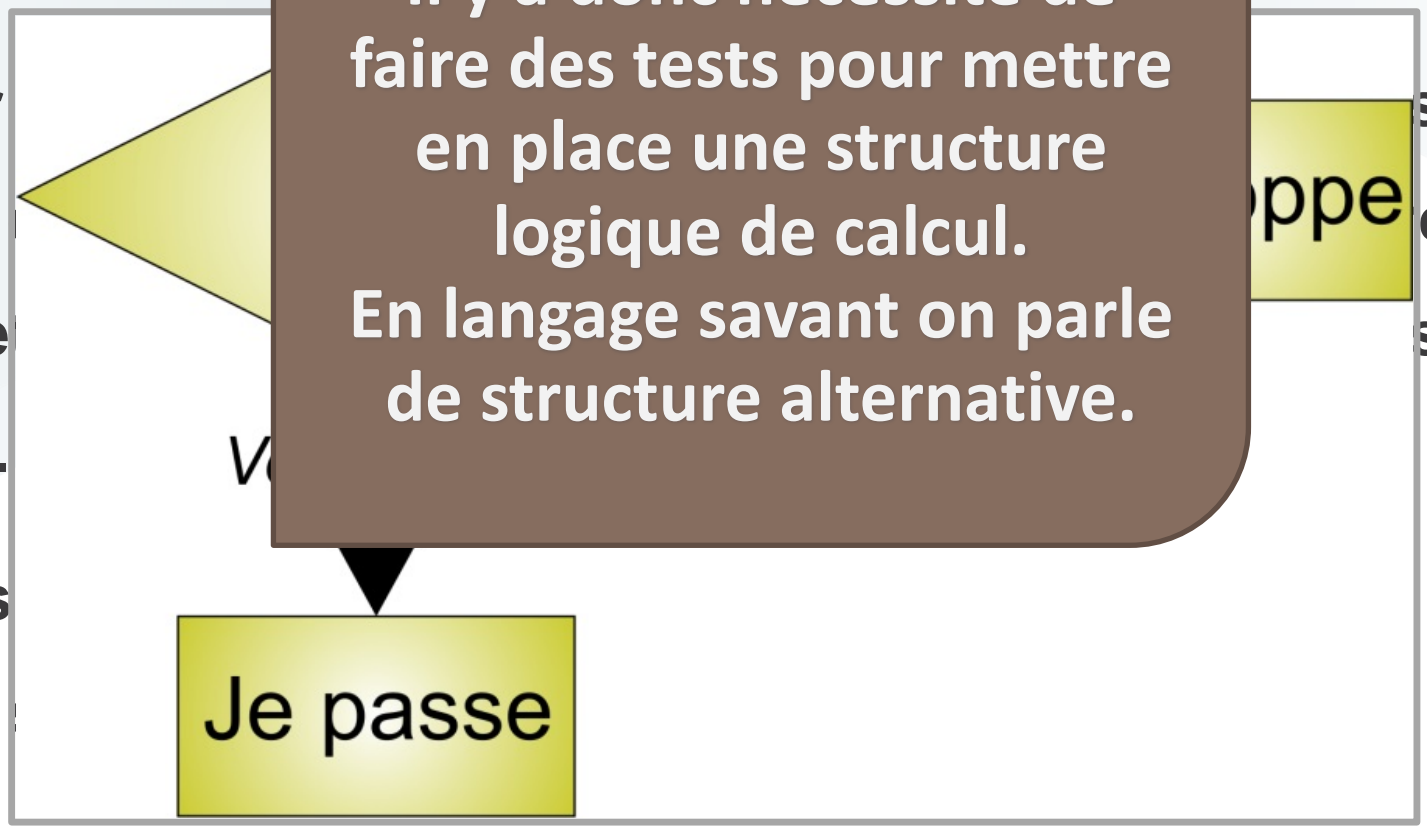
INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES TOULOUSE

Leçon sur les structures alternatives

Université de Toulouse

Où est le problème ?

- Un programme n'est pas qu'une suite unique de calcul.
- Un « être » intelligent à toujours des choix (souvent) binaires à faire pour évolu
- Bien sûr si le feux est ppe e à 20 mètres e s je passe….
- Enfin les ralentis,



Il y a donc nécessité de faire des tests pour mettre en place une structure logique de calcul.
En langage savant on parle de structure alternative.

