notebook_exemples_programmes

April 1, 2019

1 Trois exemples de programmes python pour commencer

Avant d'aborder en détail la programmation avec le langage python, voici trois exemples: - 1. le premier calcule la structure d'un atome à partir du numéro atomique et du nombre de masse - 2. le second trace une courbe de décroissance radioactive par simulation de lancés de dès - 3. le troisième propose une animation sur les ondes progressives.

1.1 1. structure de l'atome

A partir du numéro atomique et du nombre de masse, ce programme: - donne le nom et le symbole de l'atome - précise le nombre de protons, neutrons, et électrons - calcule la masse (approchée) du noyau et de l'atome entier - donne la répartition électronique par couche (pour les 18 premiers éléments) - donne la structure électronique selon la règle de Klechkowski (les exeptions n'ont pas été traitées)

Sans entrer dans la compréhension du code, je vous propose d'exécuter le programme suivant:

In [14]:
```
# programme calculant la structure de l'atome

#données
# la masse des protons et des neutrons est approchée:
masse_nucleon = 1.67e-27
# masse de l'électron
masse_electron = 9.109e-31
# liste des orbitales atomiques
liste_orbitales = ( (1, 's', 2),  (2, 's', 2),  (2, 'p', 6),
                    (3, 's', 2),  (3, 'p', 6),  (4, 's', 2),
                    (3, 'd', 10), (4, 'p', 6),  (5, 's', 2),
                    (4, 'd', 10), (5, 'p', 6),  (6, 's', 2),
                    (4, 'f', 14), (5, 'd', 10), (6, 'p', 6),
                    (7, 's', 2),  (5, 'f', 14), (6, 'd', 10),
                    (7, 'p', 6) )
# liste des éléments chimiques de la classification
elements = (
"Hydrogène H", "Hélium He", "Lithium Li", "Béryllium Be",
"Bore B", "Carbone C", "Azote N", "Oxygène O",
"Fluor F", "Néon Ne", "Sodium Na", "Magnésium Mg",
```

```
"Aluminium Al", "Silicium Si", "Phosphore P",
"Soufre S", "Chlore Cl", "Argon Ar", "Potassium K",
"Calcium Ca", "Scandium Sc", "Titane Ti", "Vanadium V",
"Chrome Cr", "Manganèse Mn", "Fer Fe", "Cobalt Co",
"Nickel Ni", "Cuivre Cu", "Zinc Zn", "Gallium Ga",
"Germanium Ge", "Arsenic As", "Sélénium Se",
"Brome Br", "Krypton Kr", "Rubidium Rb",
"Strontium Sr", "Yttrium Y", "Zirconium Zr",
"Niobium Nb", "Molybdène Mo", "Technétium Tc",
"Ruthénium Ru", "Rhodium Rh", "Palladium Pd",
"Argent Ag", "Cadmium Cd", "Indium In", "Étain Sn",
"Antimoine Sb", "Tellure Te", "Iode I", "Xénon Xe",
"Césium Cs", "Baryum Ba", "Lanthane La", "Cérium Ce",
"Praséodyme Pr", "Néodyme Nd", "Prométhium Pm",
"Samarium Sm", "Europium Eu", "Gadolinium Gd",
"Terbium Tb", "Dysprosium Dy", "Holmium Ho",
"Erbium Er", "Thulium Tm", "Ytterbium Yb",
"Lutécium Lu", "Hafnium Hf", "Tantale Ta",
"Tungstène W", "Rhénium Re", "Osmium Os",
"Iridium Ir", "Platine Pt", "Or Au", "Mercure Hg",
"Thallium Tl", "Plomb Pb", "Bismuth Bi", "Polonium Po",
"Astate At", "Radon Rn", "Francium Fr", "Radium Ra",
"Actinium Ac",  "Thorium Th", "Protactinium Pa",
"Uranium U", "Neptunium Np", "Plutonium Pu",
"Américicium Am", "Curium Cm", "Berkélium Bk",
"Californium Cf", "Einsteinium Es", "Fermium Fm",
"Mendélévium Md", "Nobélium No", "Lawrencium Lr",
"Rutherfordium Rf", "Dubnium Db", "Seaborgium Sg",
"Bohrium Bh", "Hassium Hs", "Meitnérium Mt",
"Darmstadtium Ds", "Roentgenium Rg", "Ununbium Uub",
"Ununtrium Uut", "Ununquadium Uuq", "Ununpentium Uup",
"Ununhexium Uuh", "Ununseptium Uus", "Ununoctium Uuo")

# Entrées
# demande des numéros atomiques et nombre de masse
Z = int(input('Entrer le numéro atomique Z =  '))
A = int(input('Entrer le nombre de masse A =  '))

#calcul des masses
masse_noyau = A * masse_nucleon
masse = A * masse_nucleon + Z * masse_electron

# calcul des couches électroniques
couches = ""
if Z <= 2:
    couches = couches + "(K)" + str(Z)
elif Z <=10:
    couches = couches + "(K)" + str(2) \
```

```
                        + "(L)" + str(Z-2)
    elif Z <= 18:
        couches = couches + "(K)" + str(2) \
                        + "(L)" + str(8) \
                        + "(M)" + str(Z-10)
    else:
        couches = "\nce calcul est limité à des numéros " + \
                  "atomiques inférieurs ou égal à 18"


    # calcul des orbitales atomiques
    orbitale = 0
    n_restant = Z
    structure = ""
    while n_restant > 0:
        (n, nom, ne) = liste_orbitales [orbitale]
        if n_restant < ne:
            nmin = n_restant
        else:
            nmin = ne
        structure = structure + str(n) + nom + str(nmin) + ','
        n_restant = n_restant - nmin
        orbitale = orbitale + 1

    # impression des résultats
    print ("\nRESULTATS")
    # impression de l'élément et de son symbole
    print('{:35}'.format("il s'agit de l'élément "), elements[Z-1])
    # impression de la structure de l'atome
    print('{:35}'.format('le nombre de protons est: '), Z )
    print('{:35}'.format('le nombre de neutrons est: '), A - Z)
    print('{:35}'.format("le nombre d'électrons est: "), Z)
    # impression des masses, du noyau et de l'atome
    print('{:35}'.format("la masse du noyau de l'atome est: "),
          "{0:.3e}".format(masse_noyau), ' kg')
    print('{:35}'.format("la masse de l'atome est: "),
          "{0:.3e}".format(masse), ' kg')
    print("")
    # impression des couches et structure électronique
    print('{:65}'.format("le remplissage des couches électroniques donne:"),
          couches)
    print('{:65}'.format("Selon la règle de Klechkowski,"
                         " la structure électronique est: "),
          structure)
```

```
Entrer le numéro atomique Z =  6
Entrer le nombre de masse A =  12
```

```
RESULTATS
il s'agit de l'élément              Carbone C
le nombre de protons est:           6
le nombre de neutrons est:          6
le nombre d'électrons est:          6
la masse du noyau de l'atome est:   2.004e-26  kg
la masse de l'atome est:            2.005e-26  kg

le remplissage des couches électroniques donne:                    (K)2(L)4
Selon la règle de Klechkowski, la structure électronique est:      1s2,2s2,2p2,
```

