



## Langage et algorithmique

---

**A. Malek TOUMI**

**toumiab@ensta-bretagne.fr**

ENSTA Bretagne



# Sommaire

## 1 Rappel

Notion de variables

## 2 Typage des variables

Les nombres

les types numériques

Les booléens

Portée

Opérations

## 3 Instructions du langage Java

Commentaires

Littéraux (constantes)

Mots clés

Branchements conditionnels

Boucles

## 4 Premiers programmes



# Sommaire

## 1 Rappel

Notion de variables

## 2 Typage des variables

Les nombres

les types numériques

Les booléens

Portée

Opérations

## 3 Instructions du langage Java

Commentaires

Littéraux (constantes)

Mots clés

Branchements conditionnels

Boucles

## 4 Premiers programmes



## Typage des variables

- Python : langage typé dynamiquement



## Typage des variables

- Python : langage typé dynamiquement
- ⇒ typage lors de l'affectation



## Typage des variables

- Python : langage typé dynamiquement
- ⇒ typage lors de l'affectation
- Une variable n'est qu'une référence à l'objet (entier, réel, ...)



## Typage des variables

- Python : langage typé dynamiquement
- ⇒ typage lors de l'affectation
- Une variable n'est qu'une référence à l'objet (entier, réel, ...)
  - Une variable peut changer de type au cours de l'exécution



## Typage des variables

- **Les types intégrés de Python**

- Les numériques : Entiers signés (**int**), Réels (IEEE 754) (**float** et Complexes (**complex**)
- Booléens : le type **bool**
- Chaînes de caractères : type **str**
- Listes : type **list**
- Dictionnaires : type **dict**
- Tuples : type **tuple**
- Ensembles modifiables et immuables : le type **set** et le type **frozenset**
- ...





# Typage des variables

## • Les types intégrés de Python

- Les numériques : Entiers signés (**int**), Réels (IEEE 754) (**float** et Complexes (**complex**)
- Booléens : le type **bool**
- Chaînes de caractères : type **str**
- Listes : type **list**
- Dictionnaires : type **dict**
- Tuples : type **tuple**
- Ensembles modifiables et immuables : le type **set** et le type **frozenset**
- ...

### Remarque

- Pour connaître le type d'une variable ou expression, on peut utiliser la fonction **type()**
- La fonction **print()** permet d'afficher la valeur d'une variable



## Typage des variables

### Exemple

```
>>> i = 42
>>> type(i)
<class 'int'>
>>> i = 'indices'
>>> type(i)
<class 'str'>
>>> i = 42.0
>>> type(i)
<class 'float'>
>>> print(i)
42.5
>>>
```



# Sommaire

## 1 Rappel

Notion de variables

## 2 Typage des variables

Les nombres

les types numériques

Les booléens

Portée

Opérations

## 3 Instructions du langage Java

Commentaires

Littéraux (constantes)

Mots clés

Branchements conditionnels

Boucles

## 4 Premiers programmes



# Sommaire

## 1 Rappel

Notion de variables

## 2 Typage des variables

Les nombres

les types numériques

Les booléens

Portée

Opérations

## 3 Instructions du langage Java

Commentaires

Littéraux (constantes)

Mots clés

Branchements conditionnels

Boucles

## 4 Premiers programmes



# Sommaire

## 1 Rappel

Notion de variables

## 2 Typage des variables

Les nombres

les types numériques

Les booléens

Portée

Opérations

## 3 Instructions du langage Java

Commentaires

Littéraux (constantes)

Mots clés

Branchements conditionnels

Boucles

## 4 Premiers programmes



# Les types numériques

- **Les entiers**
  - Leurs représentation n'est limitée que par la taille mémoire.



