Variable ...
Vous avez dit variable ...



Pascal Lafourcade

UNIVERSITÉ Clermont Auvergne

LIMOS

IREM Institut de Recherche sur l'Enseignement des Mathématiques Clermont-Ferrand

LIMOS LABORATOIRE D'INFORMATIQUE, DE MODÉLISATION ET D'OPTIMISATION DES SYSTÈMES

15 Janvier 2019

Définition du dictionnaire

Variable

- Qui est sujet à variations, qui peut changer souvent au cours d'une durée, selon les circonstances.
- Qui peut prendre plusieurs valeurs distinctes selon les circonstances.
- Qui est conçu, fabriqué de façon à pouvoir subir des variations.

Définition du dictionnaire

Variable

- Qui est sujet à variations, qui peut changer souvent au cours d'une durée, selon les circonstances.
- Qui peut prendre plusieurs valeurs distinctes selon les circonstances.
- Qui est conçu, fabriqué de façon à pouvoir subir des variations.

MATH., INFORMAT., PHYS.

Symbole, terme, phénomène observable auquel différentes valeurs prises dans un ensemble peuvent être attribuées.

Définition du dictionnaire

Variable

- Qui est sujet à variations, qui peut changer souvent au cours d'une durée, selon les circonstances.
- Qui peut prendre plusieurs valeurs distinctes selon les circonstances.
- Qui est conçu, fabriqué de façon à pouvoir subir des variations.

MATH., INFORMAT., PHYS.

Symbole, terme, phénomène observable auquel différentes valeurs prises dans un ensemble peuvent être attribuées.

Antonyme : constante

Vous avez dit variable ...

$A = 2\sin x + 3x$

$y = ax + b$

$\forall x \exists y, x + 3 > y$

$f(x) = sin(x)$

$$E[X] := \sum_{u \in U} X(u) \Delta P[u]$$

$x = x + 1$

if $x > 0$ *then*

```
def f(x) :
    return x+1
```

$4x^2 - 2x + 1 = 0$

L'implicite ...

En mathématiques

$$A = 2\sin x + 3x$$

L'implicite ...

En mathématiques

$$A = 2\sin x + 3x$$

La valeur de A *dépend* de la valeur de x.

L'implicite ...

En mathématiques

$$A = 2\sin x + 3x$$

La valeur de A *dépend* de la valeur de x.

- x prend ses valeurs dans un ensemble donné D.
- À *chaque* valeur de x correspond une valeur de A.
- Il faudrait écrire :

$$\forall x \in D, A(x) = 2\sin x + 3x$$

L'implicite ...

En mathématiques

$$2x^2 - 8 = 0$$

L'implicite ...

En mathématiques

$$2x^2 - 8 = 0$$

- Existe-t-il x, tel que $2x^2 - 8 = 0$?

Mais où vit x ? $\mathbb{N}$? $\mathbb{Z}$? ...

L'implicite ...

En mathématiques

$$2x^2 - 8 = 0$$

- Existe-t-il x, tel que $2x^2 - 8 = 0$?

Mais où vit x ? $\mathbb{N}$? $\mathbb{Z}$? ...

- Existe-t-il $x \in D$, tel que $2x^2 - 8 = 0$?

L'implicite ...

En mathématiques

$$2x^2 - 8 = 0$$

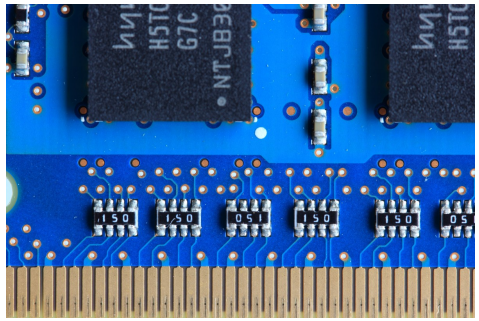
- Existe-t-il x, tel que $2x^2 - 8 = 0$?

Mais où vit x ? $\mathbb{N}$? $\mathbb{Z}$? ...

- Existe-t-il $x \in D$, tel que $2x^2 - 8 = 0$?

$$2x^2 + 8 = 0$$

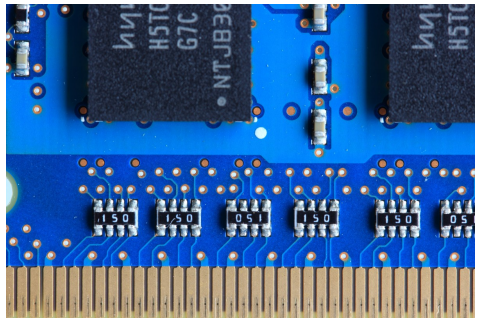
En informatique



Notion de variable

- Une variable désigne un **emplacement** de la **mémoire** qui porte un **nom**.

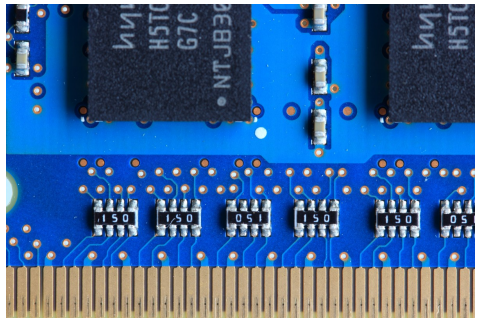
En informatique



Notion de variable

- Une variable désigne un **emplacement** de la **mémoire** qui porte un **nom**.
- Contient une **valeur**.

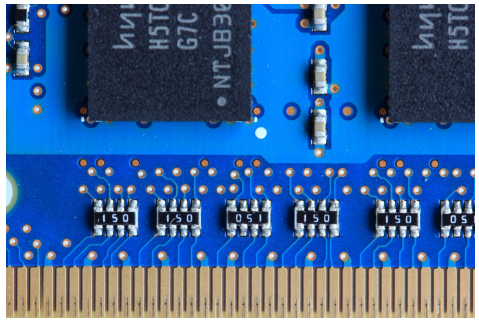
En informatique



Notion de variable

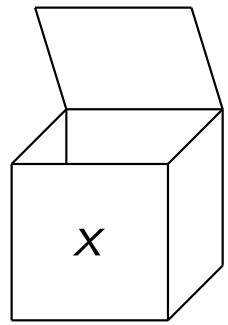
- Une variable désigne un **emplacement** de la **mémoire** qui porte un **nom**.
- Contient une **valeur**.
- Son contenu peut changer durant l'exécution d'un programme.

En informatique



Notion de variable

- Une variable désigne un **emplacement** de la **mémoire** qui porte un **nom**.
- Contient une **valeur**.
- Son contenu peut changer durant l'exécution d'un programme.





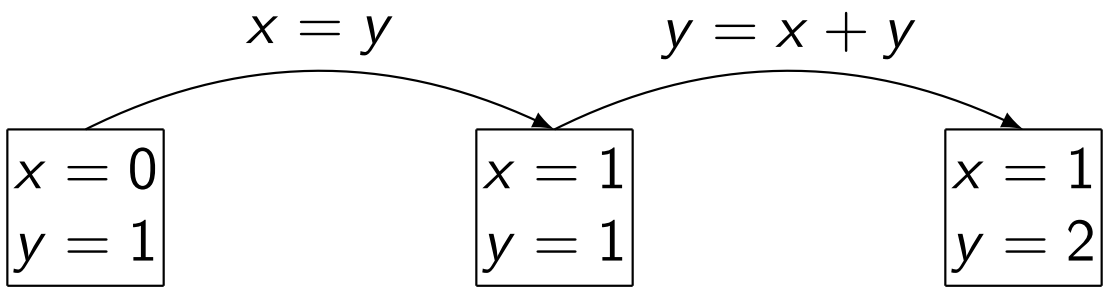
Plan

- L'état
- Type
- Nommage
- L'affectation
- Égalité
- Procédure et fonction
- Variable globale vs locale
- Récursivité
- Mémoire et complexité
- Passage des variables
- Ordinateur Quantique
- Conclusion

Plan

- L'état
- Type
- Nommage
- L'affectation
- Égalité
- Procédure et fonction
- Variable globale vs locale
- Récursivité
- Mémoire et complexité
- Passage des variables
- Ordinateur Quantique
- Conclusion

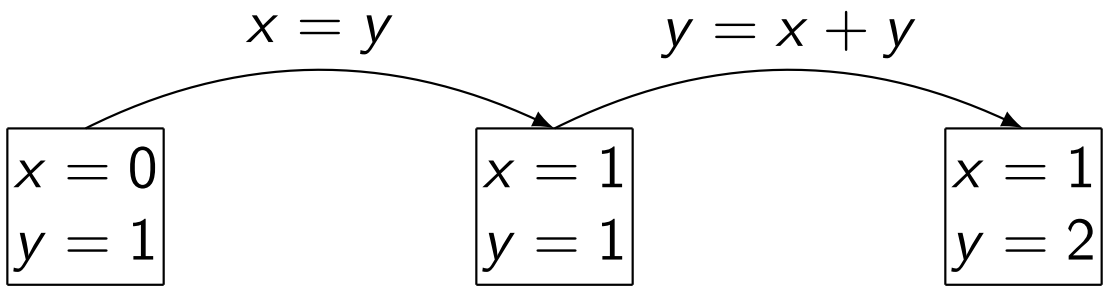
La notion d'état



état de la mémoire

L'ensemble des variables présentes en mémoire et leurs valeurs respectives

La notion d'état

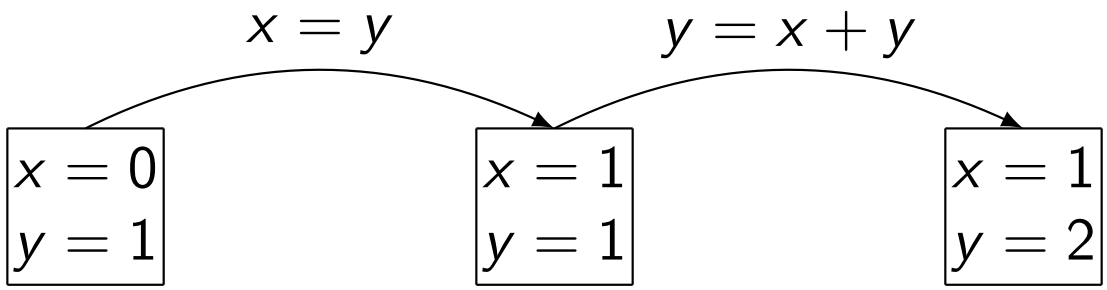


état de la mémoire

L'ensemble des variables présentes en mémoire et leurs valeurs respectives

Une instruction d'un programme ne « fait » pas quelque chose, elle « fait quelque chose à l'état ».