Vous pouvez bien sur exécuter plusieurs fois le programme, c'est là tout son intérêt.

1.2 2. Simulation d'une décroissance radioactive

Simulons une décroissance radioactive par un lancer de dès: - Un dé à 6 faces représente un atome radioactif. - Il se désintègre quand, lors d'un lancé, il indique 6.

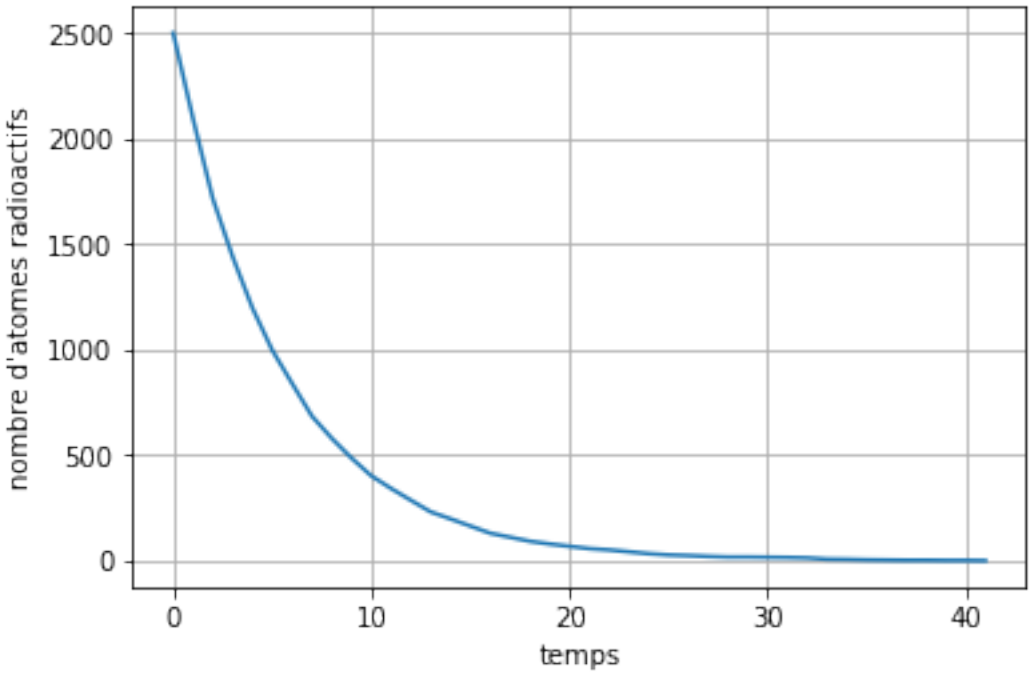
Sur un échantillon de plusieurs dés, la simulation consiste à retirer à chaque lancé les dés ayant indiqué 6. Nous traçons ensuite la courbe donnant le nombre d'atomes radioactifs en fonction du numéro de lancé.

