

YSINL2A1 - INTRODUCTION À LA POO EN PYTHON

2023-2024

Semaine 1/10 - 8/1/2023

CM1 - TP1 - TP2

Rappels bases/séquences/fichiers



Ce pictogramme indique un exemple de code sur *ecampus*
dans un fichier commençant par NUMCHAP_NUMSLIDE

Fabrice Maurel - L1 INFO/MATH/MIASH - Semestre 2
fabrice.maurel@unicaen.fr - 02 31 56 73 97 - S3-364

PRÉSENTATION SEMESTRE

- Déroulement
 - 10 semaines (2h CM, 1h TP, 2h TP)
 - 1/3 CC (FIL ROUGE ½ travail, ¼ ccf et ¼ rapport/git)
 - 2/3 CT (QCM - 1h avec 1/3 de questions de cours)
 - BOOKLET évolutif sur ecampus
- Informations TP
 - ressources.zip sur ecampus
 - fichiers_template.py (signatures + asserts)
 - sections pour aller plus loin (en rouge)
- Informations Fil Rouge
 - 3 mini applications autour d'un gestionnaire de grille
 - Structure mise en place dans les premières semaines
 - Extensions par trinôme d'étudiant (cahier des charges fonctionnel)
 - Rendu d'un rapport et d'un dépôt GIT de l'application (forge Unicaen)

- Contenu
 - Semaines 1 à 3 : programmation impérative (non évaluée)
 - Rappels
 - Mise à niveau
 - Suppléments python potentiellement utiles
 - Semaine 4 : transition de l'impératif à l'orienté objet (QCM)
 - Du module de fonctions à la classe / représentation UML
 - Semaines 5 à 7 : concepts de la programmation orientée objet (QCM)
 - Attributs, méthodes, visibilité, Association, Agrégation faible
 - Agrégation forte, Héritage simple, Redéfinition/Surcharge
 - Méthodes spéciales / Héritage multiple / Polymorphisme
 - Semaines 8 à 10 : fil rouge et suppléments POO (QCM - projet)
 - Héritage de classes *built-in* / Classe abstraite / Interface
 - Module tkinter / Spécialisation de widgets / Programmation événementielle
 - Mini-applications / Docstring et Pydoc / Séance d'échanges

LANGAGES D'IMPLÉMENTATION

Hello world!

Assembleur (x86)

```
cseg segment  
assume cs:cseg, ds:cseg  
org 100h  
main proc  
jmp debut  
mess db 'Hello world!\$'  
debut:  
mov dx, offset mess  
mov ah, 9  
int 21h  
ret  
main endp  
cseg ends  
end main
```

Basic originel

```
10 PRINT "Hello world!"  
20 END
```

Shell Unix

```
echo "Hello world!"
```

Langage C

```
#include <stdio.h>  
int main(int argc, char **argv)  
{  
    print("Hello world!\n");  
    return 0;  
}
```

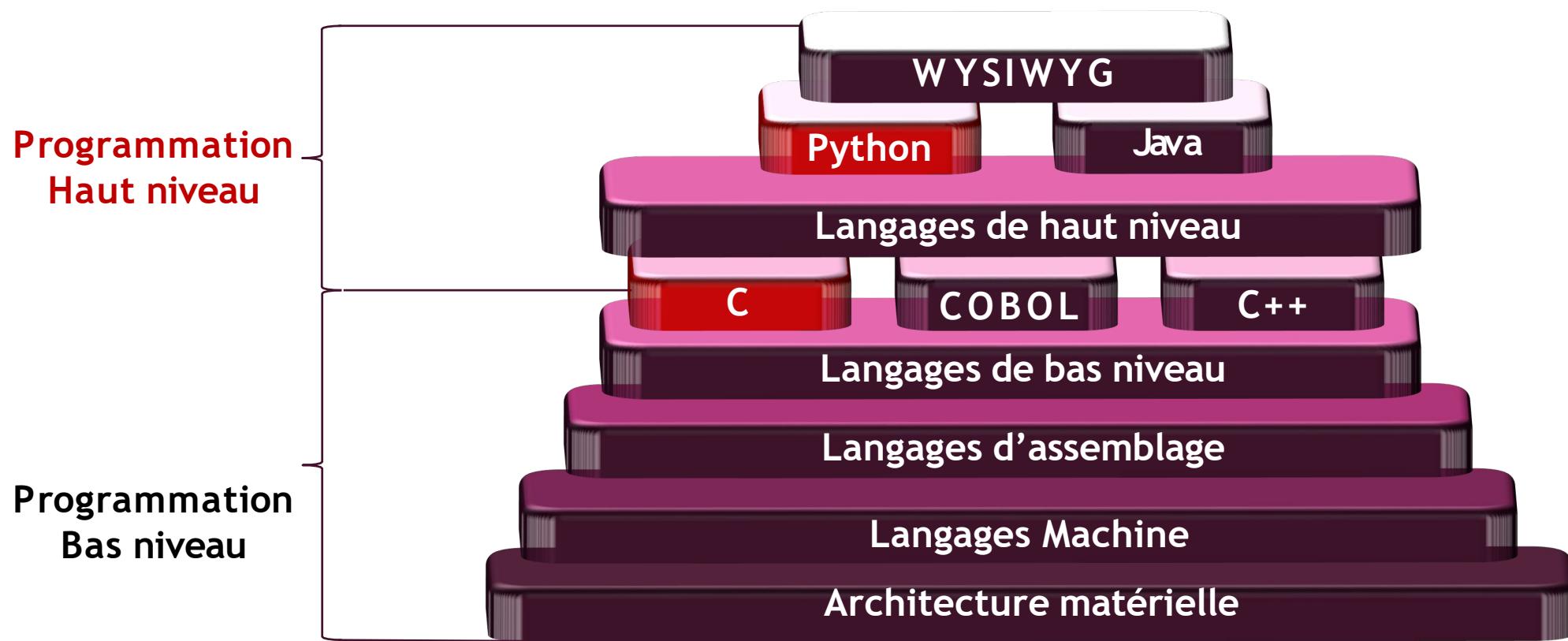
Python 2.x

```
print "Hello world!"
```

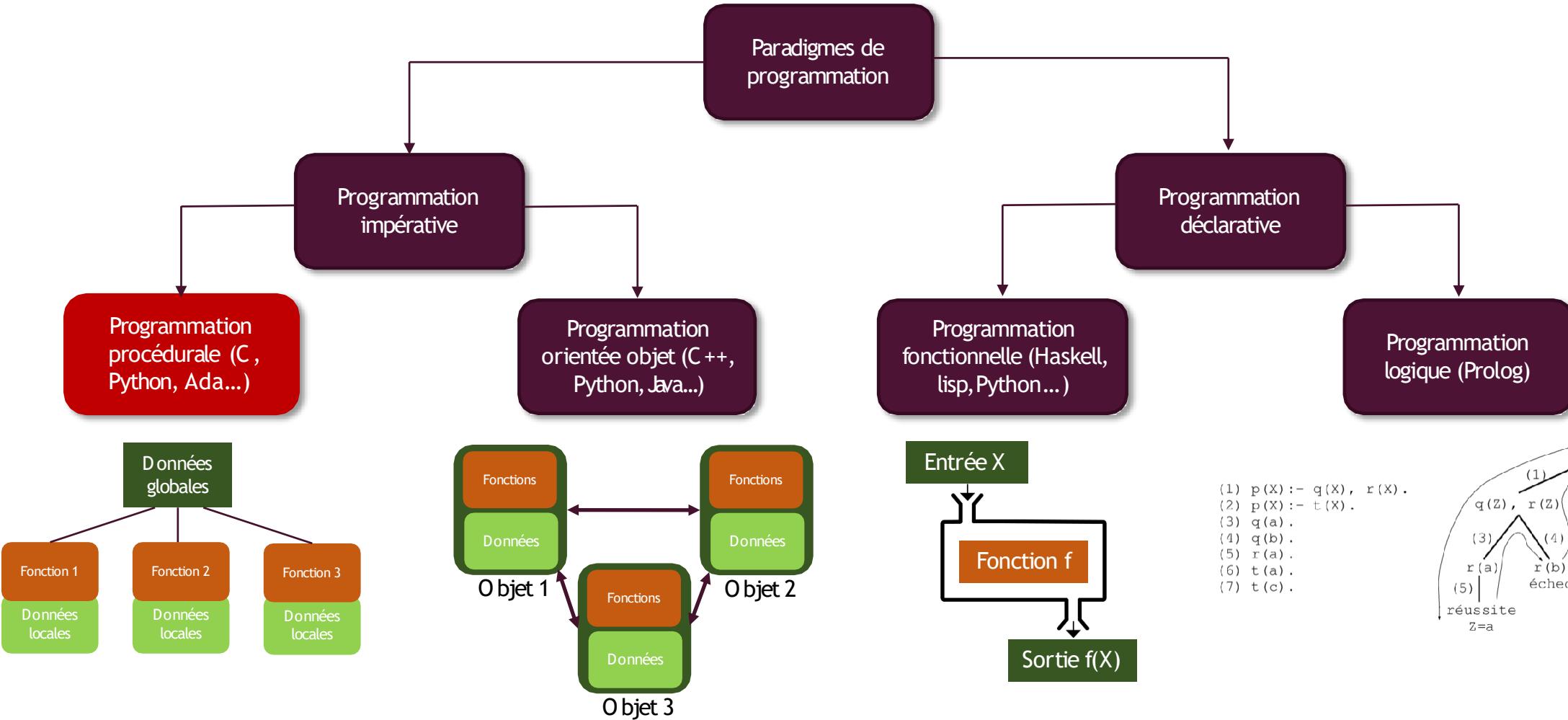
Python 3.x

```
print("Hello world!")
```

