Session #6 19/08/19 : Ma boite à outils

Obligatoire sinon Emmanuelle râle : Arriver à l'heure !

Mode :

- **1 petit groupe / 1 tableau par groupe / 1 sujet par tableau**
- **Groupes de 4-5 en fonction des présents**
- **20-30' par thème**

Fonctionnement :

- **je choisis les sujets qui m'intéressent et je m'inscris**
- **je squize le reste**

Comment on procède par session :

- **je lis le sujet (2 - 5 min) et je regarde sur le web (à limiter)**
- **je partage mes idées / approches (en mode tour de table)**
- **on décide d'un méta-code "rapide" et on le challenge (*"Last input is always your worst enemy"*)**

A éviter :

- **Que cela ne devienne pas "1 prof, N élèves" pour tous les sujets (nos animatrices de choc sont là ?).**
- **Que chacun puisse chercher un petit moment afin que la découverte d'une astuce soit assimilée par rapport aux connaissances de chacun.**

NB :

- **Toutes les réponses sont disponibles , vous avez le droit de demander la réponse (Ok, c'est du Java).**

Thème 1 : Tricks dans ma boite à outils :

- Split de chaines : **"1 2 3 4" -> [1,2,3,4]**
- Split de changement de valeurs : **"aabbccddeee" → ["aa", "bb", "cc", "dd", "eee"]**
- Génération de séquences à partir d'un mot:
  - mode autocompletion : **abcdef -> [a, ab, abc, abcd, abcde, abcdef, b, bc, bcd, bcde, bcdef, c, cd, cde, cdef, d, de, def, e, ef, f]**
  - mode sous chaine partielle : **test -> [tt, st, tet, tes, test, e, est, es, et, te, s, tst, t, ts]**
  - avec longueur fixe de sous-chaine : **(size 3) abcdef -> [abc, bcd, cde, def]**

Thème 2 : Autres tricks

- Comment gérer les heures/minutes dans mon langage ? "12h50"→"13h10" = "20 min" ?
- **DamerauLevenshtein** : je trouve la distance Levenshtein entre 2 mots. **"trois", "quatre" → 25 ?**

- Valider que 2 lettres se suivent dans l'alphabet : **"az" → false, "cd" → true**
- Affichage de nombres décimaux : **5 chiffres significatifs : 25.00010101 → 25.0001, 3.0000004 → 3.0**
- **Fibonacci** : je code la fonction des n 1ers termes de la suite.

Thème 3 : Parcours de Matrice (tableau)

- J'affiche un **tableau**
- A partir de coordonnées (x, y), je valide que je suis bien dans le tableau et je donne **tous les voisins valides**
- **Trouver toutes les positions** d'un tableau ayant une valeurs donnée : example_valeurs_dans_tableau.txt
- **Trouver toutes les positions continues** ayant la meme valeur à partir d'un point de départ : example_barrage.txt

Thème 4 : "Les constructions résursives c'est sympa mais j'ai pas 10 heures devant moi" *A.Einstein*

- **Battle Dev RegoinJob - Mars 2019**, **Exercice "Extrémites Manquantes"**. (https://www.isograd.com/FR/solutionconcours.php)
  - Ce problème enchaine 3 points techniques simples dont la combinaison le rend sympa.
  - Il nécessite 28 lignes de codes, toutes lignes comprises.
  - https://fr.wikipedia.org/wiki/Arbre_des_suffixes

Groupe 2

Thème 1 :

- Split de chaines : **"1 2 3 4" -> [1,2,3,4]**
  - "1 2 3 4".Split(' ');
  -

```
def ints():
    return list(map(int, input().split()))


def floats():

    return list(map(float, input().split()))
```

  - Pour faire l'inverse: string.Join(" ", **[1,2,3,4]**) => **"1 2 3 4"**
- Split de changement de valeurs : **"aabbccddeee" → ["aa", "bb", "cc", "dd", "eee"]**
  - Regex : (.)\1{1,}

    Split de changement de valeurs C#

```
static List<string> ValueSplit(string entry)
        {
            List<StringBuilder> returnArray = new List<StringBuilder>();
            foreach(char c in entry)
            {
                if(returnArray.Count > 0 && returnArray.Last().ToString().StartsWith(c
                {
                    returnArray.Last().Append(c);
                }
                else
                {
```

```
                    returnArray.Add(new StringBuilder(c.ToString()));
                }
            }
            return returnArray.Select(x => x.ToString()).ToList();
        }
```

