# Cours Info - 5

## Variables et instructions

D.Malka

MPSI 2017-2018

# Sommaire

# Sommaire

# Définition d'une variable

## Qu'appelle-t-on variable ?

Une variable est un emplacement mémoire réservé par un programme pour y stocker une valeur.

Accès à la variable :

- **en lecture ;**
- **en écriture.**

# Représentation de la mémoire

Représentation de la mémoire :

| *nom variable* | **valeur de la variable** |
|---|---|

Sur un exemple :

| *x* | **5.202** |
|---|---|

| *toto* | **"Hello World !"** |
|---|---|

# Etat du programme

## Etat du programme

On appelle *état du programme* l'ensemble des variables du programme et de leurs valeurs.

Etat du programme :

| | |
|---|---|
| *x* | **5.202** |
| *toto* | **"Hello World !"** |
| *y* | **1492** |
| *a* | **-9.3** |

# Evaluation d'une expression

## Quelle valeur pour une expression ?

La valeur des expressions contenant des variables dépend de l'état du programme précédant leur évaluation.

Par exemple, supposons l'état suivant :

| $x$ | **6** |
|---|---|

| $y$ | **20** |
|---|---|

- L'expression `x+y` vaut ... `26`
- L'expression `y%x+3.0` vaut ... `6.0`

# Sommaire

# Déclaration & initialisation

Avant de pourvoir utiliser une variable dans une expression, il faut la déclarer.

## Déclaration d'une variable

La déclaration d'une variable réserve un emplacement mémoire qui en contiendra la valeur.

Représentation mémoire :

# En Python : typage dynamique

Majorité des langages : déclaration du nom **et** du type (typage statique)

Exemple (en C) :

```c
int a; //declaration de la variable a et de son type int
```

Python : pas de déclaration du type $\Rightarrow$ typage à l'exécution (typage dynamique)

# Initialisation

## Initialisation d'une variable

Initialisation d'une variable = affectation d'une valeur pour la première fois.

En C, déclaration et initialisation sont distinctes :

```
1 int a;
2 a=0;
```

En Python, initialisation obligatoire à la déclaration :

```
1 a=1
```

# Suppression d'une variable ?

## Suppression d'une variable

Python : impossibilité de supprimer une variable une fois déclarée [a].

a. En fait si mais c'est une mauvaise pratique.

# Nom d'une variable

Comment nommer une variable ?

- pas d'espace $\Rightarrow$ remplacer par _, ex : `ma_variable`,
- pas de mots clé : `for, while, if, as ...`,
- ne pas commencer par un chiffre,
- pas d'accent,
- Donner un nom explicite : pour la position `x` ou `position` et non `tutu`.

# Sommaire

# Qu'est-ce qu'une instruction ?

## Affectation

Instruction modifiant la valeur d'une variable.

En Python, opérateur d'affectation : « = »

Associativité de droite à gauche !

C'est le moins prioritaire des opérateurs.

CODE PYTHON

```
In [1]: a=0 //déclaration
In [2]: a
Out[1]: 0

In [3]: a=3.0 //affectation
In [4]: a
Out[2]: 3.0
```

# Quelques exemples

CODE PYTHON

```
In [5]: age=18
In [6]: age_2=age+2
In [7]: nom="Perna"
In [8]: prenom="Louison"
```

Modification de l'état :

| *age* | **18** |
|---|---|

| *age_2* | **20** |
|---|---|

| *nom* | **"Louison"** |
|---|---|

| *prenom* | **"Perna"** |
|---|---|

# Affectation multiples

CODE PYTHON

```
In [9]: y=x=z=3
```

Modification de l'état :

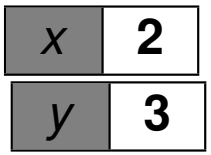

Conseil : à éviter.

# Affectation parallèle

CODE PYTHON

```
In [10]: x,y=2,3
```

Modification de l'état :




Conseil : pourquoi pas.

