600 canvas = Canvas(window, width = sizex, height = sizey)
canvas.pack() x = 100 # initial left-most edge of first ball y = 1 # initial top-most edge of first ball
r=4 # ball diameter depx=10 # displacement at each move in x direction depy=0
```

```
# create balls: no_particles= 100 dy = (sizey-2.)/(no_particles+1) # y initial separation between balls
print dy ball_list=[] for i in range(no_particles):
```

```
    ball=canvas.create_oval(x,y,x+r,y+r,fill="red")
    y = y+dy
    ball_list.append(ball)
```

```
#moves no_moves=4 # number of moves for j in range(no_moves):
```

```
    for ball in ball_list:
        canvas.move(ball, choice([-1,1])*depx, depy)
    canvas.after(1)
    canvas.update()
```

```
#analysis - histogram # see http://matplotlib.sourceforge.net/examples/api/histogram\_demo.html
xpos=[] for ball in ball_list:
```

```
    posi=canvas.coords(ball)
    xpos.append(((no_moves+1.)/no_moves)*(posi[0]-x)/depx)
    # le facteur (no_moves+1.)/no_moves permet de gérer la largeur des barres
    de l'histogramme
```

```
xh=np.array(xpos) # see http://www.scipy.org/Cookbook/BuildingArrays #print xh
```

```
fig = plt.figure() ax = fig.add_subplot(111) n, bins, patches = ax.hist(xh, (no_moves)+1,
facecolor='green', alpha=0.75) print n,bins, patches
```

```
plt.show()
```

```
#window.mainloop()
```

```
</sxh>
```

Marche aléatoire d'un grand nombre de pas

```
<sxh python; title : 06_tkinter_random_walk_many_steps_1D.py> #!/usr/bin/env python # -*- coding:
utf-8 -*-
```

```
from Tkinter import * from random import choice # http://docs.python.org/library/random.html import
numpy as np import matplotlib.pyplot as plt #
```

http://matplotlib.sourceforge.net/api/pyplot_api.html#module-matplotlib.pyplot import
 matplotlib.mlab as mlab #
http://matplotlib.sourceforge.net/api/mlab_api.html#module-matplotlib.mlab

```
window = Tk() sizex=400 sizey=400 canvas = Canvas(window, width = sizex, height = sizey)
canvas.pack() x = 200 # initial left-most edge of first ball y = 1 # initial top-most edge of first ball
r=4 # ball diameter depx=1 # displacement at each move in x direction depy=0
```

```
# create balls: no_particles= 2000 dy = (sizey-2.)/(no_particles+1) # y initial separation between
balls print dy ball_list=[] for i in range(no_particles):
```

```
ball=canvas.create_oval(x,y,x+r,y+r,fill="blue")
y = y+dy
ball_list.append(ball)
```

```
#moves no_moves=1000 for j in range(no_moves):
```

```
for ball in ball_list:
    canvas.move(ball, choice([-1,1])*depx, depy)
    canvas.after(1)
    canvas.update()
```

```
#analysis - histogram # see http://matplotlib.sourceforge.net/examples/api/histogram\_demo.html
xpos=[] for ball in ball_list:
```

```
posi=canvas.coords(ball)
xpos.append((posi[0]-x)/depx)
```

```
xh=np.array(xpos) # see http://www.scipy.org/Cookbook/BuildingArrays # compute the mean mu and
sigma from xh (and/or theoretical value from random walk result) mu=np.mean(xh) sigma=np.std(xh)
fig = plt.figure() ax = fig.add_subplot(111) # print xh n, bins, patches = ax.hist(xh, 10,
facecolor='green', alpha=0.75) print n,bins, patches # hist uses np.histogram to create 'n' and 'bins'.
cf. http://docs.scipy.org/doc/numpy/reference/generated/numpy.histogram.html
```

```
ax.set_xlabel('X positions') ax.set_ylabel('Occurences')
```

```
ax.grid(True)
```

```
plt.show()
```

```
#window.mainloop() </sxh>
```

Avec analyse de la distribution :

```
<sxh python; title : 07_tkinter_random_walk_many_steps_1D-analysis.py> #!/usr/bin/env python # -*-
coding: utf-8 -*-
```

```
from Tkinter import * from random import choice # http://docs.python.org/library/random.html import
numpy as np import matplotlib.pyplot as plt #
http://matplotlib.sourceforge.net/api/pyplot\_api.html#module-matplotlib.pyplot import
```

matplotlib.mlab as mlab #

http://matplotlib.sourceforge.net/api/mlab_api.html#module-matplotlib.mlab

```
window = Tk() size_x=400 size_y=400 canvas = Canvas(window, width = size_x, height = size_y)
canvas.pack() x = 200 # initial left-most edge of first ball y = 1 # initial top-most edge of first ball
r=4 # ball diameter dep_x=1 # displacement at each move in x direction dep_y=0
```

```
# create balls: no_particles= 1000 dy = (size_y-2)/(no_particles+1) # y initial separation between
balls #print dy ball_list=[] for i in range(no_particles):
```

```
    ball=canvas.create_oval(x,y,x+r,y+r,fill="blue")
    y = y+dy
    ball_list.append(ball)
```

```
#moves no_moves=900 for j in range(no_moves):
```

```
    for ball in ball_list:
        canvas.move(ball, choice([-1,-1,-1,-1,-1,1,1,1,1,1])*dep_x, dep_y)
    canvas.after(1)
    canvas.update()
```

```
#analysis - histogram # see http://matplotlib.sourceforge.net/examples/api/histogram\_demo.html
xpos=[] for ball in ball_list:
```

```
    posi=canvas.coords(ball)
    xpos.append(posi[0]-x)
```

```
xh=np.array(xpos) # see http://www.scipy.org/Cookbook/BuildingArrays # compute the mean mu and
sigma from xh (and/or theoretical value from random walk result) mu=np.mean(xh) sigma=np.std(xh)
fig = plt.figure() ax = fig.add_subplot(111) # print xh n, bins, patches = ax.hist(xh, 20,
facecolor='green', alpha=0.75) print mu, sigma print n,bins, patches # hist uses np.histogram to
create 'n' and 'bins'. # np.histogram returns the bin edges, so there will be ii probability # density
values in n, ii+1 bin edges in bins and ii patches. To get # everything lined up, we'll compute the bin
centers bincenters = 0.5*(bins[1:]+bins[:-1]) # add a 'best fit' line for the normal PDF yh = (bins[1]-
bins[0])*no_particles*mlab.normpdf( bincenters, mu, sigma) #
http://matplotlib.sourceforge.net/api/mlab\_api.html#matplotlib.mlab.normpdf l = ax.plot(bincenters,
yh, 'r-', linewidth=1) #print n ax.set_xlabel('X positions') ax.set_ylabel('Occurences')
```

```
ax.grid(True)
```

```
plt.show()
```

```
#window.mainloop() </sxh>
```

From:

<https://dvillers.umons.ac.be/wiki/> - **Didier Villers, UMONS - wiki**

Permanent link:

https://dvillers.umons.ac.be/wiki/teaching:exos:simulations_random_walks_codes?rev=1541414113

Last update: **2018/11/05 11:35**