Ici encore, vous êtes juste invités à excécuter le programme.

```
In [15]: # programme de simulation de décroissance radioactive
         # par le jet de dés

         # importations
         import matplotlib.pyplot as plt
         from random import randint

         # entrée du nombre d'atomes radioactifs pour la simulation
         n = int(input("pour combien d'atomes radioactifs "
                       "voulez-vous faire la simulation? : "))

         # initialisation des données
         nombrelance = 0
         temps = [0]
         radioactifs = [n]

         # coeur du programme
         while n > 0:
                 desintegration = 0
                 for i in range(n):
                     tirage = randint(1,6)
                     if tirage == 6:
                         desintegration = desintegration + 1
                 n = n - desintegration
                 nombrelance = nombrelance + 1
```

```
            temps.append(nombrelance)
            radioactifs.append(n)

        # affichage
        plt.figure()
        plt.plot(temps, radioactifs)
        plt.grid()
        plt.xlabel("temps")
        plt.ylabel("nombre d'atomes radioactifs")
        plt.show()
        plt.close()
```

```
pour combien d'atomes radioactifs voulez-vous faire la simulation? : 2500
```



1.3 3. Animation pour une onde progressive

(code sous licence creative commun CC BY-NC-SA BY Gaëlle Rebolini) Ce programme propose l'animation d'une onde progressive à partir des valeurss: - de l'amplitude - de la période - de la longueur d'onde

(Une fois lancée, patientez un peu pour voir l'animation)

```
In [16]: import numpy as np
         import matplotlib.pyplot as plt
         from matplotlib import animation, rc
```

```
Ymax=float(input("Quelle est la valeur de "
                 "l'amplitude de l'onde en m ? : "))
print("L'amplitude de l'onde vaut ",Ymax,"m")
T=float(input("Quelle est la valeur de "
              "la période de l'onde en s ? : "))
print("La période de l'onde vaut ",T,"s")
l=float(input("Quelle est la valeur de la "
              "longueur d'onde de l'onde en m ? : "))
print("La longueur d'onde de l'onde vaut ",l,"m")
print("Calcul de l'animation en cours, "
      "merci de patienter...")
xmin=0
xmax=3*l
nbx=100

fig=plt.figure(figsize=(12,10))
line = plt.plot([], [],'bo-')
plt.xlim(xmin,xmax)
plt.ylim(-Ymax,Ymax)
plt.grid()
plt.xlabel("x(m)")
plt.ylabel("y(m)")
plt.title("animation : propagation d'une "
          "onde le long d'une corde")

def init():
    line[0].set_data([], [])
    return (line)

def animate(i):
    dt=0.03
    t=i*dt
    x = np.linspace(xmin, xmax, nbx)
    y = Ymax*np.cos(2 * np.pi * (x/l - t/T))
    line[0].set_data(x, y)
    return (line)

anim = animation.FuncAnimation(
    fig,
    animate,
    init_func=init,frames=100,
    interval=30,
    blit=True,
    repeat=False)

plt.close()
```

```
        rc('animation', html='jshtml')  # lignes de code à remplacer par plt.show() sur un éd
        anim

        # patience, c'est un peu long à s'afficher...
```

```
Quelle est la valeur de l'amplitude de l'onde en m ? : 1
L'amplitude de l'onde vaut  1.0 m
Quelle est la valeur de la période de l'onde en s ? : 0.5
La période de l'onde vaut  0.5 s
Quelle est la valeur de la longueur d'onde de l'onde en m ? : 3
La longueur d'onde de l'onde vaut  3.0 m
Calcul de l'animation en cours, merci de patienter...
```

```
Out[16]: <matplotlib.animation.FuncAnimation at 0x7f339840a1d0>
```

1.4 Conclusion

Ces trois exemples de programme vous ont montré quelques premières possibilités de la programmation Python appliquée à la Physique-Chimie. Il s'agit par la suite de comprendre progressivement les bases de cette programmation.

```
In [ ]:
```