La notion d'état



état de la mémoire

L'ensemble des variables présentes en mémoire et leurs valeurs respectives

Une instruction d'un programme ne « fait » pas quelque chose, elle « fait quelque chose à l'état ».

Pour l'enseignement

Expliciter et **concrétiser** cette notion d'état pour les élèves.

Plan

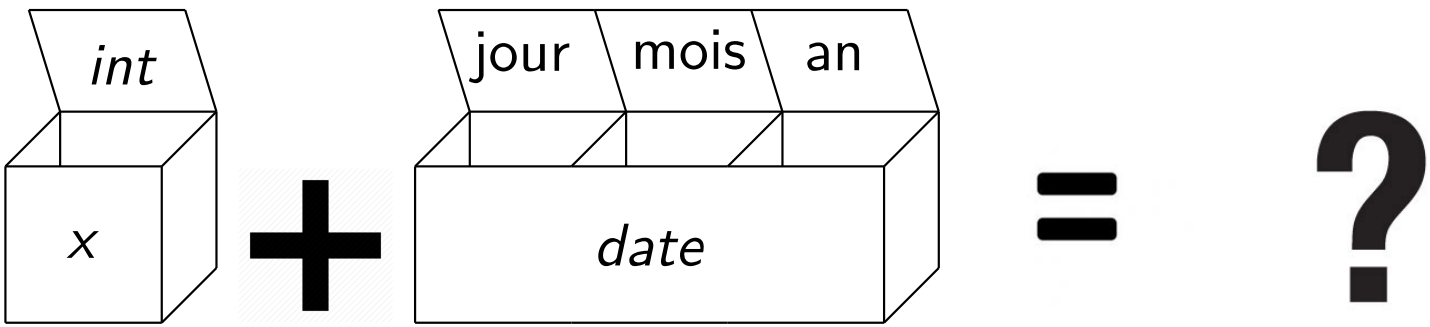
- L'état
- Type
- Nommage
- L'affectation
- Égalité
- Procédure et fonction
- Variable globale vs locale
- Récursivité
- Mémoire et complexité
- Passage des variables
- Ordinateur Quantique
- Conclusion

Des choux et des carottes



Des choux et des carottes





Taille des boîtes prédéfinies ...

... espace mémoire fini (2^{64}, tables vs listes chaînées).

Types de base

`int` : 42 -12	+ - * / mod
`float` : -3.1 0. 1.3e-4	+. -. *. /.
`char` : 'a' 'B'	char_of_int, int_of_char
`string` : "abc"	.
`bool` : true false	&& || not

Autres Types

Type produit

type complexe = float * float

Type énuméré

type couleur = Rouge | Vert | Bleu

Tableaux de taille fixe (tuple)

x = (4,5)

Listes chainée

l=[]
l.append (x)

Type des fonctions

Des fonctions

- $f(x) = x^2$

Type des fonctions

Des fonctions

- $f(x) = x^2 : \mathbb{R} \to \mathbb{R}$

Type des fonctions

Des fonctions

- $f(x) = x^2 : \mathbb{R} \to \mathbb{R}$ ou `float` $\to$ `float` ou `int` $\to$ `int`
- `estplusgrand(a,b)` :

Type des fonctions

Des fonctions

- $f(x) = x^2 : \mathbb{R} \to \mathbb{R}$ ou `float` $\to$ `float` ou `int` $\to$ `int`
- `estplusgrand(a,b)` : `int` * `int` $\to$ `bool`

Type des fonctions

Des fonctions

- $f(x) = x^2 : \mathbb{R} \to \mathbb{R}$ ou `float` $\to$ `float` ou `int` $\to$ `int`
- `estplusgrand(a,b)` : `int * int` $\to$ `bool`
  ou `float * float` $\to$ `bool`

Type des fonctions

Des fonctions

- $f(x) = x^2 : \mathbb{R} \to \mathbb{R}$ ou `float` $\to$ `float` ou `int` $\to$ `int`
- `estplusgrand(a,b)` : `int * int` $\to$ `bool`
  ou `float * float` $\to$ `bool`
  ou `char * char` $\to$ `bool`

Type polymorphe α

Type de la fonction estegale(x,y)

Type polymorphe α

Type de la fonction estegale(x,y)

`int * int` $\rightarrow$ `bool`

Type polymorphe α

Type de la fonction estegale(x,y)

`int * int` $\rightarrow$ `bool`

`float * float` $\rightarrow$ `bool`

Type polymorphe α

Type de la fonction estegale(x,y)

`int * int` $\rightarrow$ `bool`

`float * float` $\rightarrow$ `bool`

`char * char` $\rightarrow$ `bool`

Type polymorphe α

Type de la fonction estegale(x,y)

`int * int` $\rightarrow$ `bool`

`float * float` $\rightarrow$ `bool`

`char * char` $\rightarrow$ `bool`

α `*` α $\rightarrow$ `bool`

Typage

Statique (en C)

```
int a ;
a=5 ;
```

Dynamique (en Python)

```
def identite(x) :
    return x
```

Conversions de type

Explicites (cast)

```
int a;
a=(int)3.14;
print("a=%d",a);
```

```
a=3
```

Implicites

```
int  i = 17;
char c = 'c'; /* ascii value is 99 */
float sum;
sum = i + c;
printf("Value of sum : %f\n", sum );
```

```
Value of sum : 116.000000
```

Types en Scratch

Il a 2 types :

- nombre (number)
- chaîne de caractères (string)





Types en Scratch

Il a 2 types :

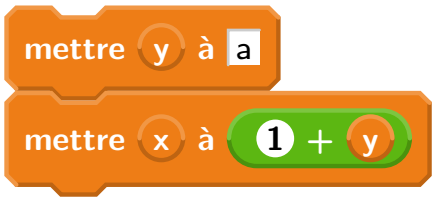
- nombre (number)
- chaîne de caractères (string)





Remarques

- Les opérateurs sont typés
- Les variables ne sont pas typées



Types en Scratch

Il a 2 types :

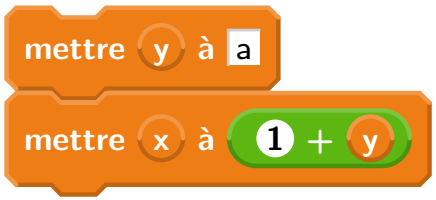
- nombre (number)
- chaîne de caractères (string)





Remarques

- Les opérateurs sont typés
- Les variables ne sont pas typées



Que vaut x ?

Types en Scratch

Il a 2 types :

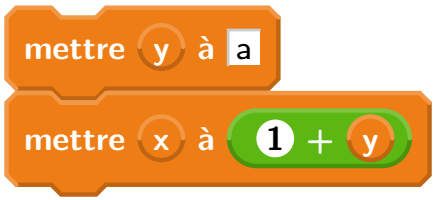
- nombre (number)
- chaîne de caractères (string)





Remarques

- Les opérateurs sont typés
- Les variables ne sont pas typées



Que vaut x ? 1

Plan

- L'état
- Type
- Nommage
- L'affectation
- Égalité
- Procédure et fonction
- Variable globale vs locale
- Récursivité
- Mémoire et complexité
- Passage des variables
- Ordinateur Quantique
- Conclusion

Choix des noms des variables :

En mathématiques

- n, p, q pour des entiers
- i, j, k pour des indices
- x pour un réel
- z pour un complexe
- f, g, h pour des fonctions
- ...

Choix des noms des variables :

En mathématiques

- n, p, q pour des entiers
- i, j, k pour des indices
- x pour un réel
- z pour un complexe
- f, g, h pour des fonctions
- ...

Il y a que 26 lettres !

Choix des noms des variables :

En mathématiques

- n, p, q pour des entiers
- i, j, k pour des indices
- x pour un réel
- z pour un complexe
- f, g, h pour des fonctions
- ...

Il y a que 26 lettres !

α, β... ou x_1, x_2... ou encore $\alpha_1, \alpha_2 \ldots$

Choix des noms des variables :

En informatique

- *cpt* pour un compteur
- *cote*, *somme*, *resultat*

Choix des noms des variables :

En informatique

- *cpt* pour un compteur
- *cote*, *somme*, *resultat*

Conventions de nommage

- **Par définition** : Pas d'espaces, ni symboles réservés, pas d'accents
- Des "vrais" noms explicites !
- Plus de restriction de taille !
- Attention aux majuscules (CONSTANTES, Classes etc...)

Plan

- L'état
- Type
- Nommage
- L'affectation
- Égalité
- Procédure et fonction
- Variable globale vs locale
- Récursivité
- Mémoire et complexité
- Passage des variables
- Ordinateur Quantique
- Conclusion

Action élémentaire : l'affectation

État initial

a	b	x	somme	...
2	39	0	128	...

