DS 1
Informatique
Correction

Simon Dauguet
simon.dauguet@gmail.com

Mercredi 25 Septembre 2024

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'appuie 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 :

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

```
while conditions :
    instructions
```

3. La syntaxe complète de la structure conditionnelle est :

```
if condition1 :
    instructions1
elif condition2 :
    instruction2
elif condition3 :
    instruction3
...
else :
    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 dans la boucle,
   et on renverra directement p.
4          N=Conway(N)        # terme suivant
5      return(N)              #On renvoie le résultat.
```

Exercice 3 (Syntaxe) :
On donne directement la version corrigé :

```
print("Entrer un entier naturel")
n = int(input())
som = 0
for i in range(1,n) :
    som = som + i
print(f"La somme vaut {som}")
```

Exercice 4 (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

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

2. Puis la fonction

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

3. Et enfin :

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

Exercice 5 (Chaînes de caractères) :

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

2.

```python
def bn0cc2(mot:str, phrase:str) -> int :
    cmpt = 0 # compteur
    n = len(mot)
    for k in range(len(phrase)-n) :
        if phrase[k:k+n] == mot :
            cmpt = cmpt+1
    return(cmpt)
```

Exercice 6 (Bonux) :

1.

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

2.

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