PARADIGMES DE LA PROGRAMMATION : NIVEAU



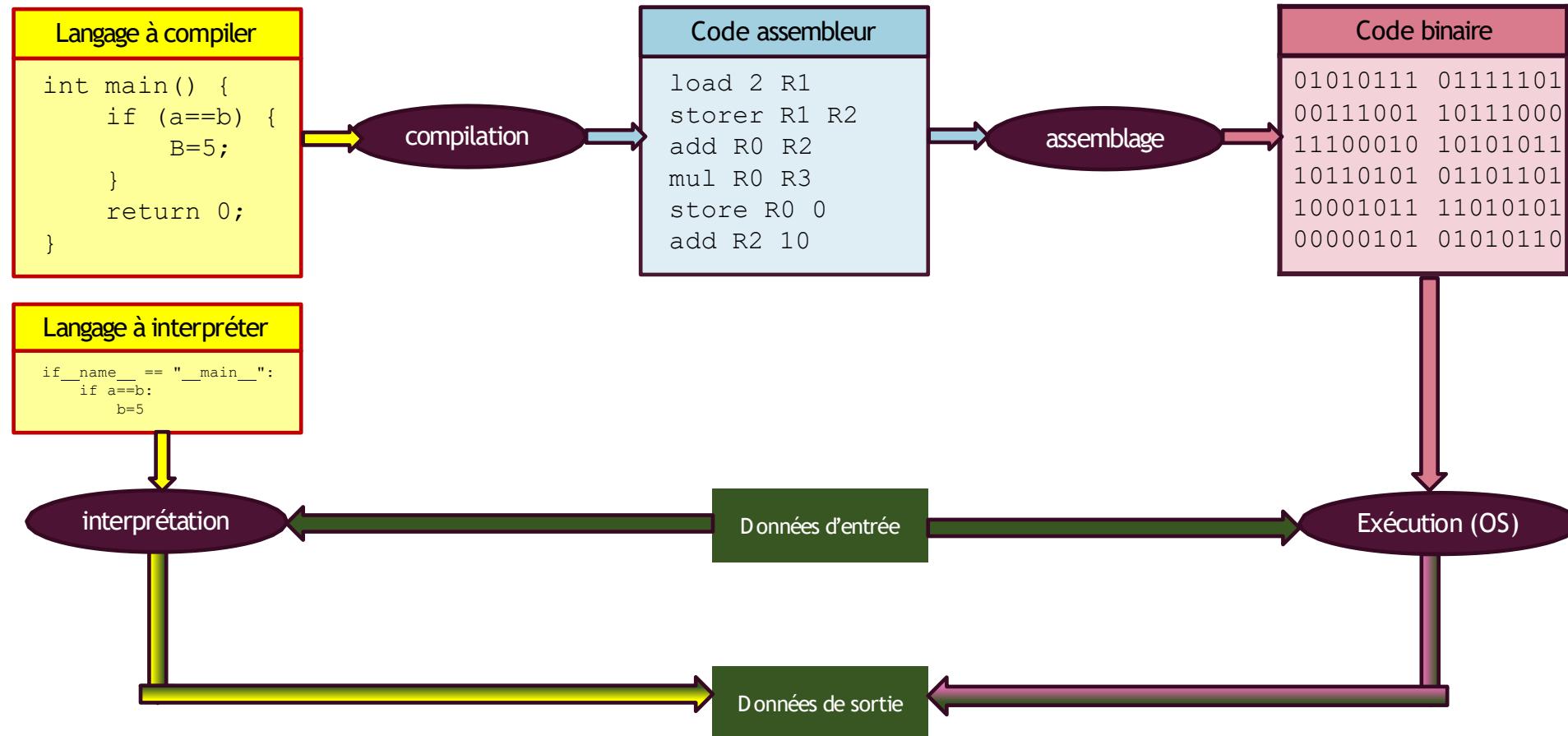
PARADIGMES DE LA PROGRAMMATION : STRUCTURATION



PARADIGMES DE LA PROGRAMMATION : TYPAGE

| Typage | Fort | Faible | Absent |
|-----------|----------------------|------------|--------------|
| Statique | Ada, Java, Pascal | C, C++ | Code binaire |
| Dynamique | Python, Ruby | Javascript | |

PARADIGMES DE LA PROGRAMMATION : COMPILETION VS. INTERPRÉTATION



COMPARAISON DE DEUX LANGAGES (C ET PYTHON)

Factorielle d'un entier naturel n

| En C | En Python |
|---|--|
| <pre>int factorielle(int n) { if (n < 2) {return 1;} else {return n * factorielle(n - 1);} }</pre> | <pre>def factorielle(n): <.....>if n < 2: <.....><.....>return 1 <.....>else: <.....><.....>return n * factorielle(n - 1)</pre> |

- Impératif - compilé
- Typage explicite, statique et faible
- Gestion mémoire explicite (pointeurs, libération, indexation)
- Indentation libre
- **Usage** : maîtrise des ressources matérielles, programmation embarquée sur microcontrôleurs, calculs intensifs, rapidité de traitement importante
- **Exemples** : noyau de grands systèmes d'exploitation comme Windows et Linux, moteurs graphiques des jeux vidéo,...

- Impératif/objet/fonctionnel - interprété
- Typage implicite, dynamique et fort
- Gestion mémoire implicite (ramasse miettes)
- Indentation structurante
- **Usage** : licence libre, multiplateforme, optimise la productivité des programmeurs en offrant une syntaxe simple séparée des mécanismes de bas niveau, initiation aisée aux bases de la programmation
- **Exemples** : développement Web, data science (apprentissage, analyse, visualisation...), *scripting*

ENVIRONNEMENT DETRAVAIL

C



Python

9

- Stations de travail de l'université, ordinateurs exploités par une distribution Linux UBUNTU
- Ordinateurs personnels OS X /MacOS ➔ fonctionne avec un système UN IX
- Ordinateurs personnels Windows :
 - Installer un logiciel de virtualisation (par exemple VirtualBox d'Oracle est libre et opensource, <https://www.virtualbox.org/>)
 - Télécharger ubuntu desktop :<https://ubuntu.com/download/desktop>
 - L'installer comme une machine virtuelle grâce à l'outil de virtualisation
 - Les deux systèmes (windows et ubuntu) vont cohabiter
- Editeurs /Compilateurs /Débugger :
 - Université :gedit,bluefish,geany,kwrite,emacs... /gcc /gdb
 - Mac :xcode,codeLite... /gcc /gdb
 - Windows :notepad++, ... /minGW
 - IDE Gratuits (solution 3 en 1 - multiplateformes) :Code::blocks, Clion (licence étudiante),...

- Stations de travail de l'université, ordinateurs exploités par une distribution Linux UBUNTU
- Langage multiplateforme
- Editeurs /Interpréteur /Débugger :
 - Université :gedit,bluefish,geany,kwrite,emacs,... /python 3.x
 - Mac :xcode,codeLite... /python 3.x
 - Windows :notepad++,... /python 3.x
 - IDE Gratuits (solution 3 en 1 - multiplateformes) :idle, spyder, Pycharm (licence étudiante),...

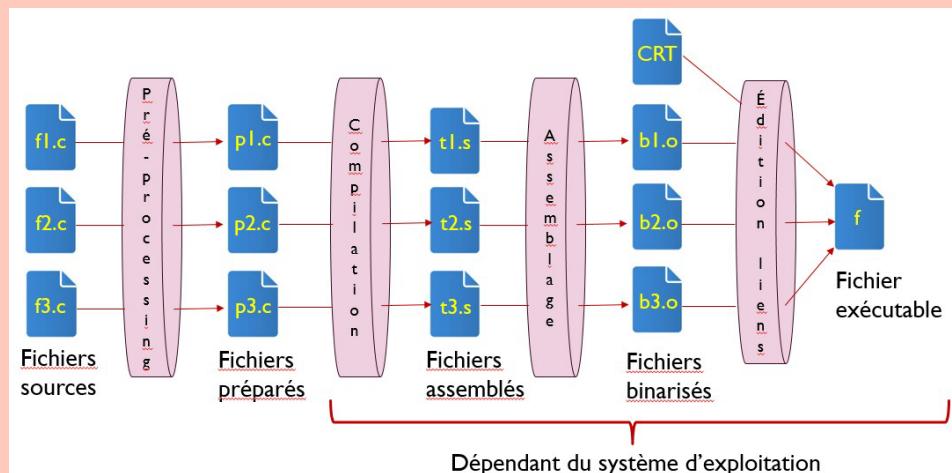
EXÉCUTION D'UN PROGRAMME

La commande `gcc` permet de transformer le code source `code.c` en binaire exécutable

Généralement :

Production du binaire `code.o` à partir du source `code.c` →
`gcc -c code.c`

Production de l'exécutable `code` à partir du binaire `code.o`
→ `gcc code.o -o code`