# Sommaire

# Qu'est-ce qu'une instruction ?

## Instruction

Une instruction modifie l'état du programme.

L'affectation est une d'instruction particulière.

En général, en Python, une instruction n'a pas de valeur.

# Instruction vs expression

## Instruction $\neq$ expression

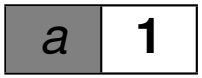
Ne pas confondre instruction et expression.

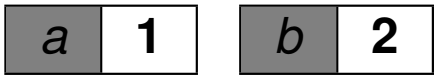
Tester sur `http://www.pythontutor.com/` :

```
1 a=1
2 b=a+1
3 b*3
4 print(a)
5 print(b)
```

Quelles sont les lignes ci-dessus qui sont des expressions ? des instructions ? Que valent `a` et `b` finalement ?

# Instruction vs expression

1. 

2. 

3. `b*3` n'est pas une instruction : état non modifié
4. `print(a)` affiche 1
5. `print(b)` affiche 2

# Modification de l'état par un jeu d'instructions

```
1 x=3
2 y=x
3 x=3*x*x
4 y=x+2
5 print(x);print(y)
```

Quelles sont les lignes ci-dessus qui sont des expressions ? des instructions ? Que valent `x` et `y` finalement ?

# Modification de l'état par un jeu d'instructions

1. | $x$ | **3** |
2. | $x$ | **3** | | $y$ | **3** |
3. | $x$ | **27** | | $y$ | **3** |
4. | $x$ | **27** | | $y$ | **29** |
5. | $x$ | **27** | | $y$ | **29** |
6. | $x$ | **27** | | $y$ | **29** |

On peut aussi représenter l'état sous la forme d'un tableau.

| Instruction/Variable | x | y |
|---|---|---|
| 1 | 3 | |
| 2 | 3 | 3 |
| 3 | 27 | 3 |
| 4 | 27 | 29 |
| 5 | 27 | 29 |
| 6 | 27 | 29 |

Finalement $x = 27$ et $y = 29$.

# Incrémentation/décrémentation

```
1 x=4
2 y=0
3 x=x+1
4 y=y-2
5 x+=1 //equivalent a x=x+1
6 y-=2 //equivalent a y=y-2
```

Que valent `x` et `y` ?

# Incrémentation/décrémentation

1. | $x$ | 4 |
2. | $x$ | 4 | | $y$ | 0 |
3. | $x$ | 5 | | $y$ | 0 |
4. | $x$ | 5 | | $y$ | -2 |
5. | $x$ | 6 | | $y$ | -2 |
6. | $x$ | 6 | | $y$ | -4 |

Finalement $x = 6$ et $y = -4$.

# Sommaire

# 1er (tout petit) algorithme

## Spécification

Ecrire un algorithme permettant de permuter les valeurs de `x` et `y`.

# Algorithme naïf

Tester ce programme :

```
1 x=2
2 y=4
3 x=y
4 y=x
```

1. | x | 4 |
2. | x | 2 | | y | 4 |
3. | x | 4 | | y | 4 |
4. | x | 4 | | y | 4 |

Ne fonctionne pas !

# Algorithme (langage naturelle)

En langage naturelle :

1. stocker la valeur de `x` dans une variable `temp`,
2. affecter la valeur de `y` à `x`,
3. affecter la valeur de `temp` à `y`.

# Implémentation en Python

```
1 x,y,temp=2,4,0
2 temp=x
3 x=y
4 y=temp
```

1. | *x* | **2** | *y* | **4** | *temp* | **0** |
2. | *x* | **2** | *y* | **4** | *temp* | **2** |
3. | *x* | **4** | *y* | **4** | *temp* | **2** |
4. | *x* | **2** | *y* | **4** | *temp* | **2** |

L'algorithme fonctionne ! $x = 4$ et $y = 2$.

En Python, encore plus élégant : `x,y=y,x`.