Devoir surveillé n°2

Durée 2h

■ ***Consignes :***

- Les calculatrices sont interdites
- Lors de l'écriture des programmes , les indentations doivent être respectées.
- Des commentaires doivent expliquer les instructions.
- Le candidat répondra directement sur la feuille de l'énoncé aux questions 1 et 2.

1ère partie : Conversions

Donner le résultats des conversions suivantes :

00111010_2 en base 10	58_{10}
80_{10} en base 2	1010000_2
Donner la représentation binaire sur 4 bits de $(-11)_{10}$ en utilisant le complément à 2.	On code 2^3-11=8-11 impossible
Donner en base 10 la valeur de 1111_2 considéré comme un entier signé en machine sur 4 bits.	$1111_2=(1+2+4+8=15)_{10}=(2^4-1)_{10}$ rep :-1

2ème partie : Bases de programmation

On définit les variables suivantes :

- `s = "Bonjour Charlie" ;`
- `seq = ['3','9',6.2,'0',8,8 < 3,'8 < 3'].`

2.1. Remplir le tableau ci-dessous (des messages d'erreur sont possibles) :

Instruction dans la console	Interprétation
`>>>s+s`	'Bonjour CharlieBonjour Charlie'
`>>>seq[5]`	False
`>>>len(s)`	15
`>>>'9' in seq`	True
`>>>s[0]+seq[2]`	erreur

2.2 Préciser le type des variables dans le tableau ci-dessous :

Instruction python	Type de la variable
`seq[2]`	float
`seq`	liste
`seq[6]`	Str
`s`	str

2.4 Dans les boucles suivantes :

- ***préciser les valeurs successives de i jusqu'à sa valeur finale comprise ;***
- ***Le nombre d'itérations effectuées .***

Instructions	Valeurs successives de i, nombre d'itérations
`for i in range(1,10) :` `  i = i + 2`	Le changement de valeur de i dans la boucle ne change pas sa valeur dans la liste des valeurs crées par l'instruction range(1, 10). Les valeurs prises successivement par i sont : **(1,3,2,4,3,5, 4,6 5,7 6,8 7,9 8,10, 9,11)** . Il y a **9** itérations.

`i = 1`
`while i < 10:`
`  i = i + 2`
	i prend successivement les valeurs : **1, 3, 5, 7 , 9,11** .
	Il y a **5** itérations

`i = 1`
`while i < 10:`
`  if i % 2 == 0 :`
`  i = i + 2`
	C'est une **boucle infinie** car en partant de i=1, on n'obtient jamais i % 2 == 0 donc i n'est pas incrémenté.
	i garde toujours la valeur 1.

3ème partie : jeu des chiffres et des lettres

Madame Anna Gramme (un peu âgée) aime beaucoup le jeu « des chiffres et des lettres » et veut s'entraîner au jeu des lettres. Bien qu'un peu âgée, elle connaît le langage Python et souhaite écrire un programme lui donnant la liste de toutes les sous chaînes extraites d'une liste de lettres, sans tenir compte de l'ordre.

Par exemple, avec la liste ['n', 'a', 'g'], *le programme donnera* ['g', 'a', 'n', 'ag', 'ng', 'na', 'nag']

Indications : Après avoir chargé la bibliothèque **random**, via **`from random import *`**, on pourra utiliser l'instruction **`choice`**. *Par exemple :* **`choice(['a','b','c'])`** renvoie aléatoirement **`'a'`**, **`'b'`** ou **`'c'`**.

1. Écrire la fonction **`liste_lettres(n)`** qui renvoie une liste de **`n`** lettres toutes différentes et choisies aléatoirement.

```
from random import *
#1. On crée une liste le n lettres sans répétition
def liste_lettres(n):#on définit la fonction
    liste=[]#on génère une liste vide
    while len(liste)<n:#on va remplir la liste tant qu'elle a un nombre de valeurs inférieur à n
        i=choice(['a','b','c','d','e','f','g','h','i','j','k','l','m','n','o','p','q','r','s','t','u','v','w','x','y','z'])
#on choisit aléatoirement une lettre
        if i in liste :
            liste=liste #si la lettre est dans la liste on ne l'ajoute pas
        else :
            liste=liste+[i]#sinon on l'ajoute
    return(liste)
```

2. Anna souhaite tester son programme en affectant à la variable **`L`** une liste de 5 lettres . L'exécution de son test renvoie : **`L=['b','s','u','p','d']`**. Quelle instruction est à l'origine de cette affichage ?

```
#2) test
n=5
L=liste_lettres(n)
print('L=',L)
```

Ces préliminaires étant terminés et disposant de la liste des lettres nommée ***`L (ci-dessus)`****, Anna cherche à en obtenir toutes les sous-chaînes, sans tenir compte de l'ordre des lettres. Elle pense alors une méthode utilisant l'écriture des nombres binaires sur n bits..Pour convertir un nombre entier en binaire, elle utilise la fonction* **`bin`** *de python ayant pour paramètre un entier et renvoyant* une chaîne de caractères contenant les 2 caractères 0 et b puis l'entier écrit en binaire.

