TD 7 : Prog.

1. Exécuter les instructions ou évaluer les expressions suivantes :

```
l = [1, 2, 3, 4, 5, 6, 7, 8, 9]
len(l)  9
l[0]  1
l[len(l)] IndexError
l[5] 6
l.append(10) None
l[-1] 9
l[-len(l)] 1
l[1:6]  [2,3,4,5,6].
l[:4] [1,2,3,4].
l[4:20][4,5,6,7,8,9].
l[::3] [1,4,7]
l[1:5:-1]  []
l[len(l)-2: 0: -2] -> [7,5,3,1]
l[1,6,2] TypeError.
```

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,-4,5]	elem	res
"	3	3
"	-4	3
"	5	5.

[-3,-4,-5]	elem	res
"	-3	~~0~~
"	-4	0
"	-5	0

fonct° pas correcte, il faut mettre res = l[0]

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

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 ?

s	car	res
'ami'	a	a*
"	m	a*m*
"	i	a*m*i*

Return res[:-1]

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.

```
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
```

$$$ maxi([])
0
$$$ maxi([7,-2,7,93])
93

$$$ deco('')
''
$$$ deco('salut')
"s*a*l*u*t"

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.

```
def nom_jour(n:int)->str:
    """
    Précond: 0 ≤ n ≤ 6.
    Exemples:
    $nom_jour(3)
    'jeudi'
    $nom_jour(6)
    'dimanche'
    """
    return ["lundi","mardi","mercredi","jeudi","vendredi",
            "samedi","dimanche"][n],

def lendemain(n:int)->str:
    """
    Précond: 0 ≤ n ≤ 6.
    $$$ lendemain(6)
    'lundi'
    $$$ lendemain(0)
    'mardi'
    """
    return nom_jour((n+1)%7).
```

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]
```

6. list(s) renvoie la liste de tous les caractères de s, pas la liste associée à s. (python est bête).

7.
```
def chaine2liste(s:str)->list:
    '''
    Précond: /
    $$$ chaine2liste(`[1,2,3,4]')
    [1,2,3,4]
    $$$ chaine2liste(``)
    [].
    '''
    ret=[].
    buf=``
    for char in s[1:-1]:
        if char==",":
            ret.append(int(buf))
        else:
            buf+=char.
    if buf!="":
        ret.append(buf)
    Return ret
```

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'`

8.
```
def nom_domaine(extensions: list[str], deuxieme_niveaux: list[str]) -> list[str]:
    """
        Précond : /
        Exemples : oui
    """
    Ret = [].
    for extension in extensions:
        for deuxieme_niveau in deuxieme_niveaux:
            Ret.append(extension + deuxieme_niveau).
    Return Ret.
```