Fiche de TP no 5

La mémoire d'un programme

Exercice 1. Écrivez un script `fact.py` contenant deux définitions de la fonction factoriel. La première définition, nommée `fact_rec`, utilisera la récursivité pour calculer le factoriel d'un nombre ; la deuxième définition, nommée `fact_while`, utilisera les boucles obtenir le même résultat.

Exercice 2. Allez sur le tutor python :

http://pythontutor.com/visualize.html#mode=edit

et faites un copier-coller du script `fact.py` dans la fenêtre du code. Ajoutez dans la fenêtre du code cette ligne :

```
fact_rec(4)
```

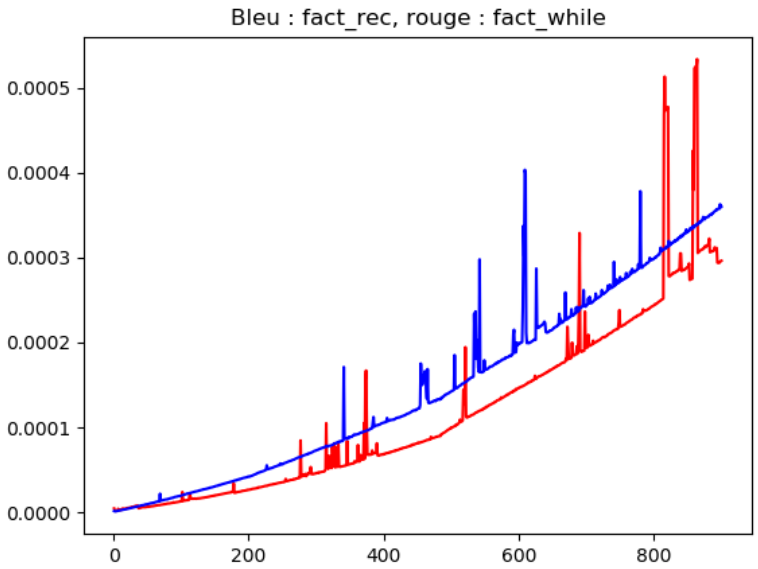
Visualisez l'exécution du code. Modifiez ensuite le contenu de la fenêtre du code comme suit :

```
# fact_rec(4)
fact_while(4)
```

Visualisez l'exécution du code. Que peut on conclure ?

Exercice 3. On souhaite comparer le comportement des deux fonctions. Ajoutez, dans le script `fact.py`, des instructions pour calculer tous les valeurs du factoriel de n, avec $n = 5, \ldots, 1500$, d'abord en utilisant `fact_rec` et ensuite en utilisant `fact_while`. Que peut on conclure ? Comment modifier la limite au nombre des appels récursifs possibles (cherchez le mot clé `recursionlimit` dans la doc du module `sys`) ?

Exercice 4. Ensuite, on souhaite comparer la performance des deux fonctions. On ajoutera dans le fichier `fact.py` la définition d'une fonction qui affiche le graphique du temps d'exécution des deux fonctions `fact_rec` et `fact_while`, pour tous les valeurs de $n = 1, \ldots, 900$. Voici ce que l'on souhaite obtenir :



— Pour dessiner le graphique, on utilisera le module `mathplotlib`. Pour l'exemple ci-dessus, on a utilisé le modèle trouvé ici.

— Pour mesurer le temps d'exécution de chaque fonction, on utilisera le module `timeit` et sa fonction `timeit`. La documentation du module :

https://docs.python.org/3/library/timeit.html

Voici un exemple d'utilisation de la fonction `timeit.timeit` depuis un script

```
import timeit

def do_something(number):
    pass

number=33
print(timeit.timeit('do_something(number)', globals=globals()))
```

Attention : le paramètre `global=globals()` permet de récupérer la valeur de `number` de l'espace global. Il faudra donc réfléchir si utiliser `import fact` ou bien `from fact import *` lors de l'utilisation de cette fonction.

Débogage post-mortem

Exercice 5. Considérez le script suivant :

```
def split(text):
    words, word = [], ''
    for char in text:
        if char == ' ':
            words.append(word)
            word = ''
        else:
            word += char
    words.append(word)
    return words

def first_letter(word):
    """Cette fonction demande
    une chaine non vide en paramètre"""

    if not word: # si word est vide
        raise ValueError
    else:
        return word[0]

def first_letters(text):
    letters = []
    words = split(text)
    for word in words:
        letters.append(first_letter(word))
    return letters

print(first_letters('Hello beautiful  world'))
```

Ce script ne marche pas, car il lève `ValueError`.

— Pouvez vous deviner où la bogue se trouve, sans utiliser le déboguer ?

— Utilisez le déboguer pour identifier la nature de l'erreur : consultez la pile des appels et les valeurs des variables dans le contexte (cadre d'appel) approprié.

— Corrigez ensuite la bogue.

Débogage pas à pas

Exercice 6. Le script suivant sert à tester le bon fonctionnement de la procédure `delete_key_from_alist` pour retirer d'une liste d'association toute occurrence d'une clé donnée :

```
import random
random.seed()

def delete_key_from_alist(key, assoc_list):
    i=0
    for key_, _ in assoc_list:
        if key_ == key:
            assoc_list.pop(i)
        i += 1
    return assoc_list

def face_ou_pile(n):
    if random.randint(0,1) == 0:
        return 'face'
    else:
        return 'pile'

def rand_assoc_list(size):
    return [(face_ou_pile(n), n) for n in range(size)]

assoc_list = rand_assoc_list(10)
print(delete_key_from_alist('face', assoc_list))
```

La procédure ne marche pas, car le script affiche le résultat suivant :

```
> python face_ou_pile.py
[('pile', 1), ('pile', 3), ('face', 5), ('face', 7), ('pile', 8)]
```

— Identifiez les lignes les plus douteuses de ce script ; placez un ou plusieurs points d'arrêts sur ces lignes.
— Utilisez le déboguer pour inspecter les valeurs des variables lors de l'exécution de ces linges.
— Identifiez clairement c'est laquelle la bogue.
— Corrigez la bogue.