L'algorithme

- Alternative s

Le cas sinon ne prend jamais de condtion ! **C'est explicitement l'inverse de si**

Alternative complète

```
Si (expression) Alors
   instr
Finsi
```

```
Si (expression) Alors
```

Suite
ser
e

qui ne
ue si
sse

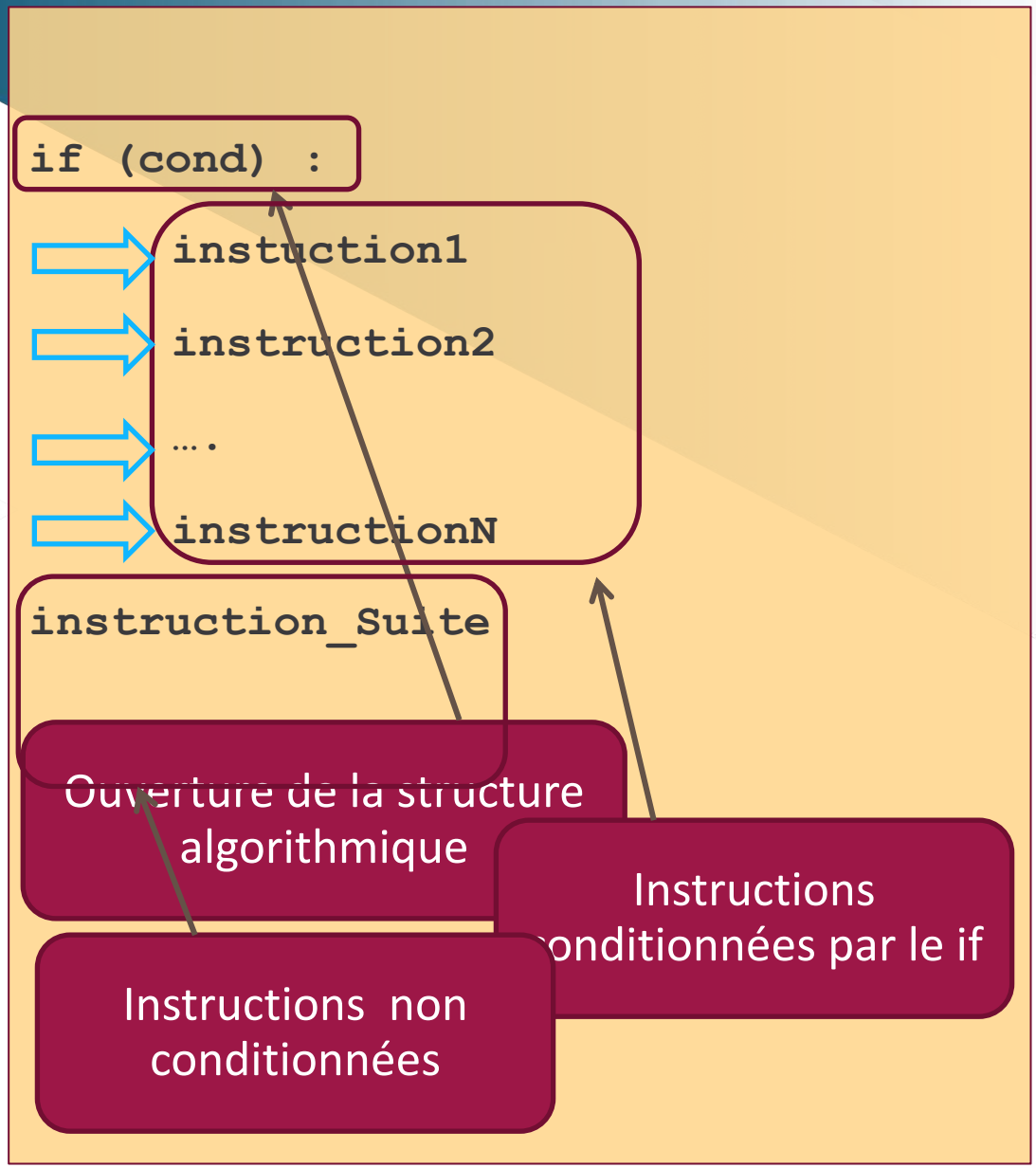
« expression » donne une valeur qui est obligatoirement binaire : VRAIE (1) ou FAUSSE (0)

Cette expression est :

- ☑ *Une variable de type bool*
- ☑ *Une condition explicite*

Codage alternative simple

- **Codage de l'altenative simple**
  - Le if et les : ouvre la structure
  - Les instructions suivantes qui sont indentées appartiennent à la structure
  - La première instruction qui revient au même niveau que le if ferme la structure.