C



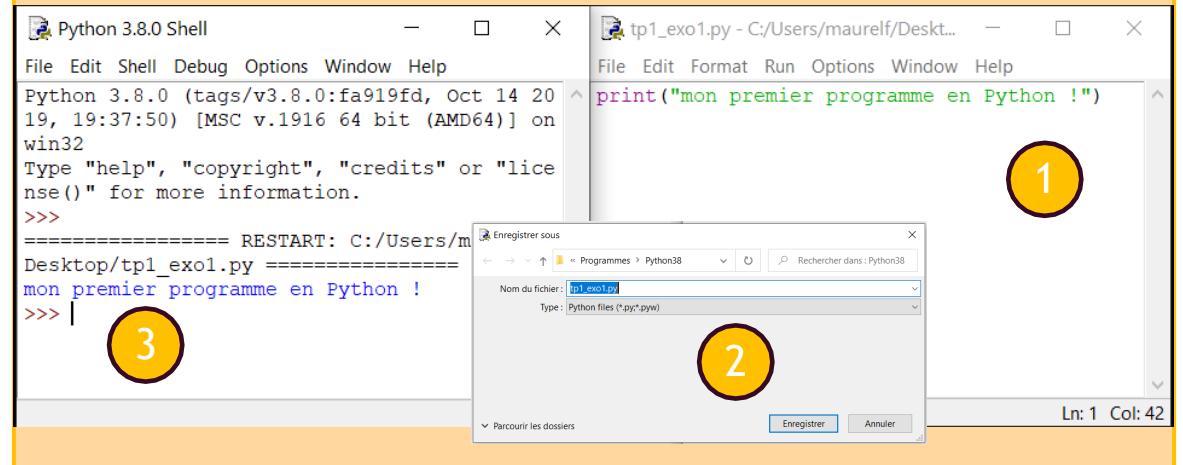
Python

10

Directement dans un terminal Linux : `python code.py`

Exemple en passant par un IDE : `idle` (pour Python 3.x)

1. Ecrire un programme dans l'éditeur intégré
2. Sauver le fichier `nom.py`
3. Exécuter un programme avec affichages dans l'interpréteur de commande (raccourci F5)



VARIABLES ET MÉMOIRE

C



Python

11

```
1. #include <stdio.h>
2.
3. int main() {
4.     /* my first program in C */
5.     int i;
6.     i = -3*2;
7.     printf("Val : %d \n", i);
8.     return 0;
9. }
```

Ligne 1 :fichier de définition

Ligne 2 :saut de ligne (purement esthétique)

Ligne 3 :début de l'exécution du bloc de programme principal

Ligne 4 :commentaire non pris en compte par le compilateur

Ligne 5 :définition de variable

Ligne 6 :calcul et d'affectation

Ligne 7 :affichage

Ligne 8 :fin du programme principal avec la valeur 0

Ligne 9 :marque de fin de bloc de programme principal

```
1. # my first program in Python
2. i = -3 * 2
3. print(f"Val : {i}") # utilisation des f-strings
```

Ligne 1 :commentaire non pris en compte par le compilateur

Ligne 2 :définition de variable, calcul et affectation

Ligne 3 :affichage

VARIABLES ET MÉMOIRE

C



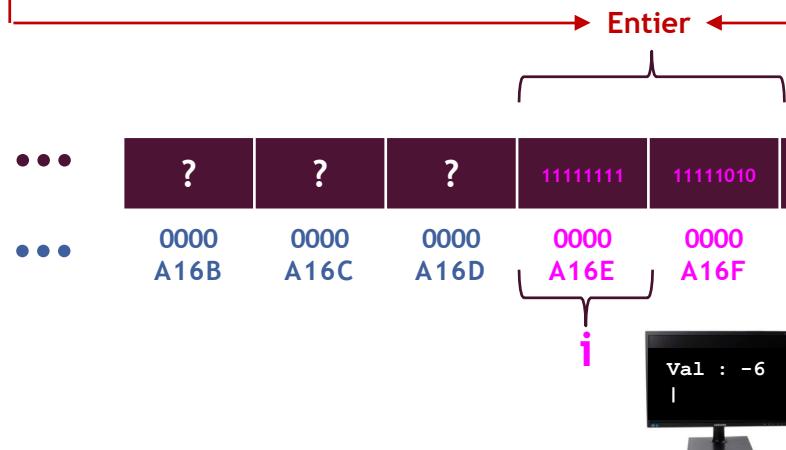
Python

12

```
1. #include <stdio.h>
2.
3. int main() {
4.     /* my first program in C */
5.     int i;
6.     i = -3*2;
7.     printf("Val : %hd \n", i);
8.     return 0;
9. }
```

```
1. # my first program in Python
2. i = -3 * 2
3. print(f"Val : {i}")
```

- Interprétation (pas de compilation mais performances diminuées)
- Gestion de mémoire dynamique et transparente (Garbage collector)
- Typage fort et implicite



Valeurs
Adresses
Labels

Mémoire 32 bits
(4Go = 2^{32} = 4 294 967 296 cases
de 00000000 à FFFFFFFF)

DÉCLARATION DE VARIABLES ET DE CONSTANTES

C



Python

13

| Déclaration | Python | C |
|-------------|---|--|
| Variables | Implicit : <code>i = 2020</code> | <code>[[un]signed] type maVariable [= valeur] ;</code> <code>short i ;</code> <code>signed short i ;</code> <code>unsigned short i = 2020 ;</code> |
| Constantes | Pas de déclarations spécifiques - Natives : <code>True, False, None</code> | <code>const [[un]signed] type maVariable [= valeur] ;</code> <code>const double PI = 3.1416</code> <code>#define MA_CST val</code> <code>#define PI 3.1416</code> |

Contraintes de nommage des labels :

- 63 premiers caractères considérés
- commencent par une lettre
- sensibles à la casse de caractère
- excluent les mots réservés au langage
- excluent certains caractères ambigus

Conventions de nommage des labels :

- pas d'accents
- uneVariableOuFonction (convention CamelCase)
- une_variable_ou_fonction (convention snake_case)
- UNE_CONSTANTE
- s_chaine_de_caractere

CONSTRUCTION D'EXPRESSIONS

C



Python

14

- Opérande** : constante, variable, littéral, expression
- Opérateur** : arithmétique, décomparaison, logique (booléen), binaire, d'assignation (composite)
- Expression** : combinaison d'opérandes et d'opérateurs
- Évaluation** : selon priorité, associativité (droite ou gauche) et regroupements forcés (parenthèses, crochets, accolades) des opérateurs

| Arithmétiques | Python | C | Associativité |
|------------------|--------|---|---------------|
| addition | + | + | gauche |
| soustraction | - | - | gauche |
| multiplication | * | * | gauche |
| division | / | / | gauche |
| division entière | // | | gauche |
| modulo | % | % | gauche |
| exponentiation | ** | | droite |

| Assignation | Python | C | Associativité |
|---------------------------|--------|----|---------------|
| assignation simple | = | = | droite |
| addition combinée | += | += | droite |
| soustraction combinée | -= | -= | droite |
| multiplication combinée | *= | *= | droite |
| division combinée | /= | /= | droite |
| division entière combinée | //= | | droite |
| modulo combiné | %= | %= | droite |
| incrémentation | | ++ | droite |
| décrémentation | | -- | droite |

| Logiques | Python | C | Associativité |
|----------|--------|----|---------------|
| et | and | && | gauche |
| ou | or | | gauche |
| non | not | ! | droite |

| Binaires | Python | C | Associativité |
|-----------------|--------|----|---------------|
| et | & | & | gauche |
| ou | | | gauche |
| ou exclusif | ^ | ^ | gauche |
| complément | ~ | ~ | droite |
| décalage gauche | << | << | gauche |
| décalage droite | >> | >> | gauche |

| Comparaison | Python | C | Associativité |
|------------------|--------|----|---------------|
| égalité | == | == | gauche |
| différence | != | != | gauche |
| inférieur strict | < | < | gauche |
| supérieur strict | > | > | gauche |
| inférieur | <= | <= | gauche |
| supérieur | >= | >= | gauche |

| | | | |
|------|--|--|--|
| ** | | | |
| * | | | |
| / | | | |
| % | | | |
| // | | | |
| + | | | |
| - | | | |
| << | | | |
| >> | | | |
| == | | | |
| != | | | |
| < | | | |
| > | | | |
| <= | | | |
| >= | | | |
| not, | | | |
| ! | | | |
| and, | | | |
| && | | | |
| or, | | | |
| | | | |
| & | | | |
| - | | | |
| ~ | | | |
| = | | | |
| += | | | |
| -= | | | |
| *= | | | |
| /= | | | |
| %= | | | |
| //= | | | |
| ++ | | | |
| -- | | | |

Priorités croissantes ↑

TYPES

- **Numériques**

- short, int, long, long long (tous *signed* or *unsigned*)
- Float, double, long double (tous *signed* or *unsigned*)

- **Alphanumériques**

- char (*signed* or *unsigned*)
- char *

- **Enumérés**

- bool

- **Composites**

- [] *

- struct



C



Python

15

- **De base**

- Entiers (*int*) → i=3
- flottants (*float*) → f=3.1416
- Booléens (*bool*) → b=True

- **Séquences**

- Chaines de caractère (*str*) → s="Bonjour" ou 'Bonjour'
- Listes (*list*) → l=[1,3,"bonjour",[4.2,5],False]
- Tuples(*tuple*) → t=(1,3,"bonjour",[4.2,5],False)

