Solutions TP 2

Exercice 1 :

La plupart des années contiennent 365 jours, cependant la Terre tourne autour du Soleil en un temps un petit peu plus long. Pour remédier à ce problème, notre calendrier contient un jour supplémentaire, le 29 février, toutes les années bissextiles. On détermine si une année est bissextile selon les règles suivantes :

- Toute année divisible par 400 est bissextile.
- Parmi les autres, toute année divisible par 100 n'est pas bissextile.
- Parmi les autres, toute année divisible par 4 est bissextile.
- Toute autre année n'est pas bissextile.

Écrire un programme qui détermine si unes année est bissextile.

Solution 1:

```
annee = input("Saisissez une annee : ")
annee = int(annee)
bissextile = False

if annee % 400 == 0: # Si l'annee est divisible par 400
    bissextile = True
elif annee % 100 == 0: # Si l'annee est divisible par 100
    bissextile = False
elif annee % 4 == 0: # Si l'annees est divisible par 4
    bissextile = True

if bissextile:
    print("L'annee", annee, "est bissextile")
else:
    print("L'annee", annee, "n'est pas bissextile")
```

Exercice 2 :

On cherche à évaluer le nombre d'années à attendre avant que le contenu d'un compte en banque n'atteigne une certaine somme. Demander à l'utilisateur trois valeurs : le montant initial (on suppose un seul dépôt), le taux annuel d'intérêt (par exemple 0.04 pour 4%), ainsi que le montant final désiré. Déterminer le nombre d'années nécessaires pour atteindre ce dernier montant.

Par exemple, si le montant initial est de 500 € et le montant souhaité de 535 € pour un taux annuel d'intérêt de 4%, le compte en banque contiendra 500 €×1.04=520 € après un an, puis 520 €×1.04=540.80 € après deux ans, de sorte qu'il aura fallu attendre deux ans.

Solution 2:

```
initial = float(input("Montant initial: "))
taux = float(input("Taux annuel (en percentage) : "))
final = float(input("Montant final desire: "))

annees = 0
compte = initial
while compte < final:
    annees += 1
    compte *= (1+taux*0.01)
    print (compte)

print("Il faut attendre", annees)
```

Exercice 3 :

On dit souvent qu'une année supplémentaire pour un chien est équivalente à 7 années supplémentaires pour un humain. Cependant, cette simple conversion ne tient pas compte des deux premières années de l'animal au cours desquels il atteint l'âge adulte. On dit parfois qu'il est mieux de compter les deux premières années de l'animal comme équivalant à 9 années humaines, puis les suivantes comme 6.

Écrire un programme qui converti les années canines en années humaines.

Écrire un programme qui converti les années humaines en années canines.

Solution 3:

```
canin = float(input('Age canin: '))

humain = 0
if canin > 2:
    humain = 2*9 + (canin-2)*6
else:
    humain = canin*9

print('En age humain cela vaut %d annees.' % humain)

humain = float(input('Age humain: '))
canin = 0

if humain > 2*9:
    canin = 2 + (humain-2*9)/6
else:
    canin = humain/9

print('En age canin cela vaut %d annees.' % canin)
```

Exercice 4 :

Écrire un programme qui évalue la validité d'un mot de passe entré par l'utilisateur. Le mot de passe devra :

- Contenir au moins un nombre entre 0 et 9.
- Contenir au moins un caractère parmi [$#@].
- Contenir au moins une lettre majuscule et une lettre minuscule.
- Contenir au moins 6 caractères.
- Contenir au plus 16 caractères.

