Suite de Conway:

Exemple:

1

11

21

1211

111221

312211

13112221

...

Plan

I ) Introduction

II) Partie Algorithmique

III) Programme en Python

IV) Tests

V) Amélioration possible et conclusion

I] Introduction

Description:

La suite de Conway est une suite de nombre. On part par exemple du nombre 1 et on cherche le nombre suivant.

Pour avoir le terme suivant, il faut compter chaque caractère qui se répète successivement et écrire la lecture de ce comptage.

Introduction:

La suite de Conway est une suite audio-active. Il n'y a pas de formule, à premiére vue, qui puisse donner le terme suivant. Il faut donc écrire un algorithme qui écrit une nouvelle phrase à partir d'une autre phrase. Il faut donc considérer chaque terme, non comme un nombre, mais comme une chaine de caractères. On va décrire plusieurs algorithmes successifs qui permettront l'implémentation d'un programme.

II] Partie Algorithmique

Algorithme 1:

**On appelle P, une phrase de n caractères**

**Tant qu'il y a des caractères dans P faire:**

**Compter le nombre k de caractères C qui se répètent**

**Ecrire à la suite le nombre k et le caractére C**

**FinTantQue**

Commentaires:

la première difficulté est d'interpréter la phrase **"Tant qu'il y a des caractères dans P**".

* Doit on supprimer des caractères dans P ?
* Fait on une boucle sur l'ensemble des caractères de P et utiliser le rang de chaque caractère ? le problème c'est de faire une boucle qui va regarder tous les indices de P et une boucle qui regarde les indices pour lesquelles les caractères se répètent

Les deux idées impliquent deux algorithmes différents. La deuxième idée est plus compliquée à visualiser, on est obligé de créer un indice tampon. On va s'intéresser à la première car elle est facilement lisible et assez intuitive.

* Le mot "**Compter**" implique un compteur ou une formule de comptage

Problème: A partir de quoi compte-t-on ?

* "**Ecrire à la suite**" doit être interpréter comme une chaine que l'on rempli au fur et à mesure. Il faut donc commencer par une chaine vide que l'on remplira

Algorithme 2.1: On va supprimer des caractères dans P

**On appelle P, une phrase de n caractères**

**Psuivant est une chaine vide**

**Tant qu'il y a des caractères dans P faire:**

**Compter le nombre k de fois que le 1er caractère C de P se répètent**

**Dans Psuivant écrire k puis C**

**Enlever les k premiers caractères à P**

**FinTantQue**

**Afficher Psuivant**

Commentaires:

la difficulté est d'interpréter la phrase " **Compter le nombre k de fois que le 1er caractère C se répètent"**

On ne connait pas le nombre de caractère répété, on doit donc faire un **TantQue**.

On peut faire un **Pour** si on remarque qu'il ne peut pas y avoir plus de trois caractères répétés Cela serait à démontrer.

Il est plus efficace de créer une fonction au cas où cette fonction sera utiliser plusieurs fois. Cela favorise aussi une lisibilité de l'algorithme.

Algorithme 2.2:

**Fonction Comptage1erCaractereRepete(Chaine)**

**Indice=0**

**Tant que Chaine(Indice)=Chaine(Indice+1) faire**

**Indice=Indice+1**

**FinTantQue**

**Retourner la valeur Indice+1**

#Programme principal

**On appelle P, une phrase de n caractères**

**Psuivant est une chaine vide**

**Tant qu'il y a des caractères dans P faire:**

**k=Comptage1erCaractereRepete(P)**

**Dans Psuivant écrire k puis le premier caractère de P**

Peut être serait il nécessaire d'écrire des fonctions qui font ces deux actions ?

