TD Informatique TSI2

GRAPHS : STACKS and QUEUES

EXERCICE 01

Création d'une pile et fonctions de base

1. Taper les fonctions suivantes

```
In [2]: def cree_pile():
   ...:     return []
In [3]: def empile(x,p):
   ...:     p.append(x)
In [4]: def depile(p):
   ...:     assert(len(p)>0)
   ...:     return p.pop()
```

2. En utilisant les trois fonctions précédentes, créer une pile `p` puis y empiler les vingt premiers carrées d'entiers (à partir de 1). Afficher ensuite `p`. Puis dépiler `p` de ses trois derniers éléments. Afficher de nouveau `p`

   Indications *: On tape* `p = cree_pile()` *puis dans une boucle* `for`*, on utilise* `empile` *autant de fois que nécessaire. Puis on refait une boucle* `for` *et on utilise* `depile` *autant de fois que nécessaire*

EXERCICE 02

Utilisation d'une pile pour tester le parenthésage

On dispose d'une chaîne de caractères, dont certains sont des parenthèses (ouvrantes ou fermantes). On souhaite vérifier que le parenthésage est cohérent. Pour cela, on parcourt la chaîne de caractères. Lorsque l'on rencontre une parenthèse ouvrante, on l'empile sur une pile. Lorsque l'on rencontre une parenthèse fermante, on dépile. Le parenthésage est cohérent si, et seulement si, à la fin du parcours, la pile ne contient plus aucune parenthèse et qu'au cours du parcours, on n'a jamais tenté de dépiler alors que la pile était vide (ce qui reviendrait à trouver une parenthèse fermante dans la chaîne de caractères, alors que toutes les parenthèses déjà ouvertes ont été refermées). Par ailleurs, on considère ici la fonction :

```
In [10]: def ist_leer(p):
    ...:     return (len(p) == 0)
```

1. Construire une procédure Python `parenthesage_correct(texte)` qui fait cela et qui renverra `True` si le parenthésage est correct et `False` sinon.

   Indications *: On commence par créer une pile* `p` *avec* `cree_pile` *de l'exercice 01. Puis on fait une boucle du type* `for c in texte :` *et dans cette boucle si* `c == '('` *alors on empile* `c` *dans* `p` *et sinon si* `c == ')'` *il y a deux possibilités : soit* `p` *est vide (que l'on teste avec* `ist_leer`*) et on retourne* `False` *soit* `p` *n'est pas vide et on utilise* `depile(p)` *de l'exercice 01.*
   Il reste à retourner `ist_leer(p)` *après la boucle conditionnelle et la fonction renvoie True s'il y a un bon parenthesage et False sinon*

2. Faire fonctionner le programme avec :

   ce (petit) texte est ((vraiment) super) bien parenthésé

   Puis essayer avec un texte qui ne marche pas.

T.S.V.P →

EXERCICE 03

Création d'une file et fonctions de base

1. Taper les fonctions suivantes

```
In [2]: def cree_file():
   ...:     return []
In [3]: def enfile(x,f):
   ...:     f.append(x)
In [4]: def defile(f):
   ...:     return f.pop(0)
```

2. En utilisant les trois fonctions précédentes, créer une file `f` puis y enfiler les 14 premiers carrées d'entiers (à partir de 1). Afficher ensuite `f`. Puis défiler `f` de ses trois derniers éléments. Afficher de nouveau `f`

   Indications *: On tape `f = cree_file()` puis dans une boucle `for`, on utilise `enfile` autant de fois que nécessaire. Puis on refait une boucle `for` et on utilise `defile` autant de fois que nécessaire*

EXERCICE 04

Utilisation d'une file pour résoudre le problème de Flavius Josèphe (67 apr. J.C)

On considère le problème de Flavius Josèphe (le jour où il failli mourir, voir sa biographie dans le corrigé) : une assemblée de `n` personnes, numérotées de `1` à `n` forme un cercle. On décide de parcourir le cercle dans l'ordre croissant des numéros et d'éliminer une personne sur deux (qui sort alors du cercle), et ainsi de suite jusqu'à ce qu'il ne reste plus qu'une personne dans le cercle. On note `josephe(n)` la fonction Python qui, à chaque entier `n` (entier non nul) associe le numéro de la dernière personne restant après élimination de tous les autres candidats.

1. Faire un dessin du cercle initial en numérotant les personnes pour `n = 4` et trouver avec ce dessin ce que renvoie `josephe(4)`.
   Faire de même avec `josephe(7)`

   Indications *: On remarque qu'à la fin du premier tour, toutes les personnes aux positions paires ont èté éliminées.*

2. Taper (si ce n'est déjà fait) les procédures `cree_file` qui crée une file, puis `enfile(x,f)` qui enfile `x` dans la file `f` puis `defile(f)` qui enlève le premier de liste de la file.

3. (a) Écrire en Python la fonction `josephe(n)` qui donne le numéro de cette dernière personne. On usera d'une file et on utilisera les trois fonctions précédentes.

   Indications pour construire la fonction `josephe(n)` *: On commence par créer une file nommée `candidats` puis on insère tous les candidats dans l'ordre de leur numéro avec une boucle `for` Ensuite, tant que la file n'est pas vide, c'est-à-dire tant que `len(candidat)>0` on extrait le candidat de tête (avec `defile`) que l'on replace en fin de file (avec `enfile`), c'est le candidat non éliminé et on extrait le candidat suivant (avec `defile`) que l'on ne remet pas dans la file, il est éliminé ! Puis, à la fin, on retourne le dernier candidat défilé car il est devenu seul et gagnant.*

   (b) Faire le cas `n=5` à la main puis retrouver ce résultat avec Python. Retrouver aussi avec Python `josephe(4)` et `josephe(7)`

   (c) On sait (voir la biographie de Flavius Josèphe) qu'il avait 40 personnes avec lui quand il a dû faire ce terrible jeu. Sachant qu'il a triché, quel numéro a-t-il pris pour se retrouver le dernier, c'est-à-dire le survivant ?