Action élémentaire : l'affectation

État initial

a	b	x	somme	...
2	39	0	128	...

L'affectation change l'état de la mémoire

Variable $\leftarrow$ expression

Autres notations :

```
a := b
a prend la valeur b
b -> a
```

Action élémentaire : l'affectation

État initial

a	b	x	somme	...
2	39	0	128	...

L'affectation change l'état de la mémoire

Variable $\leftarrow$ expression

Autres notations :

```
a := b
a prend la valeur b
b -> a
```

Exemples : x $\leftarrow$ 1

$p \leftarrow 3 + b$

En Python : `x = 1`

`p = 3 + b`

Action élémentaire : l'affectation

État initial

a	b	x	somme	...
2	39	1	128	...

L'affectation change l'état de la mémoire

Variable $\leftarrow$ expression

Autres notations :

```
a := b
a prend la valeur b
b -> a
```

Exemples : $x \leftarrow 1$

$p \leftarrow 3 + b$

En Python :
```
x = 1
p = 3 + b
```

Action élémentaire : l'affectation

État initial

a	b	x	somme	p	...
2	39	1	128	42	...

L'affectation change l'état de la mémoire

Variable $\leftarrow$ expression

Autres notations :

```
a := b
a prend la valeur b
b -> a
```

Exemples : x $\leftarrow$ 1

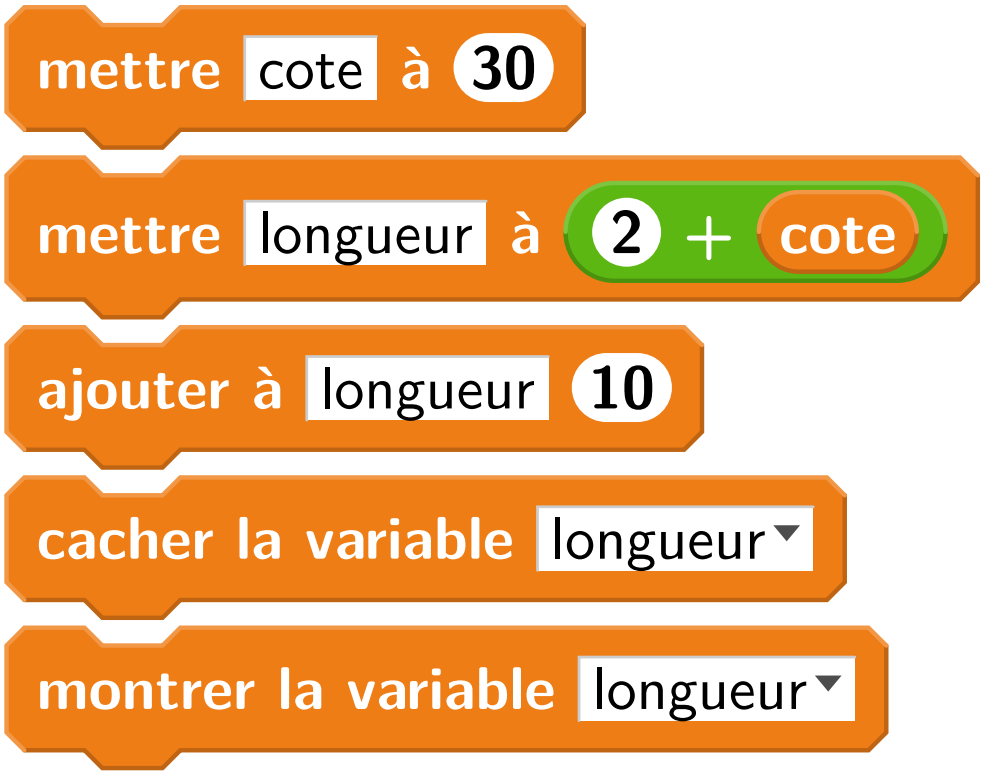
$p \leftarrow 3 + b$

En Python : `x = 1`

`p = 3 + b`

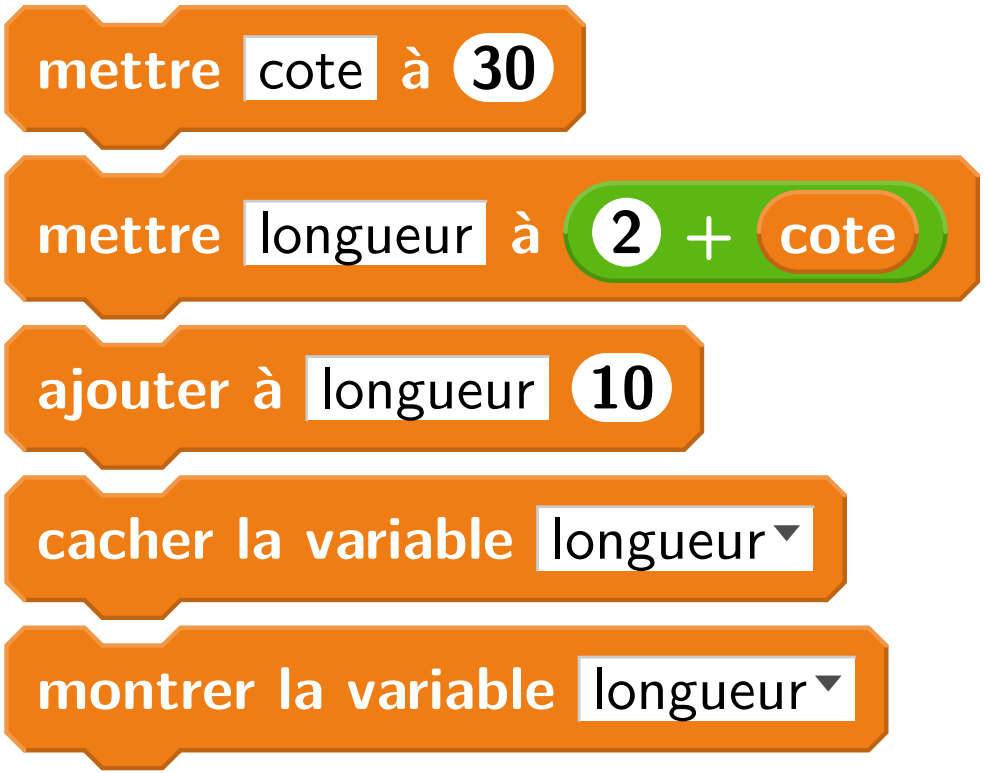
Dans Scratch

- Le menu variable est au départ vide.
- Il faut créer une variable pour avoir les blocs de commande.



Dans Scratch

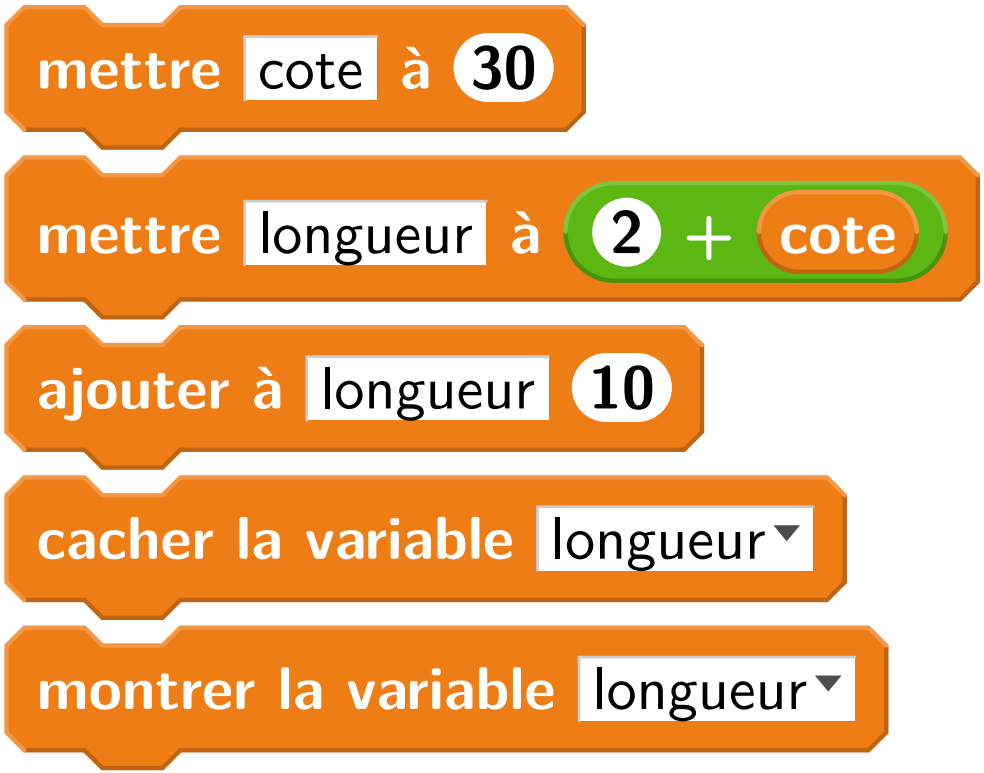
- Le menu variable est au départ vide.
- Il faut créer une variable pour avoir les blocs de commande.



Quelle valeur est affichée pour longueur ?