- **Dictionnaires (*dict*)**

- d={0:1,6:"bonjour","trois":False}

- **Ensembles (*set* et *frozenset*)**

- e={3, 12, 17, 157, 6}

BOOLÉENS

C



Python

16

```
int flag = 1;  
  
while (flag) {  
    ... flag = 0; ...  
}
```

```
bool flag = true;  
  
while (flag) {  
    ... flag = false; ...  
}
```

```
! (A % 400) || !(A % 4) && A % 100;
```

```
flag = True  
  
while flag:  
    ... flag = False ...
```

```
not(A % 400) or not(A % 4) and A % 100;
```

MUTABILITÉ DES TYPES



Python

17

Pas de modifications directes des valeurs des variables. Passage obligé par une nouvelle affectation de la variable.

```
n = 3.4  
n = -1.2  
s = "bonjour"  
s = s + " à tous !"
```

Types non mutables

Modifications directes possibles des valeurs des variables.

```
l[3] = "toto"  
d["trois"] = True
```

Types mutables

- **De base**

- Entiers (*int*) → `i = 3`
- flottants (*float*) → `f=3.1416`
- Booléens (*bool*) → `b=True`

- **Séquences**

- Chaines de caractère (*str*) → `s="Bonjour" ou 'Bonjour'`
- Tuples(*tuple*) → `t=(1, 3, "bonjour", [4.2, 5], False)`

- Listes (*list*) → `l=[1, 3, "bonjour", [4.2, 5], False]`

- **Dictionnaires(*dict*)**

- `d={0:1, 6:"bonjour", "trois":False}`

CHAINES (NON MUTABLES) : "STR"



Python

| | | | | | |
|----|----|----|----|----|----|
| 0 | 1 | 2 | 3 | 4 | 5 |
| -6 | -5 | -4 | -3 | -2 | -1 |

- `s="Chaine"` ou `s='Chaine'` →

| | | | | | |
|---|---|---|---|---|---|
| C | h | a | i | n | e |
|---|---|---|---|---|---|
- `s + ' Python'` →

| | | | | | | | | | | | | |
|---|---|---|---|---|---|--|---|---|---|---|---|---|
| C | h | a | i | n | e | | P | y | t | h | o | n |
|---|---|---|---|---|---|--|---|---|---|---|---|---|
- `s[2]` équivalent à `s[-4]` →

| |
|---|
| a |
|---|
- `s[1:4]` équivalent à `s[-5:-2]` →

| | | |
|---|---|---|
| h | a | i |
|---|---|---|
- `s[2:]` équivalent à `s[-4:]` →

| | | | |
|---|---|---|---|
| a | i | n | e |
|---|---|---|---|
- `s[:3]` équivalent à `s[:-3]` →

| | | |
|---|---|---|
| C | h | a |
|---|---|---|
- `s[:]` fait une copie de `s` →

| | | | | | |
|---|---|---|---|---|---|
| C | h | a | i | n | e |
|---|---|---|---|---|---|
- Caractères spéciaux : '`\n`' (nouvelle ligne), '`\t`' (tabulation), '`\b`' (backslash) ...

LISTES (NON MUTABLES) : (TUPLES,)



Python

19

- Fonctionnement identique aux chaînes de caractères mais listes potentiellement hétérogènes

| | | | | | |
|----|----|----|----|----|----|
| -6 | -5 | -4 | -3 | -2 | -1 |
| 0 | 1 | 2 | 3 | 4 | 5 |

- $l = ('P', 'y', 't', 'h', 'o', 'n') \rightarrow$

| | | | | | |
|---|---|---|---|---|---|
| P | y | t | h | o | n |
|---|---|---|---|---|---|

| | | | | | | | |
|----|----|----|----|----|----|----|----|
| -8 | -7 | -6 | -5 | -4 | -3 | -2 | -1 |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |

- $l + (' ', 3) \rightarrow$

| | | | | | | | |
|---|---|---|---|---|---|-----|---|
| P | y | t | h | o | n | ' ' | 3 |
|---|---|---|---|---|---|-----|---|

Paquetage automatique
 $t = 1, 2, 3$

Dépaquetage automatique
 $a, b, c = t$

LISTES (MUTABLES) : [LIST]



Python

20

- Fonctionnement identique aux tuples mais type **mutable**

- `l = ['L','i','s','t','e']` →

| | | | | |
|----|----|----|----|----|
| -5 | -4 | -3 | -2 | -1 |
| L | i | s | t | e |
- `l + ['P','y','t','h','o','n',3]` →

| | | | | | | | | | | | | |
|---|---|---|---|---|--|---|---|---|---|---|---|---|
| L | i | s | t | e | | P | y | t | h | o | n | 3 |
|---|---|---|---|---|--|---|---|---|---|---|---|---|
- `del(l[3])` →

| | | | |
|---|---|---|---|
| L | i | s | e |
|---|---|---|---|
- `l[1:3]=['a','c','h']` →

| | | | | |
|---|---|---|---|---|
| L | a | c | h | E |
|---|---|---|---|---|
- `l+=['u','r']` →

| | | | | | | |
|---|---|---|---|---|---|---|
| L | a | c | h | e | u | r |
|---|---|---|---|---|---|---|
- `l[:]` fait une copie de l



Attention aux méthodes de modification « sur place » (documentation)

e.g : méthode de liste **sort** vs primitive **sorted**

GESTION DES ENTRÉES / SORTIES

C



Python

21

```
int main()
{
    char nom[100];
    unsigned short age;

    printf("Quel est votre nom ? ");
    scanf("%s", nom);
    printf("Quel est votre âge ? ");
    scanf("%hu", &age);
    printf("%s - %hu ans.\n", nom, age);
    return 0;
}
```

Transtypepage explicite parfois nécessaire pour 1 et 2 :

input retourne toujours une chaîne de caractère (str)

! Typage fort → age + 1 produirait une erreur !

```
nom = input("Quel est votre nom ? ")
age = input("Quel est votre âge ? ")
```

6 possibilités d'affichage au résultat visuel équivalent :

1. print(nom, "-", age, "ans.")
2. print(nom + " - " + age + " ans.")
3. print("{n} - {a} ans.".format(n=nom, a=age))
4. print("{1} - {0} ans.".format(age, nom))
5. print("{} - {} ans.".format(nom, age))
6. print(f"{nom} - {age} ans.")



lbis. print(nom, "-", int(age) + 1, "ans.")

2bis. print(nom + " - " + str(int(age) + 1) + "ans.")

PRINCIPALES FONCTIONS UTILES DE BASE DE PYTHON



Python

22

Intégrées au langage (built-in functions - fonctions natives)

Transtypage

```
int(val)  
float(val)  
bool(val)  
str(val)  
tuple(val)  
frozenset(val)  
list(val)  
dict(val)  
set(val)
```

Calcul numérique/logique

```
min(intseq)  
max(intseq)  
abs(num)  
eval(str)  
pow(base, num, mod)  
round(num)  
sum(intseq)  
all(boolseq)  
any(boolseq)
```

Séquences, Infos, I/O

```
range(deb, fin, pas)  
zip(seq, seq)  
reversed(seq)  
sorted(seq)  
len(seq)  
type(val)  
print(val)  
input(str)  
open(str, mode, encodage)
```

Documentation

```
dir(val)  
help(val)
```

Associées à un type avec l'opérateur . (méthodes)

```
{un_dict}.keys()  
{un_dict}.values()  
{un_dict}.items()  
{un_dict}.get(cle, defaut)  
{un_dict}.fromkeys(seq, val)
```

```
"une_str".format(vals...)  
"une_str".split(str)  
"une_str".join(strseq)  
"une_str".lower()  
"une_str".upper()
```

```
[une_list].extend(seq)  
[une_list].append(val)  
[une_list].pop()  
[une_seq].count(val)  
[une_seq].index(val)
```

BRANCHEMENTS CONDITIONNELS

C



```
if (a%400 == 0) {  
    printf("%d bissextile", a);  
} else if (a%4 == 0) {  
    if (a%100 == 0) {  
        printf("%d non bissextile", a);  
    } else {  
        printf("%d bissextile", a);  
    }  
} else {  
    printf("%d non bissextile", a);  
}
```

Python

```
if a%400 == 0:  
    print(a, "bissextile")  
elif a%4 == 0:  
    if a%100 == 0:  
        print(a, "non bissextile")  
    else:  
        print(a, "bissextile")  
else:  
    print(a, "non bissextile")
```

Opérateur ternaire : ma_var = val1 if <cond> else val2

BOUCLES

C



Python

24

```
#include <stdio.h>

int main(){

    char reponse = "n";
    while (reponse == "n") {
        printf("Terminé o/n ?");
        reponse = getchar();
    }
    return 0;
}
```

```
#include <stdio.h>

int main(){
    char reponse;
    do {
        printf("Terminé o/n ?");
        reponse = getchar();
    } while (reponse == "n");
    return 0;
}
```

| | |
|---|---|
| <pre>for (int i=0, i<5, i++) {</pre> | <pre>for (int i=5, i>0, i--) {</pre> |
| <pre> printf("%d\n", i);</pre> | <pre> printf("%d\n", i);</pre> |
| <pre>}</pre> | <pre>}</pre> |

Pas de do-while

BOUCLES SUPPLÉMENTAIRES



Python

25

```
for lettre in "Bonjour":  
    print(lettre, end="-")
```

>>> B-o-n-j-o-u-r-

```
for i in range(len("Bonjour")):  
    print(i, end="-")
```

