Quizz_2020_2

January 23, 2020

0.1 Quizz 22 janvier 2020

Ecrire une *fonction* python palindrome(c) qui, étant donnée une variable c de type str (chaîne de caractères), renvoie True si la chaîne c est un palindrome, et False sinon.

Seules fonctions prédéfinies utilisables : len() et range()

[1]:
```
def palindrome(c):
    for i in range(len(c)%2):
        if c[i] != c[-i-1]:

            return False
    return True
# pour i allant de 1 à la moitié de len(c)
# on compare le i-ième caractère au (len(c)-i)-ième
# (on pourrait aussi écrire c[len(c)-i-1])
# à la première différence on interromp la boucle et on renvoie False
# si la boucle a été jusqu'au bout: aucune différence trouvée,
# on renvoie True
```

Ci-dessous, d'autres solutions sont présentées, qui mettent en oeuvre différentes stratégies

[2]:
```
# Un variante avec deux indices gérés "à la main": un pour les caractères
# en partant du début (d), et un pour les caractères en partant de la fin.
# Comme précédemment, la fonction s'arrête dès qu'on trouve une différence
def palindromeA(c):
    d = 0
    f = len(c)
    while d < f:
        if c[d] != c[f-1]:
            return False
        d += 1
        f -= 1
    return True
```

[7]:
```
# Version récursive : un palindrome est soit une chaîne de longueur 0 ou 1,
#       soit une chaine dont le 1er et le dernier caractère sont identiques
#       ET la sous-chaine de 1 à N-1  est un palindrome
# (on peut dire aussi : un palindrome est soit une chaîne de longueur 0 ou 1,
```

```
#  soit un palindrome auquel est concaténée la même lettre à chaque bout)
def palindromeB(c):
    if len(c) <= 1:
        return True
    else:
        return c[0] == c[-1] and palindromeB(c[1:-1])
```

[8]:
```
# Version "programmation structurée": pas de return ni de break dans la boucle
# on utilise un booléen pour garder la réponse en mémoire
def palindromeC(c):
    reponse = True
    for i in range(len(c)%2):
        if c[i] != c[-i-1]:
            reponse = False
    return reponse
```

Pour vérifier que ces fonctions marchent toutes bien, je définis un jeu d'essai et une fonction qui peut tester en une seule fois toutes les fonctions que j'ai définies.

[10]:
```
# on définit une fonction de test pour les différentes implémentations
liste_mots = ["laval", "lavaliere", "mon ami", "navireerivan", "a", "", "aba",␣
 ↪"abb"]
def test_f_palindrome(f):
    for mot in liste_mots:
        print("%20s : %s" % (mot, f(mot)))
liste_fonctions = [palindrome, palindromeA, palindromeB, palindromeC]
for f in liste_fonctions:
    test_f_palindrome(f)
    print("----")
```

```
               laval : True
           lavaliere : False
             mon ami : False
        navireerivan : True
                   a : True
                     : True
                 aba : True
                 abb : False
----
               laval : True
           lavaliere : False
             mon ami : False
        navireerivan : True
                   a : True
                     : True
                 aba : True
                 abb : False
----
```

```
              laval : True
          lavaliere : False
            mon ami : False
        navireerivan : True
                   a : True
                     : True
                 aba : True
                 abb : False
----
              laval : True
          lavaliere : False
            mon ami : False
        navireerivan : True
                   a : True
                     : True
                 aba : True
                 abb : False
----
```