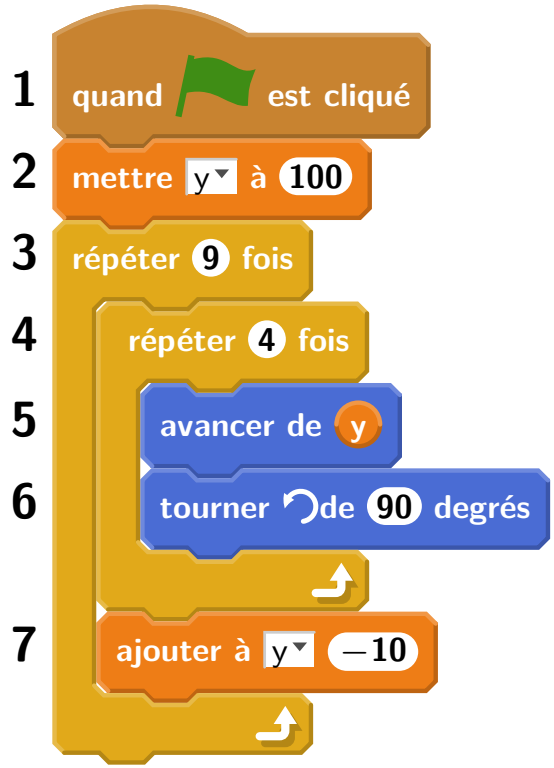
Dans Scratch

- Le menu variable est au départ vide.
- Il faut créer une variable pour avoir les blocs de commande.



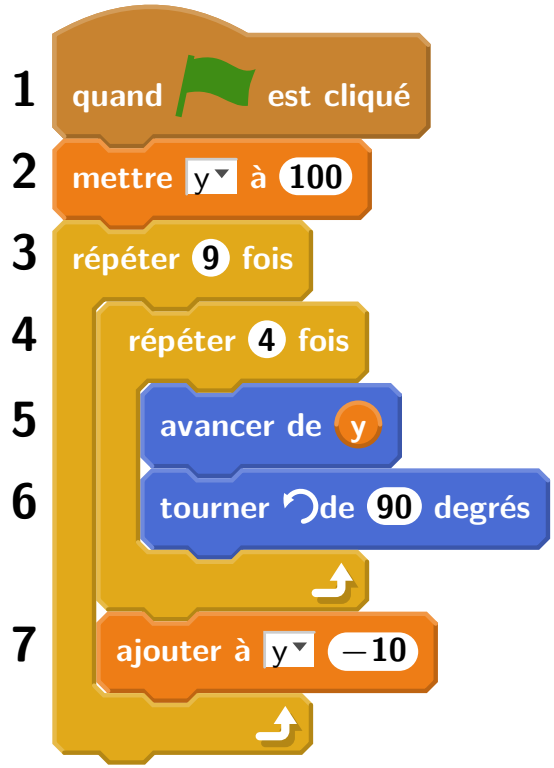
Quelle valeur est affichée pour longueur ? 42

Exercice



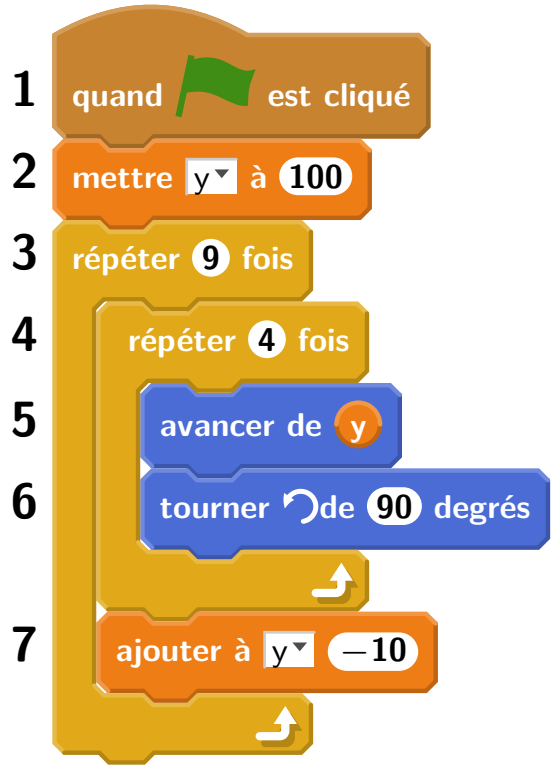
- Que vaut la variable y après l'exécution du programme ?

Exercice



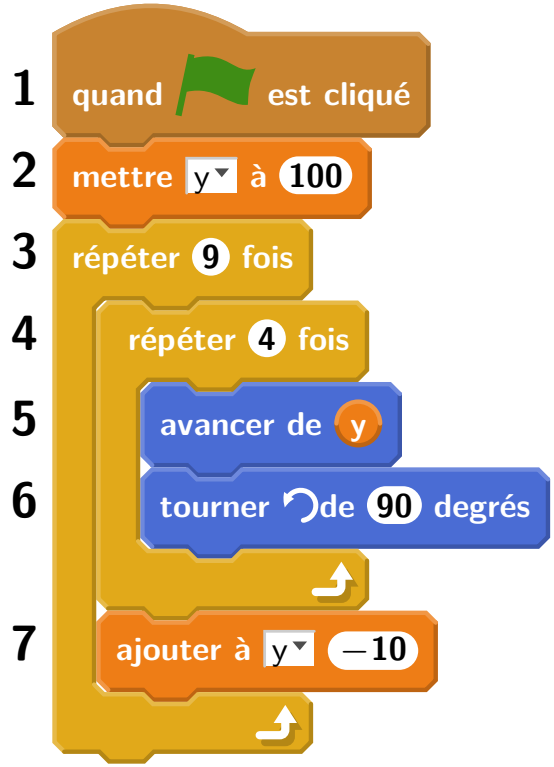
- Que vaut la variable y après l'exécution du programme ? 10

Exercice

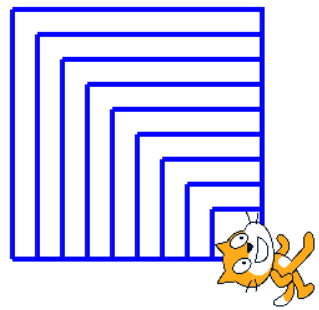


- Que vaut la variable y après l'exécution du programme ? 10
- Que fait ce programme ?

Exercice



- Que vaut la variable y après l'exécution du programme ? 10
- Que fait ce programme ?



Si on baisse le stylo ;-)

Plan

- L'état
- Type
- Nommage
- L'affectation
- Égalité
- Procédure et fonction
- Variable globale vs locale
- Récursivité
- Mémoire et complexité
- Passage des variables
- Ordinateur Quantique
- Conclusion

En mathématiques :

3 égalités différentes

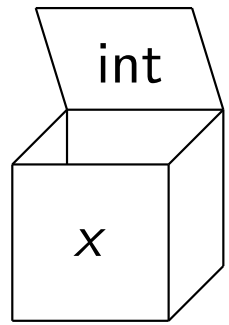
- $\forall x \in \mathbb{R}, f(x) = 2x + 1$ (égalité de définition)
- $\forall x \in \mathbb{R}, 1 = sin(x)^2 + cos(x)^2$ (propriété)
- $\exists x \in \mathbb{Z}, 2x + 1 = 0$? (équation)

En informatique :

Une autre notion d'égalité

- Comparaison des valeurs dans un état de la mémoire.
  $== : \alpha * \alpha \rightarrow bool$

Exemple : $3 == x$



Dans certains langages (Scratch), le test d'égalité est noté $x = 3$.

Plan

- L'état
- Type
- Nommage
- L'affectation
- Égalité
- Procédure et fonction
- Variable globale vs locale
- Récursivité
- Mémoire et complexité
- Passage des variables
- Ordinateur Quantique
- Conclusion

Procédure vs fonction

- Une fonction renvoie une valeur
- Une procédure ne renvoie rien

Notion de fonction en informatique

Motivation

- Organiser le code
- Isoler des parties réutilisables
- Faciliter le développement et la maintenance

Notion de fonction en informatique

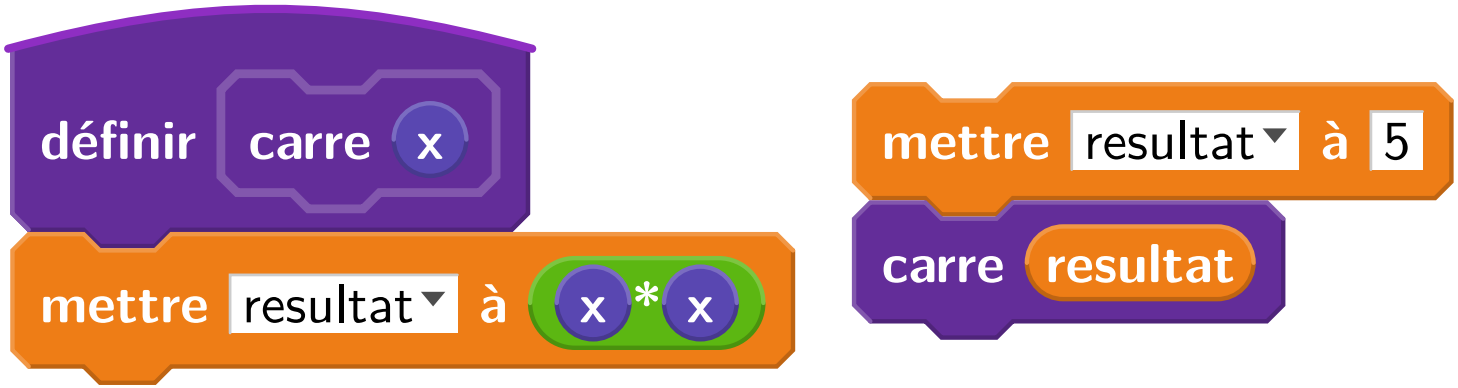
Motivation

- Organiser le code
- Isoler des parties réutilisables
- Faciliter le développement et la maintenance

Comme en mathématiques, une fonction :

- posséde un ou plusieurs arguments (ou paramètres) ;
- pour chaque valeur possible de ses arguments, renvoie une valeur.