Exemple alternative simple

Dans cette version la variable M n'est créé que si N est multiple de 7…

=
est différent de
==

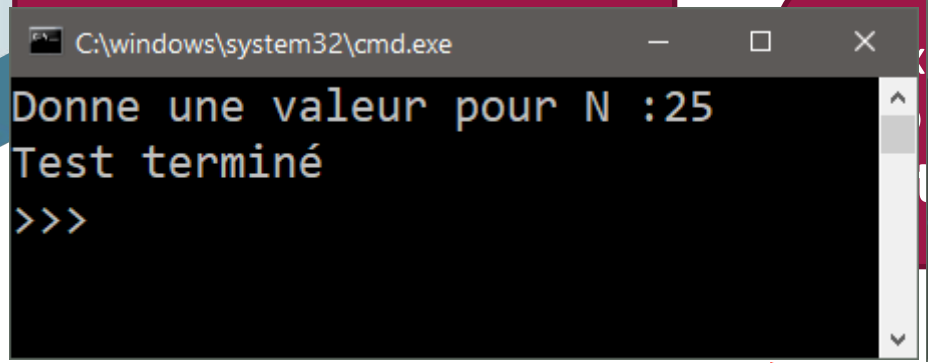
Division entière
% : opérateur qui renvoie le reste
// : opérateur qui renvoie le quotient

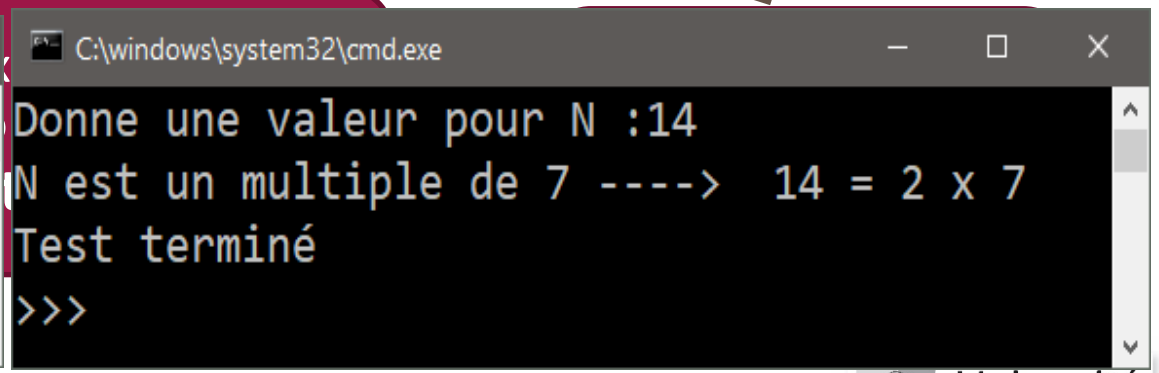
Lire un nombre entier N
Si le nombre est multiple de 7
M = N/7
Afficher(M)
FinSi
Afficher « Test terminé »

```
N = int(input("Donne une valeur pour N :"))
reste = N%7
if (reste == 0 ):
    M = N//7
    print("N est un multiple de 7 ----> ", N,"=",M, "x 7" )

print ("Test terminé")
```