Exemple : **`bin(7)`** *renvoie* **`'0b111'`** .

3. Écrire une fonction **`str_bin (nb,n)`** ayant pour paramètre un entier nb , un nombre de bits n et renvoyant une chaîne de caractères correspondant à l'entier naturel nb écrit en binaire sur n bits .

```
#3) on écrit sur n bits un entier
def str_bin(nb,n):
    b=bin(nb) #conversion en binaire
    # on enlève les deux premier caractères
    b=b[2:len(b)]
    #on ajoute des zéros au début pour avoir le bonne taille
    while len(b)<n:
        b='0'+b
    return(b)
```

4. Pouvez-vous aider Anna ? Pour la suite de son programme, elle se demande : "Combien d'entiers non nuls peut-on écrire en binaire avec n bits?"

Sur n bits, on peut écrire 2^n-1 entiers non nuls

5. Écrire une fonction **`liste_bin (n)`** renvoyant la liste de tous les entiers que l'on peut écrire sur n bits. Pour n = 5, le programme renvoie :

```
['00001', '00010', '00011', '00100', '00101', '00110', '00111', '01000', '01001',
'01010', '01011', '01100', '01101', '01110', '01111', '10000', '10001', '10010',
'10011', '10100', '10101', '10110', '10111', '11000', '11001', '11010', '11011',
'11100', '11101', '11110', '11111']
```

```
#5)On fabrique une liste contenant la liste des entiers non nuls que l'on peut écrire sur n bits
def liste_bin(n):
    """liste des entiers de 1 à 2**taille-1 en binaire de taille donnée"""
    liste=[]#on commence par une liste vide
    max=2**n#on se sert de cette valeur pour déterminer le nombre de valeurs converties
    for k in range(1,max):
        liste=liste+[str_bin(k,n)]
    return(liste)
```

6. Anna continue son programme et écrit les instructions suivantes :

```
1 n=5 #on affecte 5 à la lettre n
2 def bin_mot(b,L): '''si on fait correspondre un nombre binaire sur n bits et une liste de n lettres , cette fonction renvoie la liste des lettres correspondant à 1 du nombre binaire'''
3     taille=len(L) #taille est la longueur de L
4     mot=''#on définit une chaine de caractères vide
5     for k in range(taille):#k va prendre les valeurs de 0 à taille-1
6        if b[k]=='1':#si l'élément k de b est 1
7           mot=mot+L[k]#on rajoute la lettre de rang k de L
8     return(mot)#la fonction renvoie mot
9  b=liste_bin(n)[10]
10 print(bin_mot(b,L))
```

a) Documenter la fonction et commenter les lignes 1 à 8 du programmes.

b) D'après la ligne 9, que vaut b ? **'01011'**

c) Qu'affiche l'exécution de la ligne 10 ? **spd**

7. Anna perfectionne son programme. Écrire une fonction **sous_liste(L)** ayant pour paramètre une liste **L** de lettres et qui ,tenant compte du programme précédent, renvoie la liste de tous les mots possibles à partir d'une liste de lettres L.

```
#7)On définit une fonction qui donne la liste des mots possibles
def sous_liste(L):
    liste=[]
    n=len(L)
    bin=liste_bin(n)
    for b in bin: #on parcourt la liste bin, chaque élément donne un mot
        liste=liste+[bin_mot(b,L)]
    return(liste)
```

8. Anna est perfectionniste, elle veut encore améliorer son programme. Écrire une fonction **mots_taille(L,taille)** ayant pour paramètre une liste **L** et un entier **taille** qui renvoie la liste de tous les mots possibles de L pour un nombre de lettres (taille) déterminé.

```
Lmots=sous_liste(L)
print(Lmots)
print(sorted(Lmots))
#8)on veut la liste des mots de taille déterminée
def mots_taille(L,taille):
    liste=[]
    liste_mots=sous_liste(L)
    for mot in liste_mots:
        if len(mot)==taille:
            liste=liste+[mot]
    return(liste)
```

9. Après tout ce travail, Anna se dit que l'ordre des lettres est important ! Écrire , en python ou en langage algorithmique, une fonction **`vrais_mots (mot)`** qui étant donné une chaîne de caractères mot de taille k renvoie toutes les combinaisons possibles de taille k en éliminant celles qui sont identiques.