while_prolongements

October 21, 2019

Exemple 3 : localisation

Ce programme affiche un itinéraire entre Reims et Marseille qui a été enregistré dans une liste "itineraire"

[0]:
```
itineraire =␣
 ↪["Reims","Châlons-en-Champagne","Troyes","Dijon","Mâcon","Lyon","Valence","Avignon","Marsei
k = 0                    #l'indice de la liste itineraire commence à 0
fin = len(itineraire)-1  #la variable fin est affectée de la valeur de la␣
 ↪taille de la liste -1
while k != fin:
    print("Vous êtes dans la ville de ", itineraire[k])
    k = k+1
print("Vous êtes arrivé à", itineraire[k])
```

On peut prévoir un temps de suspension du programme après chaque affichage d'une ville.

[0]:
```
import time  #importe la bibliothèque time
itineraire =␣
 ↪["Reims","Châlons-en-Champagne","Troyes","Dijon","Mâcon","Lyon","Valence","Avignon","Marsei
k = 0                    #l'indice de la liste itineraire commence à 0
fin = len(itineraire)-1  #la variable fin est affectée de la valeur de la␣
 ↪taille de la liste -1
while k != fin:
    print("Vous êtes dans la ville de ", itineraire[k])
    time.sleep(4) #suspend l'exécution du programme pendant 4 secondes
    k = k+1
print("Vous êtes arrivé à", itineraire[k])
```

Exercice : Ecrire un programme qui demande à l'utilisateur sa ville d'arrivée puis affiche l'itinéraire jusqu'à la ville d'arrivée. Prévoir un temps de suspension de 5 secondes entre chaque affichage.

Exemple 4 : remplissage d'une liste en vue d'un traitement ultérieur

[0]:
```
liste=[]
entree='go!'                  # pour forcer l'entrée dans la boucle
while entree!= 'fin':
  entree=input('Entrer la valeur entière à ajouter ou le mot "fin" si vous␣
 ↪souhaitez arrêter le remplissage de la liste :     ')
  if entree!= 'fin':
    entree=int(entree)
    liste.append(entree)
```

```
print('Vous pouvez utiliser la variable liste = ',liste)
```

Exemple 5 : choix au hasard d'un pixel

En traitement d'image, on peut être amené à choisir au hasard un pixel dans une certaine zone de l'image, par exemple un pixel situé à une distance inférieure à 100 pixels du centre de l'image.

Ou plus simplement, dans la partie supérieure droite de l'image (convenons que c'est celle pour laquelle x>y).

[0]:
```
def choix_pixel():
  from random import randint
  x,y=randint(0,99),randint(0,199)     # choix au hasard d'un entier entre 0 et 99 pour x, entre 0 et 199 pour y (image 100*200 pixels
  while x<=y:                          # on refait le tirage au hasard tant que x est inférieur ou égal à y, ce que nous ne voulons pas !
      x,y=randint(0,99),randint(0,199)
  return x,y

pixel_choisi=choix_pixel()
print(pixel_choisi)
```

1 2) Une erreur à éviter !!

Attention ! Lorsque l'on rentre dans le corps de la boucle, la condition située après le "while" doit y être modifiée sous peine de boucle infinie...

Exemple :

PREPAREZ-VOUS A INTERROMPRE LE PROGRAMME !!

[0]:
```
n=10
while n<20:              #  On soustrait 1 à chaque fois en partant de 10 : n<20 est TOUJOURS vrai.
  print('n vaut ', n)
  n=n-1
```

Contre-exemple :

Parfois justement, on veut une boucle infinie, dans le cas par exemple en robotique de " l'écoute d'un événement " (on capte un obstacle, ou un seuil de température,...). Ce peut être aussi en attente d'un événement du type clic de souris, survolement d'une zone par la souris,... Cela est uilisé dans les jeux (notamment avec le module Pygame).

Voici un exemple un peu artificiel (on peut faire autrement comme ci-dessus (d)) qui affiche un message et sort du flux du corps de la boucle grâce à l'instruction break :

[0]:
```
while True:                    # on voit aussi souvent while 1 :
  rep=input('tu sors ?')
  if rep in ['o','y','O','Y','oui','OUI','yes','YES','ben voui']:
    break
print('tu es sorti')
```

2 3) FOR et WHILE : même combat ?

En fait, on peut souvent remplacer une boucle For par une boucle While, est-ce souhaitable ?

Les différences se font sentir lorsque le nombre de données manipulées est très grand ; ci-dessous deux programmes qui tous deux calculent la somme des N premiers entiers (1+2+3+...+(N-1)+N). La différence est probante pour 10**8 (10 puissance 8) : essayez.

[0]:
```
N=10**7
import time
t=time.clock()
s=0
for i in range(N):
  s=s+i

print(time.clock()-t)
```

[0]:
```
t=time.clock()
s=0
i=0
while i<N:
  s=s+i
  i=i+1
print(time.clock()-t)
```