INFO-H-100 - Informatique

Séance d'exercices 7
Calcul itératif et numérique

Université libre de Bruxelles
École polytechnique de Bruxelles

2013-2014

Rappel de base - Composition de tests

Il y a trois opérateurs logiques : `and` (et), `or` (ou) et `not` (non).

On construit des expressions booléennes en utilisant les opérateurs de comparaison et logiques.

```
>>> x = 5
>>> 0 < x and x < 10
True
>>> x % 2 == 0 or x % 3 == 0
False
>>> not x > 10
True
```

Rappel de base - Algèbre booléenne

- `a and b` est vrai si et seulement si `a` est vrai et `b` est vrai
- `a or b` est faux si et seulement si `a` est faux et `b` est faux

a	b	not b	a and b	a or b
True	True	False	True	True
True	False	True	False	True
False	True	False	False	True
False	False	True	False	False

Loi de De Morgan :

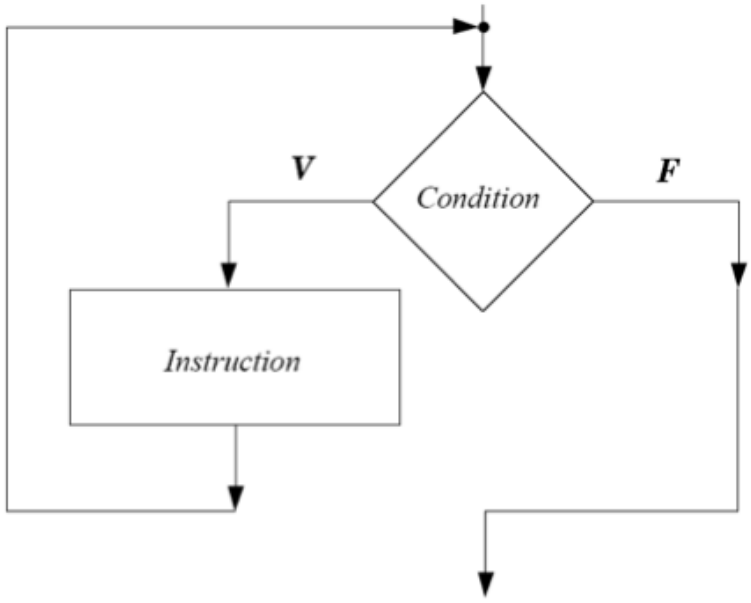
- `not(a and b)` est équivalent à `(not a) or (not b)`
- `not(a or b)` est équivalent à `(not a) and (not b)`

Exemple d'expressions équivalentes :

```
not(0 >= x or x >= 1000)
0 < x and x < 1000          #plus lisible
```

Rappel de base - Boucle `while`

La structure itérative `while` permet de répéter un bloc d'instructions tant qu'une condition est vérifiée.



1. La condition est évaluée.
2. Si la condition est vérifiée, on exécute les instructions et on retourne au point 1.

   Si la condition n'est pas vérifiée on "sort" de la boucle.

Condition de terminaison et De Morgan

```
def find1(seq, val):
    """
    Renvoie l'indice de la premiere apparition de val dans
    la sequence seq, ou -1 si le caractere n'apparait pas
    """
    pos = 0
    while pos < len(seq) and seq[pos] != val:
        pos += 1
    # Assertion: Ici, la condition est fausse!
    # Donc pos >= len(seq) OU seq[pos] == val
    if pos < len(seq):
        # Ici pos < len(seq) ET (donc) seq[pos] == val -> Trouve!
        result = pos
    else:
        result = -1
    return result
```

```
>>> find1("bonjour", "j")
3
>>> find1([4, 3, 1, 6, 8], 9)
-1
```

Calcul itératif : exemple

Principe : Utiliser, lors d'une itération, les résultats de l'itération précédente

Le nombre e peut être défini comme la limite d'une série :

$$e = 1 + \frac{1}{1!} + \frac{1}{2!} + \frac{1}{3!} + \ldots$$

On désire arrêter les calculs quand le dernier terme ajouté est inférieur à un epsilon donné ou quand un certain nombre de termes donné a été pris en compte.

```
EPS = 0.000001
NB_TERMS = 1000

def e():
    n = 1
    term = 1.0
    res_sum = 1
    while term >= EPS and n < NB_TERMS:
        term = term/n
        res_sum += term
        n += 1
    return res_sum
```

Exercices

1, 3, 4, 6, 7, 9, 11