Informatique MPSI - Devoir surveillé n°2 - Vendredi 19/01/2024 - 2 heures 30
Calculatrices interdites.

Les programmes demandés seront écrits en Python. **Le travail devra bien sûr être soigné, les codes clairs et commentés autant que vous le jugez nécessaire. Veillez à réserver du temps, après la rédaction de chaque programme, pour le vérifier pour éviter les erreurs de syntaxe.**

Dans chaque exercice, il est bien évidemment possible de faire appel à des fonctions ou procédures créées dans les questions antérieures.

Le sujet comporte **6 pages**. Il est composé d'un exercice et d'un problème indépendants qui seront rendus sur des **copies séparées**. **Vous devez commencer par l'exercice qui sera ramassé au bout de 45 minutes au maximum.**

Exercice.

Dans cet exercice, on se propose d'écrire un algorithme pour décomposer un entier en produit de nombres premiers. On rappelle la définition de la valuation p-adique de n (notée $v_p(n)$) : si p est un nombre premier et n entier naturel non nul :

- Si p divise n, on note $v_p(n)$ le plus grand entier k tel que p^k divise n.
- Si p ne divise pas n, on pose $v_p(n) = 0$.

Q1. Écrire une fonction booléenne `est_premier(n)` qui prend en argument un entier naturel non nul `n` et qui renvoie le booléen `True` si `n` est premier et le booléen `False` sinon.

On pourra utiliser le critère suivant : un entier $n \geq 2$ qui n'est divisible par aucun entier $d \geq 2$ tel que $d^2 \leq n$, est premier.

Q2. En déduire une fonction `liste_premiers(n)` qui prend en argument un entier naturel non nul `n` et renvoie la liste des nombres premiers inférieurs ou égaux à `n`.

Q3. Pour calculer la valuation 2-adique de 40 , on peut utiliser la méthode suivante :

- 40 est divisible par 2 et le quotient vaut 20
- 20 est divisible par 2 et le quotient vaut 10
- 10 est divisible par 2 et le quotient vaut 5
- 5 n'est pas divisible par 2.

La valuation 2-adique de 40 vaut donc 3.

Écrire une fonction `valuationp_adique(n,p)` **non récursive** qui implémente cet algorithme. Elle prend en arguments un entier naturel `n` non nul et un nombre premier `p` et renvoie la valuation `p`-adique de `n`.

Par exemple, puisque $40 = 2^3 \times 5$, `valuationp_adique(40,2)` renvoie 3 , `valuationp_adique(40,5)`renvoie 1 et `valuationp_adique(40,7)` renvoie 0 .

Q4. Écrire une deuxième fonction, cette fois-ci **récursive**, `val(n, p)` qui renvoie la valuation `p`-adique de `n`.

Q5. En déduire une fonction `decomposition_facteurs_premiers(n)`, où `n` est un entier supérieur ou égal à 2.

Cette fonction doit renvoyer la liste des couples $\big(\mathtt{p}, v_{\mathtt{p}}(\mathtt{n})\big)$ pour tous les nombres premiers `p` qui divisent `n`.

Par exemple, `decomposition_facteurs_premiers(40)` renvoie `[(2,3), (5,1)]`.

Problème : Élection présidentielle

I. Introduction

On s'intéresse à une élection présidentielle à laquelle participent $n \geq 2$ candidats. En Python, chaque candidat est représenté par son prénom (de type str) et par un numéro `i` $\in [|0, n-1|]$ (de type int). Tout au long de cet exercice, on manipulera une liste `C` (liste de candidats) de taille `n` telle que pour tout i $\in [|0, n-1|]$, `C[i]` est le prénom du candidat numéro `i`.
Par exemple, si `C` = `C0` où :

`C0` = `["Jean-Luc", "Yannick", "Emmanuel", "Valérie", "Marine", "Eric"]`

cela signifie que le numéro du candidat "`Emmanuel`" est le 2 et que celui de la candidate "`Valérie`" est le 3.

Q1. Quelle instruction Python permet d'obtenir la valeur de `n` à partir de la variable `C` ?