# Les types numériques

- **Les entiers**

- Leurs représentation n'est limitée que par la taille mémoire.  
⇒ pas de risque de débordement



# Les types numériques

- **Les entiers**

- Leurs représentation n'est limitée que par la taille mémoire.  
⇒ pas de risque de débordement

## Exemple

```
>>> i = 42
>>> i
42
>>> math.factorial(i)
1405006117752879898543142606244511569936384000000000
```





# Les types numériques

- **Les entiers**
- **Les réels**
  - Sont représentés en en double précision (64 bits).  
⇒ le plus petit nombre qu'il est possible de distinguer de 1 est  $2.22E - 16$



# Les types numériques

- **Les entiers**
- **Les réels**
  - Sont représentés en en double précision (64 bits).  
⇒ le plus petit nombre qu'il est possible de distinguer de 1 est  $2.22E - 16$
  - Pour utiliser la simple précision, on peut faire appel à la bibliothèque `numpy`.



## Les types numériques

- Les entiers
- Les réels
  - Sont représentés en en double précision (64 bits).  
⇒ le plus petit nombre qu'il est possible de distinguer de 1 est  $2.22E - 16$
  - Pour utiliser la simple précision, on peut faire appel à la bibliothèque `numpy`.

### Exemple

```
>>> freq = 10e9
>>> freq
10000000000.0
```



# Les types numériques

- **Les entiers**
- **Les réels**
- **Les complexes**
  - Sont formés d'un couple de réels qui composent la partie réelle et la partie imaginaire



# Les types numériques

- **Les entiers**
- **Les réels**
- **Les complexes**
  - Sont formés d'un couple de réels qui composent la partie réelle et la partie imaginaire
  - Un suffixe **j** pour regrouper deux valeurs composant la partie réelle et la partie imaginaire



# Les types numériques

- Les entiers
- Les réels
- Les complexes
  - Sont formés d'un couple de réels qui composent la partie réelle et la partie imaginaire
  - Un suffixe `j` pour regrouper deux valeurs composant la partie réelle et la partie imaginaire
  - Un nombre complexe possède deux attributs en lecture seule `.real` et `.imag` et une méthode `.conjugate()`



## Les types numériques

- Les entiers
- Les réels
- Les complexes

### Exemple

```
>>> nb = 10 + 5j
>>> nb.real
10.0
>>> nb.conjugate()
(10-5j)
>>> print(complex(10, 5))
(10+5j)
```



# Sommaire

- 1 Rappel
  - Notion de variables
- 2 **Typage des variables**
  - Les nombres
  - les types numériques
  - Les booléens**
  - Portée
  - Opérations
- 3 **Instructions du langage Java**
  - Commentaires
  - Littéraux (constantes)
  - Mots clés
  - Branchements conditionnels
  - Boucles
- 4 Premiers programmes





# Les booléens



# Sommaire

## 1 Rappel

Notion de variables

## 2 Typage des variables

Les nombres

les types numériques

Les booléens

Portée

Opérations

## 3 Instructions du langage Java

Commentaires

Littéraux (constantes)

Mots clés

Branchements conditionnels

Boucles

## 4 Premiers programmes



## Portée, bloc

### Portée

---

- Portée : zone de visibilité d'une variable
- Notion fondamentale
- Utilise la notion de bloc



## Portée, bloc

### Portée

- Portée : zone de visibilité d'une variable
- Notion fondamentale
- Utilise la notion de bloc

### Définition (Bloc)

Ensemble des instructions entre une paire balancée d'accolades (accolade ouvrante et accolade fermante correspondante).



## Exemple de blocs

### Exemple (Bloc)

```
public class MaClasse
{
    // debut bloc 1
    private int variable1;
    public void essai()
    {
        // debut bloc 2
        int i, j;
    }
    // fin bloc 2
    public static void main(String args[])
    {
        // debut bloc 3
        int i;
    }
    // fin bloc 3
}
// fin bloc 1
```



## Effet de bord

### Attention

Limiter la portée des variables au maximum : attention aux effets de bord



## Effet de bord

### Attention

Limiter la portée des variables au maximum : attention aux effets de bord

### Définition (Effet de bord)

Effet de bord vient de **side effect** (effet secondaire). Modification d'une variable globale par une fonction.



