3 Boucles imbriquées

July 26, 2024

[1]:
```
def sontEgales(ch1:str, ch2:str)->bool:
    """
    Retourne True si les deux chaînes ch1 et ch2 sont égales, False sinon.
    """
    if len(ch1) != len(ch2):
        return False
    else:
        for i in range(len(ch1)):
            if ch1[i] != ch2[i]:
                return False
        return True
```

[2]:
```
sontEgales('abcghjkv', 'abcf')
```

[2]: False

[3]:
```
def find(mot:str, texte:str)->int:
    """
    Recherche de l'emplacement de la sous chaîne mot dans la chaîne texte.
    """
    for i in range(len(texte)):
        if sontEgales(mot, texte[i:i+len(mot)]):
            return i
    return None
```

[4]:
```
print(find('ra', 'bra'))
```

1

[5]:
```
def find_all(mot:str, texte:str)->[int]:
    """
    Retourne la liste des indices des emplacements de la sous chaîne mot dans␣
 ↪la chaîne texte.
    """
    li = []
    for i in range(len(texte)):
        if sontEgales(mot, texte[i:i+len(mot)]):
            li.append(i)
```

```
    return li
```

[6]:
```
find_all('ra', 'abracadrabra')
```

[6]:
```
[2, 7, 10]
```

[7]:
```
def voisins(li:[int])->(int, int):
    """
    Retourne un couple d'éléments de li les plus proches.
    """
    m = abs(li[0] - li[1])
    couple = li[0], li[1]
    for i in range(len(li)):
        for j in range(len(li)):
            if i != j and abs(li[i] - li[j]) < m:
                m = abs(li[i] - li[j])
                couple = li[i], li[j]
    return couple
```

[8]:
```
def voisins(li:[int])->(int, int):
    """
    Retourne un couple d'éléments de li les plus proches.
    Version améliorée : on ne parcourt que la moitié de la matrice.
    """
    m = abs(li[0] - li[1])
    couple = li[0], li[1]
    for i in range(len(li)):
        for j in range(i+1, len(li)):
            if abs(li[i] - li[j]) < m:
                m = abs(li[i] - li[j])
                couple = li[i], li[j]
    return couple
```

[9]:
```
from random import randint
```

[17]:
```
help(randint)
```

```
Help on method randint in module random:

randint(a, b) method of random.Random instance
    Return random integer in range [a, b], including both end points.
```

[10]:
```
def alea(n:int, N:int)->[int]:
    """
    Retourne une liste de n entiers aléatoires compris entre 0 et N.
    """
    return [randint(0, N) for i in range(n)]
```

[18]:
```
def alea(n:int, N:int)->[int]:
    """
    Retourne une liste sans doublons de n entiers aléatoires compris entre 0 et␣
 ↪N.
    """
    assert N > n, 'Tirage sans doublons impossible'
    li = []
    for i in range(n):
        a = randint(0, N)
        while a in li:
            a = randint(0, N)
        li.append(a)
    return li
```

[19]:
```
li = alea(10, 30)
li
```

[19]: [30, 14, 0, 25, 1, 16, 8, 3, 12, 23]

[20]:
```
voisins(li)
```

[20]: (0, 1)

[21]:
```
def tamis(li:[int])->[int]:
    """
    Applique la remontée d'une bulle :
    on échange deux éléments consécutifs s'il ne sont pas dans l'ordre.
    """
    for i in range(len(li)-1):
        if li[i+1] < li[i]:
            li[i+1], li[i] = li[i], li[i+1]
    return li
```

[22]:
```
def bubble_sort(li:[int])->[int]:
    """
    Retourne la liste li triée avec l'algorithme du tri à bulles.
    """
    for _ in range(len(li)):
        li = tamis(li)
    return li
```

[23]:
```
li = alea(10, 20)
print(li)
bubble_sort(li)
```

```
[2, 3, 8, 9, 1, 7, 4, 10, 18, 16]
```

[23]: [1, 2, 3, 4, 7, 8, 9, 10, 16, 18]

[ ]:

[ ]:

[ ]: