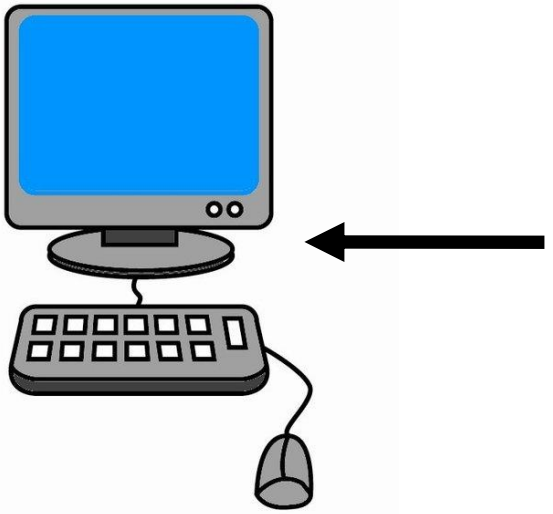
Introduction à la programmation

Patrick Fuchs (Pierre Poulain)

DU Bioinformatique intégrative 2019

Programmation

Définition



```
allAtom=Rigidbody(atomicName)
sys.stderr.write("Load atomic file %s with %d atoms \n" %(atomicName, allAtom.Size()))

#extract all 'atoms' objects
atomList=[]
for i in xrange(allAtom.Size()):
        atom = allAtom.CopyAtom(i)
        # look for residue or base type conversion
        resName = atom.GetResidType()
        if resName in resConv.keys():
                atom.SetResidType( resConv[resName] )
        # look for atom type conversion
        atomTag = atom.GetResidType() + '-' + atom.GetType()
        if atomTag in atomConv.keys():
                atomName = atomConv[atomTag].split('-')[1]
                atom.SetType( atomName )
        atomList.append(atom)

#count residues
residueTagList=[]
coarseResList=[]
for atom in atomList:
        resName = atom.GetResidType()
        # create a unique identifier for every residue
        # resTag is for instance "LEU-296-A"
        resTag = resName + '-'+ str(atom.GetResidId()) + '-' + atom.GetChainId()
        if resTag not in residueTagList:
                if resBeadAtomModel.has_key(resName):
                        residueTagList.append(resTag)
                        # add a pattern residue to the list of coarse residues for the protein
                        # beware of the hugly list copy: use copy.deepcopy() !
                        coarseResList.append(copy.deepcopy(resBeadAtomModel[resName]))
                else:
                        sys.stderr.write("WARNING: residue %s is unknown the residues <-> beads <-> atom
                        sys.stderr.write("        : residue %s will not be reduced into coarse grain\n" %
sys.stderr.write("Number of residues: %i\n" %(len(residueTagList)))
```

Donner des ordres

But

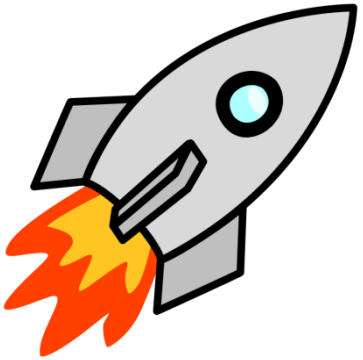
- stocker de l’information
- manipuler / extraire de l’information
- répéter des tâches
- faire des calculs

etc

L’ordinateur

Qualités

Défauts



Très rapide

Beaucoup de mémoire

Mais bête !

« Fondamentalement, l'ordinateur et l'homme sont les deux opposés les plus intégraux qui existent. L'homme est lent, peu rigoureux et très intuitif. L'ordinateur est super rapide, très rigoureux et complètement con... »

Gérard Berry, informaticien, médaille d'or du CNRS 2014, professeur au collège de France
Le Nouvel Observateur, 26/08/2016

https://www.nouvelobs.com/rue89/rue89-le-grand-entretien/20160826.RUE7684/gerard-berry-l-ordinateur-est-completement-con.html

Notion d’algorithme

Wikipedia: Un **algorithme** est une suite finie et non ambiguë d’opérations ou d'instructions permettant de résoudre un problème.

Recette de cuisine: Comment faire un gâteau au yaourt ?