## Effet de bord

### Attention

Limiter la portée des variables au maximum : attention aux effets de bord

### Définition (Effet de bord)

Effet de bord vient de **side effect** (effet secondaire). Modification d'une variable globale par une fonction.

Le comportement d'un programme change selon l'ordre d'appel des fonctions à effet de bord.





## Transtypage / cast

- Objectif : convertir d'un type vers un autre.
- Exemple : `double f = (double) retourInt ()`;
- Attention : risque de perte de données (ex : **long** vers **int**)



## Transtypage / cast

- Objectif : convertir d'un type vers un autre.
- Exemple : `double f = (double) retourInt ()`;
- Attention : risque de perte de données (ex : **long** vers **int**)
- Transtypage implicite



## Transtypage / cast

Type initial	Transtypage
byte	short, char, int, long, float, double
short	int, long, float, double
char	int, long, float, double
int	long, float, double
long	float, double
float	double



## Transtypage / cast

- Objectif : convertir d'un type vers un autre.
- Exemple : `double f = (double) retourInt ();`
- Attention : risque de perte de données (ex : **long** vers **int**)
- Transtypage implicite
- Ex : `if ('a' < 3) ...`



## Transtypage / cast

- Objectif : convertir d'un type vers un autre.
- Exemple : `double f = (double) retourInt ()`;
- Attention : risque de perte de données (ex : **long** vers **int**)
- Transtypage implicite
- Ex : `if ('a' < 3) ...`
- 'a' est converti en entier



# Sommaire

## 1 Rappel

Notion de variables

## 2 Typage des variables

Les nombres

les types numériques

Les booléens

Portée

Opérations

## 3 Instructions du langage Java

Commentaires

Littéraux (constantes)

Mots clés

Branchements conditionnels

Boucles

## 4 Premiers programmes



# Affectation

## Définition (Affectation)

Affectation : donner une valeur à une variable. Réalisée par l'opérateur “=”.



# Affectation

## Définition (Affectation)

Affectation : donner une valeur à une variable. Réalisée par l'opérateur “=”.

## Exemple (Affectations)

```
int a = 2;  
a = a+1;  
boolean b=(a>2);  
char c = 'a'+3;
```





## Pré et post incrémentation

### Exemple (Forme condensée)

```
a += 2;
```

```
b *= 3;
```



## Pré et post incrémentation

### Exemple (Forme condensée)

```
a += 2;
```

```
b *= 3;
```

- Affectation très utile :  $i += 1$  (ou  $i = i+1$ )



## Pré et post incrémentation

### Exemple (Forme condensée)

```
a += 2;
```

```
b *= 3;
```

- Affectation très utile :  $i += 1$  (ou  $i = i+1$ )
- $i++$ ;



## Pré et post incrémentation

### Exemple (Forme condensée)

```
a += 2;
```

```
b *= 3;
```

- Affectation très utile :  $i += 1$  (ou  $i = i+1$ )
- $i++$ ;  $i--$ ;



## Pré et post incrémentation

### Exemple (Forme condensée)

```
a += 2;
```

```
b *= 3;
```

- Affectation très utile :  $i += 1$  (ou  $i = i+1$ )
- $i++$ ;  $i--$ ;
- $++i$ ;



## Pré et post incrémentation

### Exemple (Forme condensée)

```
a += 2;
```

```
b *= 3;
```

- Affectation très utile :  $i += 1$  (ou  $i = i+1$ )
- $i++$ ;  $i--$ ;
- $++i$ ;  $--i$ ;



## Pré et post incrémentation

### Exemple (Forme condensée)

```
a += 2;
```

```
b *= 3;
```

- Affectation très utile :  $i += 1$  (ou  $i = i+1$ )
- $i++$ ;  $i--$ ;
- $++i$ ;  $--i$ ;

### Exemple

```
int j, i=0;
```

```
j = i++;
```



## Pré et post incrémentation

### Exemple (Forme condensée)

```
a += 2;
```

```
b *= 3;
```

- Affectation très utile :  $i += 1$  (ou  $i = i+1$ )
- $i++$ ;  $i--$ ;
- $++i$ ;  $--i$ ;