- Génération de séquences à partir d'un mot:
  - mode autocompletion : **abcdef -> [a, ab, abc, abcd, abcde, abcdef, b, bc, bcd, bcde, bcdef, c, cd, cde, cdef, d, de, def, e, ef, f]**
  - mode sous chaine partielle : **test -> [tt, st, tet, tes, test, e, est, es, et, te, s, tst, t, ts]**
  - avec longeur fixe de sous-chaine : **(size 3) abcdef -> [abc, bcd, cde, def]**

Thème 2 :

- Comment gérer les heures/minutes dans mon langage ? "12h50"→"13h10" = "20 min" ?
  - split + conversion en minute
    - python : h,m ="12h50".split("h") ; minute= 60*int(h)+int(m)
  - C# : Convertir en TimeSpan puis faire le différence entre les 2

    - var entry = "12h50";

      int hours = int.Parse(entry.Substring(0, 2));
      int minutes = int.Parse(entry.Substring(3, 2));

      var now = new TimeSpan(hours, minutes, 0);

- **DamerauLevenshtein** : je trouve la distance Levenshtein entre 2 mots. **"trois", "quatre" → 25 ?**

  Levenshtein C#

```
public static Int32 Compute(String s, String t)
        {
            int n = s.Length;
            int m = t.Length;
            int[,] d = new int[n + 1, m + 1];

            // Step 1
            if (n == 0)
            {
                return m;
            }

            if (m == 0)
            {
                return n;
            }

            // Step 2
            for (int i = 0; i <= n; d[i, 0] = i++)
            {
            }

            for (int j = 0; j <= m; d[0, j] = j++)
            {
            }

            // Step 3
            for (int i = 1; i <= n; i++)
            {
                //Step 4
                for (int j = 1; j <= m; j++)
                {
                    // Step 5
                    int cost = (t[j - 1] == s[i - 1]) ? 0 : 1;
```

```
                // Step 6
                d[i, j] = Math.Min(
                    Math.Min(d[i - 1, j] + 1, d[i, j - 1] + 1),
                    d[i - 1, j - 1] + cost);
            }
        }
        // Step 7
        return d[n, m];

    }
```

- Valider que 2 lettres se suivent dans l'alphabet : **"az" → false, "cd" → true**
  - Transformer chaque lettre en toLower puis en ASCII
    - Python : ord('a')+1==ord('b') et inversement
    - C# (int) char
- Affichage de nombres décimaux : **5 chiffres significatifs : 25.00010101 → 25.0001, 3.0000004 → 3.0**
  - C# : Math.Round(float, 5).ToString("G29")
  - python : print('{0:.3f}'.format(n))
- **Fibonacci** : je code la fonction des n 1ers termes de la suite

Fibonacci C#

```
    public static int Fibonacci(int position)
    {
        if (position == 0)
            return 1;
        if (position == 1)
            return 1;

        return Fibonacci(position - 2) + Fibonacci(position - 1);
    }
```

Fibonacci Python

```
def fn(rang):
    if rang > 2:
        return fn(rang - 1) + fn(rang - 2)
    elif rang > -1:
        return max(0, rang - 1)
    print('error')
```

Thème 3 :

Parcours de Matrice (tableau)

- J'affiche un **tableau**

Display Matrix

```
        public static void DisplayMatrix<T>(T[,] arr)
        {
            int rowLength = arr.GetLength(0);
            int colLength = arr.GetLength(1);

            for (int i = 0; i < rowLength; i++)
            {
                for (int j = 0; j < colLength; j++)
                {
                    Console.Write(string.Format("{0} ", arr[i, j]));
                }
```

```
                Console.Write(Environment.NewLine);
            }
        }
```

```
python: print(matrice)
```

- A partir de coordonnées (x, y), je valide que je suis bien dans le tableau et je donne **tous les voisins valides**
  - **voisin**

```
def voisin(a,b,matrice):
    for i,j in ((0,1),(0,-1),(-1,0),(1,0)):
        if 0<=a+i<len(matrice[0]) and 0<=b+j<len(matrice) :
            print(matrice[a+i][b+j])
```