Gâteau au yaourt ?

français

- Mélanger jaunes d'oeufs et sucre jusqu'à obtenir une pâte blanche
- Ajouter farine et bien remuer
- Ajouter yaourt et bien remuer
- Ajouter un peu d'huile si vous l'aimez moelleux
- Monter les blancs en neige et incorporez les à la pâte
- Préchauffez le four à 180°C et cuire la pâte à 180°C pendant 30'

algorithme

```
pate <- pate + jaunes + sucre
Tant que pate n'est pas blanche:
    mélanger()
pate <- pate + farine
mélanger()
pate <- pate + yaourt
mélanger()
Si je veux un gateau moelleux:
    pate <- pate + huile
    mélanger()
monter blancs en neige()
pate <- pate + blanc
préchauffer le four(180°)
cuire la pate(30', 180°)
```

L'ordre des opérations est important !

Autre exemple

Combien ai-je d’alanine dans cette séquence ?

GWGAWILAGAGA

Algorithme humain

GWGAWILAGAGA

1 2 3 4

Pour chaque acide aminé de la séquence,

si l'acide aminé est A

alors on compte une alanine de plus.

Algorithme humain

Pour chaque acide aminé de la séquence,

si l'acide aminé est A

alors on compte une alanine de plus.

Variables

Données

Tests / Boucles (actions)

Opérations

➔ Retrouvé dans tous les langages de programmation !

Algorithme Python

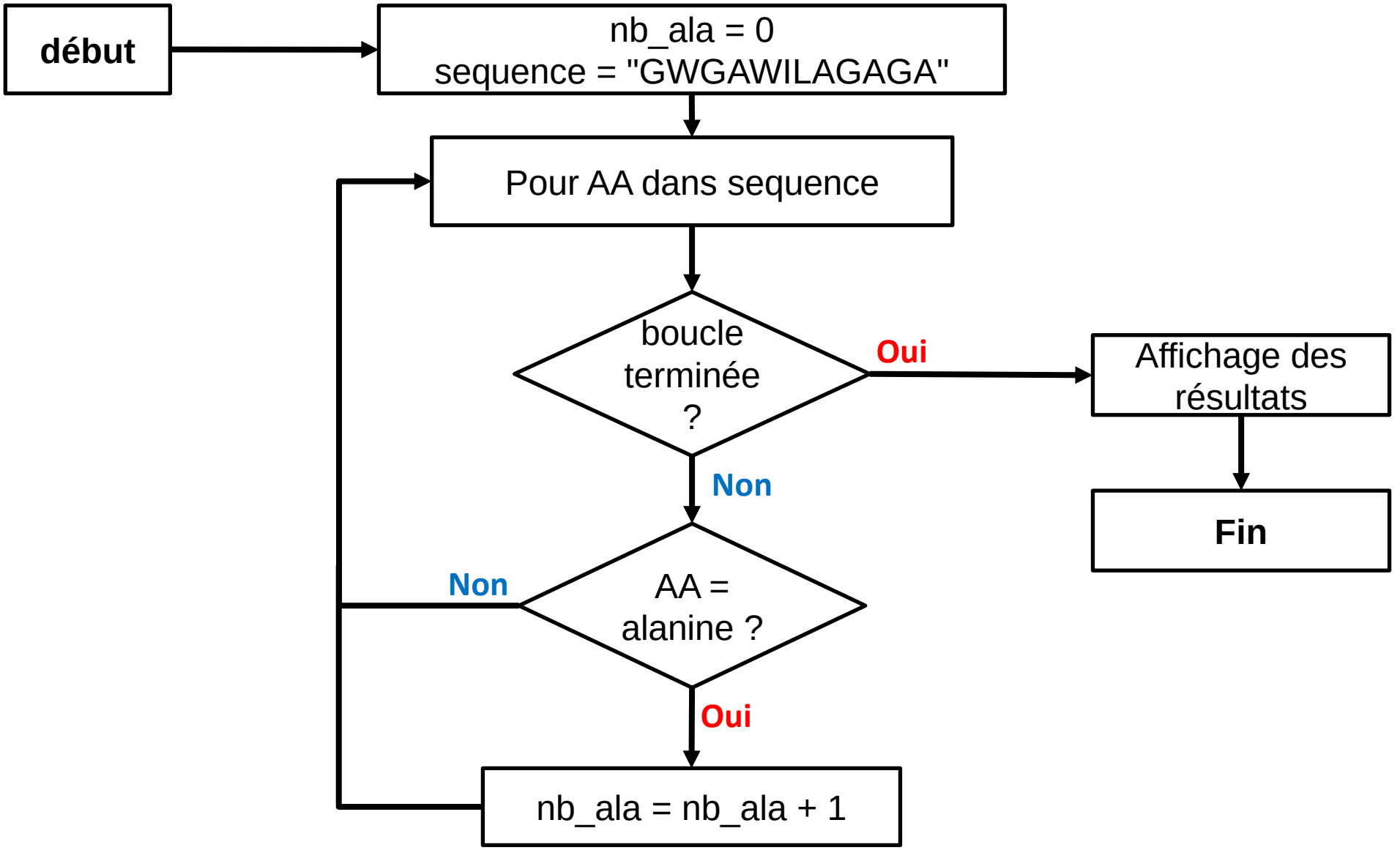
```
sequence = "GWGAWILAGAGA"
nb_ala = 0

for acide_amine in sequence:
    if acide_amine == "A":
        nb_ala = nb_ala + 1


print(nombre_ala)
```

