



DS 1

Informatique

Correction

Simon Dauguet
simon.dauguet@gmail.com

Mercredi 24 Septembre 2025

Exercice 1 (Questions de cours) :

1. La commande d'interactivité est la commande `input("texte")`. Au compilage, la chaîne de caractères `"texte"`. L'ordinateur récupère ce qui est tapé par l'utilisateur (jusqu'à l'appui sur la touche "Entrée") sous la forme d'une chaîne de caractères.

2. Il y a deux boucles : une boucle `for` et une boucle `while` dont les syntaxes sont :

```
1 for i in range(n,m,k) :  
2     instructions
```

```
1 while conditions :  
2     instructions
```

3. La syntaxe complète de la structure conditionnelle est :

```
1 if condition1 :  
2     instructions1  
3 elif condition2 :  
4     instruction2  
5 elif condition3 :  
6     instruction3  
7 ...  
8 else :  
9     instruction_fin
```

4. L'opérateur `\` correspond au quotient dans la division euclidienne ; `%` est le reste dans la division euclidienne ; `+=` est une incrémentation automatique (`x+=1` correspond à `x=x+1`) ; `**` correspond à la puissance.

5. L'opérateur `+` peut être une addition (entre deux flottants, par exemple) ou une concaténation (entre deux listes, par exemple).

6. Il n'y a que deux booléens : `True` et `False`. Le comparateur d'égalité (dont le résultat est donc un booléen) est `==`.

7. La fonction `print` correspond à un affichage (à titre informatif) et ne peut être réutiliser. Le `return` s'utilise en général à la fin d'une fonction et sert à renvoyer une ou des variables pour une utilisation ultérieure.

8. Une fonction et une procédure sont des algorithmes. Les fonctions se terminent par un `return` (et donc renvoie un résultat qui peut être réutilisé), une procédure est un algorithme qui ne renvoie rien.

9. Pour mettre la valeur d'une variable `var` dans une chaîne caractères (et ne pas obtenir la chaîne de caractères `"var"`), il faut utiliser la fonction `str` : `str(var)` correspond à la chaîne de caractères dont les caractères sont de la valeur de la variable `var`.

Exercice 2 (Lecture de code et cours) :

On considère le code suivant :

```
1 def Conway(n) :
2     """n -- entier naturel"""
3     N=str(n)
4     L=""
5     k=0
6     while k<len(N) :
7         c=1
8         P=True
9         while k+c<len(N) and P==True :
10             if N[k+c]==N[k] :
11                 c=c+1
12             else :
13                 P=False
14             L=L+str(c)+str(N[k])
15             k=k+c
16     return(int(L))
```

1. La ligne 2 est le manuel de la fonction. C'est ce qui sera affichée à l'appel de `help(Conway)`.

2. La commande `N[k]` veut dire que l'on considère le k -ème caractère de la chaîne de caractère N . Et `len(N)` correspond au nombre de caractère de la chaîne de caractère N . On rappelle que python commence à compter à 0, donc le premier caractère est d'indice 0 et le dernier est d'indice `len(N)-1` pour qu'il y ait en tout `len(N)` caractère.

3. La fonction `str` est une fonction de transtypage permettant de transformer un objet ou la valeur d'une variable en chaîne de caractères. À la ligne 14, on s'intéresse à la valeur de variable c et le caractère en position k de la chaîne N . En écrivant `L+"c"+N[k]` on aurait concaténé la chaîne L avec la chaîne `"cN[k]"` et donc on aurait pas concaténé avec les valeurs des variables. C'est une erreur classique que de confondre les caractères du nom d'une variable avec la valeur de la variable.

4. La variable locale P est là simplement pour s'assurer de la terminaison de la deuxième boucle `while`. Sans elle, on pourrait avoir une boucle infini. En effet, si on l'enlevait et que les deux premiers caractères étaient différents, alors on n'entre par dans la structure `if`. Et comme il n'y a rien dans le `else`, rien n'est fait. Donc le booléen de la condition du second `while` ne change jamais de valeur et donc il est indéfiniment vrai. On a alors une boucle infini.

La variable c est un compteur. Il compte le nombre de caractère successif identique.

5. En calculant, on a `Conway(1)=11`, `Conway(Conway(1))=Conway(11)=21` et `Conway(Conway(Conway(1)))=Conway(21)=1211`.

Donc a priori, on devrait obtenir (et c'est le cas), `Conway(223)=2213`.

Ce code fait référence à la suite audio-descriptive de John Conway (1937-2020). Chaque terme décrit le précédent à haute voix. Dans 223, il y a : deux 2, un 3.