Q2. **a.** Écrire une fonction `appartient(L, x, i0)` qui prend en arguments une liste `L`, un objet `x` et un entier `i0` et qui renvoie `True` si x est présent dans `L` à un indice supérieur ou égal à `i0` et `False` sinon.
Par exemple, `appartient(C0, "Emmanuel", i0)` :

- Renvoie `True` lorsque `i0` vaut 0, 1 ou 2.
- Renvoie `False` lorsque `i0` ≥ 3.

b. Ecrire alors une fonction `tousDiff(L)` qui prend en argument une liste `C` qui renvoie `True` si les éléments de la liste `L` sont tous différents et `False` sinon.

Si plusieurs candidats avec le même prénom se présentent, l'élection est annulée et personne n'est élu. Dans la suite, on supposera que les candidats ont des prénoms différents.

II. Premier tour du vote

Lors du premier tour du scrutin, chaque électeur choisit un candidat, écrit son prénom sur un bulletin de vote et met le bulletin dans une urne. On note `V` la liste des votes c'est à dire la liste des prénoms qui apparaissent dans l'urne.
Par exemple, si `V = V0` avec :

`V0` = `["Emmanuel", "Marine", "Superman", "Emmanuel", " ", "Marine", "Emmanuel", "Jean-Luc"]`

cela signifie que :

- Trois personnes ont voté pour "Emmanuel".
- Deux personnes ont voté pour "Marine".
- Une personne a voté pour "Jean-Luc".
- Personne n'a voté pour "Yannick", "Valérie" ou "Éric".
- Une personne a voté blanc (le vote " ").
- L'un des votes est nul (le vote pour "Superman").

Lors du dépouillement, on comptabilise le nombre de voix pour chaque candidat et on construit une liste d'entiers `R` (appelée liste de résultats) telle que pour chaque `i` $\in [|0, n-1|]$, l'entier `R[i]` est le nombre de voix pour le candidat numéro `i`.
Dans l'exemple précédent, `R` vaut $[1, 0, 3, 0, 2, 0]$. On propose deux méthodes pour construire `R`.

Méthode 1.

Q3. **a.** Écrire une fonction `nbOcc(V,s)`qui prend en argument une liste de votes `V` et une chaine de caractère `s` et qui renvoie le nombre d'occurrences de `s` dans `V` (le nombre d'occurrences est le nombre d'apparitions).
Par exemple, dans `V0`, le nombre d'occurrences de "`Emmanuel`" est `3`, le nombre d'occurrences de "`Marine`" est `2` et le nombre d'occurrences de "`Yannick`" est `0`.

b. En déduire une fonction `makeR(C,V)` qui prend en argument une liste de candidats `C` et une liste de votes `V` et qui renvoie la liste des résultats `R` correspondante. On utilisera la fonction `nbOcc`.

Méthode 2. Dans la méthode 1 implémentée à la question précédente, la liste `V` est parcourue plusieurs fois (une fois pour chaque appel à la fonction `nbOcc`). On essaye maintenant de construire `R` en ne parcourant `V` qu'une seule fois.

Q4. **a.** Écrire une fonction `nomToNum(C,s)` qui prend en argument une liste de candidats `C` et une chaine de caractère `s` et qui renvoie le numéro du candidat dont le prénom est `s`. Si `s` n'est le prénom d'aucun candidat, votre fonction devra renvoyer `None`. L'utilisation de la méthode `index` de Python n'est pas autorisée.

b. Écrire une fonction `makeRBis(C,V)` qui prend en argument une liste de candidats `C` et une liste de votes `V` et qui renvoie la liste des résultats `R` correspondante. La liste `V` ne devra être parcourue qu'une seule fois.

Dans la suite, on pourra utiliser au choix la fonction `makeR` ou `makeRBis`.

Résultats du premier tour.

