Labo d'introduction à l'informatique

pour les mathématiques

Yann Thorimbert

Semaine 5
Tableaux de valeurs et boucles for

Un problème

- Comment regrouper des valeurs d'une même nature ?
- Exemple : liste des âges au sein d'une population.
- Exemple : liste des coefficients d'un polynôme.

- Mauvaise solution : déclarer des variables différentes à chaque fois !

```
age_1 = 12
age_2 = 43
…
age_13953 = 19
```

Un problème et une solution

- Mauvaise solution : déclarer des variables différentes à chaque fois !

```
age_1 = 12
age_2 = 43
…
age_15 = 19
```

- Bonne solution : **regrouper les valeurs au sein d'une même variable**.

```
ages = [12, 43, …, 19]
```

Cela va nous permettre d'effectuer un **traitement systématique** des valeurs.

Qu’est-ce qu’un tableau de valeurs ?

- Stocke une **séquence ordonnée** de valeurs au sein d’**une variable**.
- Tableau : “array” en anglais.
- L’emplacement des éléments s’appelle l’**indice**.
- Exemple :

Indice	1	2	3	4	5	6
Valeur	4	-8	16	2	2	1

Ci-dessus, l’indice correspondant à la valeur 4 est 1.
Les indices correspondant à la valeur 2 sont 4 et 5.

Déclarer un tableau en Matlab

- Le tableau suivant …

Indice	1	2	3	4	5	6
Valeur	4	-8	16	2	2	1

- … se déclare de façon **littérale** comme suit :

```
a = [4, -8, 16, 2, 2, 1];
```

Déclarer un tableau en Matlab

- Se déclare de façon **littérale** comme suit :

```
a = [4, -8, 16, 2, 2, 1];
```

- Comme n'importe quelle variable, on peut lui attribuer le nom que l'on veut.

- On peut **générer** un tableau rempli de 0 ou de 1.

```
b = zeros(1, 15); % une ligne remplie de quinze 0.
a = ones(1, 4); % une ligne remplie de quatre 1.
```

1 car on veut une seule ligne (on verra les matrices + tard dans le semestre)

Accéder aux éléments d’un tableau

- On accède à un élément avec des parenthèses. Exemple :

```
a = [6, 7, 8, 9];
x = a(2); % On accède à l’élément d’indice 2
disp(x); % Affiche la valeur 7
```

Accéder aux éléments d’un tableau

- On accède à un élément avec des parenthèses. Exemple :

```
a = [6, 7, 8, 9];
a(2) = 3; % On assigne la valeur trois à l’élément n°2
disp(a); % Affiche [6, 3, 8, 9]
```

- La fonction `length()` retourne le nombre d’éléments d’un tableau. Exemple:

```
a = [6, 7, 8, 9];
a(length(a)) = 3; % On assigne une valeur au dernier el.
disp(a); % Affiche  [6, 7, 8, 3]
```

Ajouter un élément

- La fonction `length()` retourne le nombre d'éléments d'un tableau.

```
a = [6, 7, 8, 9];
a(length(a)+1) = 3; % On ajoute 3 à la fin
disp(a); % Affiche  [6, 7, 8, 9, 3]
```

- Autre façon de faire :

```
a = [6, 7, 8, 9];
a(end + 1) = 3; % On ajoute 3 à la fin
disp(a); % Affiche  [6, 7, 8, 9, 3]
```

Analogie avec les vecteurs

- Matlab permet un traitement dit “vectoriel”. Exemple :

```
v = [2, 1, 1]; % Représente la vitesse 3D d’un objet
dt = 2; % Représente une durée
x = v * dt; % Position 3D de cet objet après un temps dt (MRU)
disp(x) % x est aussi un tableau, et il vaut [4, 2, 2]
```

- `x(1)` représente la première coordonnée, `x(2)` la seconde, etc.

- Pour l’instant, nous ignorons ce type de traitement.