1) initialisation

2) algorithme principal

3) affichage résultats

Organigramme de programmation



Nombre premiers ?

Problème: J'ai un nombre N = 7, comment déterminer s'il est premier ?

français

Un nombre premier n'est divisible que par lui-même et par 1.

On se promène sur tout les nombres de 1 à N=7. A chaque nombre, si celui-ci est un diviseur de N alors je compte un diviseur de plus.

A la fin, si j'ai 2 diviseurs alors N est premier.

algorithme

```
N <- 7 # premier ?

nb_div <- 0
Pour div de 1 à N:
    Si div est un diviseur de N:
        nb_div <- nb_div + 1

Si nb_div est égal à 2:
    Afficher(N est premier)
```

Recherche de gènes

J’ai un fichier genbank, comment récupérer le nombre de gènes sur le brin direct et sur le brin complémentaire?

```
[...]
                     /db_xref="SGD:S000121252"
     gene            complement(<1807..>2169)
                     /gene="PAU8"
                     /locus_tag="YAL068C"
[...]
                     KPAISSALSKDGIYTIAN"
     gene            <2480..>2707
                     /locus_tag="YAL067W-A"
[...]
```

➔ NC_001133.gbk (Saccharomyces cerevisiae S288c chromosome I, complete sequence)

Recherche de gènes

```
# initialisation variables
nb_genes_direct <- 0
nb_genes_compl <- 0
# on se ballade sur toutes les lignes
Pour chaque ligne du fichier:
    Si la ligne commence par "     gene                ":
        Si la ligne contient "complement":
            nb_genes_compl <- nb_genes_compl + 1
        Sinon:
            nb_genes_direct <- nb_genes_direct + 1

# Affichage résultats
Afficher(Le nb de gènes directs est nb_genes_direct)
Afficher(Le nb de gènes directs est nb_genes_compl)
```

Apprentissage

L'apprentissage de la programmation va nécessiter :

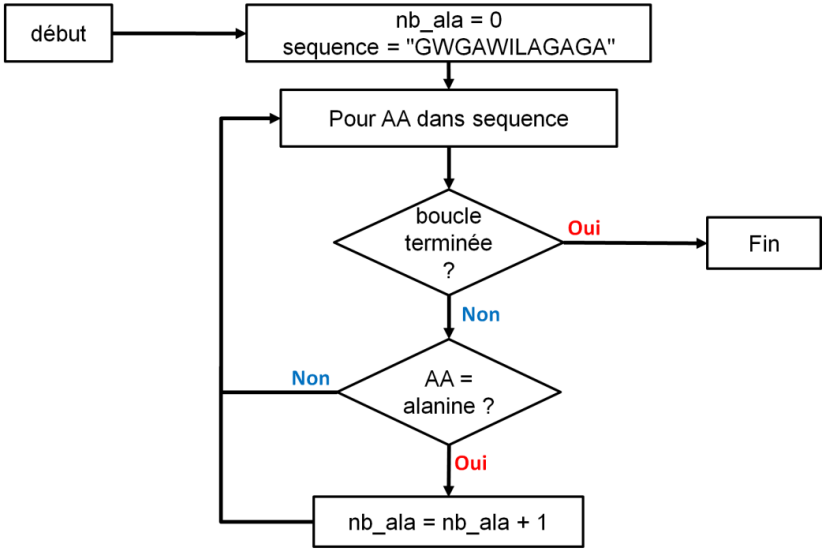
1) Syntaxe du langage Python

```
sequence = "GWGAWILAGAGA"
nombre_ala = 0

for acide_amine in sequence:
    if acide_amine == "A":
            nombre_ala = nombre_ala + 1

print(nombre_ala)
```

