

1 Indexations

1. Exécuter les instructions ou évaluer les expressions suivantes :

```
l = [1, 2, 3, 4, 5, 6, 7, 8, 9]
len(l)
l[0]
l[len(l)]
l[5]
l.append(10)
l[-1]
l[-len(l)]
l[1:6]
l[:4]
l[4:20]
l[:3]
l[1:5:-1]
l[len(l)-2: 0: -2]
l[1,6,2]
```

2 Lecture de code

On donne le code des fonctions suivantes :

```
def maxi(l:list[int]) -> int:
    res = 0
    for elem in l:
        if elem > res:
            res = elem
    return res

def deco(s:str) -> str:
    res = ''
    for car in res:
        res = res + car + '*'
    return res
```

2. La fonction `maxi` n'a pas de précondition et est censée renvoyer le plus grand élément de la liste. Dérouler l'appel à `maxi([3, -4, 5])` puis `maxi([-3, -4, -5])`. Que pensez-vous de la correction de cette fonction ? Voyez-vous quoi corriger ?
3. La fonction `deco` sans précondition est censée intercaler entre chaque paire de caractères consécutifs de son paramètre une étoile. Par ex on souhaite que `deco('ami')` renvoie `'a*m*i'`. Dérouler l'appel à `deco('ami')`. Que pensez-vous de la correction de cette fonction ? Voyez-vous quoi corriger ?

3 Écriture de tests

Écrire des test pour chaque fonction de la section “Écriture de code”, en pensant bien à prévoir :

- un ou plusieurs cas nominaux avec des itérables de taille au moins 3, exhibant les comportements essentiels de la fonction
- les cas des itérables les plus petits qui satisfont la précondition.

4 Écriture de code

4.1 Semaine

On associe à chaque jour de la semaine son numéro entre 1 et 7, le numéro 1 étant le lundi.

4. Écrire une fonction `nom_jour` qui prend en paramètre un numéro de jour et renvoie le jour de la semaine associé.
5. Écrire une fonction `lendemain` qui prend en paramètre un numéro de jour et renvoie le lendemain du jour de la semaine associé.

4.2 Des listes et des chaînes

On considère une chaîne de caractères contenant la représentation Python d'une liste :

```
>>> l = [1, 2, 3, 4]
>>> s = str(l)
>>> s
'[1, 2, 3, 4]'
```

On souhaite recalculer la liste d'origine.

6. Pourquoi l'expression `list(s)` ne répond-elle pas au problème ?
7. Écrire une fonction `chaine2liste` qui prend en paramètre une chaîne de caractères produite par la conversion d'une liste d'entiers par `str` et renvoie la liste d'origine.

```
>>> chaine2liste(str([1, 281, 3, 4]))
[1, 281, 3, 4]
```

NB la conversion par la fonction `int()` d'une chaîne de caractères représentant un entier en entier accepte les caractères d'espacement :

```
>>> int('    45    \t\n')
45
```

4.3 Noms de domaine

Un nom de domaine est composé de 3 éléments. Par exemple, pour `www.univ-lille.fr` on trouve :

- l'extension `.fr`
- le domaine de deuxième niveau (en anglais *Second-Level Domain* ou SLD) `univ-lille`
- le sous-domaine `www`

Un gérant d'entreprise souhaite acheter des noms de domaine pour lesquels il a choisi des extensions et des domaines de deuxième niveau.

8. Écrire une fonction `nom_domaines` qui prend en paramètre une liste de chaînes de caractères `extensions` non vide et une liste de chaînes de caractères `deuxieme_niveaux` et qui renvoie la liste des noms de domaines pour le sous-domaine `www`.

Par exemple `nom_domaines(['fr', 'net', 'eu'], ['mondomaine', 'mon_domaine_a_moi'])` renvoie la liste contenant dans cet ordre :

- `'www.mondomaine.fr'`
- `'www.mondomaine.net'`
- `'www.mondomaine.eu'`
- `'www.mon_domaine_a_moi.fr'`
- `'www.mon_domaine_a_moi.net'`
- `'www.mon_domaine_a_moi.eu'`

4.4 Nombre max d'occurrences successives

9. Écrire une fonction `max_identiques` qui prend en paramètre une liste d'entiers qui renvoie le nombre maximum d'éléments consécutifs identiques.

```
>>> max_identiques([1, 2, 2, 2, 1, 6, 6, 5])
3
```