Différences entre Matlab et Python

Si vous avez l'habitude de langages comme Python, il est tentant de faire le parallèle entre les tableaux de valeurs et les listes de Python. Voici certaines différences :

Liste en Python	**Tableau en Matlab**
Le premier élément porte l'indice 0	Le premier élément porte l'indice 1
Accession aux éléments avec les crochets [ ]	Accession aux éléments avec les parenthèses ( )
Peut contenir des objets de types différents	Contient uniquement des valeurs numériques

Un exemple important : recherche dans un tableau

On veut écrire un code qui affiche un message si une valeur plus grande que 9 figure dans un tableau x.

On peut effectuer la recherche “manuellement” :

```
if x(1) > 9:
    disp(“Il y a un élément plus grand que 9”)
elseif x(2) > 9:
    disp(“Il y a un élément plus grand que 9”)
.
.
.
end
```

Un exemple important : recherche dans un tableau

- La recherche “manuelle” est la plupart du temps la mauvaise solution, car s’il y a plus qu’un ou deux éléments dans le tableau :
  - Elle est longue et peu pratique à écrire, et favorise les erreurs humaines dans le code.
  - On ne connaît pas forcément la longueur du tableau, par exemple si c’est l’argument d’une fonction !
- Les **boucles** sont une bonne façon d’**itérer** sur les valeurs d’un tableau.
- Les boucles constituent pour nous une nouvelle **structure de contrôle**.
- Itérer : parcourir les éléments les uns après les autres.
- Il existe deux types de boucles : `for` et `while`.

Sujet d’aujourd’hui !

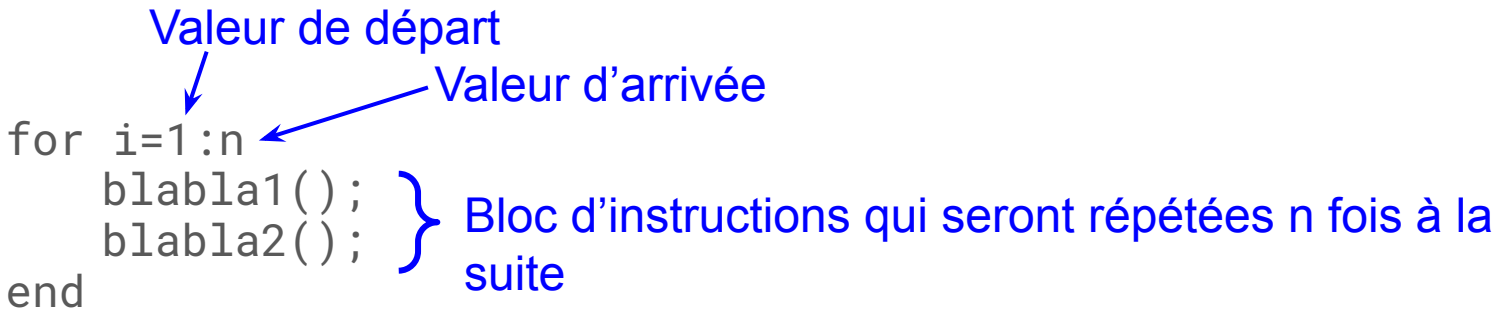
Un exemple important : recherche dans un tableau

Solution du problème du nombre plus grand que 9 :

```
x = [4, 12, -8, … , 4]; % Tableau de longueur quelconque !
n = length(x); % n contient le nombre d'éléments du tableau
for i=1:n % On ordonne de répéter les instructions n fois
    if x(i) > 9
        disp("Il y a un nombre plus grand que 9 !")
    end
end
```

Fonctionnement d'une boucle | **Variable de boucle**

- Une boucle for se sert d'une **variable de boucle** (souvent nommée i, mais le nom dépend du choix du programmeur !)
- La variable de boucle va prendre successivement toutes les valeurs, de sa valeur de départ à sa valeur d'arrivée.



