Ch1. Introduction

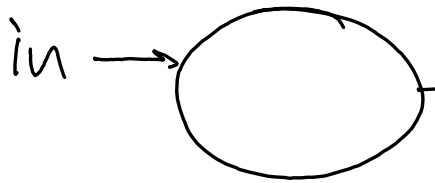
Prog. + { Algorithmique
{ Structures de données

- Tris
- Structures linéaires : listes (piles, files)
- Mots : recherche de facteurs
- Arbres

1. Notion d'algorithme

Church Turing

Def : suite ordonnée d'actions à effectuer pour passer de données à un résultat.



contrôle

test
boucle
sous-programme

jeux d'instructions.

Recettes de cuisine

- jeu d'instructions

expert
- "faire un roux"
- blondir les oignons
- monter des blancs

novice
- mettre la poêle sur le feu moyen.
-

def "faire un roux" :

{
—
—
—
—

sort()

import nltk

2. Pseudo-code

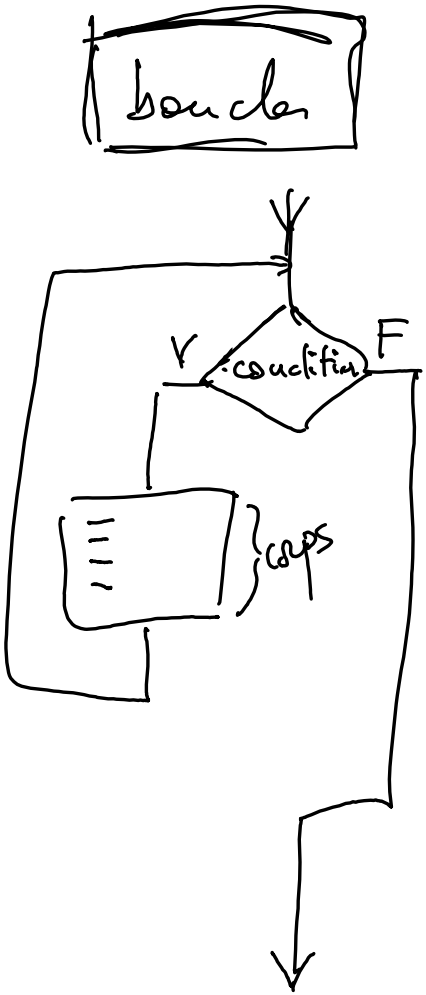
```
def toto():
```

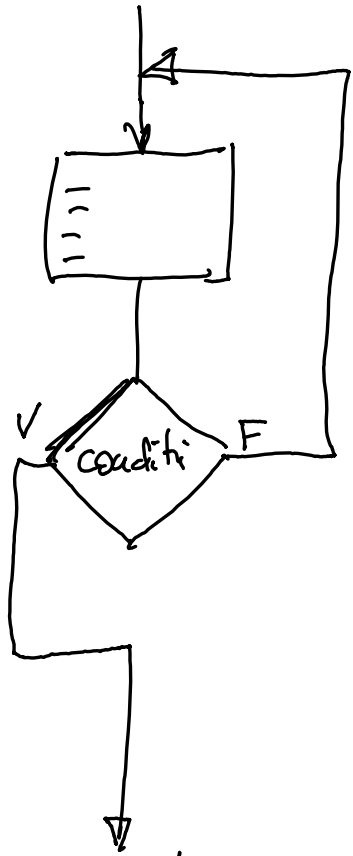
```
void toto()
{
```

- objets : constantes / variables / expressions
- structures de données :
  listes - dictionnaires - strings
- instructions élémentaires
  - affectation

```
x = 7
x = x + 1
```

  - entrées / sorties :
    input() - print()
- structures de contrôle
  - test
  - boucles
  - fonctions





while / for

"test en tête"

repeat until

test en queue

3 Qualité d'un algorithme

3.1 Qualité d'écriture

Lisibilité

- commentaires
- identificateurs
  - noms parlants
  - convention.
- mise en forme
- choix des structures de contrôle. for / while

Modularité

- tests unitaires.

3.2 Validité

"preuve de programme" -

banc d'essai. benchmark

test de non regression

3.3 Performance

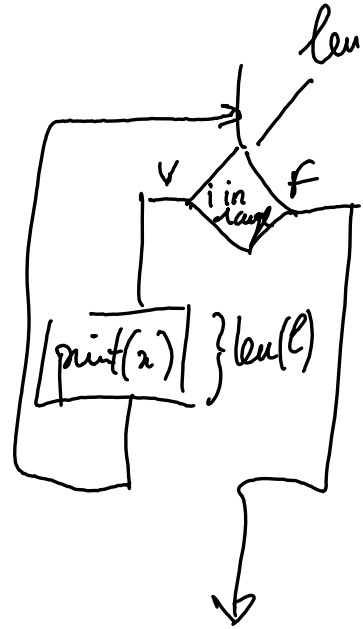
coût
- en mémoire
- en temps ←

Complexité : nombre d'opérations

```
for i in range(len(l)):
    print(x)
```

$len(l) = n$

$print(x) : \alpha$



coût : $n \times \alpha + (n+1)\beta$

$= n(\alpha+\beta) + \beta$

linéaire