Portail Descartes
Introduction à la science info : TD 2
Algorithmes en Python

amU Faculté des sciences Aix Marseille Université

Exercice 1 (Variables, affectations et séquence)

1. Exécuter la séquence d'instructions suivante, en remplissant une table qui montre le déroulement du calcul. Combien d'opérations exécute-t-on au total ?

```
a = 3 + 7
b = a * 2
c = a + b
a = a + 3
```

2. Est-ce que la séquence d'instructions suivante est légitime ? Pourquoi ?

```
a = 3 + 7
b = c * 2
c = a + b
a = a + 3
```

Exercice 2 (Définition et exécution de fonctions)

1. Écrivez une fonction `cube_plus_1(x)` qui renvoie le cube de x plus 1.
2. Montrez, à l'aide d'une table, l'exécution du calcul `cube_plus_1(3)`. Combien d'opérations exécute-t-on ?

Exercice 3 (Fonctions qui appellent d'autres fonctions)

1. Étant donné :

```
def carré(x):
    y = x * x
    return y

def discriminant(a, b, c):
    d = carré(b) - 4*a*c
    return d
```

   calculez `discriminant(2, 4, 2)` en montrant les étapes du calcul avec des tables. Combien d'opérations exécute-t-on au total ?

2. Écrivez *en utilisant la fonction* `carré` une autre fonction `somme_carrés(x, y)` qui renvoie $x^2 + y^2$; puis exécutez le calcul `somme_carrés(2, 3)` en montrant les étapes sous forme de tables et comptez le nombre d'opérations exécutées.

Exercice 4 (Conditions)

1. Écrivez une fonction `maximum(x, y)` qui prend en entrée deux nombres et en renvoie le maximum.
2. Exécutez les calculs `maximum(3, 2)` et `maximum(1, 4)` en montrant les étapes à l'aide de tables et comptez le nombre d'opérations exécutées.

Exercice 5 (Conditions sans alternative) Réécrivez la fonction `maximum(x, y)` sans utiliser `else`. Expliquez pourquoi cela fonctionne correctement dans les deux cas (quand x est le maximum et quand y est le maximum).

Exercice 6 Écrivez une fonction `num_solutions(a, b, c)` qui renvoie le nombre de solutions réelles de l'équation $ax^2 + bx + c = 0$ à l'aide des fonctions `discriminant(a, b, c)`, de deux façons différentes :
— d'abord en utilisant la structure `if ...elif ...else` ;
— puis en exploitant la fonction `sgn(x)`, sans utiliser de conditions explicites dans votre code.

Exercice 7 (Itération avec la boucle `while`) Considérez l'algorithme suivant, qui prend en entrée deux entiers naturels :

```
def mystère(x, y):
    z = y
    r = 0
    while z >= x:
        r = r + z
        z = z - 1
    return r
```

1. Exécutez le calcul de `mystère(2, 5)` en montrant toutes les étapes de calcul à l'aide d'une table. Combien d'opérations exécute-t-on ? Quel est le résultat du calcul ?
2. Qu'est-ce que cet algorithme renvoie, en général, sur une entrée x, y quelconque ?

Exercice 8 (Terminaison d'un algorithme)

1. Montrez que l'algorithme `mystère` de l'exercice précédent termine pour n'importe quelles valeurs (deux entiers naturels) de x et y.
2. Montrez que l'algorithme `mystère_bis` suivant termine pour n'importe quelles valeurs (deux entiers naturels) de x et y.

```
def mystère_bis(x, y):
    z = y
    r = 0
    while z >= x:
        r = r + z
        z = z - 2
        x = x - 1
    return r
```

Exercice 9 (Non-terminaison d'un algorithme) Montrez que la fonction `mystère_ter` suivante ne termine pas pour certaines valeurs (deux entiers naturels) de x et y.

```
def mystère_ter(x, y):
    z = y
    r = 0
    while z >= x:
        r = r + z
        z = z - 1
        x = x - 1
    return r
```

Exercice 10 Exécuter l'algorithme suivant sur l'entrée 4 en montrant les étapes de calcul avec une table. Quel est le résultat et combien d'opérations exécute-t-on ?

```
def énigme(n):
    j = 12
    for i in range(n):
        j = j - i
    return j
```