Solution 4:

```
import string

minuscules = string.ascii_lowercase
majuscules = string.ascii_uppercase
nombres = [str(i) for i in range(10)]
spec = ["\$","@","#"]

mdp = input('Entrer un mot de passe valide: ')

isnombre=False
for c in nombres:
    if c in mdp:
        isnombre = True
        break
isminu=False
for a in minuscules:
```

```
    if a in mdp:
        isminu = True
        break
ismaju=False
for a in majuscules:
    if a in mdp:
        ismaju = True
        break
isspec=False
for a in spec:
    if a in mdp:
        isspec = True
        break

if len(mdp)<6 :
    print('Mot de passe non valide: trop court')
elif len(mdp)>16 :
    print('Mot de passe non valide: trop long')
elif not isnombre :
    print('Mot de passe non valide: doit contenir un chiffre')
elif not isminu :
    print('Mot de passe non valide: doit contenir une minuscule')
elif not ismaju :
    print('Mot de passe non valide: doit contenir une majuscule')
elif not isspec :
    print('Mot de passe non valide: doit contenir un caractere special')
else:
    print('Mot de passe valide!')
```

Exercice 5 :

Écrire un programme qui :

- initialise une variable avec un texte que vous voulez crypter (par exemple *'mon message secret'*).
- crée une première chaîne de caractères avec les caractères du texte en position paire.
- crée une deuxième chaîne de caractères avec les caractères du texte en position impaire.
- crée une variable qui concatène ces deux listes et l'affiche.

Dans le même code, ajouter un programme qui permet de décrypter la chaîne résultante (i.e. qui effectue le processus permettant de revenir au texte initial).

Solution 5:

```
s='mon message secret'
if len(s)%2 !=0 :
        s=s+' '
sp=s[::2]
si=s[1::2]
print('texte initial:\n',s)
print('caracteres en position paire\n',sp)
print('caracteres en position impaire\n',si)

sc=sp+si
print('texte code\n',sc)

n=len(sc)
print('longueur de la chaine cryptee: ',n)

n=n//2
sg=sc[:n]
sd=sc[n:]
sf=''
for c, x in enumerate(sg) :
```

```
        sf=sf+sg[c]+sd[c]
print('texte decode: ',sf)
```

Exercice 6 : *Fibonacci ou la démultiplication des lapins*

Fibonacci est un mathématicien italien du 13ème siecle qui publia en 1202 le Liber Abaci, où il introduisit pour la première fois en Occident le calcul décimal à l'aide des notations indo-arabes, à une époque où ses contemporains utilisaient encore les chiffres romains. Fibonacci s'est en particulier inspiré des traités de Al-Khwârizmî (9ème siècle) dont le nom a donné le mot "algorithme".

Fibonnaci s'intéresse entre autre à l'évolution d'une population idéale de lapins et crée une suite mathématique permettant d'évaluer le nombre total de couples de lapins après n phases de procréation. Le problème de Fibonacci est à l'origine de la suite dont le n-ième terme correspond au nombre de couples de lapins au n-ième mois. Dans cette population (idéale), on suppose que :

- au (début du) premier mois, il y a juste un couple de lapereaux ;
- les lapereaux ne procréent qu'à partir du (début du) troisième mois ;
- chaque (début de) mois, tout couple susceptible de procréer engendre effectivement un nouveau couple de lapereaux ;
- les lapins ne meurent jamais (donc la suite de Fibonacci est croissante).

Notons $\mathcal{F}_n$ le nombre de couples de lapins au début du mois n. Jusqu'à la fin du deuxième mois, la population se limite à un couple (ce qu'on note : $\mathcal{F}_1 = \mathcal{F}_2 = 1$). Dès le début du troisième mois, le couple de lapins a deux mois et il engendre un autre couple de lapins ; on note alors $\mathcal{F}_3 = 2$.

Plaçons-nous maintenant au mois n et cherchons à exprimer ce qu'il en sera deux mois plus tard, soit au mois $n+2$: $\mathcal{F}_{n+2}$ désigne la somme des couples de lapins au mois $n+1$ et des couples nouvellement engendrés. Or, n'engendrent au mois $n+2$ que les couples pubères, c'est-à-dire ceux qui existent deux mois auparavant. On a donc, pour tout entier n strictement positif :

$$\mathcal{F}_{n+2} = \mathcal{F}_{n+1} + \mathcal{F}_n \tag{1}$$

On choisit alors de poser $\mathcal{F}_0 = 0$, de manière que cette équation soit encore vérifiée pour n = 0. Les termes de cette suite sont appelés "nombres de Fibonacci".

