interro_rapide_20_minutes_2014_09

July 1, 2022

1 1A.e - Correction de l'interrogation écrite du 26 septembre 2014

chaîne de caractères, tri, fonction

[1]:
```
from jyquickhelper import add_notebook_menu
add_notebook_menu()
```

[1]:
```
<IPython.core.display.HTML object>
```

1.0.1 Enoncé 1

Q1 Ecrire une fonction qui retourne 1 si n est un multiple de 2014.

[2]:
```
def mul2014(n):
    return 1 if n % 2014 == 0 else 0

print(mul2014(2014), mul2014(2015))
```

```
1 0
```

Pour cette question, quelques élèves ont vérifié que `n` était plus petit que 2014 d'abord. Ce n'est pas vraiment la peine.

Q2 Calculer le minimum des cosinus des entiers de 1 à 10.

[3]:
```
import math
min ( math.cos(i) for i in range(1,11) )
```

[3]:
```
-0.9899924966004454
```

`range(0,10)` va de 0 à 10 exclu.

[4]:
```
list(range(0,10))
```

[4]:
```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

Q3 Ecire une fonction qui vérifie qu'une chaîne de caractères est symétrique (ex: `kayak`).

[5]:
```
def symetrie(s):
    i=0
    j=len(s)-1
    while i < j :
```

```
        if s[i] != s[j] : return False
        i += 1
        j -= 1
    return True

print(symetrie("kayak"), symetrie("kakaks"))
```

```
True False
```

Sur cette question, on fait régulièrement deux erreurs lorsqu'on commence à programmer :

- Le dernier indice est `len(s)-1` et non `len(s)`. Le premier indice est 0.
- Il ne sert à rien que `i` aille de 0 à `len(s)-1`, aller de 0 à `len(s)//2` suffit.

1.0.2 Enoncé 2

Q1 Ecrire une fonction qui retourne 1 si une chaîne de caractères contient un A, 0 sinon.

[6]:
```
def contient_A(s):
    return 1 if "A" in s else 0

print(contient_A("e"))
```

```
0
```

Utiliser une boucle n'était pas nécessaire, un simple test suffit.

Q2 Calculer $\sum_{i=1}^{10} \frac{x^{2i}}{\sin i}$.

[7]:
```
import math
x = 0.3
sum ( x**(2*i) / math.sin(i) for i in range(1,11) )
```

[7]:
```
0.12093467645909634
```

Q3 Ecrire qu'une fonction qui vérifie si un tableau est trié.

[8]:
```
def est_trie(tab):
    for i in range(1,len(tab)):
        if tab[i-1] > tab[i] : return False
    return True

est_trie( [1]),est_trie( [1,2,3]),est_trie( [1,2,3,0])
```

[8]:
```
(True, True, False)
```

Quelques élèves ont écrit quelque chose comme :

[9]:
```
def est_trie(tab):
    res = tab.copy()
    res.sort()
    return res == tab
```

Bien que ceci soit tout-à-fait correct, le fait de trier une copie du tableau nécessite des calculs inutiles. Le coût d'un tri est en $O(n \ln n)$ alors que tester si le tableau est trié est au pire de $O(n)$ puisqu'il suffit de le parcourir une fois. J'ajoute une dernière remarque : sans copie la fonction `est_trie` retourne toujours vrai.

[10]:
```
def est_trie_nocopy(tab):
    res = tab
    res.sort()
    return res == tab

t = [ 0,1,2 ]
print( est_trie(t), est_trie_nocopy(t) )

t = [ 0,1,2,0 ]
print( est_trie(t), est_trie_nocopy(t) )
```

```
True True
False True
```

La raison est l'instruction `res = tab` crée une autre variable `res` mais l'instruction n'implique pas la copie de la liste. Elle ne fait que donner un autre nom à la même liste. Ainsi :

[11]:
```
t = [0,1,2]
t2 = t
t2[0] = 1000000000

t,t2
```

[11]:
```
([1000000000, 1, 2], [1000000000, 1, 2])
```

Modifier la liste `t` revient à modifier la liste `t2` puisque ce sont les mêmes. Cela est dû au fait que le langage Python ne copie les listes, les dictionnaires ou les instances de classes que si ces copies sont demandées explicitement. Par défaut, le langage évite toute copie.

[12]:
```

```