Un candidat est élu lors du premier tour s'il obtient la majorité absolue, c'est à dire strictement plus de 50% des voix exprimées (une voix exprimée est un vote qui n'est ni blanc ni nul). Si aucun candidat n'a obtenu la majorité absolue, les deux meilleurs candidats s'affrontent lors d'un second tour.
Dans l'exemple précédent, il y a 6 voix exprimées et il faut au minimum 4 voix pour obtenir la majorité absolue, il y aura donc un second tour entre "Emmanuel" et "Marine".

Q5. **a.** Écrire une fonction `nbVotes(R)` qui prend en argument une liste de résultats `R` et renvoie le nombre de voix exprimées.

b. Écrire une fonction `majAbs(R)` qui prend en argument une liste de résultats `R` et renvoie le nombre minimum de voix pour obtenir la majorité absolué.

Q6. **a.** Écrire une fonction `vainqueurTour1(R)` qui prend en argument une liste de résultats `R` et qui renvoie le numéro du candidat qui a obtenu la majorité absolue. Dans le cas où aucun candidat n'a obtenu la majorité absolue votre fonction renverra `None`.

b. Écrire une fonction `deuxMeilleurs(R)` qui prend en argument une liste de résultats `R`, et qui renvoie un couple d'entiers `(i1,i2)` où `i1` est le numéro du candidat ayant obtenu le plus de votes et `i2` est le numéro du candidat ayant obtenu le plus de votes après `i1`. Afin de traiter le cas où un troisième candidat a obtenu le même nombre de voix que `i2`, on demande que `i1` et `i2` soient les plus petits possibles. De plus, si les candidats `i1` et `i2` ont le même nombre de voix, on fera en sorte que $\texttt{i1} < \texttt{i2}$. On rappelle enfin qu'au début de l'exercice, le nombre de candidats a été supposé supérieur ou égal à 2.

III. Second tour du vote

On se place dans le cas où aucun candidat n'a obtenu la majorité absolue au premier tour et on note `i1` et `i2` les numéros des deux candidats sélectionnés pour le second tour comme dans la question précédente. On fait l'hypothèse que, lors de ce second tour, tous les électeurs ont la même stratégie : ils votent pour le candidat ayant le programme le plus proche du candidat pour lequel ils ont voté au premier tour. Il se trouve que la liste `C` est supposée organisée de telle manière que si un électeur vote pour le candidat numéro $\texttt{i} \in [|0, \texttt{n}-1|]$ lors du premier tour alors, lors du second tour :

- Il vote pour le candidat numéro `i1` si $|\texttt{i} - \texttt{i1}| \leqslant |\texttt{i} - \texttt{i2}|$.
- Il vote pour le candidat numéro `i2` sinon.
- De plus, une personne qui a voté blanc votera de nouveau blanc et une personne qui a voté nul votera de nouveau nul.

Dans l'exemple précédent, une personne qui a voté "`Jean-Luc`", "`Yannick`", "`Emmanuel`" ou "`Valérie`" au premier tour votera "`Emmanuel`" au second tour et une personne qui a voté "`Marine`" ou "`Éric`" au premier tour votera pour "`Marine`" au second tour.

Finalement, le candidat élu est le candidat ayant obtenu le plus de voix au second tour. Dans le cas où les deux candidats ont le même nombre de voix, c'est le candidat avec le plus petit numéro qui l'emporte.

Q7. **a.** A l'aide d'une instruction conditionnelle, écrire une fonction `valAbs(x)` qui prend en entrée un nombre réel `x` et renvoie sa valeur absolue.

b. Ecrire une fonction `vainqueurTour2(R, i1, i2)` qui prend en argument une liste de résultats `R` et les deux entiers `i1` et `i2` correspondant aux deux meilleurs candidats du 1er tour (comme définis question 6.b.) et qui renvoie le numéro du vainqueur du second tour.

