Python G3 - DAD

TP3

To create a function:

```
def carre_cube(n):
    return n**2,n**3

a,b = carre_cube(5)
a,b
```

Note that you don't need to give names to outputs: just get what `return` produces.

One can also get both outputs in one tuple only:

```
c = carre_cube(5)
c
```

You can also affect default values to parameters:

```
def oiseau(voltage=100, etat="allume", action="danser la java"):
    print("Ce perroquet ne pourra pas", action)
    print("si vous le branchez sur", voltage, "volts !")
    print("L'auteur de ceci est completement", etat)
```

In that case, you can give input parameters in any order as in:

```
oiseau(etat="givre", voltage=250, action="vous approuver")|
```

Exercise 1 *Brownian motion*
The main purpose of this work is to implement the simulation of a Brownian motion in 2 dimensions. Then we will prepare a movie which illustrates its behaviour. To this aim, we will work within the unitary disk $\mathcal{D}$ of radius 1 and centered in 0. The Brownian motion describes a random walk with independent and identically distributed Gaussian increments. It has been used as a model for the displacement of a big particle through a media made of a large number of small particles. It is also connected to the diffusion process (Einstein).

Let $\mathcal{D}$ the 2 dimensional disk of radius 1 and cenered in 0.
We define the following simulation scheme of a random walk in discrete time. For any $\mathbf{x} = (x_1, x_2) \in \mathcal{D}$, the initial condition is

$$\mathbf{W}_0 = \mathbf{x}, \qquad \text{(in 2 dimensions).} \tag{1}$$

Let δ a sufficiently small time step (for instance $\delta = 0.001$). We will work with the following recursive definition of a discrete time approximation of the Brownian motion. At time $n\delta$:

$$\mathbf{W}_n = \mathbf{W}_{n-1} + \sqrt{\delta}\mathbf{G}_n, \quad n = 1, 2, \dots \tag{2}$$

where $\mathbf{G}_n$ is a family of independent random variable distributed according to a standardized normal law $\mathcal{N}(0, \mathbf{I})$ (variance equal to 1 along x and y axes).

1. Make a function

   `def brownianmotion(X,delta,...):...`

   which

   - simulates $\mathbf{W_x}$ from some $\mathbf{x} \in \mathcal{D}$ until $\mathbf{W_x}$ reaches the boundary of $\mathcal{D}$,
   - interpolates linearly between the two last positions to determine where the trajectory actually crossed the boundary,
   - yields both the whole random walk `W` and this borderline point `Xinter` as an output.

2. Show an example of a Brownian trajectory originating from $\mathbf{x} = (0.2, 0.4)$ (until it reaches the boundary of $\mathcal{D}$). Display the unit circle on the same figure.

3. Represent on the same figure a reasonable collection of Brownian trajectories originating from the same point $\mathbf{X}$ until they reach the boundary of $\mathcal{D}$.

Exercise 2 *2D graphics & FFT to filter images*
We are going to see what happens when we separate low-frequencies from high-frequencies in an image. To begin with, load the image called `stinkbug.png`, and show it with its corresponding colormap.

1. Try various colormaps using `colormap`.

2. Each inner list represents a pixel. Here, with an RGB image, there are 3 values. Since it?s a black and white image, R, G, and B are all similar. We currently have an RGB image. Since R, G, and B are all similar, we can just pick one channel of our data: `lum_img = img[:,:,0]`

3. Compute the discrete 2D Fourier transform of this image and show its modulus in logarithmic scale.
   `https://docs.scipy.org/doc/numpy/reference/routines.fft.html`

   (a) Where is the null frequency ?

   (b) Read help about `fftshift` and use it before representing the image FFT again. Where is the null frequency now ?

4. Create a function `lowpassfilter2D.py` in 2D for an image which keeps low frequencies below some cut frequency `fc` only and puts others to zero (in the Fourier domain). Be careful of the use of `fft` and `ifft` as well as `fftshift` and `ifftshift`.The cut frequency will be chosen equal to 25 for instance.

Useful : for images, use mpimg & PIL (for pillow to read/write images in various formats)

Natively, matplotlib only supports PNG images. The commands shown below fall back on Pillow if the native read fails.

$\longrightarrow$ `https://matplotlib.org/users/image_tutorial.html`

```
import matplotlib.image as mpimg
```

```
from PIL import Image
```

See also: `mpimg.imread,plt.imshow, plt.colorbar(), Image.open, np.fft.fft2, ...`