### Exemple

```
int j, i=0;
```

```
j = i++;      (j vaut 0, i vaut 1)
```





## Pré et post incrémentation

### Exemple (Forme condensée)

```
a += 2;
```

```
b *= 3;
```

- Affectation très utile :  $i += 1$  (ou  $i = i+1$ )
- $i++$ ;  $i--$ ;
- $++i$ ;  $--i$ ;

### Exemple

```
int j, i=0;
```

```
j = i++;      (j vaut 0, i vaut 1)
```

```
j = ++i;
```



## Pré et post incrémentation

### Exemple (Forme condensée)

```
a += 2;  
b *= 3;
```

- Affectation très utile :  $i += 1$  (ou  $i = i+1$ )
- $i++$ ;  $i--$ ;
- $++i$ ;  $--i$ ;

### Exemple

```
int j, i=0;  
j = i++;      (j vaut 0, i vaut 1)  
j = ++i;      (j vaut 2, i vaut 2)
```



## Comparaisons

Opérateur	Exemple	Signification
>	a > 10	strictement supérieur
<	a < 10	strictement inférieur
>=	a >= 10	supérieur ou égal
<=	a <= 10	inférieur ou égal
==	a == 10	égal à
!=	a != 10	différent de



## Opérations logiques

Opérateur	Exemple	Signification
!	!a	NON logique
&&	a && b	ET logique : évaluation paresseuse
	a    b	OU logique : évaluation paresseuse
&	a & b	ET logique
^	a ^ b	OU exclusif logique
	a   b	OU logique



## Opérations logiques

Opérateur	Exemple	Signification
!	!a	NON logique
&&	a && b	ET logique : évaluation paresseuse
	a    b	OU logique : évaluation paresseuse
&	a & b	ET logique
^	a ^ b	OU exclusif logique
	a   b	OU logique

Remarque : évaluation paresseuse



## Opérations arithmétiques

- Opérations de base : + - \* /
- Modulo : %
- Attention à la division entière :
  - $1/2$  vaut 0
  - $1/2.0$  vaut 0.5



## Opérations arithmétiques

- Opérations de base : + - \* /
- Modulo : %
- Attention à la division entière :
  - $1/2$  vaut 0
  - $1/2.0$  vaut 0.5

Réels particuliers :

- Double.NaN
- Double.POSITIVE\_INFINITY



## Opérations arithmétiques

- Opérations de base : + - \* /
- Modulo : %
- Attention à la division entière :
  - $1/2$  vaut 0
  - $1/2.0$  vaut 0.5

Réels particuliers :

- Double.NaN
- Double.POSITIVE\_INFINITY
- Comportement défini par IEEE 754





## Opérations arithmétiques

X	Y	X/Y	X%Y
valeur finie	0	$+\infty$	NaN
valeur finie	$\pm\infty$	0	x
0	0	NaN	NaN
$\pm\infty$	valeur finie	$\pm\infty$	NaN
$\pm\infty$	$\pm\infty$	NaN	NaN



## Opérateurs sur les bits

Opérateur	Exemple	Signification
<<	<code>a &lt;&lt; n</code>	décalage à gauche de $n$ bits
>>	<code>a &gt;&gt; n</code>	décalage à droite de $n$ bits
>>>	<code>a &gt;&gt;&gt; n</code>	décalage à droite non signé de $n$ bits
~	<code>~a</code>	NON binaire
&	<code>a &amp; b</code>	ET binaire
^	<code>a ^ b</code>	OU exclusif binaire
	<code>a   b</code>	OU binaire



## Priorité des opérateurs

les parenthèses	( )
les opérateurs d'incrémentation	++, --
les opérateurs de multiplication, division, et modulo	*, /, %
les opérateurs d'addition et soustraction	+, -
les opérateurs de décalage	<<, >>
les opérateurs de comparaison	<, >, <=, >=
les opérateurs d'égalité	==, !=
l'opérateur OU exclusif	^
l'opérateur ET	&
l'opérateur OU	
l'opérateur ET logique	&&
l'opérateur OU logique	
les opérateurs d'affectation	=, +=, -=