- En exploitant la relation de récurrence (1), créer une liste contenant les nombres de Fibonacci de 0 à n, où n est demandé à l'utilisateur. Vérifier que $\mathcal{F}_{25} = 75025$.

Solution 6:

```
N=int(input("N = "))
F=[0,1]

for i in range(N+1):
    if i>1:
        a=F[i-1]+F[i-2]
        F.append(a)

print("F_n =",F[-1])
```

Exercice 7 :

Les hydrogénoides (hydrogène, ou atomes ionisés à un électron) émettent de la lumière à des longueurs d'ondes précises déterminées par la formule de Rydberg, que vous trouverez à la page suivante : `https://en.wikipedia.org/wiki/Rydberg_formula`

— Après avoir lu avec attention la section Formule de Rydberg pour l'hydrogène de la page wikipédia, écrire un programme qui imprime à l'écran la longueur d'onde des 4 premières raies de la série de Lyman.
— Généraliser ce programme pour imprimer la longueur d'onde des 4 premières raies des séries de Lyman, Balmer et Pashen.

Le spectre d'émission de la nova (explosion d'étoile) V630 Sgr est montré ci-après. On se focalise pour l'instant sur les raies H, qui correspondent à l'hydrogène.

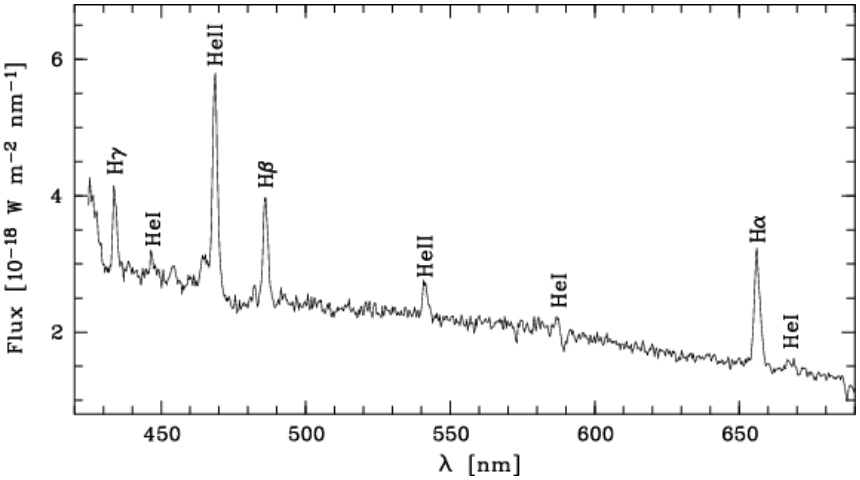

FIGURE 1 – *Spectre d'émission de la nova V630 Sgr. Figure tirée de "Spectroscopic analysis of tremendous-outburst-nova candidates", A&A 432, 199-205 (2005).*

— À quelle série et quelles transitions ces raies de l'hydrogène correspondent-elle ?

Vous souhaitez généraliser votre programme pour aussi traiter le cas des hydrogénoides, en particulier l'hélium simplement ionisé (noté HeII en astronomie, HeI étant l'hélium neutre) :

— Après avoir lu avec attention la section Généralisation aux hydrogénoïdes, généralisez votre programme précédent pour traiter le cas Z=2.
— Déterminez à quelles transitions correspondent les raies HeII observées dans le spectre de la nova V630 Sgr. Qu'en est-il des raies HeI ?

