Initiation à la programmation, Leçon 5 : numpy et pyplot

Numpy

1. **Création d'un array**

   Dans l'exemple suivant, vous créez d'abord une liste Python, et puis un array numpy de cette liste.

```
# Import the numpy package as np
import numpy as np

# Create a new list height
height = [1.87,  1.87, 1.82, 1.91, 1.90, 1.85]

# Create a numpy array from height
np_height = np.array(height)
print np_height
```

   Les tableaux peuvent avoir plusieurs dimensions. Essayez de créer un arrays de la liste des listes, comme [[1,1], [2,4], [3,7]].

2. **Pyramide des âges: Création d'un array**

   Télécharger un fichier age-2013.txt de Moodle, leçon 5. Ce fichier a été généré par l'INSEE et contient des statistiques démographiques pour la France. La première ligne contient les catégories, le sexe et l'âge, séparés par des virgules. Chacune des lignes suivantes contient un nom de la région et le nombre d'habitants de chaque région appartenant à chaque catégorie. Votre tâche consiste à écrire une fonction qui lit le fichier et retourne un tuple de quatre arrays numpy: arr_categories, arr_regions, arr_men, arr_women. Les types de données pour les arrays arr_categories et arr_regions doivent être str, les types de données pour les tableaux arr_men, arr_women doivent être int.

```
import numpy as np
def read_data(filename):
    lines = open(filename).readlines()
    #Your code goes here
    return (arr_categories, arr_regions, arr_men, arr_women)
```

3. **Quelques statistiques sur les arrays**

   Nous avons beaucoup d'opérations unaires, comme le calcul de la somme de tous les éléments d'un array.

```
import numpy as np
weight_kg = np.array([81.65, 97.52, 95.25, 92.98, 86.18, 88.45])
print weight_kg.sum() #sum of all values
print weight_kg.max() #min value
print weight_kg.size() #size
```

   Pour les arrays bidimensionnels, il est possible de calculer des statistiques par lignes ou par colonnes.

```
import numpy as np
weight_height = np.array([[180,81.65], [160,97.52], [200,95.25]])
print weight_height.max(axis=0) #max by columns
print weight_height.max(axis=1) #max by rows
```

4. **Pyramid of ages: Population by region**

   Ecrivez une fonction qui retourne un array `total`. Chaque valeur doit contenir le nombre total d'habitants de la région correspondante en float. Pour convertir les éléments d'un tableau d'int en float, utilisez `astype(float)`.

```
def total_by_region(arr_women, arr_men):
```

```
    #Your code goes here
    total = total.astype(float)
    return total
```

5. **Opérations par élément**

   Les opérateurs arithmétiques, les fonctions mathématiques et les opérateurs booléens sur les arrays s'appliquent par élément. Un nouveau array est créé et rempli avec le résultat.

```
import numpy as np
a = np.array( [20,30,40,50] )
b = np.array([0, 1, 2, 3])
print a-b
print a*b
print 10*np.sqr(a)
print a<35
```

6. **Pyramide des âges: statistiques par région**

   Ecrivez une fonction percentage_by_region qui calcule deux tableaux, le pourcentage d'hommes dans chaque région et le pourcentage de femmes dans chaque région.

7. **Sélectionner des entrées**

   Une autre caractéristique intéressante des arrays est la capacité de sélectionner un sous-ensemble des entrées. Par exemple, si vous voulez savoir quels poids dans notre array weights_kg sont au-dessus de 90, nous pourrions rapidement le faire de la façon suivante.

```
import numpy as np
weight_kg = np.array([81.65, 95.25, 92.98, 86.18,97.52,88.45])
over_90 = weight_kg > 90
print over_90, weight_kg[over_90]
```

8. **Pyramide des âges: Pourcentage d'hommes et de femmes par région**

   Trouvez la région la plus masculine et la plus féminine.

9. **Produit scalaire**

   Imaginez que vous dirigez un magasin de bagel. Vous avez plusieurs recettes de bagels, et pour chacun vous stockez les ingrédients nécessaires dans une table (en kg). Vous connaissez également le prix de chaque ingrédient: le fromage coûte 15 € / kg, le jambon 40 € / kg et la tomate 5 € / kg. Votre tâche consiste à calculer le prix de chaque bagel.