Fonctionnement d'une boucle | **Variable de boucle**

- Une boucle for se sert d'une **variable de boucle** (souvent nommée i, mais le nom dépend du choix du programmeur !)
- La variable de boucle va prendre successivement toutes les valeurs, de sa valeur de départ à sa valeur d'arrivée.

```
for i=1:n
    disp(i); % Affiche successivement 1, 2, ... n
end
```

Fonctionnement d'une boucle *for* | **Bloc répété**

- Une boucle for permet d'**automatiser la répétition d'instructions** un certain nombre de fois, tout en **gardant trace** de la variable de boucle.
- Par exemple, nous allons automatiser ce code :

```
disp("Salut");
disp(2);
disp("Salut");
disp(4);
disp("Salut");
disp(6);
disp("Salut");
disp(8);
```

On voit que les instructions:
`disp("Salut")` et `disp(x)` sont répétées, où x "saute" de 2 en 2.

Fonctionnement d'une boucle *for* | **Bloc répété**

- Une boucle for permet d'**automatiser la répétition d'instructions** un certain nombre de fois, tout en gardant trace de la variable de boucle.

- Solution :

```
for i=1:4
    disp("Salut");
    disp(2 * i);
end
```

Fonctionnement d'une boucle *for* | **Incrément**

- On peut également indiquer le "pas" de la variable de boucle explicitement :

```
for i=2:2:8 % Signifie de 2 à 8, avec un pas de 2
    disp("Salut");
    disp(i); % Dans ce cas on affiche i au lieu de 2*i
end
```

- Ce pas est nommé l'**incrément**. Par défaut, il vaut 1.

Retour au problème de la recherche dans un tableau

Comment afficher **où se trouve** le nombre plus grand que 9 au sein du tableau ?

```
x = [4, 12, -8, … , 4]; % tableau de longueur quelconque !
n = length(x); % n contient le nombre d'éléments du tableau
for i=1:n
    if x(i) > 9
        fprintf("Il y a un grand nombre ici."); % ???
    end
end
```

Retour au problème de la recherche dans un tableau

Grâce à la variable de boucle, on sait à quelle itération s'affiche le message.

```
x = [4, 12, -8, … , 4]; % tableau de longueur quelconque !
n = length(x); % n contient le nombre d'éléments du tableau
for i=1:n
    if x(i) > 9
        fprintf("Il y a un grand nombre ici : %d", i);
    end
end
```

Fonctionnement d'une boucle *for* | **L'arrêt de la boucle**

Et si l'on veut arrêter la recherche au milieu de la boucle ?

```
x = [4, 12, -8, … , 4]; % tableau de longueur quelconque !
n = length(x); % n contient le nombre d'éléments du tableau
for i=1:n
    if x(i) > 9
        fprintf("Il y a un grand nombre ici : %d", i);
        break; % Avec cette instruction, le flux est "coupé".
    end
end
```

Terminologie “boucle for”

- Se nomme ainsi parce qu’on itère **pour une variable de boucle** donnée, d’une valeur de départ à une valeur finale.

- Pensez à vous représenter “for i=a:b” comme “**répéter** de i=a jusqu’à i=b”.

- On verra bientôt la boucle `while`, qui répète un bloc d’instruction tant que certaines conditions sont remplies, plutôt qu’en fonction d’une variable de boucle.

Un exemple important : fonction retournant un tableau

On veut créer une fonction `parite(x)` qui prend en entrée un tableau `x` et retourne un tableau `y` de même taille, où l'élément `y(i)` = 1 si `x(i)` est pair, et `y(i)` = 0 sinon.

Exemple : si la fonction reçoit le tableau `[1,2,4,3]` elle retourne `[0,1,1,0]`.

Un exemple important : fonction retournant un tableau

On veut créer une fonction `parite(x)` qui prend en entrée un tableau `x` et **retourne un tableau** `y` de même taille, où l'élément `y(i)` = 1 si `x(i)` est pair, et `y(i)` = 0 sinon.

Solution :

```
function y = parite(x)
    n = length(x);
    y = zeros(1,n); % y est un tableau de n éléments.
    for i=1:n
        if mod(x(i),2) == 0
            y(i) = 1
        end
    end
end
```