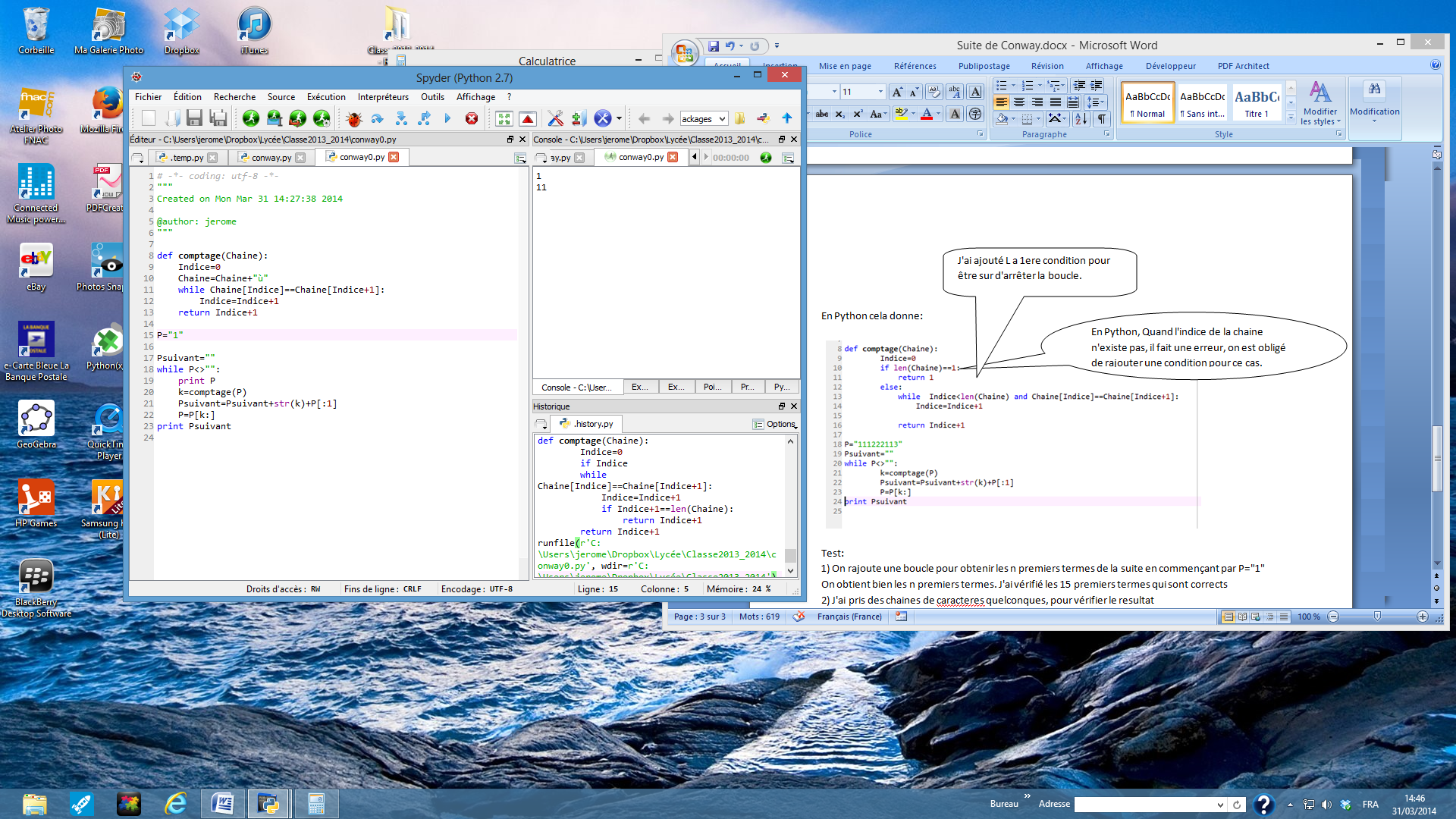
**Enlever les k premiers caractères à P**

**FinTantQue**

**Afficher Psuivant**

III] Programme Python

En Python cela donne:



En Python, Quand l'indice de la chaine n'existe pas, il fait une erreur J'ai rajouté un caractère improbable pour éviter l'erreur.

IV] Tests:

1) On rajoute une boucle pour obtenir les n premiers termes de la suite en commençant par P="1"

On obtient bien les n premiers termes. J'ai vérifié les 10 premiers termes qui sont corrects

2) J'ai pris des chaines de caractères quelconques, pour vérifier le résultat comme "aaaavvvddjhdss"

3) Il se passe quelque chose de bizarre si j'introduis le caractère "ù" mais il n'y a pas d'erreur.

V] Amélioration et conclusion

Amélioration possible:

Faire un programme qui pourrait faire la suite des termes pour tout type de caractère même avec "ù"

Essayer un algorithme qui gérerait les indices plutôt que de couper la chaine et comparer en terme d'efficacité les deux algorithmes

conclusion:

C'est un algorithme qui permet de comprendre la manipulation des chaines de caractère. On s'aperçoit qu'il peut servir à compresser des chaines de caractère avec une compression de type LRE

exemple: "aaabbbbbbcccd" devient "3a6b3c1d". il serait alors intéressant de faire un algorithme qui fait la décompression, c'est à dire qui trouve la chaine de caractère précédente.

Ou encore faire une méga compression hypothétique qui reviendrait au terme initiale de la chaine de Conway. On peut être essayer d'évaluer ce terme.