	Fromage	Jambon	Tomate
Bagel 1	0.1	0.1	0.05
Bagel 2	0.05	0.1	0.1
Bagel 3	0.2	0	0.1

   Le prix de Bagel 1 est 15*0.1 + 40*0.1 + 5*0.05 = 5.75. Le prix de Bagel 2 est 15*0.05 + 40*0.1 + 5*0.1 = 5.25. Le prix de Bagel 3 est 15*0.2 + 40*0 + 5*0.1 = 3.5. Nous pouvons écrire les prix comme un array [5.75, 5.25, 3.5]. On dit que cet array est le résultat du produit scalaire du array des ingrédients et du array des prix.

```
import numpy as np
ingredients = np.array([[0.1, 0.1, 0.05], [0.05,0.1,0.1], [0.2,0,0.1]])
prices_ingredients = np.array([15, 40, 5])
prices_bagels = np.dot(ingredients, prices_ingredients)
```

10. **Pyramide des âges: Âge moyen par région**

    En supposant que l'âge de chaque personne dans le groupe 0-19 est de 10, dans le groupe 20-39 est de 30, dans le groupe 40-59 est de 50, dans le groupe 60-74 est de 67 et dans le

groupe 70+ est de 80, calculer la moyenne âge dans chaque région. Trouvez les régions ayant les âges moyens les plus élevés et les plus petits.

Pyplot

11. Pyramide des âges: camembert des groupes d'âge féminin et masculin en Alsace

Voici un exemple de camembert dessiné à l'aide de matplotlib.

```
import matplotlib.pyplot as plt
import numpy as np

plt.figure(1, figsize=(6,6))
labels = 'Frogs', 'Hogs', 'Dogs', 'Logs'
fracs = np.array([15,30,45, 10])
plt.pie(fracs,  labels=labels, autopct='%1.1f%%',shadow=True)
plt.title('Raining Hogs and Dogs')
plt.savefig('raining.png')
plt.show()
```

Votre tâche est de dessiner un camembert des groupes d'âge féminins et masculins en Alsace.

12. Pyramide des âges: Diagramme à barres des populations masculines et féminines selon l'âge

Voici un exemple de diagramme à barres pour l'utilisation du langage de programmation.

```
import matplotlib.pyplot as plt
import numpy as np

languages = np.array(['Python', 'C++', 'Java', 'Perl', 'Scala', 'Lisp'])
number_of_users = np.array([10,8,6,4,2,1])

plt.title('Programming language usage')
x_pos = np.arange(len(languages))
bc = plt.bar(x_pos-0.4, number_of_users, color='b')
plt.xticks(x_pos, languages)
plt.ylabel('Usage')

plt.savefig('programming_languages.png')
plt.show()
```

Votre tâche consiste à desinner un diagramme à barres des populations masculines et féminines selon l'âge. Pour ajouter une légende à votre graphique, utilisez `plt.legend((bc_men[0], bc_women[0]), ('Men','Women'))`.

13. **Tracés de fonctions "scientifiques"**

Dessiner un tracé d'une fonction qui est égal à $-x-5$ sur $[-10,-5]$, à x^2-25 sur $[-5,4]$, et à $\sin(x-4)-9$ sur $[4,10]$.

```
import matplotlib.pyplot as plt
import numpy as np
x = np.linspace(-5,5,101) # 101 equally spaces ticks from -5 to 5
y = np.sin(x)
plt.plot(x,y)
plt.show()
```

14. **(*) L'ensemble de Mandelbrot**

L'ensemble de Mandelbrot est une fractale qui est définie comme l'ensemble des points c du plan

complexe pour lesquels la suite récurrente définie par : $z_{n+1} = z_n^2 + c$ et la condition $z_0 = 0$ ne tend pas vers l'infini (en module). Si nous reformulons cela sans utiliser les nombres complexes, en remplaçant z_n par le couple (x_n, y_n) et c par le couple (a, b) alors nous obtenons: $x_{n+1} = x_n^2 - y_n^2 + a$ et $y_{n+1} = 2x_n y_n + b$.

Il peut être démontré que dès que le module de z_n est strictement plus grand que 2 (z_n étant sous forme algébrique, quand $x_n^2 + y_n^2 > 2$), la suite diverge vers l'infini, et donc c est en dehors de l'ensemble de Mandelbrot. Cela nous permet d'arrêter le calcul pour les points ayant un module strictement supérieur à 2 et qui sont donc en dehors de l'ensemble de Mandelbrot. Pour les points de l'ensemble de Mandelbrot, i.e. les nombres complexes c pour lesquels z_n ne tend pas vers l'infini, le calcul n'arrivera jamais à terme, donc il doit être arrêté après un certain nombre d'itérations déterminé par le programme.

Écrire un script qui affiche (une approximation de) l'ensemble de Mandelbrot.