2 instructions





Université de Toulouse

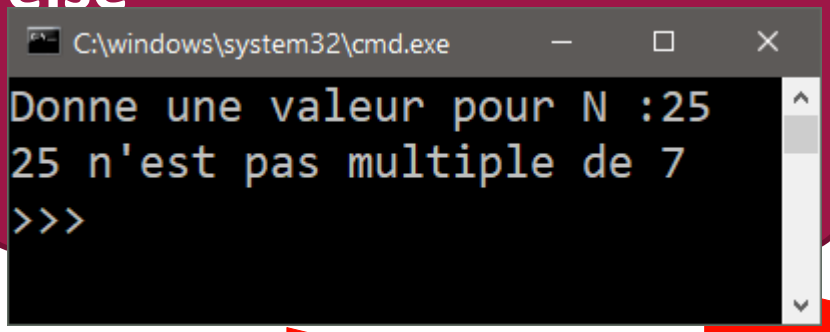
Extension à l'alternative complète

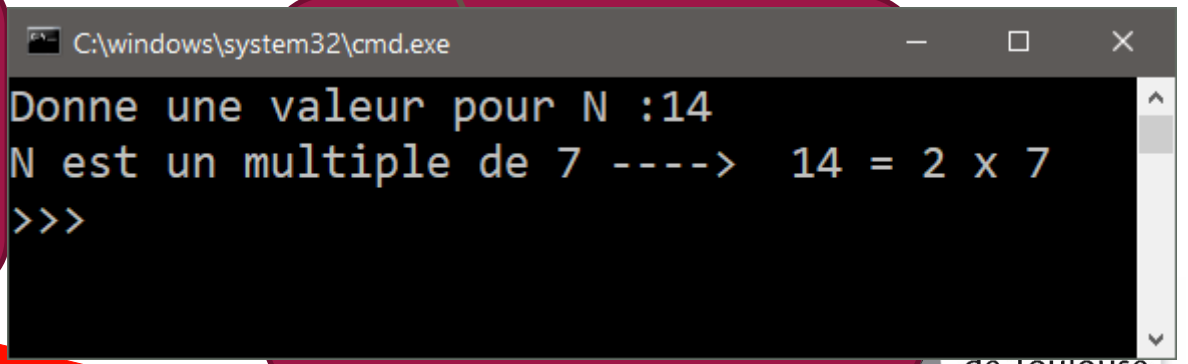
La variable M n'est créé toujours que si N est multiple de 7…

```
Lire un nombre entier N
Si le nombre est multiple de 7
    M = N/7
    Afficher(M)
Sinon
    Afficher ('Non multiple de7)
FinSi
```

```
N = int(input('Donne une valeur pour N :'))
reste = N%7
if (reste == 0 ):
    M = N//7
    print("N est un multiple de 7 ----> ", N,"=",M, "x 7" )
else:
    print(N,'n\'est pas multiple de 7')
```

else





Les tests possibles

- **Tests classiques directs**
  - opérateurs de comparaison
- *égal à :* ==
- *différent de :* !=
- *plus petit que :* <
- *plus grand que :* >
- *plus petit ou égal* <=
- *plus grand ou égal* >=

- **Quelques tests « math »**

  retourne un booléen
- isfinite(A)
  - retourne vrai si A est “fini”
- isinf(A)
  - retourne vrai si A vaut ±inf
- isnan(A)
  - retourne vrai si A vaut Nan
- isclose(A,B, rel_tol) :
  - renvoie vrai si |A-B| <rel_tol

Application directe

```
if (cond):
    instructions_vrai
else:
    instructions_faux
```

Ecrire un programme qui demande à l'utilisateur de saisir une chiffre entier et qui le mette au carré s'il est inférieur à 1 et qui en calcule la racine carrée si il est supérieur ou égal à 1

Evolution

```
if (N < 0) :
    print('Le nombre est négatif')
if (N == 0) :
    print('Le nombre nul')
if (N > 100) :
    print('Le nombre est >100')
if (N > 100) :
    print('Le nombre est >10')
```

```
if (N < 0) :
    print('Le nombre est négatif')
if (N == 0) :
    print('Le nombre nul')
if (N >100) :
    print('Le nombre est >100')
else :
    print('Le nombre est >10')
```

Avec N = 111 ….?

```
Le nombre est >100
Le nombre est >10
```

```
Le nombre est >100
```

Evolution 2

```
if (N < 0) :
    print('Le nombre est négatif')
if (N == 0) :
    print('Le nombre nul')
if (N > 100) :
    print('Le nombre est >100')
else :
    print('Le nombre est >10')
```

Avec N = -1 ….?

```
Le nombre est négatif
Le nombre est >10
```

```
if (N < 0) :
    print('Le nombre est négatif')
else :
    if (N == 0) :
        print('Le nombre nul')
    else :
        if (N > 100) :
            print('Le nombre est >100')
        else :
            print('Le nombre est >10')
```

Pourquoi faire tous les tests si on sait déjà que N < 0 ?