>>> 0-1-2-3-4-5-6-

```
for val in [1, 2, 3, 4]:  
    print(val, end="-")
```

>>> 1-2-3-4-

```
for cle in {"Fabrice":49, "Isabelle":49, "Julien":20 ,  
           "Samuel":18}: print(cle, end="-")
```

>>> Fabrice-Isabelle-Julien-Samuel-

```
for nom, age in {"Fabrice":49, "Isabelle":49, "Julien":20 ,  
                 "Samuel":18}.items(): print(nom, "a", age, "ans")
```

>>> Fabrice a 49 ans
>>> Isabelle a 49 ans
>>> Julien a 20 ans
>>> Samuel a 18 ans

UN PEU PLUS SUR LES LISTES

Certaines **fonctions natives** sont très utiles pour créer des listes ou lorsqu'elles utilisent des listes en paramètre :

| | | | |
|--|---|---|------------------------------------|
| >>> len([1, 2, 3]) 3 | >>> zip([1, 2, 3], [4, 5, 6], [7, 8, 9]) [(1, 4, 7), (2, 5, 8), (3, 6, 9)] | >>> dir([1]) les méthodes | >>> range(3) range(0, 3) |
| >>> reversed([2, 1, 3]) <list_reverseiterator object> | >>> sorted([2, 1, 3]) [1, 2, 3] | >>> 'a*b*c'.split('*') ['a', 'b', 'c'] | >>> list('abc') ['a', 'b', 'c'] |



Les listes possèdent également de nombreuses **méthodes** qui peuvent s'avérer très pratiques. On appelle méthode une fonction qui agit sur l'**objet** auquel elle est attachée par un **:**

```
>>> l = [1, 2, 3]
>>> l += [4]
>>> l.append(5)
>>> l
>>> [1, 2, 3, 4, 5]
>>> l = ['a', 'b', 'c', 'b']
>>> l.count('b')
2
```

Certaines méthodes sont programmées

```
>>> l = [1, 2, 3]
>>> l.extend([4, 5, 6])
>>> l
[1, 2, 3, 4, 5, 6]
```

s pour ne pas retourner de valeur : `append`, `extend`, ... d'autres si : `count`, `join`, ...

```
>>> l = ['a', 'b', 'c']
>>> ''.join(l)
'a-b-c'
```

Beaucoup d'autres **fonctions** ou **méthodes** exploitant les listes existent dans de nombreux modules qu'il suffit d'importer. Elles peuvent aussi être attachées au module par l'opérateur `:` si seul le nom est importé

```
>>> import random
>>> l = [1, 2, 3, 4, 5]
random.shuffle(l)
>>> l
[2, 3, 6, 4, 1, 5]
>>> random.choice(l)
3
```

FONCTIONS

```
#include <stdio.h>
#include <math.h>

double f(double x) {
    double y = pow(x, 2)-2*x+1;
    return y;
}

int main() {
    printf("Entrer valeur x : ");
    scanf("%lf", &val);
    double y = f(val);
    printf("y = %f", y);
    return 0
}
```

C



Python

28

```
def f(x):
    y = x**2-2*x+1
    return y

val = input("Entrer valeur x : ")
y = f(float(val))
print("y =", y)
```

VARIABLES LOCALES ET GLOBALES



Python

29

```
nb_erreur = 0

def increment_erreur(val=1):
    nb_erreur = nb_erreur + val

def a_perdu():
    return nb_erreur == 10

>>> nb_erreur = 10
>>> print(nb_erreur)
10
>>> print(a_perdu())
False
>>> increment_erreur()
UnboundLocalError: local variable
'nb_erreur' referenced before assignment
```

VARIABLES LOCALES ET GLOBALES



Python

30

Création variable globale

```
nb_erreur = 0
```



Création/accès variable locale

```
def increment_erreur(val=1):  
    nb_erreur = nb_erreur + val
```

Accès variable globale

```
def a_perdu():  
    return nb_erreur == 10
```

Modification variable globale

```
>>> nb_erreur = 10
```

Accès variable globale

```
>>> print(nb_erreur)
```

```
10
```

```
>>> print(aPerdu())
```

```
True
```

```
>>> increment_erreur()
```

```
UnboundLocalError: local variable
```

```
'nb_erreur' referenced before assignment
```

VARIABLES LOCALES ET GLOBALES



Python

31

Création variable globale

```
nb_erreur = 0
```

Accès/Modification variable globale

```
def increment_erreur(val=1):  
    global nb_erreur  
    nb_erreur = nb_erreur + val
```

Accès variable globale

```
def a_perdu():  
    return nb_erreur == 10
```

Modification variable globale

```
>>> nb_erreur = 10
```

Accès variable globale

```
>>> print(nb_erreur)
```

10

```
>>> print(aPerdu())
```

True

```
>>> increment_erreur()
```

```
>>> print(nb_erreur)
```

11

PAS DE PROCÉDURES



Python

32

Retour explicite

```
def f(x) :  
    return x**2 - 2*x + 1  
  
>>> y = f(3)  
>>> print(y)  
4
```

Retour implicite : **None**

```
! def f(x) :  
    print(x**2 - 2*x + 1)  
  
>>> y = f(3)  
4  
>>> print(y)  
None
```

À l'exception de fonctions spécifiques d'affichage, une fonction **ne devrait pas** contenir d'instruction `print`.

Retour paqueté possible (type `tuple` implicite)

```
def f(x) :  
    return x, x**2 - 2*x + 1  
  
>>> a, b = f(3)  
>>> print(a, b)  
(3, 4)
```

Permet de facilement faire des fonctions qui retournent **plusieurs valeurs**

PARAMÈTRES OBLIGATOIRES VS. OPTIONNELS



Python

33

Les paramètres sont déclarés dans l'ordre : d'abord tous les paramètres obligatoires ; puis les paramètres optionnels (avec une valeur par défaut)

Les paramètres sont récupérés par la fonction en respectant l'ordre de l'appel, sauf lorsque les paramètres optionnels sont explicitement indiqués

```
def f(p1, p2, d1=10, d2=20):  
    print(p1, p2, d1, d2)  
  
>>> f(1, 2, 3, 4)  
1 2 3 4  
>>> f(1, 2, 3)  
1 2 3 20  
>>> f(1, 2)  
1 2 10 20  
>>> f(1)  
TypeError: f() missing 1 required positional argument: 'p2'  
>>> f(1, 2, d2=4)  
1 2 10 4
```

PASSAGE DES PARAMÈTRES PAR RÉFÉRENCE ET NON PAR VALEUR



Python

34

1. >>> l1 = [1, 2, 3]



PASSAGE DES PARAMÈTRES PAR RÉFÉRENCE ET NON PAR VALEUR



Python

35

```
1. >>> l1 = [1, 2, 3]  
2. >>> l2 = l1
```



PASSAGE DES PARAMÈTRES PAR RÉFÉRENCE ET NON PAR VALEUR



Python

36

```
1. >>> 11 = [1, 2, 3]
2. >>> 12 = 11
3. >>> 11[0] = 100
```



PASSAGE DES PARAMÈTRES PAR RÉFÉRENCE ET NON PAR VALEUR



Python

37

```
1. >>> l1 = [1, 2, 3]
2. >>> l2 = l1
3. >>> l1[0] = 100
4. >>> print(l2)
[100, 2, 3]
```



PASSAGE DES PARAMÈTRES PAR RÉFÉRENCE ET NON PAR VALEUR



Python

38

```
1. >>> l1 = [1, 2, 3]
2. >>> l2 = l1
3. >>> l1[0] = 100
4. >>> print(l2)
[100, 2, 3]
```



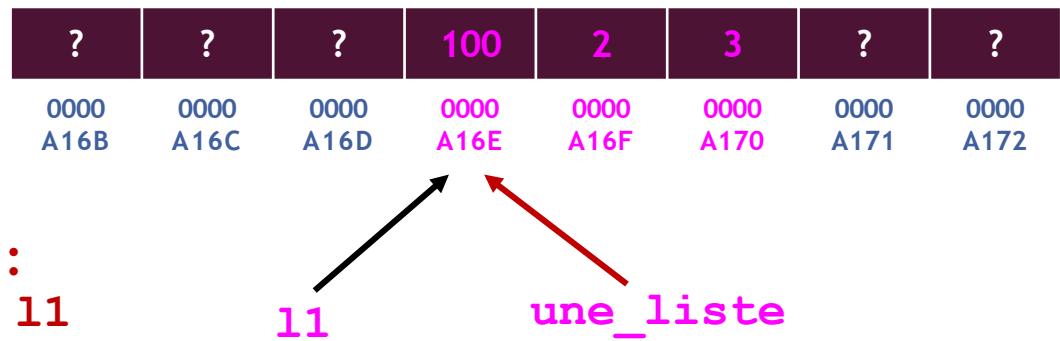
```
def f(une_liste):
    une_liste[0] = 100
    return True

>>> l1 = [1, 2, 3]
>>> print(f(l1))
True
>>> print(l1)
[100, 2, 3]
```



Prendre garde au passage de variables de type mutable (listes ou dictionnaires) en paramètre de fonctions

