Solubilité en fonction du pH et de la température

Interface en ligne de commande et graphiques matplotlib

Nécessite ce fichier de données (à décompresser).

<sxh python; title : evolution_solubilite_pH_T.py> #!/usr/bin/env python # -*- coding: utf-8 -*- “”“ Solubilité en fonction du pH et de la température Basé sur le travail de ML et PT, ba2 chimie 2013-2014 ”“”

Importation des données import csv #Importe le module pour lire une liste externe ifile = open(“Bibliotheque.csv”, “rb”) #ouvre le fichier .csv reader = csv.reader (ifile, delimiter= ';') #lit le fichier .csv

precipites = [] #initialise la liste avec tout les précipités for row in reader: #pour toutes les colonnes

```
  precipites.append(row) #ajoute la liste de chaque précipité
```

Choix du précipité

m = 0 #nombre servant à attribuer à chaque précipité un nombre pour faciliement le sélectionner precipite = [] #initialise la liste avec un précipité particulier for precipite in precipites: #pour chaque précipité de la liste reprenant tout les précipités

```
  print m, "--->", precipite [0]  #Affiche le numéro du précipité ainsi que
le précipité
  m = m + 1   #augmente m de 1 à chaque tour
```

p = input(“choisissez le précipité dont vous désirez étudier la solubilité ”) #permet à l'utilisiteur de choisir le précipité dont il veut étudier la solubilité print precipites[p] #mettre le s à précipité car c'est dans la liste précipités que l'on choisit l'élément

Donnée pour le calcul de la solubilité # Extraction des données pour le précipité Ks = float(precipites[p][4]) #valeur de Ks, float permet de rendre le nombre utilisable dans les calculs en les rendant “flotants” ka1 = float(precipites[p][5]) #valeur de ka1, Lorsque le précipité n'a pas de ka1, la valeur arbitraire 0 a été mise pour permettre au programme de continuer ka2 = float(precipites[p][6]) #valeur de ka2, Lorsque le précipité n'a pas de ka2, la valeur arbitraire 0 a été mise pour permettre au programme de continuer ka3 = float(precipites[p][7]) #valeur de ka3, Lorsque le précipité n'a pas de ka3, la valeur arbitraire 0 a été mise pour permettre au programme de continuer if ka3 ==0:

```
  precipites[p].pop() #la valeur 0 pose problème puisque l'on divise par les
ka, On la retire donc de la liste
```

if ka2 ==0:

```
  precipites[p].pop() #idem
```

if ka1 ==0:

```
  precipites[p].pop() #idem
```

a = float (precipites[p][1]) #nombre de moles de cations libérés en solution b = float (precipites[p][2]) #nombre de moles d'anions libérés en solution

G = float (precipites[p][3]) #Energie libre de Gibbs de formation des composés R = 8.314 #Constante des gaz parfaits Ti = 273.15 + 25 #Converti les degré Celsius en degré Kelvin

#Choix des températures z = 1 #nombre permettant de à la boucle ci dessous de s'appliquer Tlist = [] #initialise une liste avec les différentes températures while z == 1: #boucle de commande

```
  Tlist.append(float(input ("A quelle température voulez-vous étudiez la solubilité en Celsius? ")+273.15)) #Converti la température en degré Kelvin
  z = input ("Voulez étudiez la solubilité à une autre température ? ==> oui ->1, non ->0  ")  #si l'utilisateur rentre la valeur 1, on recommance la boucle, si 0, le programe continu
```

Mise en graphique #Importation des modules supplémentaires import matplotlib as plt from pylab import * import numpy as np

#les données plt.figure() #Directive pour créer la fenêtre, les axes, les échelles, ...

serie_pH = [] #initialise une liste vide pour la liste des points des abscisses for pH in range (0,140,1): # va passer successivement des valeurs 0 à 14, la valeur 15 n'est pas comprise dans la liste, besoin d'être des nombres entiers (nombre multiplié par 10)

```
  serie_pH.append(pH/10.) #ajoute chaque valeur à la liste serie_pH, (divisé par 10 pour obtenir les bonnes valeurs de pH)
```

for T in Tlist: #pour chaque Température dans la liste Tlist

```
  Kst = Ks * np.exp((G/R)*((1/T)-(1/Ti)))  #Relation de Van't Hof
  if len(precipites[p]) == 5:  #si la longueur de la liste est de 5, cette formule sera appliquée (anion non-acide)
       Solubilite = [(Kst/((a**a)*(b**b))**(1./(a+b))) for pH in serie_pH]
  if len(precipites[p]) == 6:         #Si la longueur de la liste est de 6, cette formule est appliquéé (anion monoacide)
       Solubilite = [(Kst/((a**a)*(b**b))*((1+(10**-pH)/ka1)**b))**(1./(a+b)) for pH in serie_pH]
  if len(precipites[p]) == 7:    #si la longueur de la liste est de 7, cette formule est appliquée (anion diacide)
       Solubilite = [(Kst/((a**a)*(b**b))*((1+(10**-pH)/ka1)+((10**-pH)**2)/(ka1*ka2))**b)**(1./(a+b)) for pH in serie_pH]
  if len(precipites[p]) == 8:   # si la longueur de la liste est de 8, cette formule est appliquée (anion triacide)
       Solubilite = [(Kst/((a**a)*(b**b))*((1+(10**-pH)/ka1)+((10**-pH)**2)/(ka1*ka2)+((10**-pH)**3/(ka1*ka2*ka3)))**b)**(1./(a+b)) for pH in serie_pH]
```

#plot de la ligne

```
  plt.plot(serie_pH, Solubilite, label= T-273.15) #initialise le graphe en fonction des données,  indication pour la légende, Température exprimée en Celsius
```

plt.title(u“Evolution de la solubilite en fonction du pH a temperature donnee”) #directive pour donner un titre au graphique plt.xlabel(“pH”) #directive pour nommer l'axe des abscisses plt.ylabel(“Solubilite”) #directive pour nommer l'axe des ordonnées plt.legend() #directive pour inclure la légende au graphique plt.show() #directive pour afficher la courbe

###Sources : #http://python.physique.free.fr/graphiques.html # Code inspiré des exemples de code du cours </sxh>