➔Connaitre les instructions Python pour effectuer chaque tâche

Sans oublier que l'ordinateur est bête !!!

2) Algorithmie

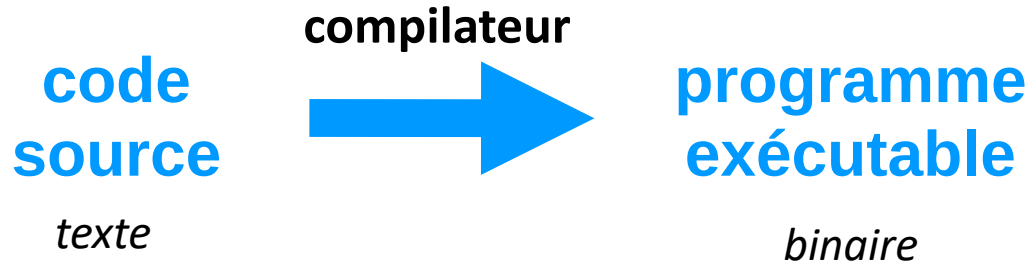


➔ Traduire un problème en algorithme informatique

Différents types de langages

Compilés : C, C++, C#, Fortran

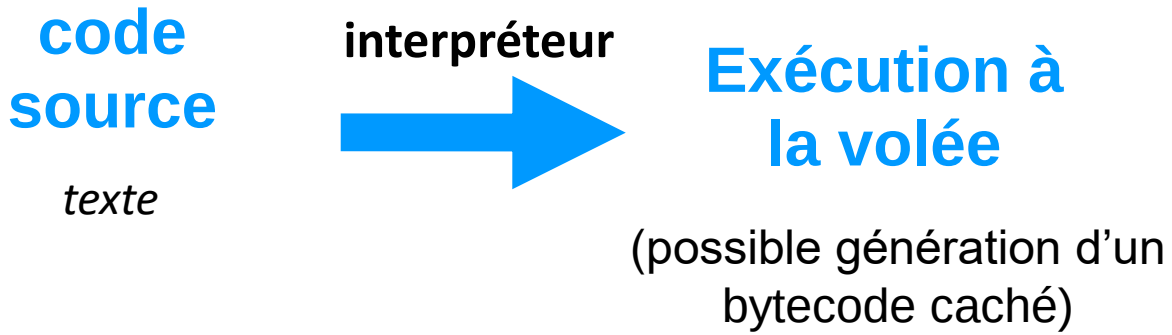
Bas niveau



Rapide, + lourd à développer

Interprétés : Java, Perl, Python (etc)

Haut niveau



Lent, facile à développer

Différents types de langages

Compilés **: C, C++, C#, Fortran**

Bas niveau

code source

texte

compilateur

programme exécutable

binaire

Rapide, + lourd à développer

Interprétés **: Java, Perl, Python**

Haut niveau

code source

texte

interpréteur

Exécution à la volée

(possible génération d'un bytecode caché)

Lent, facile à développer

Python

Python

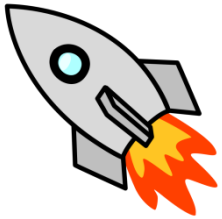
Crédits graphiques

https://www.educol.net

https://en.wikiversity.org/wiki/Python

TurboMilk (Findicons)

Wilsoninc (Findicons)



Nicobunu (Openclipart.org)

VisualPharm (Findicons)