# Sommaire

- 1 **Rappel**
  - Notion de variables
- 2 **Typage des variables**
  - Les nombres
  - les types numériques
  - Les booléens
  - Portée
  - Opérations
- 3 **Instructions du langage Java**
  - Commentaires
  - Littéraux (constantes)
  - Mots clés
  - Branchements conditionnels
  - Boucles
- 4 **Premiers programmes**



## Règles de base

- Java est sensible à la casse



## Règles de base

- Java est sensible à la casse
- les blocs de code sont encadrés par des accolades. Chaque instruction se termine par un caractère ';'



## Règles de base

- Java est sensible à la casse
- les blocs de code sont encadrés par des accolades. Chaque instruction se termine par un caractère ';'
- une instruction peut tenir sur plusieurs lignes



## Règles de base

- Java est sensible à la casse
- les blocs de code sont encadrés par des accolades. Chaque instruction se termine par un caractère ';'
- une instruction peut tenir sur plusieurs lignes
- l'indentation est ignorée du compilateur mais elle permet une meilleure compréhension du code par le programmeur





# Sommaire

- 1 **Rappel**
  - Notion de variables
- 2 **Typage des variables**
  - Les nombres
  - les types numériques
  - Les booléens
  - Portée
  - Opérations
- 3 **Instructions du langage Java**
  - Commentaires
  - Littéraux (constantes)
  - Mots clés
  - Branchements conditionnels
  - Boucles
- 4 **Premiers programmes**



## Commentaires

- Commentaires : remarques en langage naturel



## Commentaires

- Commentaires : remarques en langage naturel
- Ignorés par le compilateur



## Commentaires

- Commentaires : remarques en langage naturel
- Ignorés par le compilateur
- Trois types de commentaires :



## Commentaires

- Commentaires : remarques en langage naturel
- Ignorés par le compilateur
- Trois types de commentaires :

### Exemple (Commentaire sur une ligne)

```
// zone de declaration des variables  
double x; // abscisse
```



## Commentaires

- Commentaires : remarques en langage naturel
- Ignorés par le compilateur
- Trois types de commentaires :

### Exemple (Commentaire sur plusieurs ligne)

```
/* debut de commentaire  
 * suite du commentaire  
 * et fin du commentaire */
```



## Commentaires

- Commentaires : remarques en langage naturel
- Ignorés par le compilateur
- Trois types de commentaires :

### Exemple (Documentation automatique)

```
/**  
 * commentaire de la methode  
 * @param val la valeur a traiter  
 * @since 1.0  
 * @return Rien  
 * @deprecated Utiliser la nouvelle methode XXX  
 */
```



# Sommaire

- 1 **Rappel**
  - Notion de variables
- 2 **Typage des variables**
  - Les nombres
  - les types numériques
  - Les booléens
  - Portée
  - Opérations
- 3 **Instructions du langage Java**
  - Commentaires
  - Littéraux (constantes)
  - Mots clés
  - Branchements conditionnels
  - Boucles
- 4 **Premiers programmes**





## Types de littéraux

- Entiers : base 10

### Exemple (Littéral)

42, -7



## Types de littéraux

- Entiers : base 10 ,base 16

### Exemple (Littéral)

0x9AE3



## Types de littéraux

- Entiers : base 10 ,base 16 ,base 8

### Exemple (Littéral)

020 (= 16)



## Types de littéraux

- Entiers : base 10 ,base 16 ,base 8
- Réels : double précision

### Exemple (Littéral)

3.14, 6.02E23 (=  $6.02 \cdot 10^{23}$ )



## Types de littéraux

- Entiers : base 10 ,base 16 ,base 8
- Réels : double précision simple précision

### Exemple (Littéral)

1.5f



## Types de littéraux

- Entiers : base 10 ,base 16 ,base 8
- Réels : double précision simple précision
- Booléens