Quelques lignes de listing en plus mais **correcte + gain de temps.**

Evolution 3

```
if (N < 0) :
    print('Le nombre est négatif')
else :
    if (N == 0) :
        print('Le nombre nul')
    else :
        if (N > 10) :
           print('Le nombre est >10')
        else :
           print('Le nombre est >100')
```

```
if (N < 0) :
    print('Le nombre est négatif')
elif(N == 0) :
    print('Le nombre nul')
elif (N > 10) :
    print('Le nombre est >10')
else :
   print('Le nombre est >100')
```

« Contraction » possible du **else …if** en **elif**

Code plus compact et tout aussi rapide

Utilité du test combiné ?

Signe du produit de 2 nombres sans calculer le produit ni avoir la fonction math.sign….

Il y a du combinatoire au sens logique du terme là dedans …

```
flag = 0
Si (X > 0) alors
     Si (Y > 0) alors
          flag = +1
     end
Sinon
     Si (Y  < 0)
          flag = +1
     end
FinSi
Si (flag = 1)
     Ecrire "Leur produit est positif"
Sinon
     Ecrire "Leur produit est négatif"
FinSi
```

flag vaut 1 si :

((X >0 ET Y > 0) OU (X < 0 ET Y < 0)

Cette combinatoire existe dans tout langage de programmation !!

Composition de base

- **Le ET logique**

  si (CondA ET condB) alors

  Instructions_oui

  FinSi

- **Le OU logique**

  si (CondA OU condB) alors

  Instructions_oui

  FinSi

- **Le « not » logique**

  si (non(CondA)) alors

  Instructions_oui

  FinSi

```
if (CondA and CondB):
    instructions_vrai
```

⟺

```
if (CondA) :
    if (CondB):
        instructions_vrai
```

```
if (CondA or CondB):
    instructions_vrai
```

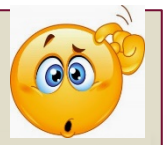
⟺

```
if (CondA) :
    instructions_vrai
if (CondB):
    instructions_vrai
```

```
if (not(CondA)):
    instructions_vrai
```

⟺

```
if (CondA) :
else :
    instructions_vrai
```



Imbrication vs combinaison

```
Ecrire "Entrez la température du patient:"
Lire Temp
Si (Temp <= 36) Alors
        Ecrire "Hypothermie ou mort"
FinSi
Si ((Temp < 37.5) ET (Temp > 36)) Alors
        Ecrire "Tout va bien "
FinSi
Si (Temp >= 37.5) Alors
                Ecrire "Fièvre !!"
Finsi
```

ICI : l'imbrication simplifie la combinaison

Il n'y a jamais ni de solution unique ni de solution systématique.

```
Ecrire "Entrez la température du patient:"
Lire Temp
Si (Temp <= 36) Alors
     Ecrire "Hypothermie ou mort"
Sinon
        Si (Temp < 37.5) Alors
             Ecrire "Tout va bien "
        Sinon
             Ecrire "Fièvre !!"
        Finsi
FinSi
```

Même test !!
=> Simplification possible

L'art de compliquer…

- Il est toujours possibilité de compliquer les choses….

si ((a == 1) ET ((b < 10) OU (c > a)) OU (c ≠ b))

- Il est important d'exprimer CLAIREMENT sa condition…
- …et de savoir le traduire (test simple ou combinaison)

```
si Wof ∈ [a,b]
⇔
si ((Wof >= a) ET (Wof <= b))
```

```
si Wof ∉[a,b[
⇔
si ((Wof < a) OU (Wof >= b))
```

Perd pas la Bool !

L'algèbre de Bool peut aider à formuler des conditions équivalentes…et simplifier certaines expressions :

$$\overline{A.B} = \overline{A} + \overline{B}$$
$$A.B = \overline{\overline{A} + \overline{B}}$$
$$\overline{A + B} = \overline{A}.\overline{B}$$
$$A + B = \overline{\overline{A}.\overline{B}}$$

Choisir son camp….

Si (note < 10 ET (cheque < 1000)) alors
Ecrire « Etudiant recalé »
⇔
Si (note < 10 ET Pas(cheque ≥ 1000)) alors
Ecrire « Etudiant recalé »
⇔
Si (Pas(note ≥10) ET (cheque < 1000)) alors
Ecrire « Etudiant recalé »
⇔
Si Pas((note ≥10 OU (cheque ≥ 1000)) alors
Ecrire « Etudiant recalé »
⇔ …

Il n'y a pas de « meilleure » solution .
Elles sont toutes équivalentes

A faire en fonction de votre façon de raisonner.