Définition d'une procédure (bloc) en Scratch



- resultat est une variable globale
- x est une variable locale

Définition d'une fonction

En Python :

```
def nomdelafonction(arguments):
␣␣␣␣...
␣␣␣␣return valeur
```

Exemple :valeur absolue

```
def abs(x):
    if x < 0:
        return -x
    else:
        return x
```

Exercice : maximum

- Écrire un algorithme qui calcule le maximum de 2 entiers.

Exercice : maximum

- Écrire un algorithme qui calcule le maximum de 2 entiers.

Une solution

Si A $>$ B alors
    Max $\leftarrow$ A
Sinon
    Max $\leftarrow$ B

Exercice : maximum

- Écrire un algorithme qui calcule le maximum de 2 entiers.

Une solution

Si A $>$ B alors
    Max $\leftarrow$ A
Sinon
    Max $\leftarrow$ B

- Écrire une fonction `max2` qui calcule le maximum de 2 entiers.

Exercice : maximum

- Écrire un algorithme qui calcule le maximum de 2 entiers.

Une solution

Si A $>$ B alors
    Max $\leftarrow$ A
Sinon
    Max $\leftarrow$ B

- Écrire une fonction `max2` qui calcule le maximum de 2 entiers.

Une solution en Python

```
def max2 (x,y):
    if x > y:
        return x
    else:
        return y
```

Maximum de A, B, C et D.

Écrire un algorithme qui calcule le maximum de 4 entiers.

Maximum de A, B, C et D.

Écrire un algorithme qui calcule le maximum de 4 entiers.

Solution 1 (exhaustive)

Si A$\geq$B et A$\geq$C et A$\geq$D Alors Max $\leftarrow$ A
Sinon
    Si B$\geq$A et B$\geq$C et B$\geq$D alors Max $\leftarrow$B
    Sinon
        Si C$\geq$A et C$\geq$B et C$\geq$D alors Max $\leftarrow$ C
        Sinon Max $\leftarrow$ D

Maximum de A, B, C et D.

Écrire un algorithme qui calcule le maximum de 4 entiers.

Solution 1 (exhaustive)

Si A≥B et A≥C et A≥D Alors Max ← A
Sinon
    Si B≥A et B≥C et B≥D alors Max ←B
    Sinon
        Si C≥A et C≥B et C≥D alors Max ← C
        Sinon Max ← D

Solution 2 (la même sans comparaisons inutiles)

Si A≥B et A≥C et A≥D Alors Max ← A
Sinon
    Si B≥C et B≥D alors Max ←B
    Sinon
        Si C≥D alors Max ← C
        Sinon Max ← D

Maximum de A, B, C et D.

- Une solution avec seulement trois comparaisons ?

Maximum de A, B, C et D.

- Une solution avec seulement trois comparaisons ?

Solution 3 : à l'aide d'une variable supplémentaire

Max $\leftarrow$ A
Si B $>$ Max alors
    Max $\leftarrow$ B
Si C $>$ Max alors
    Max $\leftarrow$ C
Si D $>$ Max alors
    Max $\leftarrow$ D

Maximum de A, B, C et D.

- Une solution avec seulement trois comparaisons ?

Solution 3 : à l'aide d'une variable supplémentaire

Max $\leftarrow$ A
Si B $>$ Max alors
    Max $\leftarrow$ B
Si C $>$ Max alors
    Max $\leftarrow$ C
Si D $>$ Max alors
    Max $\leftarrow$ D

Solution 4 : Par composition de fonctions, en utilisant max2

M1 = max2(A,B) ; M2 = max2(C,D) ; Max = max2(M1,M2)

Maximum de A, B, C et D.

- Une solution avec seulement trois comparaisons ?

Solution 3 : à l'aide d'une variable supplémentaire

Max $\leftarrow$ A
Si B $>$ Max alors
    Max $\leftarrow$ B
Si C $>$ Max alors
    Max $\leftarrow$ C
Si D $>$ Max alors
    Max $\leftarrow$ D

Solution 4 : Par composition de fonctions, en utilisant max2

M1 = max2(A,B) ; M2 = max2(C,D) ; Max = max2(M1,M2)

ou encore max2(max2 (A, B), max2 (C, D))

Plan

- L'état
- Type
- Nommage
- L'affectation
- Égalité
- Procédure et fonction
- Variable globale vs locale
- Récursivité
- Mémoire et complexité
- Passage des variables
- Ordinateur Quantique
- Conclusion

Qu'affiche ce programme ?

```
def f(z):
    x=z+a
    return x

x=3
a=3
print (f(x))
a=4
print (f(x))
```

Qu'affiche ce programme ?

```
def f(z):
    x=z+a
    return x

x=3
a=3
print (f(x))
a=4
print (f(x))
```

6 puis 7

Qu'affiche ce programme ?

```
def f(z):
    x=z+a
    return x

x=3
a=3
print (f(x))
a=4
print (f(x))
```

6 puis 7

a est une variable globale

z est une variable locale

Que fait ce programme ?

```
def f(z):
    y=x+z
    x=42
    return y

x=18
print f(x);
```

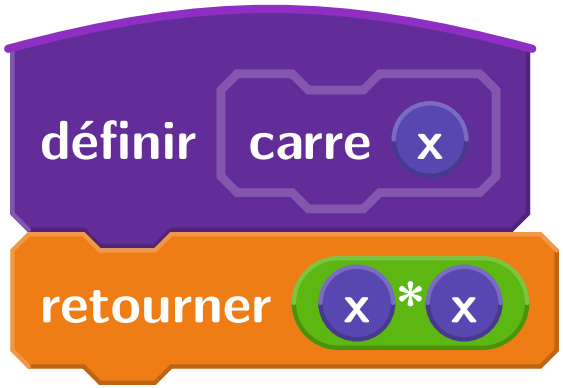
Que fait ce programme ?

```
def f(z):
    y=x+z
    x=42
    return y

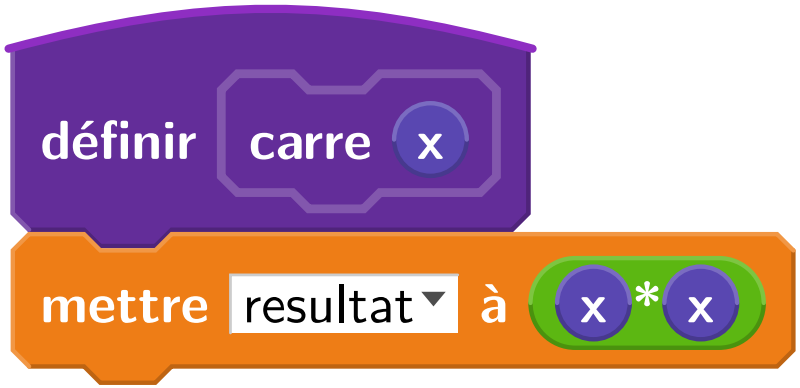
x=18
print f(x);
```

ERREUR, x non défini dans y=x+z

Scratch



Que des procédures en Scratch



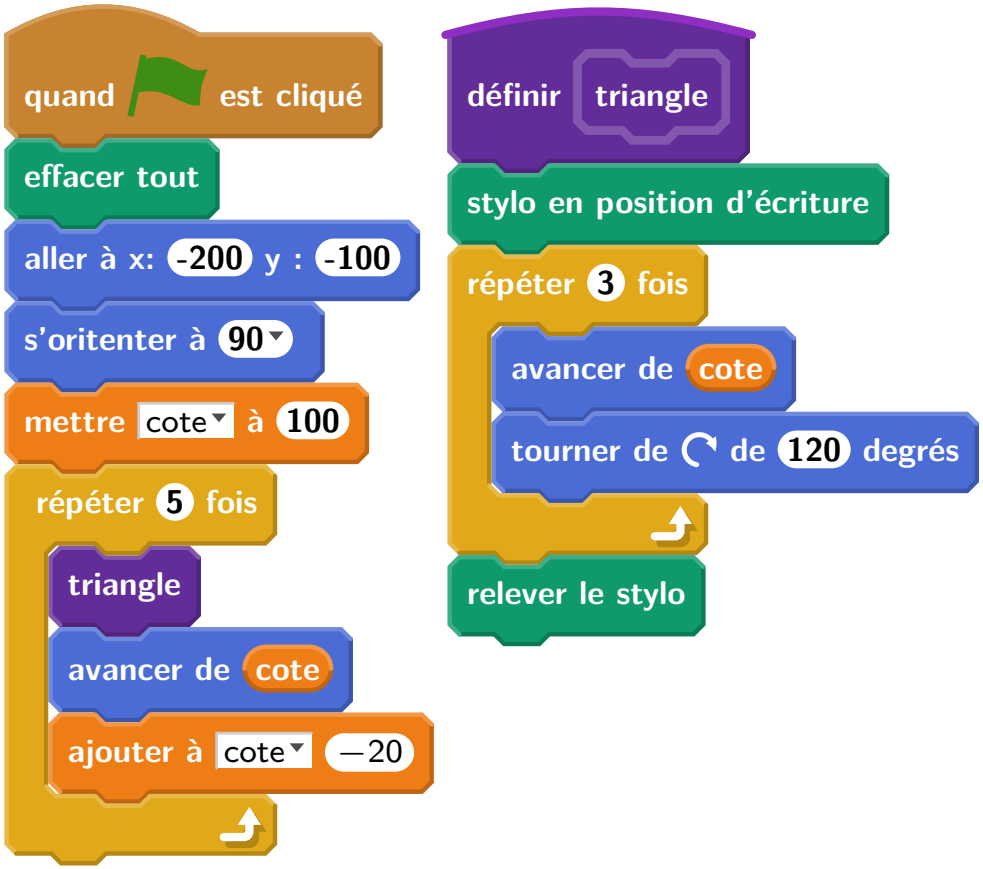


PAS DE FONCTION ! Utiliser Snap !