Q8. Écrire une fonction `vainqueurElection(C,V)`qui prend en arguments une liste de candidats `C` et une liste de votes `V`, et qui renvoie le prénom de la personne élue. En particulier, votre fonction devra vérifier que tous les prénoms sont différents et si ce n'est pas le cas, elle devra renvoyer `None.`

IV. Test des fonctions

Dans cette partie, le but est de tester les fonctions `makeR` et `makeRBis` sur des listes générées aléatoirement. Pour cela, on crée d'abord deux listes aléatoires `C` et `R`, puis on construit `V` à partir du couple `(C,R)` et pour finir on vérifie que les appels à `makeR(C,V)` et `makeRBis(C,V)` renvoient bien `R`. Voici des fonctions utiles pour la suite :

- La fonction `randint` du module `random` prend en entrée deux entiers a, b et renvoie un entier aléatoire de l'intervalle $[|a, b|]$.
- La fonction `chr` prend en entrée un entier et renvoie un caractère (c'est à dire une chaine de caractères de taille 1). La fonction `ord` est la fonction réciproque de `chr`.

  Par exemple:

i	97	98	99	...	121	122
chr(i)	'a'	'b'	'c'	...	'y'	'z'

c	'a'	'b'	'c'	...	'y'	'z'
ord(c)	97	98	99	...	121	122

Q9. Quelle instruction préalable est nécessaire en Python pour pouvoir utiliser la fonction `randint`?

Dans la suite, on suppose que cette étape préalable a été effectuée.

Génération de la liste C

Q10. Dans le but de générer aléatoirement la liste `C`, on écrit les fonctions suivantes :

```
def f(m):
    s = ''
    i1 = ord('a')
    i2 = ord('z')
    for k in range(m):
        i = random.randint(i1,i2)
        s = s + chr(i)
    return s
```

```
def g(n):
    C = [ ]
    while len(C) < n :
        m = random.randint(4,10)
        s = f(m)
        if not appartient(C,s,0):
            C.append(s)
    return C
```

a. Expliquer ce que renvoie la fonction `f`. Préciser le type de son argument et de la valeur renvoyée.

b. Même question pour la fonction `g`.

Génération de la liste R

Soit $m \in \mathbb{N}^*$ et $s \in \mathbb{N}^*$ deux entiers fixés. On souhaite générer m entiers aléatoires $(a_1, a_2, \ldots, a_m) \in \mathbb{N}^m$ vérifiant $\sum_{i=1}^{m} a_i = s$. Pour cela on propose deux méthodes.

Méthode 1. Chaque a_i est choisi aléatoirement grâce à la fonction `random.randint` de la manière suivante :

- On choisit aléatoirement a_1 dans l'intervalle $[|0, s|]$
- On choisit aléatoirement a_2 dans l'intervalle $[|0, s - a_1|]$
- On choisit aléatoirement a_3 dans l'intervalle $[|0, s - a_1 - a_2|]$
- ...
- On choisit aléatoirement a_{m-1} dans l'intervalle $\left[\left|0, s - \sum_{i=1}^{m-2} a_i\right|\right]$
- On pose $a_m = s - \sum_{i=1}^{m-1} a_i$.

Q11. Écrire une fonction `listeAlea1(m,s)` qui prend en argument deux entiers naturels non nuls `m` et `s` et qui renvoie une liste `A` contenant les entiers a_i obtenus par la méthode 1.

Méthode 2. Le problème avec la fonction `listeAlea1` est qu'un entier situé au début de la liste aura plus de chances d'être grand qu'un entier situé en fin de liste. *Par exemple*, a_1 a environ 50% de chance de vérifier $a_1 \geqslant s/2$, et si $a_1 = s/2$, tous les autres éléments de la liste vérifient $a_i \leqslant s/2$. On propose donc une seconde méthode qui évite ce problème.

Afin de générer les entiers a_i, on construit une liste contenant $(s+m-1)$ valeurs dans laquelle on place de manière aléatoire $m-1$ fois la valeur `1`, les autres valeurs étant toutes égales à `0`. *Par exemple*, pour $m = 6$ et $s = 20$, on peut obtenir une liste comme ci-dessous :

$$[0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0]$$

On définit alors :

- a_1 le nombre de zéros se trouvant avant le premier 1.
- a_2 le nombre de zéros se trouvant entre le premier et le deuxième 1.
- a_3 le nombre de zéros se trouvant entre le deuxième et le troisième 1.
- ...
- a_{m-1} le nombre de zéros se trouvant entre le $(m-2)^{\text{ème}}$ et le $(m-1)^{\text{ème}}$ 1.
- a_m le nombre de zéros se trouvant après le $(m-1)^{\text{ème}}$ 1.