Implicitement:
`une_liste = l1`



COPIE DE SÉQUENCE



Python

39

```
>>> 11 = [1, 2, 3]
```



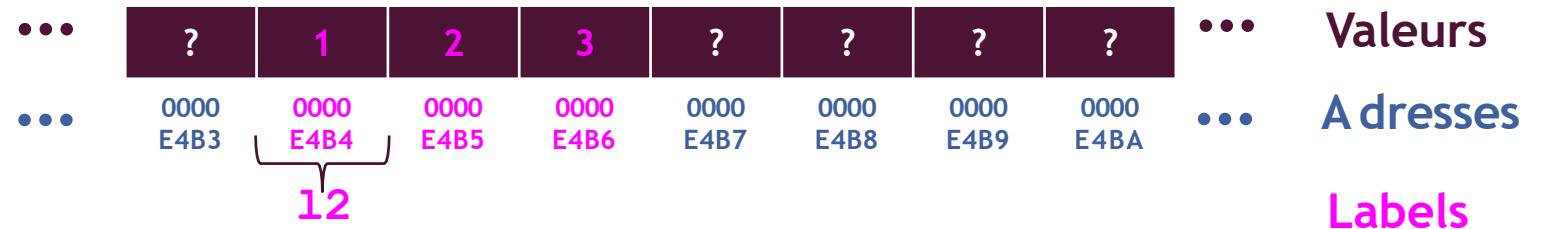
COPIE DE SÉQUENCE



Python

40

```
>>> l1 = [1, 2, 3]
>>> l2 = l1[:]
```



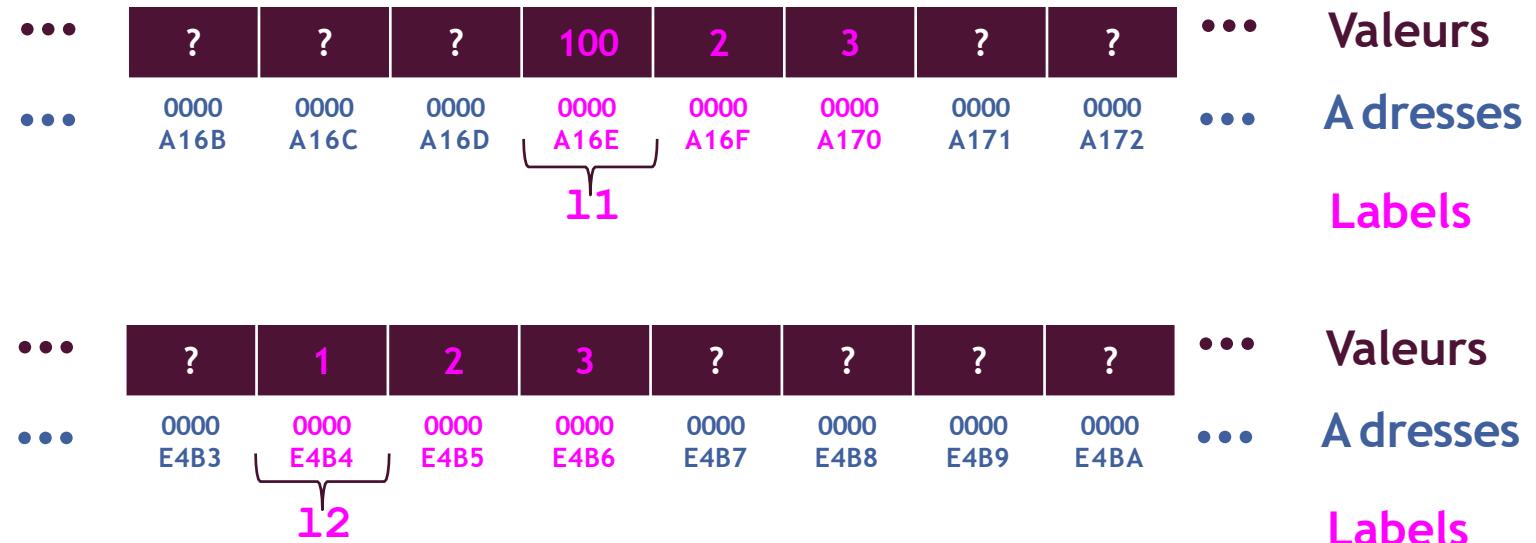
COPIE DE SÉQUENCE



Python

41

```
>>> l1 = [1, 2, 3]
>>> l2 = l1[:]
>>> l1[0] = 100
>>> print(l1)
[100, 2, 3]
>>> print(l2)
[1, 2, 3]
```



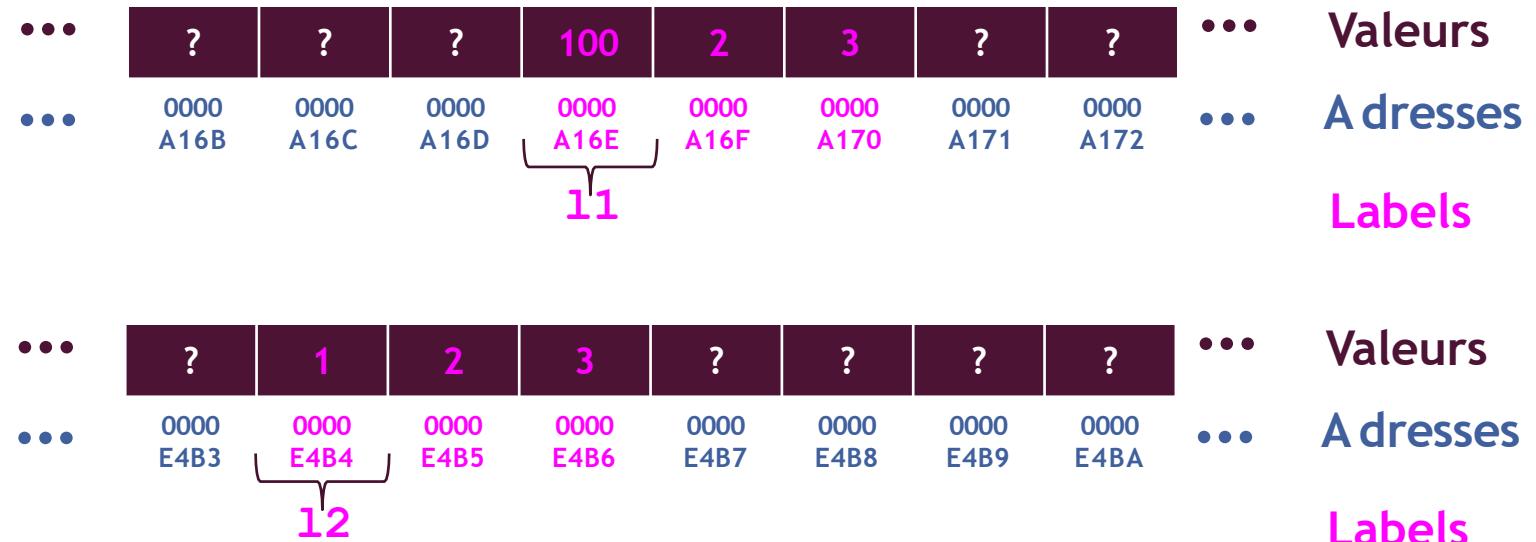
COPIE DE SÉQUENCE



Python

42

```
>>> l1 = [1, 2, 3]
>>> l2 = l1[:]
>>> l1[0] = 100
>>> print(l1)
[100, 2, 3]
>>> print(l2)
[1, 2, 3]
```



```
>>> from copy import deepcopy
>>> l1 = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
>>> l2 = deepcopy(l1)
```

! L'utilisation du *slicing* ne suffit pas pour faire une copie correcte des séquences imbriquées

LE JEU DE NIM : INCLUDE, GET_INT ET MAIN



```
#include <stdio.h>
#include <time.h>
#include <stdlib.h>

int get_int(char mess[]) {
    int n;
    printf("%s", mess);
    scanf("%d", &n);
    return n;
}

int main() {
    srand(time(NULL));
    partie_nim();
    return 0;
}
```

C



Python

43

```
from random import randint
```

```
partie_nim()
```

LE JEU DE NIM : PRINT_PLATEAU



```
void print_plateau(int allumettes) {
    for (int i=0; i<allumettes; i++) {
        printf("|");
    }
    if (allumettes>0) {
        printf("- Il reste %d allumettes\n", allumettes);
    } else {
        printf("Plus d'allumettes - Jeu terminé\n");
    }
    printf("\n");
}
```

C



Python

44

```
def print_plateau(allumettes):
    print("|" * allumettes, end="")
    if allumettes > 0:
        print("- Il reste", allumettes, "allumettes")
    else:
        print("Plus d'allumettes - Jeu terminé")
    print()
```

LE JEU DE NIM : LES CHOIX HUMAIN ET ORDI



C



Python

45

```
int choix_humain(int allumettes) {
    int choix;
    do {
        choix = get_int("Humain - le nombre d'allumettes
que je retire (de 1 à 3) !");
    } while (choix < 0 || choix > 3 || choix >
allumettes); print_plateau(allumettes-choix);
    return choix;
}
```

```
int choix_ordinateur(int allumettes) {
    int choix = allumettes%4;
    if (choix == 0) {
        choix = rand()%3 + 1;
    }
    printf("Ordi - J'enlève %d allumettes.\n",
    choix); print_plateau(allumettes-choix);
    return choix;
}
```

```
def choix_humain(allumettes):
    choix = 0
    while choix not in range(1, 4) or choix > allumettes:
        choix = int(input("Humain - le nombre
d'allumettes
que je retire (de 1 à 3) !"))
    print_plateau(allumettes - choix)
    return choix
```

```
def choix_ordinateur(allumettes):
    choix = allumettes % 4
    if choix == 0:
        choix = randint(1, 3)
    print("Ordi - J'enlève", choix, "allumettes.")
    print_plateau(allumettes - choix)
    return choix
```