command, reporter, predicate

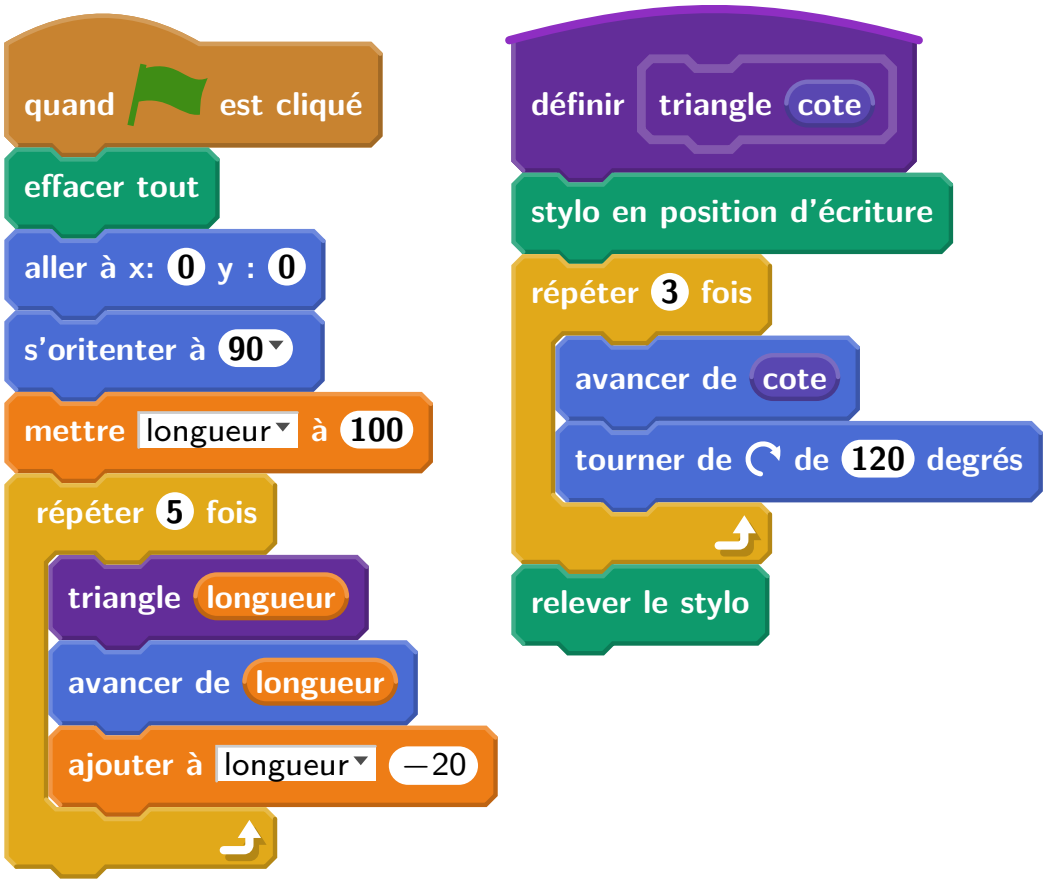
Perdre de mauvaises habitudes des variables globales !

Avec une variable globale



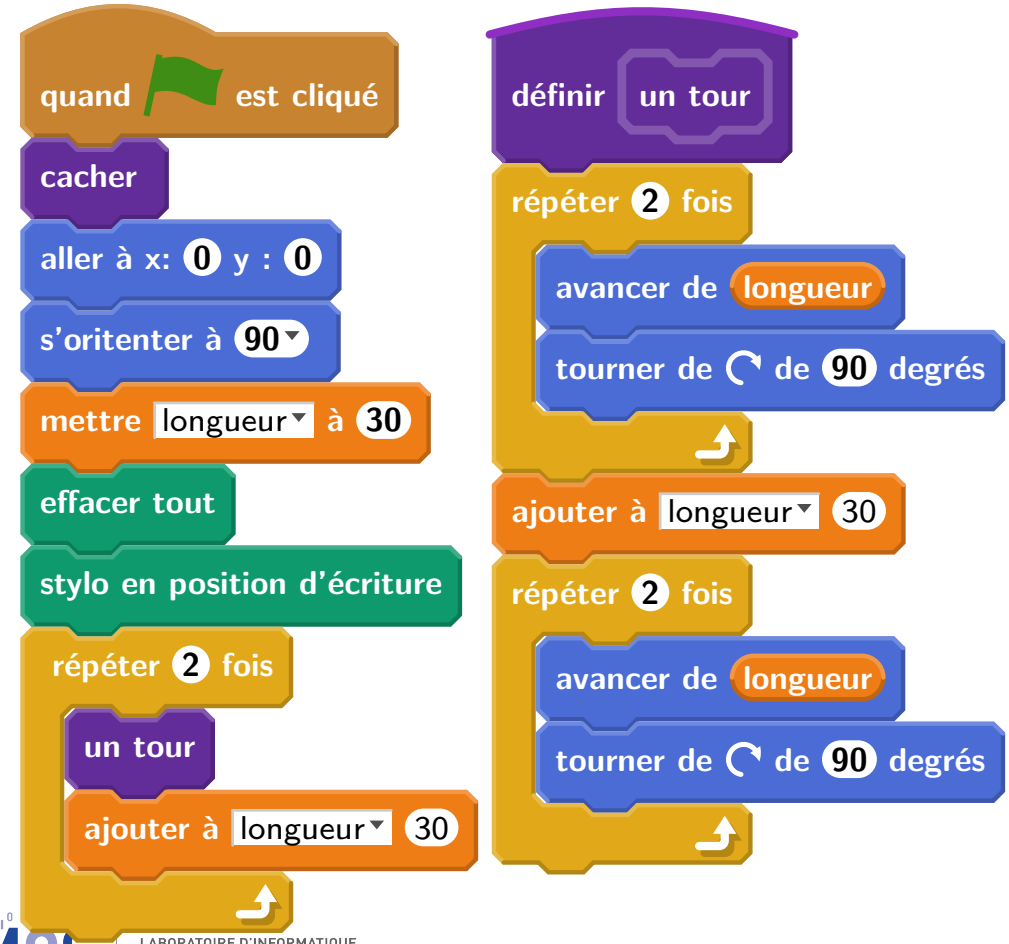
Prendre de bonnes habitudes avec les variables locales

Avec variable locale

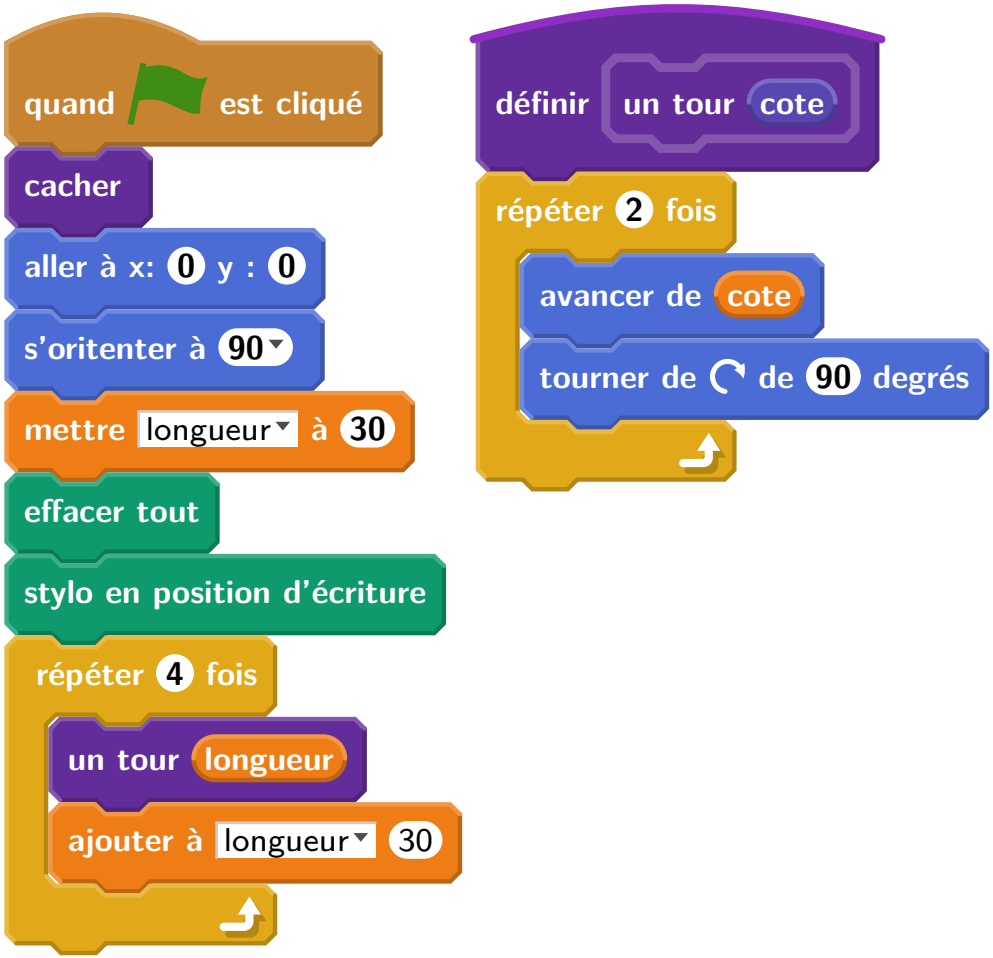


À vous de jouer

Métropole – La Réunion – Antilles-Guyane, 14 septembre 2017, Exercice 3, 6 points sur 50. Transformer ce programme en un programme avec des variables locales.



