Permutations de liste avec l'option liste.pop()

La fonction pop() permet de supprimer un élément dans une liste (par défaut le dernier élément) et renvoie la valeur de cet élément.

Exemple 1 :

```
liste = [25, 10, 36, 100, 777]
test = liste.pop()
print(liste)
print(test)
```

On obtient :

```
[25, 10, 36, 100]
777
```

Exemple 2 :

```
liste = ['chocolat', 'café', 'thé', 'bière', 'eau-de-vie']
a = liste.index('thé')
test = liste.pop(a)
print(a,liste, test)
```

On obtient :

```
2 ['chocolat', 'café', 'bière', 'eau-de-vie'] thé
```

Exemple 3 :

```
PIB_2021 = {'US': 25, 'UE': 17, 'Chine': 19, 'Japon': 5}
Japon_pib = PIB_2021.pop('Japon')
print(PIB_2021)
print(Japon_pib,"trillions de $")
```

On obtient :

```
{'Chine': 19, 'UE': 17, 'US': 25}
5 trillions de $
```

Exemple 4 :

```
aList = [123, 'xyz', 'zara', 'abc']
print ("A List : ", aList.pop())          # print ("A List : ", aList.pop(-1))
print ("B List : ", aList.pop(2))
print("aList : ", aList)
```

On obtient :

```
A List :  abc
B List :  zara
aList : [123, 'xyz']
```

Copie avec [:]

Quand on copie des listes, il faut ajouter [:] pour que les modifications de la deuxième liste ne se répercutent pas dans la première liste.

Exemple 5 :

```
def permutation(L):
    p = [L]
    print(p)
    z = p[0] [:]
    print(z)
    print(z[0])

perm = []
perm = permutation([1,2,3])
```

On obtient :
```
[[1, 2, 3]]
[1, 2, 3]
1
```

```
def permutation(L):
    p = [L]
    print(p)
    z = p [:]
    print(z)
    print(z[0])

perm = []
perm = permutation([1,2,3])
```

On obtient :
```
[[1, 2, 3]]
[[1, 2, 3]]
[1, 2, 3]
```

On comprend toute l'importance de la ligne 3 : **z = p[0] [:]**.

Exercice 1 : On cherche les permutations de la liste $[1,2,3]$.

Premier essai :

```
def permutation(L):
    p = [L]
    z = p[0] [:]
    for j in range(0,2):
        z.append(z.pop(0))
        p.append(z[:])
    return p

perm = []
perm = permutation([1,2,3])
print(perm)
```

On obtient : [[1, 2, 3], [2, 3, 1], [3, 1, 2]] !!! → La réponse est incomplète

Deuxième essai : (la version de droite est sur-détaillée)

```
def permutation(L):
    p = [L]
    z = p[0] [:]
    for i in range(0,2):
        z.append(z.pop(0))
        p.append(z[:])
    z.append(z.pop(1))
    p.append(z[:])
    for j in range(0,2):
        z.append(z.pop(0))
        p.append(z[:])
    return p

perm = []
perm = permutation([1,2,3])
print(perm)
```

```
def permutation(L):
    p = [L]
    z = p[0] [:]
    for i in range(0,2):
        z.append(z.pop(0))
        p.append(z[:])
        print(p)
    z.append(z.pop(1))
    p.append(z[:])
    print(p)
    for j in range(0,2):
        z.append(z.pop(0))
        p.append(z[:])
        print(p)
    return p
```

On obtient :
[[1, 2, 3], [2, 3, 1], [3, 1, 2], [3, 2, 1], [2, 1, 3], [1, 3, 2]]

```
perm = []
perm = permutation([1,2,3])
print(perm)
```

On obtient :
[[1, 2, 3], [2, 3, 1]]
[[1, 2, 3], [2, 3, 1], [3, 1, 2]]
[[1, 2, 3], [2, 3, 1], [3, 1, 2], [3, 2, 1]]
[[1, 2, 3], [2, 3, 1], [3, 1, 2], [3, 2, 1], [2, 1, 3]]
[[1, 2, 3], [2, 3, 1], [3, 1, 2], [3, 2, 1], [2, 1, 3], [1, 3, 2]]
[[1, 2, 3], [2, 3, 1], [3, 1, 2], [3, 2, 1], [2, 1, 3], [1, 3, 2]]

La MErci MONTPELLIER

Exercice 2 :

On cherche les permutations de la liste $[1,2,3,4]$.

```
def permutation(L):
    p = [L]
    z = p[0] [:]
    for i in range(0,3):
        z.append(z.pop(0))
        p.append(z[:])
    z.append(z.pop(1))
    p.append(z[:])
    for j in range(0,3):
        z.append(z.pop(0))
        p.append(z[:])
    return p

perm = []
perm = permutation([1,2,3,4])
print(perm)
```

On obtient :
[[1, 2, 3, 4], [2, 3, 4, 1], [3, 4, 1, 2], [4, 1, 2, 3], [4, 2, 3, 1], [2, 3, 1, 4], [3, 1, 4, 2], [1, 4, 2, 3]]

La MErci MONTPELLIER