Avec l'exemple ci-dessus , on obtiendrait :

$$a_1 = 2, \quad a_2 = 6, \quad a_3 = 4, \quad a_4 = 0, \quad a_5 = 7, \quad a_6 = 1.$$

Q12. **a.** Ecrire une fonction `tableau(m,s)` qui prend en argument deux entiers naturels non nuls `m` et `s` et qui renvoie une liste de 0 et de 1 comme décrite ci-dessus. Pour sa construction on pourra procéder ainsi :

- Initialiser une liste contenant `(m+s-1)`zéros.
- Tant que la liste contient moins de `(m-1)` fois la valeur 1
  - on choisit aléatoirement un entier j entre 0 et `(m+s-2)` et, si l'élément d'indice j de la liste n'est pas égal à 1, on le remplace par 1.

b. Écrire une fonction `listeAlea2(m,s)` qui prend en argument deux entiers naturels non nuls `m` et `s` et qui renvoie une liste `A` contenant les entiers a_i obtenus par la méthode 2 décrite ci-dessus.

On pourra écrire une fonction intermédiaire. Si tel est le cas, on précisera le but de la fonction introduite.

Q13. A l'aide de la fonction de la question précédente, écrire une fonction `genererR(n, s)` qui prend en argument deux entiers naturels non nuls `m` et `s` et qui renvoie un triplet `(R, v1, v2)` où `R` est une liste aléatoire d'entiers naturesl de taille `n` et `v1` et `v2` sont deux entiers naturels aléatoires (plus tard, `v1` sera le nombre de votes blancs et `v2` le nombre de votes nuls) tels que la somme des éléments de `R` plus `v1` plus `v2` soit égale à `s`.

Génération de la liste V

Le mélange de Fisher-Yates permet de permuter aléatoirement les éléments d'une liste `L`. En voici le principe:

- On note `n` la taille de `L`.
- Pour `i` variant de `n` -1 à 1 :
  - on choisit un entier aléatoire `j` dans $[|0, \texttt{i}|]$
  - on échange les éléments `L[i]` et `L[j]`.

Q14. Écrire une fonction `melange(L)` qui prend en entrée une liste `L` et lui applique le mélange de Fisher-Yates.

Supposons avoir à notre disposition les listes `C` et `R`, ainsi que deux entiers naturels `v1` et `v2`. Pour créer la liste de votes `V` correspondante on procède ainsi :

- On crée d'abord une liste `V0` telle que :
  - `V0` contient `R[i]` votes pour le candidat numéro `i`, `i` parcourant l'ensemble des numéros valides des candidats.
  - `V0` contient `v1` votes blancs.
  - `V0` contient `v2` votes nuls. Un vote nul est une chaine de caractères de taille 11 générée avec la fonction `f` de la question 10.
- La liste `V` est ensuite obtenue à partir de `V0` en lui appliquant un mélange de Fisher-Yates.

  remarque : L'ordre des éléments de `V0` est arbitraire (c'est à dire que vous pouvez choisir l'ordre qui vous arrange).

Q15. Écrire une fonction `genererV(C, R, v1, v2)` qui prend en argument une liste de candidats `C`, une liste de résultats `R`, et deux entiers naturels `v1` et `v2` qui génère la liste `V`.

Tests des fonctions

Q16. A l'aide des fonctions précédentes, écrire une fonction `test(n, s)` qui prend en argument deux entiers naturels non nuls `n` et `s`, génère aléatoirement des listes `C` et `R`, la liste `V` correspondante, et qui :

- renvoie `True` si `makeR(C,V)` et `makeRBis(C,V)` valent tous les deux `R`.
- renvoie `False` sinon.

L'entier `n` est le nombre de candidats et l'entier `s` est le nombre total de votes. On pourra tester l'égalité entre des listes `L1` et `L2` avec l'instruction `L1 == L2`.