Solution


quand est cliqué
cacher
aller à x: 0 y : 0
s'oritenter à 90
mettre longueur à 30
effacer tout
stylo en position d'écriture
répéter 4 fois
un tour longueur
ajouter à longueur 30
définir un tour cote
répéter 2 fois
avancer de cote
tourner de de 90 degrés

Plan

- L'état
- Type
- Nommage
- L'affectation
- Égalité
- Procédure et fonction
- Variable globale vs locale
- Récursivité
- Mémoire et complexité
- Passage des variables
- Ordinateur Quantique
- Conclusion

Composition de fonction permet la récursive

Ecrire une fonction récursive que calcule factorielle : $n!$

- $0! = 1$
- $n! = n * (n-1)!$

Composition de fonction permet la récursive

Ecrire une fonction récursive que calcule factorielle : $n!$

- $0! = 1$
- $n! = n * (n-1)!$

En Python :

```
def fact (n) :
    if n == 0:
        return 1
    else:
        return n * fact (n-1)
```

Composition de fonction permet la récursive

Ecrire une fonction récursive que calcule factorielle : $n!$

- $0! = 1$
- $n! = n * (n-1)!$

En Python :

```
def fact (n) :
    if n == 0:
        return 1
    else:
        return n * fact (n-1)
```

Exemple :

fact(3) = 3*fact(2) = 3*2*fact(1) = 3*2*1*fact(0)= 3*2*1*1= 6

Seconde version : Récursivité terminale

En Python :

```
def facto (n,res) :
    if n == 0:
        return res
    else:
        return facto (n-1, res*n)
def fact (n):
    return facto(n,1)
```

Seconde version : Récursivité terminale

En Python :

```
def facto (n,res) :
    if n == 0:
        return res
    else:
        return facto (n-1, res*n)
def fact (n):
    return facto(n,1)
```

Exemple : fact(3) =

facto(3,1) = facto(2,3*1)= facto(1,2*3) = facto(0,6*1) = 6

Plan

- L'état
- Type
- Nommage
- L'affectation
- Égalité
- Procédure et fonction
- Variable globale vs locale
- Récursivité
- Mémoire et complexité
- Passage des variables
- Ordinateur Quantique
- Conclusion

Notion de coût

Échanger a et b ?

tmp $\leftarrow$ a
a $\leftarrow$ b
b $\leftarrow$ tmp

- Coût en temps :

Notion de coût

Échanger a et b ?

tmp $\leftarrow$ a
a $\leftarrow$ b
b $\leftarrow$ tmp

- Coût en temps : 3 affectations
- Coût en espace mémoire :

Notion de coût

Échanger a et b ?

tmp $\leftarrow$ a
a $\leftarrow$ b
b $\leftarrow$ tmp

- Coût en temps : 3 affectations
- Coût en espace mémoire : 3 entiers

Échangeons

Échanger a et b en coût en espace mémoire $= 2$ (entiers) ?

Échangeons

Échanger a et b en coût en espace mémoire = 2 (entiers) ?

a $\leftarrow$ a + b
b $\leftarrow$ a - b
a $\leftarrow$ a - b

Échangeons

Échanger a et b en coût en espace mémoire = 2 (entiers) ?

	a	b
$a \leftarrow a + b$	$a_0 + b_0$	b_0
$b \leftarrow a - b$	$a_0 + b_0$	$(a_0 + b_0) - b_0 = a_0$
$a \leftarrow a - b$	$(a_0 + b_0) - a_0 = b_0$	a_0

Exercice :

Que fait cet algorithme ?

$$b \leftarrow a + b$$
$$a \leftarrow a + b$$
$$b \leftarrow a - b$$
$$a \leftarrow a - 2b$$

Exercice :

Que fait cet algorithme ?

$b \leftarrow a + b$
$a \leftarrow a + b$
$b \leftarrow a - b$
$a \leftarrow a - 2b$

	a	b
$b \leftarrow a + b$	a_0	a_0+b_0
$a \leftarrow a + b$	$2a_0+b_0$	a_0+b_0
$b \leftarrow a - b$	$2a_0+b_0$	a_0
$a \leftarrow a - 2b$	b_0	a_0

Somme des entiers de 1 à n

Version « naïve »

s=0
Pour i de 1 à n :
    s = s + i

Somme des entiers de 1 à n

Version « naïve »

```
s=0
Pour i de 1 à n :
        s = s + i
```

Complexité : n additions

Somme des entiers de 1 à n

Version « naïve »

```
s=0
Pour i de 1 à n :
        s = s + i
```

Complexité : n additions

Version « Gauss »

```
s = n*(n+1)/2
```

Somme des entiers de 1 à n

Version « naïve »

```
s=0
Pour i de 1 à n :
        s = s + i
```

Complexité : n additions

Version « Gauss »

```
s = n*(n+1)/2
```

Complexité : 2 multiplications + 1 addition

Que fait cet algorithme ?

```
Lire(x,y) ; # x et y sont des entiers >0
j ← 1
i ← 1
Tantque (i<=y) :
        j ← j * x
        i ← i + 1
Ecrire(j) ;
```

Que fait cet algorithme ?

```
Lire(x,y) ; # x et y sont des entiers >0
j ← 1
i ← 1
Tantque (i<=y) :
    j ← j * x
    i ← i + 1
Ecrire(j) ;
```

Cet algorithme calcule x^y

Que fait cet algorithme ?

```
Lire(x,y) ; # x et y sont des entiers >0
j ← 1
i ← 1
Tantque (i<=y) :
    j ← j * x
    i ← i + 1
Ecrire(j) ;
```

Cet algorithme calcule x^y

Complexité

- Coût en nombre d'opérations sur les entiers $y * 2$
- Coût en espace mémoire $= 4$ (entiers)

Que fait cet algorithme ?

```
Lire(x,y) ; # x et y sont des entiers >0
j ← 1
i ← 1
Tantque (i<=y) :
    j ← j * x
    i ← i + 1
Ecrire(j) ;
```

Cet algorithme calcule x^y

Complexité

- Coût en nombre d'opérations sur les entiers $y * 2$
- Coût en espace mémoire $= 4$ (entiers)

Peut-on faire mieux ?

Puissance rapide

Idée mathématique

$$\text{puis}(x,\, n) = \begin{cases} x, & \text{si } n = 1 \\ \text{puis}(x^2,\, n/2), & \text{si } n \text{ est pair} \\ x \times \text{puis}(x^2,\, (n-1)/2), & \text{si } n > 2 \text{ est impair} \end{cases}$$

Puissance rapide

Idée mathématique

$$\text{puis}(x,\, n) = \begin{cases} x, & \text{si } n = 1 \\ \text{puis}(x^2,\, n/2), & \text{si } n \text{ est pair} \\ x \times \text{puis}(x^2,\, (n-1)/2), & \text{si } n > 2 \text{ est impair} \end{cases}$$

```
def puisrapide(x,n):
    if (n==1) :
        return x
    else :
        if (n%2==0) :
            return puisrapide(x*x,n/2)
        else :
            return x * puisrapide(x*x, (n-1)/2)
```

Complexité logarithmique $O(log_2(n))$.

Puissance rapide

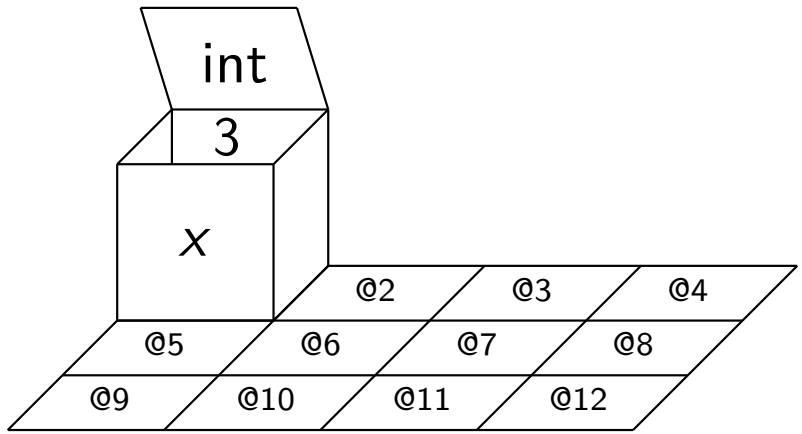
Version itérative

```
def puissrapide(x,n):
        res = 1
        while n != 0 :
            if (n%2 ==0) :
                x = x*x
                n = n/2
            else :
                res = res*x
                n= n-1
        return res
```

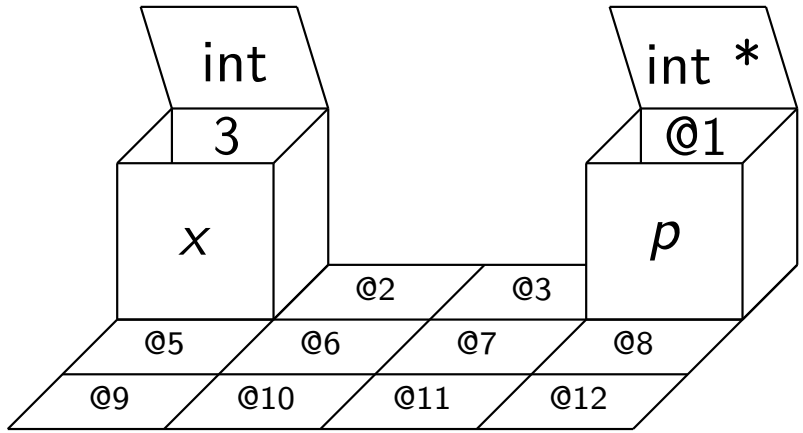
Plan

- L'état
- Type
- Nommage
- L'affectation
- Égalité
- Procédure et fonction
- Variable globale vs locale
- Récursivité
- Mémoire et complexité
- Passage des variables
- Ordinateur Quantique
- Conclusion

Variables en mémoire



Références



En Python

listes

```
a=[1,2,3]
b=a
b.append(4)
print a
```

En Python

listes

```
a=[1,2,3]
b=a
b.append(4)
print a
[1,2,3,4]
```

En Python

listes

```
a=[1,2,3]
b=a
b.append(4)
print a
[1,2,3,4]
```

- mutable : listes, dictionnaires, sets ...
- non mutable : strings, int, floats, tuples ...