LE JEU DE NIM : LA PARTIE



```
int partie_nim() {
    int allumettes = 20;
    int gagne = 0;
    print_plateau(allumettes);
    while (allumettes > 0) {
        allumettes -= choix_humain(allumettes);
        if (allumettes == 0) {
            gagne = 1;
        } else {
            allumettes -= choix_ordinateur(allumettes);
        }
    }
    if (gagne==1) {
        printf("Comment avez-vous triché ?");
    } else {
        printf("Encore l'ordinateur qui gagne...");
    }
    return allumettes;
}
```

C



Python

46

```
def partie_nim():
    allumettes, gagne = 20, False
    print_plateau(allumettes)
    while allumettes > 0:
        allumettes -= choix_humain(allumettes)
        if allumettes == 0:
            gagne = True
        else:
            allumettes -= choix_ordinateur(allumettes)
    if gagne:
        print("Comment avez-vous triché ?")
    else:
        print("Encore l'ordinateur qui gagne...")
    return allumettes
```

AUTRE EXEMPLE : LES FRACTIONS RATIONNELLES



Python

47

```
def pgcd(a, b):
    while b != 0:
        a, b = b, a % b
    return a

def affiche_rationnel(a, b):
    if b == 0:
        raise ZeroDivisionError("Dénominateur nul !")
    signe = '-' if a * b < 0 else ''
    a, b = abs(a), abs(b)
    num = '0' if a == 0 else signe + str(a)
    denom = '' if b == 1 or a == 0 else ' / ' + str(b)
    print(num + denom)
```

```
def reduction_rationnel(a, b):
    p = pgcd(a, b)
    return a // p, b // p

def somme_rationnels(n1, d1, n2, d2):
    return reduction_rationnel(n1*d2+n2*d1, d1*d2)

>>> a, b = reduction_rationnel(6, -3)
>>> c, d = reduction_rationnel(-7, 4)
>>> affiche_rationnel(*somme_rationnels(a, b, c, d))
-15 / 4
```

1. Il est possible d'arrêter le programme avec des erreurs prédéfinis et des messages d'erreur spécifiques (mécanisme de levée d'exception : `raise`)
2. Utilisation de l'opérateur ternaire de branchement conditionnel pour construire la chaîne du rationnel

1. Utilisation du dépaquetage implicite lorsqu'une fonction retourne plusieurs valeurs
2. L'opérateur `*` force le dépaquetage. Solution pratique pour fournir une valeur à chaque paramètre d'une fonction alors que ces valeurs sont stockées dans une séquence

REPRÉSENTATION DES CARACTÈRES : LE CODE ASCII 7 BITS

- 26 lettres de l'alphabet latin sans accent en minuscule et majuscule, chiffres, caractères de contrôle non imprimable, ponctuations et signe divers (ex:A = 65, a=91)
- Normalisation en 1988 sous le nom ISO -646
- Non intégration dans les nouveaux réseaux de communication → risque de disparition
- Le code ASCII pénalise les textes écrits dans d'autres langues
- Seul l'anglais, le swahili et l'indonésien ne se satisfont de cette norme

| | | | |
|--------|-----------|------|-------|
| 0 NUL | 32 espace | 64 @ | 96 ` |
| 1 SOH | 33 ! | 65 A | 97 a |
| 2 STX | 34 " | 66 B | 98 b |
| 3 ETX | 35 # | 67 C | 99 c |
| 4 EOT | 36 \$ | 68 D | 100 d |
| 5 ENQ | 37 % | 69 E | 101 e |
| 6 ACK | 38 & | 70 F | 102 f |
| 7 BEL | 39 ' | 71 G | 103 g |
| 8 BS | 40 (| 72 H | 104 h |
| 9 HT | 41) | 73 I | 105 i |
| 10 LF | 42 * | 74 J | 106 j |
| 11 VT | 43 + | 75 K | 107 k |
| 12 FF | 44 , | 76 L | 108 l |
| 13 CR | 45 - | 77 M | 109 m |
| 14 SO | 46 . | 78 N | 110 n |
| 15 SI | 47 / | 79 O | 111 o |
| 16 SLE | 48 0 | 80 P | 112 p |
| 17 CS1 | 49 1 | 81 Q | 113 q |
| 18 DC2 | 50 2 | 82 R | 114 r |
| 19 DC3 | 51 3 | 83 S | 115 s |
| 20 DC4 | 52 4 | 84 T | 116 t |
| 21 NAK | 53 5 | 85 U | 117 u |
| 22 SYN | 54 6 | 86 V | 118 v |
| 23 ETB | 55 7 | 87 W | 119 w |
| 24 CAN | 56 8 | 88 X | 120 x |
| 25 EM | 57 9 | 89 Y | 121 y |
| 26 SIB | 58 : | 90 Z | 122 z |
| 27 ESC | 59 ; | 91 [| 123 { |
| 28 FS | 60 < | 92 \ | 124 |
| 29 GS | 61 = | 93] | 125 } |
| 30 RS | 62 > | 94 ^ | 126 ~ |
| 31 US | 63 ? | 95 _ | 127 ■ |

LE CODE ASCII 8 BITS (1 OCTET)

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|----|----|----|----|----|----|----|-----|-----|----|
| 00 | | | | | | | | | |
| 01 | | | | | | | | | |
| 02 | | | | | | | | | |
| 03 | ! | " | # | \$ | % | & | ' | | |
| 04 | (|) | * | + | , | - | . | / | 0 |
| 05 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | : |
| 06 | < | = | > | ? | @ | A | B | C | D |
| 07 | F | G | H | I | J | K | L | M | N |
| 08 | P | Q | R | S | T | U | V | W | X |
| 09 | z | [| \ |] | ^ | _ | ` | a | b |
| 10 | d | e | f | g | h | i | j | k | l |
| 11 | n | o | p | q | r | s | t | u | v |
| 12 | x | y | z | (|) | ~ | ^ | ç | ü |
| 00 | 01 | 02 | 03 | 04 | 05 | 06 | 07 | 08 | 09 |
| 13 | é | â | ä | à | å | ç | ê | ë | è |
| 14 | í | î | Ã | Ã | É | æ | Æ | ô | ò |
| 15 | û | ù | ÿ | ö | Ü | ç | £ | ¥ | ƒ |
| 16 | á | í | ó | ú | ñ | ń | º | º | – |
| 17 | – | ½ | ¾ | i | « | » | III | III | |
| 18 | † | ‡ | | ŋ | ՚ | ՚ | ՚ | ՚ | ՚ |
| 19 | ‡ | ՚ | ՚ | ՚ | ՚ | ՚ | ՚ | ՚ | ՚ |
| 20 | ߱ | ߲ | ߳ | ߴ | ߵ | ߶ | ߷ | ߸ | ߹ |
| 21 | ߺ | ߻ | ߻ | ߻ | ߻ | ߻ | ߻ | ߻ | ߻ |
| 22 | ߻ | ߻ | ߻ | ߻ | ߻ | ߻ | ߻ | ߻ | ߻ |
| 23 | ߻ | ߻ | ߻ | ߻ | ߻ | ߻ | ߻ | ߻ | ߻ |
| 24 | ߻ | ߻ | ߻ | ߻ | ߻ | ߻ | ߻ | ߻ | ߻ |
| 25 | ߻ | ߻ | ߻ | ߻ | ߻ | ߻ | ߻ | ߻ | ߻ |

- **10 jeux de caractères sont standardisés :**
 - 88591 : latin 1 (français inclus ici)
 - 88592, 88593, 88594 : latin 2, 3 et 4
 - 88595 : latin/cyrillique
 - 88596 : latin/arabe
 - 88597 : latin/grec
 - 88598 : latin/hébreu
 - 88599 : latin 5
 - 885910 : latin 6

PROBLÈMES

Test avec un pangramme du français :

- Dès Noël où un zéphyr haï me vêt de glaçons würmiens je dîne d'exquis rôtis de bœuf au kir à l'Aÿ d'âge mûr & cætera !
- DÈS NOËL OÙ UN ZÉPHYR HAÏ ME VÊT DE GLAÇONS WÜRMIENS JE DÎNE D'EXQUIS RÔTIS DE BŒUFAU KIR À L'AŸ D'ÂGE MÛR & CÆTERA !

