Professeur: Nastaran FATEMI
Assistant: Christopher MEIER

Série d'exercices A
Premiers Pas

Les exercices de ce tutoriel sont classés en 2 catégories :

— Niveau 1 : Facile
— Niveau 2 : Intermédiaire

Ces exercices ne sont pas notés !

Niveau 1

1. **Déclaration de variables**

   Déclarez les variables mutables `nom` et `prenom` de type `String`, avec la valeur vide comme valeur par défaut.

2. **Affectation de variables**

   Affectez votre nom et votre prénom aux deux variables déclarées dans l'exercice précédent.

3. **Immuables vs mutables**

   Déclarez la variable immuable `dateDeNaissance` de type `String`, avec la valeur vide comme valeur par défaut.

   — Affectez votre date de naissance à la variable `dateDeNaissance`.
   — Est-ce possible ? Pourquoi ?
   — Si ça ne fonctionne pas, faites en sorte que la variable `dateDeNaissance` possède comme valeur, votre date de naissance.

4. **Affichage**

   Affichez, à l'aide de la fonction `println(...)` et des variables `nom`, `prenom` et `dateDeNaissance` le texte :

   ```
   Bonjour, je m'appelle <<prenom>> <<nom>> et je suis né le <<dateDeNaissance>>.
   ```

5. **Boucle I**

   En utilisant une boucle `while` et la variable `compteur`, affichez 5 fois le message de l'exercice 4.

6. **Boucle II**

   En utilisant une boucle `for`, affichez 5 fois le message de l'exercice 4.

7. **Boucle III**

   En utilisant une boucle `for`, et une **autre condition d'arrêt** qu'à l'exercice précédent, affichez 5 fois le message de l'exercice 4.

8. **Condition booléenne**

   En utilisant les variables `compteur`, `nom` et `prénom`, utilisez une condition booléenne pour afficher votre `nom` si le `compteur` est égal à `0`, sinon votre `prénom`.

9. **Match cases**

   Réimplémentez l'exercice précédent en remplaçant les conditions booléennes par un `match cases` **Scala**.

Niveau 2

10. **Boucle I**

Implémentez une boucle `while` qui n'affiche que les nombres impairs et multiple de 3 ou de 5, compris entre 1 et 100.
Hint : 3, 5, 9, 15, 21, 25, 27, 33, 35, 39, 45, 51, 55, 57, 63, 65, 69, 75, 81, 85, 87, 93, 95, 99

Implémentez la même condition d'affichage, en utilisant une boucle `for` et une seule ligne de code.
Hint : vous pouvez écrire for (...) println(...) sur la même ligne.

11. **Fonction I**

Implémentez une fonction, prenant deux `Int` en paramètre, qui affiche le plus grand des deux paramètres, en une seule ligne de code.
Hint : `func(3, 5)` affiche 5.

12. **Fonction II**

Implémentez une fonction qui prend un `String` en paramètre, si ce paramètre vaut

- `"Hello"`, la fonction affiche `"World"`.
- `"World"`, la fonction affiche `"Hello"`.
- Sinon, la fonction affiche `"Goodbye"`.

Sans utiliser de conditions booléennes.
Hint : match cases.

13. **Boucle II**

Implémentez la boucle ci-dessous, en **une seule boucle `for` Scala**.

```
for (int i = 0; i < 3; ++i) {
    for (int j = 0; j <= 3; ++j) {
        for (int k = 0; k < 4; ++k) {
            System.out.println("Hello World !");
        }
    }
}
```

14. **Match cases I**

Implémentez un `match cases` sur un `String`, qui vérifie dans l'ordre :

- Si la longueur du `String` est impaire, affiche la longueur.
- Si le `String` est égal à `"Hello World!"`, l'affiche.
- Sinon, affiche `"Goodbye"`.

15. **Boucle III & Match cases II**

A l'aide d'une variable mutable `compteur` valant `0`, implémentez une boucle `while` ayant pour condition, tant que le `compteur` est plus petit que `10`. Sans utiliser de conditions booléennes, incrémentez le `compteur` comme suit :

— Si le `compteur` est pair, on l'incrémente de `3`.
— Si le `compteur` est impair, on l'incrémente de `1`.

Combien de fois la boucle est-elle exécutée ? Quelle est la valeur du `compteur` après ses exécutions ?

16. **Fonctions III & Match cases III**

Implémentez une fonction qui prend un `Int` en paramètre.

— Si `x` est plus grand que `100`, retourne la valeur de `x`.
— Si `x` est un multiple de `7`, appelle la fonction avec `x + 8`.
— Si `x` est impair, appelle la fonction avec `x + 12`.
— Sinon, appelle la fonction avec `x + 1`.

Appelez cette fonction avec la variable immuable `x` ayant comme valeur `0`.

— Quelle est la valeur finale retournée par la fonction ?
— Est-ce la nouvelle valeur de `x` ?
— Combien de fois la fonction est-elle exécutée ?
— Pouvons-nous appeler la fonction à l'intérieur de celle-ci sans explicitement retourner une valeur de type `Int` ?

17. **Fonction IV**

Implémentez une fonction qui prends trois `Int` (`x`, `y`, `z`) en paramètre.

— Si `z` est pair, retournez la somme des carrés de `x` et `y`.
— Si `z` est impair, retourner le carré de la somme de `x` et de `y`.
— Si `z` vaut 0, retournez la somme des cas `z` pair et `z` impair.

Appelez la fonction avec `x = 2`, `y = 3`, et `z = 1`, puis `2`, puis `0`.

— Est-ce que le cas `z = 0` est correct ? Pourquoi ?
— S'il n'est pas correct, modifiez le code pour obtenir le bon résultat.

18. **Fonction V**

Implémentez deux versions d'une fonction qui prends deux `Int` (`x` et `y`) en paramètre, une fois avec des conditions booléennes et une fois avec des `match cases`.

- Si `x == y`, retourne `x`.
- Si `x < y`, retourne `y`.
- Si `x > y`, alors :
  - Si `x` est impair, retourne `2x + 3y`.
  - Si `y` est impair et `x` est pair, retourne `4x - 7y`.
  - Si `x` est un multiple de `3` et `y` un multiple de 4, retourne $x^2 + y^3$
- Si `x < y` et `x > 4`, retourne `2x`.
- Si `x == y` et `y == 0`, retourne `42`.
- Dans tous les autres cas, retourne `9`

Faites en sorte que **toutes** les conditions soient vérifiables.
Hint : il faut faire attention à l'ordre.

19. **Fonction VI**

Implémentez une fonction qui retourne la somme des `n` premiers nombres de Fibonacci.

Appelez cette fonction avec `n = 20`.