Quizz

Fonctions

```
def f(l):
    l.append(4)
    return 5

liste=[1,2,3]
y=f(liste)
print liste
```

Quizz

Fonctions

```
def f(l):
    l.append(4)
    return 5

liste=[1,2,3]
y=f(liste)
print liste
[1,2,3,4]
```

En C

```
int i, j;
i=3;
j=i;
```

objet	adresse	valeur
i	456000	3
j	456004	3

Pointeur en C

```
int i =3;
int *p;
p=&i;
print("*p = %d \n", *p) \\ affiche 3
*p=0;
print(i = %d \n", i) \\ affiche 0
```

objet	adresse	valeur
i	456000	3
p	456004	456000
*p	456000	3

en Ocaml

Référence

```
let x = ref 0;;
val x : int ref = {contents = 0}
x := 1;;
x;;
- : int ref = {contents = 1}
```

Lazy, name or value Evaluation

Fonctions

```
let f x = 1 ; ;
let h x = x+x ; ;
let r = ref 0 ; ;
let g x = r := !r+1 ;x ; ;
```

f g 0 vs h g 0

	f (g 0)	h (g 0)
Value		
Name		
Lazy		

Lazy, name or value Evaluation

Fonctions

```
let f x = 1 ; ;
let h x = x+x ; ;
let r = ref 0 ; ;
let g x = r := !r+1 ;x ; ;
```

f g 0 vs h g 0

	f (g 0)	h (g 0)
Value	f(0) =1 et r=1	
Name		
Lazy		

Lazy, name or value Evaluation

Fonctions

```
let f x = 1 ; ;
let h x = x+x ; ;
let r = ref 0 ; ;
let g x = r := !r+1 ;x ; ;
```

f g 0 vs h g 0

	f (g 0)	h (g 0)
Value	f(0) =1 et r=1	h(0) = 0 et r=1
Name		
Lazy		

Lazy, name or value Evaluation

Fonctions

```
let f x = 1 ; ;
let h x = x+x ; ;
let r = ref 0 ; ;
let g x = r := !r+1 ;x ; ;
```

f g 0 vs h g 0

	f (g 0)	h (g 0)
Value	f(0) =1 et r=1	h(0) = 0 et r=1
Name	f(g 0) = 1 et r=0	
Lazy		

Lazy, name or value Evaluation

Fonctions

```
let f x = 1 ; ;
let h x = x+x ; ;
let r = ref 0 ; ;
let g x = r := !r+1 ;x ; ;
```

f g 0 vs h g 0

	f (g 0)	h (g 0)
Value	f(0) =1 et r=1	h(0) = 0 et r=1
Name	f(g 0) = 1 et r=0	(g 0) + (g 0) = 0+0=0 et r=2
Lazy		

Lazy, name or value Evaluation

Fonctions

```
let f x = 1 ; ;
let h x = x+x ; ;
let r = ref 0 ; ;
let g x = r := !r+1 ;x ; ;
```

f g 0 vs h g 0

	f (g 0)	h (g 0)
Value	f(0) =1 et r=1	h(0) = 0 et r=1
Name	f(g 0) = 1 et r=0	(g 0) + (g 0) = 0+0=0 et r=2
Lazy	f(g 0) = 1 et r =0	

Lazy, name or value Evaluation

Fonctions

```
let f x = 1 ; ;
let h x = x+x ; ;
let r = ref 0 ; ;
let g x = r := !r+1 ;x ; ;
```

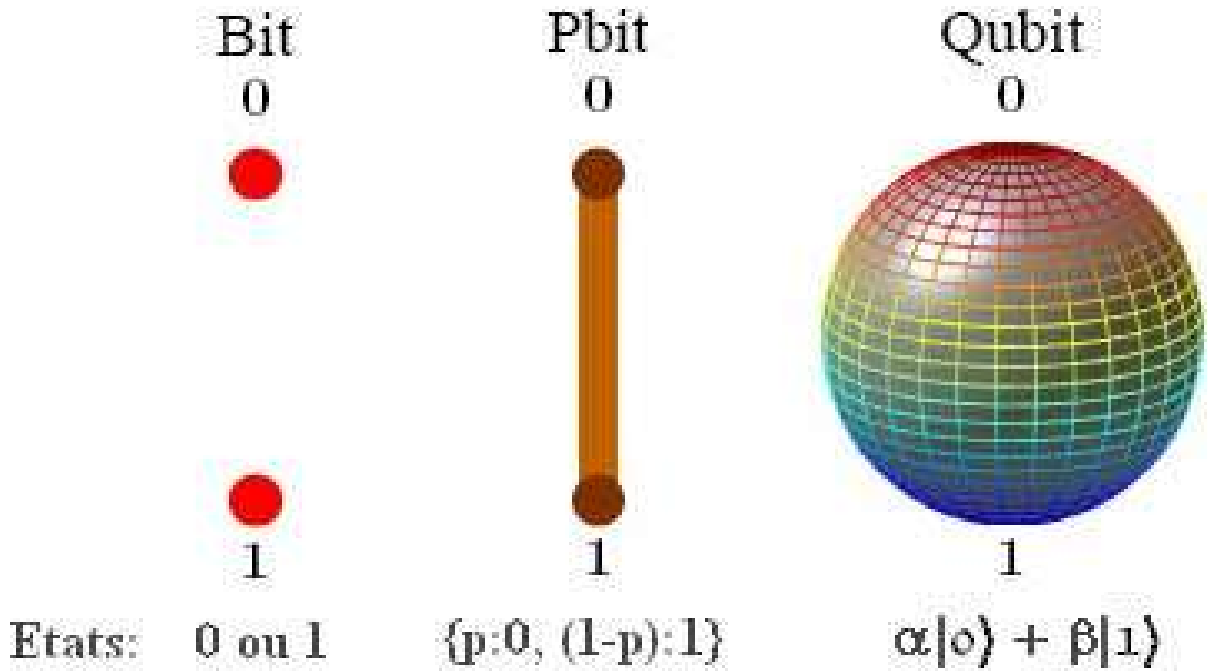
f g 0 vs h g 0

	f (g 0)	h (g 0)
Value	f(0) =1 et r=1	h(0) = 0 et r=1
Name	f(g 0) = 1 et r=0	(g 0) + (g 0) = 0+0=0 et r=2
Lazy	f(g 0) = 1 et r =0	(g 0) + (g 0) = 0+0=0 et r=1

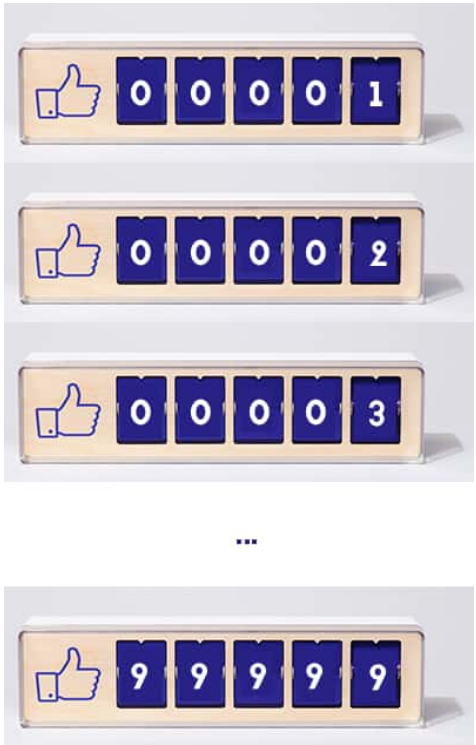
Plan

- L'état
- Type
- Nommage
- L'affectation
- Égalité
- Procédure et fonction
- Variable globale vs locale
- Récursivité
- Mémoire et complexité
- Passage des variables
- Ordinateur Quantique
- Conclusion

Qbits



Changement de paradigme





2^k tests, k boucles imbriquées

k tests

Différence, dans une variable quantique

Il y a possiblement toutes les valeurs possibles en même temps.

Plan

- L'état
- Type
- Nommage
- L'affectation
- Égalité
- Procédure et fonction
- Variable globale vs locale
- Récursivité
- Mémoire et complexité
- Passage des variables
- Ordinateur Quantique
- Conclusion

LIMOS
LABORATOIRE D'INFORMATIQUE,
DE MODÉLISATION ET D'OPTIMISATION DES SYSTÈMES

Conclusion

la notion de variable

- Notion clef en informatique
- État et affectation
- Type
- Liée à la notion de fonction :
  - locale vs globale
  - passage par paramétre ou par valeur