### Exemple (Littéral)

true, false



## Types de littéraux

- Entiers : base 10 ,base 16 ,base 8
- Réels : double précision simple précision
- Booléens
- Caractères

### Exemple (Littéral)

```
'a', ';' ;
```



## Types de littéraux

- Entiers : base 10 ,base 16 ,base 8
- Réels : double précision simple précision
- Booléens
- Caractères unicode

### Exemple (Littéral)

`'\u0065' (= 'e')`





## Types de littéraux

- Entiers : base 10 ,base 16 ,base 8
- Réels : double précision simple précision
- Booléens
- Caractères unicode
- Chaînes de caractères

### Exemple (Littéral)

" Une chaîne de caractères"



# Sommaire

- 1 **Rappel**
  - Notion de variables
- 2 **Typage des variables**
  - Les nombres
  - les types numériques
  - Les booléens
  - Portée
  - Opérations
- 3 **Instructions du langage Java**
  - Commentaires
  - Littéraux (constantes)
  - Mots clés**
  - Branchements conditionnels
  - Boucles
- 4 **Premiers programmes**

abstract	assert	boolean	break	byte
case	catch	char	class	const
continue	default	do	double	else
extends	false	final	finally	float
for	goto	if	implements	import
instanceof	int	interface	long	native
new	null	package	private	protected
public	return	short	static	super
switch	synchronized	this	throw	throws
transient	true	try	void	volatile
while				

abstract	assert	boolean	break	byte
case	catch	char	class	const
continue	default	do	double	else
extends	false	final	finally	float
for	goto	if	implements	import
instanceof	int	interface	long	native
new	null	package	private	protected
public	return	short	static	super
switch	synchronized	this	throw	throws
transient	true	try	void	volatile
while				



# Sommaire

## 1 Rappel

Notion de variables

## 2 Typage des variables

Les nombres

les types numériques

Les booléens

Portée

Opérations

## 3 Instructions du langage Java

Commentaires

Littéraux (constantes)

Mots clés

Branchements conditionnels

Boucles

## 4 Premiers programmes



### if

- Teste une condition booléenne
- Si **vrai**, exécute une partie du code
- Sinon, exécute autre partie



### if

- Teste une condition booléenne
- Si **vrai**, exécute une partie du code
- Sinon, exécute autre partie

#### Exemple (if)

```
if (expressionBooleenne1) {  
    action1  
} else if (expressionBooleenne2) {  
    action2  
} else {  
    action3  
}
```



## Exemple de test

### Exemple

```
boolean b = fonctionTest();  
int i = calculComplice();  
if ( (i>42) && b) {  
    appelFonction();  
}
```



## Remarque

- **if** peut s'écrire sans **else**

## Remarque

- **if** peut s'écrire sans **else**

## Correct mais à proscrire

```
if (condition) {  
} else {  
    action;  
}
```

## Remarque

- **if** peut s'écrire sans **else**

## Correct mais à proscrire

```
if (condition) {  
} else {  
    action;  
}
```

## Forme préférée

```
if (! condition) {  
    action;  
}
```



## switch/case

### Exemple (switch)

```
int i = genererUnEntier();
switch(i) {
case 0:
    liste d'instructions...
    break;
case 1:
    autres instructions...
    break;
default:
    instructions differentes...
}
```

## Remarque

- Variable du **switch** de type **int** ou convertie implicitement en **int** : **byte**, **short**, **char**
- Instruction **break** : ne pas traiter les autres cas
- Possibilité de regrouper les instructions pour plusieurs cas

## Remarque

- Variable du **switch** de type **int** ou convertie implicitement en **int** : **byte**, **short**, **char**
- Instruction **break** : ne pas traiter les autres cas
- Possibilité de regrouper les instructions pour plusieurs cas

## Exemple

```
switch(i) {  
  case 0: case 2: case 4:  
    ... break;  
  case 1: case 3:  
    ... break;  
}
```



## Opérateur ?

Opérateur ternaire :

`( condition ) ? valeur-vrai : valeur-faux`



## Opérateur ?

Opérateur ternaire :