- **Trouver toutes les positions** d'un tableau ayant une valeurs donnée : Session #6 19/08/19 : Ma boite à outils
- **Trouver toutes les positions continues** ayant la meme valeur à partir d'un point de départ :Session #6 19/08/19 : Ma boite à outils

  - python

```
def find(mat, bod, i, j):
    for x, y in [(1, 0), (-1, 0), (0, 1), (0, -1)]:
        if 0 <= i + x <= n - 1 and 0 <= j + y <= n - 1:
            if lines[i + x][j + y] == "#":
                if [i + x, j + y] not in mat:
                    mat.append([i + x, j + y])
                    ### Bonus on cherche si c'est un bordure
                    for xx, yy in [(1, 0), (-1, 0), (0, 1), (0, -1)]:
                        if 0 <= i + x + xx <= n - 1 and 0 <= j + y + yy <= n - 1:
                            if lines[i + x + xx][j + y + yy] == ".":
                                bod.append([i + x, j + y])
                    # on fait un récursivité
                    find(mat, bod, i + x, j + y)
```

Thème 4 :

```
chr(97)
```

Groupe 3

Thème 1 :

```
(?<=(.))(?!\\1)
```

```
# "1 2 3 4" -> [1,2,3,4]
print(list(map(int, "1 2 3 4".split())))

# "aabbccddeee" -> ["aa", "bb", "cc", "dd", "eee"]
from itertools import groupby
print(["".join(g) for k, g in groupby("aabbccddeee")])
```

Pour une solution recursive à l'extraction de sous-suite en Python

```
def rec_iter(l):
    if len(l) ==1:
        return [l]
    else:
        a = l[0]
```

```
        list_ = rec_iter(l[1:])
        return  list_ + list(map(lambda x : a +x , list_))

rec_iter("test")
```

Thème 2 :

```
#Conversion de dates python

a = '2019-12-18 00:34'

def convert(a):
    return dt.datetime.strptime(a,'%Y-%m-%d %H:%M')
```

Levenshtein

```
def levenshtein(s, t):
        ''' From Wikipedia article; Iterative with two matrix rows. '''
        if s == t: return 0
        elif len(s) == 0: return len(t)
        elif len(t) == 0: return len(s)
        v0 = [None] * (len(t) + 1)
        v1 = [None] * (len(t) + 1)
        for i in range(len(v0)):
            v0[i] = i
        for i in range(len(s)):
            v1[0] = i + 1
            for j in range(len(t)):
                cost = 0 if s[i] == t[j] else 1
                v1[j + 1] = min(v1[j] + 1, v0[j + 1] + 1, v0[j] + cost)
            for j in range(len(v0)):
                v0[j] = v1[j]

        return v1[len(t)]

#Est-ce que deux lettres se suivent

return False if ord(b) != ord(a) + 1 else True
```

Implémentation Fibonacci efficace

```
def fib(n, computed = {0: 0, 1: 1}):
    if n not in computed:
        computed[n] = fib(n-1, computed) + fib(n-2, computed)
    return computed[n]
```

Thème 3 :

```
#Affichage Table
import sys

tab = [[3,3,4],[4,2,1]]

def print_tab(tab):
    for l in tab:
        sys.stdout.write(' '.join(map(str, l)) + '\n')
print_tab(tab)
```

Voisins (on renvoie un tuple avec les index) :

```
tab = [[3,3,4],[4,2,1],[4,2,1],[4,2,1]]

def neighbor_tab(tab,i,j):
    #Index Voisin en croix
    n = len(tab)
    m = len(tab[0])

    l = []
    for k in {-1,1}:
        if (i + k) % n == i + k:
            l += [(i+k,j)]

        if (j + k) % m == j + k:
            l += [(i,j+k)]
    return l

def neighbor_diag(tab,i,j):
    #Index Voisin en diagonal
    n = len(tab)
    m = len(tab[0])

    l = []
    for k in {-1,1}:
        for il in {-1,0,1}:
            if (i + k) % n == i + k and (j  + il) % m == j + il:
                l += [(i+k,j+il)]

        if (j + k) % m == j + k:
            l += [(i,j+k)]

    return l
```

Thème 4 :