Aide-mémoire Python

1. Instructions de base

- On ne déclare pas les variables. Pas d'instructions de début et de fin ; c'est l'indentation qui joue ce rôle.
- Précéder les commentaires d'un #
- Affichage à l'écran, utiliser :
  `print("bonjour",variable,"au revoir.")`
  (si `variable=alex` l'affichage sera
  `bonjour alex, au revoir.`
- `input()` # saisie au clavier d'une chaîne de caractères à transformer en `int` ou en `float`
- On peut regrouper les instructions avec `N=int(input())`
- Les virgules des décimaux se notent avec des points.
- `a//b` affiche le quotient de a par b ;
  `a%b` affiche le reste de la division euclidienne de a par b.
- Pour générer des nombres aléatoires, on utilise la commande `import random` qui importe la librairie `random`, puis la commande `random.random()` pour générer un nombre entre 0 et 1, ou bien par exemple, la commande `random.randint(1,6)` pour tirer « au hasard » un nombre entier entre 1 et 6.

2. Structure de contrôle

- `if(N>1):` # aller à la ligne et décaler, revenir au même niveau pour le `else:` et de nouveau à la ligne en décalant.
  `elif:` correspond au sinon si (contraction de `else if`)
- On peut combiner plusieurs conditions avec le `and` le `or` et le `not`.

3. Boucles

- `while(N>1):` # aller à la ligne et décaler
- `for i in range(5):` #aller à la ligne et décaler
  l'indice i prendra les valeurs 0, 1, 2, 3 et 4.
- `for i in range(3,10,2)`
  l'indice i prendra les valeurs 3, 5, 7 et 9.
- `for i in range(4,-1,-1)`
  l'indice i prendra les valeurs 4, 3, 2, 1 et 0.

4. Listes

En Python, une liste est notée entre crochets `[` et `]` et les éléments sont séparés par des virgules.

- Pour créer une liste, voici 3 méthodes :
  - Si on veut saisir une liste complète connue, taper directement `Liste=[1,3,5,7]`.
  - Pour remplir une liste remplie de 0, on utilise `Liste=[0]*4` ou `Liste=[0 for i in range(n)]`.
  - On crée une liste vide en début de programme `Liste=[ ]`. Ensuite, on la remplit en utilisant la fonction `append` :

```
Liste=[ ]
for i in range(0,10):
        Liste.append(i)
#Crée la liste [0,1,2,3,4,5,6,7,8,9]
```

- On ne peut pas entrer directement une valeur saisie au clavier avec `.append`. Il faut faire

```
val=int(input())
Liste.append(val)
```

- Pour afficher une liste, faire `print(Tab)` ou utiliser une boucle pour avoir un élément par ligne.
- La fonction *len* donne la longueur de la liste c'est-à-dire le nombre d'éléments qu'elle contient.

```
Liste=[1,3,5,7,9]
a=len(Liste)
print(a) #donnera 5
```

- Les éléments d'une liste de `n` éléments sont numérotés à partir de `0` jusqu'à `n-1`.
- Le contenu de la case i est noté `Liste[i]`.
  `Liste[3]` contient la 4ième valeur. Dans l'exemple précédent, elle vaut 7.

5. Matrices

Une matrice est représentée en Python par une liste de listes :
On veut représenter la matrice

$$\texttt{Mat} = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix}.$$

Taper dans l'éditeur :

```
Mat=[[1,2,3],[4,5,6]]
```

`Mat[1]` contient `[4,5,6]` car c'est la ligne numéro 1 et que Python numérote à partir de 0. `Mat[0][2]` contient 3, c'est l'élément de la première ligne (`i=0`), 3ième colonne (`j=2`).

- Nombre de lignes : `n=len(Mat)` # donne à n la valeur 2.
- Nombre de colonnes : `p=len(Mat[0])` # donne à p la valeur 3.
- L'élément (i, j) de la matrice est noté `Mat[i][j]`

- Pour construire une matrice de taille $n * p$ définie, préremplie de 0, faire :

  ```
  M=[[0 for j in range (p)]for i in range(n)]
  ```

  Cela crée n listes de p éléments chacune.
  Il faut toujours initialiser une matrice avant de la remplir.

- Si on veut avoir deux matrices identiques au départ pour en modifier une et garder l'autre, il faut copier les éléments un à un à l'aide d'une double boucle.
  On veut copier `Mat` dans la matrice M :

  ```
  M=[[0 for j in range (3)]for i in range(2)]
  for i in range(2):
      for j in range(3):
         M[i][j]=Mat[i][j]
  ```

- `print(Mat)` affiche `[[1,2,3],[4,5,6]]`

- La commande

  ```
  for i in range(2):
     print(Mat[i])
  ```

  affiche

  ```
  [1,2,3]
  [4,5,6]
  ```

- Pour afficher tous les éléments les uns en dessous des autres, taper :

  ```
  for i in range(2):
     for j in range(3):
        print(Mat[i])
  ```

6. Fonctions et procédures

On utilise pour les deux la structure
`def nomfonction(param1,param2):`
puis on va à la ligne et on décale (prendre un quadratin).

Pour retourner le résultat de la fonction, on utilise l'instruction `return res` (`res` étant le nom de la variable retournée).