Marzo 15 de 2018.

Para alternar entre versiones de python:

Es posible decirle a python cuál versión queremos usar (e.g. 2.7 , 3.6).
Para ello le pedimos a conda que cree ambientes adicionales:

```
$ conda info --envs # Checar los ambientes existentes
$ conda create -n py27 python=2.7 anaconda
$ source activate py27 # To activate this environment
$ source deactivate # To deactivate
```

Para saber en cuál ambiente estoy en un momento determinado:

```
$ python --version
```

Es importante estar en el ambiente correcto antes de lanzar ipython o jupyter notebook, pues de otra manera las cosas no van a funcionar como esperamos

Ejercicios de la clase:

https://github.com/juan-pineda/Modelado-1

Los ejercicios se pueden descargar para correr localmente en jupyter notebook, o también se pueden rodar online en la plataforma: https://mybinder.org

En el repositorio dejé un archivo llamado tests.ipybn, que pueden utilizar como comodín.

NUMPY

Para ganar destreza en el uso de numpy, a continuación se ofrecen una serie de pequeños retos que les invito a resolver con las herramientas a su alcance.

Neophyte

1. Import the numpy package under the name np
2. Print the numpy version and the configuration.
3. Create a null vector of size 10
4. Create a null vector of size 10 but the fifth value which is 1
5. Create a vector with values ranging from 10 to 99
6. Create a 3x3 matrix with values ranging from 0 to 8
7. Find indices of non-zero elements from [1,2,0,0,4,0]
8. Declare a 3x3 identity matrix
9. Declare a 5x5 matrix with values 1,2,3,4 just below the diagonal
10. Declare a 10x10x10 array with random values

Novice

1. Declare a 8x8 matrix and fill it with a checkerboard pattern
2. Declare a 10x10 array with random values and find the minimum and maximum values
3. Create a checkerboard 8x8 matrix using the tile function
4. Normalize a 5x5 random matrix (between 0 and 1)
5. Multiply a 5x3 matrix by a 3x2 matrix (real matrix product)
6. Create a 10x10 matrix with row values ranging from 0 to 9
7. Create a vector of size 1000 with values ranging from 0 to 1, both excluded
8. Create a random vector of size 100 and sort it
9. Consider two random matrices A anb B, check if they are equal.
10. Create a random vector of size 1000 and find the mean value

Apprentice

1. Consider a random 100x2 matrix representing cartesian coordinates, convert them to polar coordinates
2. Create random vector of size 100 and replace the maximum value by 0
3. Declare a structured array with x and y coordinates covering the [0,1]x[0,1] area.
4. Print the minimum and maximum representable value for each numpy scalar type
5. Create a structured array representing a position (x,y) and a color (r,g,b)
6. Consider a random vector with shape (100,2) representing coordinates, find point by point distances
7. Generate a generic 2D Gaussian-like array
8. Consider the vector [1, 2, 3, 4, 5], how to build a new vector with 3 consecutive zeros interleaved between each value?

FUNCIONES

Se propone el siguiente desafío: Crear una matriz cuadrada (lado impar) con números aleatorios según una distribución normal. Después duplicar los números positivos en la matriz, y reemplazar los números negativos por su valor al cuadrado. Finalmente, ordenar los números de menor a mayor y reportar el que se encuentra justo a la mitad de la lista.

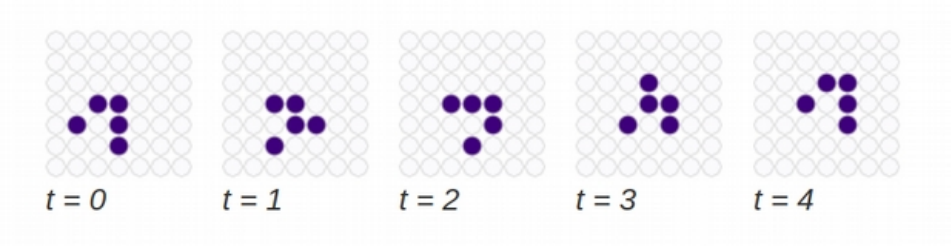
EL JUEGO DE LA VIDA

Esta va a ser la primera tarea de la clase. A continuación describo la idea, y más adelante lo que espero que nos compartan como parte de su tarea.

The universe of the Game of Life is an infinite two-dimensional orthogonal grid of square cells, each of which is in one of two possible states, live or dead. Every cell interacts with its eight neighbours, which are the cells that are directly horizontally, vertically, or diagonally adjacent. At each step in time, the following transitions occur:

1) Any live cell with fewer than two live neighbours dies, as if by needs caused by underpopulation.
2) Any live cell with more than three live neighbours dies, as if by overcrowding.
3) Any live cell with two or three live neighbours lives, unchanged, to the next generation.
4) Any dead cell with exactly three live neighbours becomes a live cell.

The initial pattern constitutes the 'seed' of the system. The first generation is created by applying the above rules simultaneously to every cell in the seed – births and deaths happen simultaneously, and the discrete moment at which this happens is sometimes called a tick. (In other words, each generation is a pure function of the one before.) The rules continue to be applied repeatedly to create further generations.



La idea es que exploren la manera de programar el juego de la vida de forma eficiente utilizando numpy. Pueden jugar con todas las variables a voluntad, explorando variantes y viendo qué pasa. Idealmente deben visualizar el tablero en diferentes momentos, y sería genial si logran concatenar las imágenes en forma de video. La entrega la pueden hacer en el formato que encuentren más conveniente, combinando código, texto, imágenes, o video a voluntad. Comentando un poco en la experimentación que efectuaron y lo que observan a partir de ella.

La idea es que su entrega la podamos compartir entre todos para que haya una realimentación. Es importante por tanto que seamos capaces de rodar (o al menos de entender) sus códigos y eventualmente reproducir sus resultados.

Para escribir sus códigos sería positivo que sigan algunas guías de estilo como las de PEP 8: https://www.python.org/dev/peps/pep-0008/

Plots con matplotlib

En paralelo con la tarea de numpy sugiero que estudien estos links, enfrentando todos los ejemplos con sus propias manos.

https://www.labri.fr/perso/nrougier/teaching/matplotlib/#other-types-of-plots

http://journals.plos.org/ploscompbiol/article?id=10.1371/journal.pcbi.1003833

Notas:

Tutorial de numpy:
http://www.labri.fr/perso/nrougier/teaching/numpy/numpy.html