Solution 7:

```
Z = 2 # le nombre de protons de l'hydrogenoide

#plus d'info, voir : https://en.wikipedia.org/wiki/Rydberg_formula
R = 1.097e-2 #Constante de Rydberg en nm^-1
for n1 in range(1,5):
  print("\nSerie pour n1 =",n1," et Z=",Z)
  for n2 in range(n1+1,n1+5):
    invlambda = R*Z**2*(1./n1**2-1./n2**2)#1/lambda en nm^{-1}
    print(int(100./invlambda)/100.,"nm")#arrondi a la deuxieme decimale


## Reponses aux questions ################################################

#Reponse - serie H
'''
Serie pour n1 = 2  et Z= 1
656.33 nm -> Halpha: Balmer 3->2
486.17 nm -> Hbeta:  Balmer 4->2
434.08 nm -> Hgamma: Balmer 5->2
```

```
'''

#Reponse - serie HeII
'''
Serie pour n1 = 3  et Z= 2
468.81 nm -> HeII: 4 -> 3

Serie pour n1 = 4  et Z= 2
1012.86 nm
656.33 nm
541.42 nm -> HeII: 7 -> 4
'''

#Reponse - HeI
'''
HeI n'est pas un hydrogenoide, donc la formule de Rydberg ne s'applique pas.
'''
### END SOLUTION
```

Exercice 8 :

En physique de la matière condensée, la constante de Madelung donne le potentiel électrique ressenti par un atome dans un solide. Cette constante dépend des charges ressenties sur les autres atomes et leurs positions dans le réseau. Considérons par exemple, le chlorure de sodium. Les atomes dans le cristal de chlorure sont disposés sur un réseau cubique, de façon à avoir une alternance entre les atomes de sodium ayant une seule charge positive $+e$ et ceux de chlore ayant une charge négative $-e$ où e est la charge de l'électron. Si nous définissons chacune des positions atomiques dans le réseau à travers trois coordonnées entières (i, j, k), alors les atomes de sodium occupent des positions pour lesquelles $i + j + k$ est pair et les atomes de chlore des positions pour lesquelles $i + j + k$ est impair. L'objectif de cet exercice est de considérer un atome de sodium dans l'origine du réseau, avec $i = j = k = 0$, et de calculer la constante de Madelung.

Si la distance entre les atomes dans le réseau est a, alors la distance de l'origine pour un atome dans le point (i, j, k) est :

$$\sqrt{(ia)^2 + (ja)^2 + (ka)^2} = a\sqrt{i^2 + j^2 + k^2}$$

Le potentiel créé dans l'origine par l'atome (i, j, k) est :

$$V(i, j, k) = \pm \frac{e}{4\pi\varepsilon_0 a\sqrt{i^2 + j^2 + k^2}}$$

Avec ε_0 représentant la permittivité dans le vide et le signe de l'expression dépend du fait que $i + j + k$ soit pair ou impair. Le potentiel total ressenti par l'atome de sodium est la somme de $V(i, j, k)$ pour tous les atomes. Assumons une boîte cubique autour de l'atome de sodium placé dans l'origine, avec L atomes dans toutes les directions. Alors nous pouvons écrire :

$$V_{\textbf{total}} = \sum_{\substack{i,j,k=-L \\ \textbf{not } i=j=k=0}}^{L} V(i, j, k) = \frac{e}{4\pi\varepsilon_o a} M$$

Où M est la constante de Madelung, ou bien une très bonne approximation pour cette quantité (en effet la constante de Madelung est définie comme le valeur de M quand L va vers l'infini, mais la valeur de M calculée avec une L très grande est déjà une très bonne approximation).

Ecrivez un programme permettant de calculer et d'imprimer la constante de Madelung pour le chlorure de sodium. Utilisez la valeur de L la plus large possible en essayant de maintenir la durée d'exécution du programme à moins d'une minute.

Solution 8:

```
from math import sqrt, pow

L=150
M=0.0
for i in range(-L,L+1):
    for j in range(-L,L+1):
        for k in range(-L,L+1):
            if (i == 0 and j == 0 and k == 0):
                continue
            if (i+j+k)%2 == 0:
                M += 1/sqrt(pow(i,2)+pow(j,2)+pow(k,2))
            else:
                M -= 1/sqrt(pow(i,2)+pow(j,2)+pow(k,2))
print(M)
```