6. On propose :

```
1 def Suite_Conway(p,n) :
2     N=p                                # N est sera le nouveau terme de la suite
3     for k in range(0,n) : # On calcul n termes successifs de la suite. Si n=0, on ne rentre pas
4         N=Conway(N)           # terme suivant
5     return(N)                 #On renvoie le résultat.
```

7. À la ligne 4, l'opérateur `*` correspond à une concaténation répétée 10 fois. Autrement dit, la ligne 4 permet de faire une liste de longueur 20 et dont tous les éléments sont nuls.

8. L'appel de `mystere(2025)` renvoie l'entier 2250.

La fonction `mystere` permet de classer les chiffres du nombre n par ordre croissant de 1 à 9 et de mettre les 0 en dernière position.

9. La condition de la ligne 9 permet d'éviter de prendre en compte les chiffres qui n'apparaissent pas dans le nombre n . Autrement dit, dans `mystere(2025)`, on ne veut pas compter les 1 qui ne sont pas dans le nombre. On ne veut que décrire les chiffres qui apparaissent effectivement dans le nombre n .

10. La variable `C` est une liste de compteurs qui compte le nombre d'apparition de chaque chiffre (de 0 à 9) dans l'écriture décimale de l'entier n . Autrement dit, si $k \in \{0, \dots, 9\}$, alors `C[k]` correspond au nombre d'apparition du chiffre k dans l'écriture décimale de n .

11. La fonction `Conway` correspond à la suite audio-descriptive de Conway, i.e. elle permet de décrire "à voie haute" le nombre n . Un ordre d'apparition des chiffres est donc conservé. La fonction `mystere` ne fait que compter les nombres d'occurrences de chaque chiffres, indépendamment de leur ordre relatif. La fonction `Conway` décrit l'entier n en tenant compte de l'ordre d'apparition des chiffres; la fonction `mystere` ne fait que réordonner les chiffres de n par ordre (presque) croissant.

12. Plusieurs version sont possibles pour la fonction `somme`. On en propose une :

```
1 def somme(n:int) -> int :
2     m = str(mystere(n)) #
3     p = len(m)//2 # len(m) est paire. On compte deux en deux. m[0] est un compteur, m[1] un chiffre.
      m[2] un compteur et m[3] un chiffre. Etc.
4     S = 0 # somme des chiffres de n
5     for k in range(p) :
6         S += int(m[2*k])*int(m[2*k+1])
7     return(S)
```

Exercice 3 (Quelques fonctions) :

On considère la suite $(u_n)_{n \in \mathbb{N}}$ définie par $u_0 = 4$ et $\forall n \in \mathbb{N}, u_{n+1} = u_n^2 - 3$.

1. On propose la fonction

```
1 def suite(n:int) -> float :
2     u=4
3     for k in range(0,n) :
4         u=u**2-3
5     return(u)
```

2. Puis la fonction

```
1 def suiteListe(n:int) -> list :
2     L=[4]
3     while len(L)<n :
4         L = L + [L[-1]**2-3]
5     return(L)
```

3. Et enfin :

```
1 def depassement(M:float) -> int :
2     u=4
3     n=0
4     while u<M :
5         u=u**2-3
6         n+=1
7     return(n)
```

Exercice 4 (Chaînes de caractères) :

1. C'est une question de compteur. Il suffit de s'inspirer de la fonction `Conway` au dessus.

```
1 def nbOcc(let:str, phrase:str) -> int :
2     cmpt = 0 # compteur
3     for k in range(len(phrase)) :
4         if phrase[k] == let :
5             cmpt += 1
6     return(cmpt)
```

2. Là aussi, il faut faire intervenir un compteur, mais sur une plage entière de caractères.

```
1 def bnOcc2(mot:str, phrase:str) -> int :
2     cmpt = 0 # compteur
3     n = len(mot)
4     for k in range(len(phrase)-n) :
5         if phrase[k:k+n] == mot :
6             cmpt = cmpt+1
7     return(cmpt)
```

Exercice 5 (Bonus) :

1. Plusieurs solutions sont possibles. Comme toujours. On propose :

```
1 def carre(n:int) -> None :
2     print("*"*n)
3     for k in range(2,n) :
4         print(' '*k+'*'*(n-2)+'*')
5     if n>1 :
6         print("*"*n)
```

2. C'est un problème de gestion du nombre d'espaces entre deux étoiles. On propose :

```
1 def croix(n:int) -> None :
2     for k in range(0,n//2) :
3         print(" "*k+"*"+ " "*(n-2*k-2)+"*"+ " "*k)
4     print(" "*(n//2)+"*"+ " "*(n//2))
5     for k in range(0,n//2) :
6         print(" "*(n//2-k-1)+"*"+ " "*(2*k+1)+"*"+ " "*(n//2-k-1))
```