`( condition ) ? valeur-vrai : valeur-faux`

### Exemple

```
// i et j sont des entiers  
int max;  
max = (i>j) ? i : j;
```





# Sommaire

- 1 **Rappel**
  - Notion de variables
- 2 **Typage des variables**
  - Les nombres
  - les types numériques
  - Les booléens
  - Portée
  - Opérations
- 3 **Instructions du langage Java**
  - Commentaires
  - Littéraux (constantes)
  - Mots clés
  - Branchements conditionnels
  - Boucles
- 4 **Premiers programmes**



## Principe

- Boucle = structure de contrôle
- But : exécuter certaines opérations plusieurs fois
- Il existe plusieurs types de boucles en Java
- Toutes les boucles sont équivalentes



## while

```
while(condition) {  
    actions  
}
```



## while

```
while(condition) {  
    actions  
}
```

### Exemple (boucle while)

```
int i=0;  
while (i<10) // pas de ";"  
{  
    System.out.println(i);  
    i++; // i=i+1  
}
```



## do-while

### Exemple

```
char c;  
do {  
    c = lireUnCaractereAuClavier();  
} while (c != ' '); // ici ";"
```

Différence avec le **while** : le bloc est exécuté au moins une fois.



## for

Utilisation : le nombre d'itérations est connu. Syntaxe :

```
for (initialisation ; condition ; modification)
```



## for

Utilisation : le nombre d'itérations est connu. Syntaxe :

```
for (initialisation ; condition ; modification)
```

- **initialisation** exécuté avant de commencer la boucle ;



### for

Utilisation : le nombre d'itérations est connu. Syntaxe :

```
for (initialisation ; condition ; modification)
```

- **initialisation** exécuté avant de commencer la boucle ;
- **condition** détermine s'il est possible de continuer la boucle ;





### for

Utilisation : le nombre d'itérations est connu. Syntaxe :

```
for (initialisation ; condition ; modification)
```

- **initialisation** exécuté avant de commencer la boucle ;
- **condition** détermine s'il est possible de continuer la boucle ;
- **modification** modifie les données pour l'itération suivante.



## for

### Exemple (Équivalent du while)

---

```
int i;  
for (i=0 ; i<10 ; i++) {  
    System.out.println(i);  
}
```



# for

### Exemple (Équivalent du while)

```
for (int i=0 ; i<10 ; i++) {  
    System.out.println(i);  
}
```

### Remarque

Possibilité de déclarer l'indice de boucle lors de l'initialisation  
⇒ variable locale à la boucle



## Sortie de boucle

Possibilité de modifier l'exécution d'une boucle avec **break** et **continue**. Mais ne pas en abuser.

### Exemple

```
while( condition1 ) {  
    if (condition2)  
        break;  
    ...  
}
```



## Sortie de boucle

Possibilité de modifier l'exécution d'une boucle avec **break** et **continue**. Mais ne pas en abuser.

### Exemple

```
while (condition1 && !condition2) {  
    ...  
}
```



# Sommaire

- 1 **Rappel**
  - Notion de variables
- 2 **Typage des variables**
  - Les nombres
  - les types numériques
  - Les booléens
  - Portée
  - Opérations
- 3 **Instructions du langage Java**
  - Commentaires
  - Littéraux (constantes)
  - Mots clés
  - Branchements conditionnels
  - Boucles
- 4 **Premiers programmes**



# Hello World

```
/**  
 * Premier programme Java : hello world  
 */  
public class Hello {  
    public static void main(String args[]) {  
        System.out.println("Hello");  
    }  
}
```



## Remarques

- Nom du fichier et nom du programme doivent coïncider
- Ex : le fichier doit s'appeler **Hello.java**
- Compilation : **javac Hello.java**
- Exécution : **java Hello**





## Passage de paramètres au main

- Passage de paramètres par la variable **args**
- Ex : **java Hello 1234 xyz**
  - $\text{args}[0] \leftarrow 1234$
  - $\text{args}[1] \leftarrow \text{xyz}$