Problèmes de l'ASCII 8 bits :

- Ligature œ et Œ en latin-1 (iso-8859-15)
- Accessibilité des caractères : Estonien (latin-4) écrivent avec latin-1 (moins 2 caractères) ou latin-2 (moins 1 caractère)
- Difficultés si plusieurs systèmes d'écritures pour gérer plusieurs normes de codage en même temps
➔ idée d'un jeu de caractères dit universel : ISO-10646 ou UNICODE

NORME UNICODE (1989)

- Basé sur 32 bits soit 4 octets (4 294 967 296 de caractères)
- 256 groupes de 256 plans de 256 colonnes
- Chaque cellule contient 256 caractères
- Un seul plan disponible pour l'instant (16 bits soit 65 536 caractères) : Basic Multilingual Plane (BMP)
- 38 885 caractères actuellement codés pour transcrire la plupart des langues écrites
- Les 256 premiers codes = ISO-8859

PRINCIPE SUR 2 OCTETS

| | 000 | 001 | 002 | 003 | 004 | 005 | 006 | 007 |
|---|------|-----|-----|-----|-----|-----|-----|-----|
| 0 | NULL | DEL | SP | 0 | @ | P | ` | p |
| 1 | STX | DC1 | ! | 1 | A | Q | a | q |
| 2 | SOH | DC2 | “ | 2 | B | R | b | r |
| 3 | EOT | DC3 | # | 3 | C | S | c | s |
| 4 | EOA | DC4 | \$ | 4 | D | T | d | t |
| 5 | ENQ | NAK | % | 5 | E | U | e | u |
| 6 | ACK | SYN | & | 6 | F | V | f | v |
| 7 | BEL | ETB | ! | 7 | G | W | g | w |
| 8 | BS | CAN | (| 8 | H | X | h | x |
| 9 | HT | EM |) | 9 | I | Y | i | y |
| A | LF | SUB | * | : | J | Z | j | z |
| B | VT | ESC | + | ; | K | [| k | { |
| C | FF | FS | , | < | L | \ | l | |
| D | CR | GS | - | = | M |] | m | } |
| E | SO | RS | . | > | N | ^ | n | ~ |
| F | SI | US | / | ? | O | _ | o | DEL |

| | 008 | 009 | 00A | 00B | 00C | 00D | 00E | 00F |
|---|------|------|-------|------|------|------|------|------|
| 0 | CTRL | CTRL | NB SP | o | À | Đ | à | ð |
| 1 | 0080 | 0090 | 00A0 | 00B0 | 00C0 | 00D0 | 00E0 | 00F0 |
| 2 | CTRL | CTRL | ! | ± | Á | Ñ | á | ñ |
| 3 | 0081 | 0091 | 00A1 | 00B1 | 00C1 | 00D1 | 00E1 | 00F1 |
| 4 | CTRL | CTRL | ¢ | 2 | Â | Ò | â | ð |
| 5 | 0082 | 0092 | 00A2 | 00B2 | 00C2 | 00D2 | 00E2 | 00F2 |
| 6 | CTRL | CTRL | £ | 3 | Ã | Ó | ã | ó |
| 7 | 0083 | 0093 | 00A3 | 00B3 | 00C3 | 00D3 | 00E3 | 00F3 |
| 8 | CTRL | CTRL | ¤ | ‘ | À | Ô | ä | ô |
| 9 | 0084 | 0094 | 00A4 | 00B4 | 00C4 | 00D4 | 00E4 | 00F4 |
| A | CTRL | CTRL | ¥ | μ | Ã | Ó | å | ó |
| B | 0085 | 0095 | 00A5 | 00B5 | 00C5 | 00D5 | 00E5 | 00F5 |
| C | CTRL | CTRL | ¦ | ¶ | Æ | Ö | æ | ö |
| D | 0086 | 0096 | 00A6 | 00B6 | 00C6 | 00D6 | 00E6 | 00F6 |
| E | CTRL | CTRL | § | · | Ç | × | ç | ÷ |
| F | 0087 | 0097 | 00A7 | 00B7 | 00C7 | 00D7 | 00E7 | 00F7 |
| G | CTRL | CTRL | ” | „ | È | Ø | è | ø |
| H | 0088 | 0098 | 00A8 | 00B8 | 00C8 | 00D8 | 00E8 | 00F8 |
| I | CTRL | CTRL | © | 1 | É | Ù | é | ù |
| J | 0089 | 0099 | 00A9 | 00B9 | 00C9 | 00D9 | 00E9 | 00F9 |
| K | CTRL | CTRL | ¤ | ¤ | Ê | Ú | ê | ú |
| L | 0090 | 009A | 00AA | 00BA | 00CA | 00DA | 00EA | 00FA |
| M | CTRL | CTRL | « | » | È | Û | ë | û |
| N | 009B | 009B | 00AB | 00BB | 00CB | 00DB | 00EB | 00FB |
| O | CTRL | CTRL | ¬ | ¼ | Ì | Ü | ì | ü |
| P | 009C | 009C | 00AC | 00BC | 00CC | 00DC | 00EC | 00FC |
| Q | CTRL | CTRL | - | ½ | Í | Ý | í | ý |
| R | 009D | 009D | 00AD | 00BD | 00CD | 00DD | 00ED | 00FD |
| S | CTRL | CTRL | ® | ¾ | Î | Þ | î | þ |
| T | 009E | 009E | 00AE | 00BE | 00CE | 00DE | 00EE | 00FE |
| U | CTRL | CTRL | - | ½ | Ì | Þ | í | þ |
| V | 009F | 009F | 00AF | 00BF | 00CF | 00DF | 00EF | 00FF |

- Représentation en base 16 : 0000 à FFFF
- le caractère numéro 233 est le 00E9 : on trouve dans la table le « é »
- Chaque entrée UNICO DE offre une cinquantaine d'informations

AVANTAGES ET INCONVÉNIENTS

- **l'UNICODE représente tous les signes de toutes les langues du monde avec un numéro unique pour chacun !**
- Problèmes :
 - Pas toujours facile à manier pour les programmeurs
 - 2 octets : 2 fois plus volumineux que l'ASCII, alors que généralement peu de caractères non ASCII sont nécessaires dans une langue occidentale donnée...
 - 2 octets probablement insuffisants : évolution vers 4 octets
- Encodages différents : UCS-2, UCS-4, UTF-32, UTF-16, UTF-8...

LE GRAND GAGNANT ACTUEL :UTF-8

- Efficacité d'ASCII + étendu de UNICODE
 - Codage variable du caractère
 - ASCII en général
 - Si non ASCII : caractère spécial indiquant que le caractère suivant relève d'UNICODE
- Texte brut en français : **Bienvenu chez Sébastien** (23 octets ASCII ou 46 octets UNICODE)
- Texte encodé avec la norme UTF-8 :
 - **Bienvenu chez Sébastien** (28 octets UTF-8)
 - **Bienvenu chez Sébastien** (29 octets UTF-8)

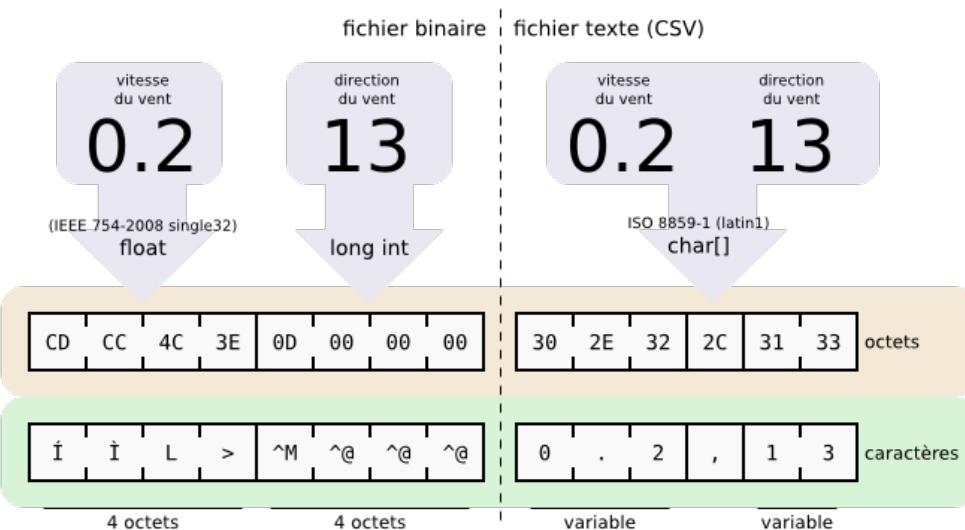
LES FICHIERS : OUVERTURE, FERMETURE, ENCODAGE

Astuce : le module **os** permet de gérer des fichiers hors du répertoire courant

```
>>> from os import getcwd, chdir, mkdir
>>> print(getcwd())
/home/fabrice
>>> chdir('/tps_intro_prog/tp_10')
>>> print(getcwd())
/home/fabrice//tps_intro_prog/tp_10
```

La gestion des fichiers comprend une **fonction native** d'ouverture de fichier et des **méthodes** de fermeture, lecture et écriture de fichier

```
with open(<fichier>, <mode>, encoding=<encodage>) as <reference> :
    <reference>.read() #ici le fichier est disponible
#ici le fichier n'est plus disponible (fermeture automatique)
```



```
with open('file.txt', 'r', encoding='utf-8') as f:
    texte = f.read()
```

4 modes d'ouverture (fichier texte - **encodage** doit être spécifié) :

- '**r**' : lecture (**mode** par défaut)
 - '**w**' : écriture (**fichier** créé ou écrasé)
 - '**a**' : écriture fin de **fichier** existant (création si non existant)
 - '**r+**' : lecture ET écriture
- Combinables avec '**b**' (fichier binaire - pas d'**encodage** unique) :

- '**rb**', '**wb**', '**ab**', '**r+b**'

Exemple de duplication d'un fichier texte

```
with open('copie.txt', 'w', encoding='utf-8') as f_out:
    with open('original.txt', encoding='utf-8') as f_in:
        f_out.write(f_in.read())
```

LES FICHIERS : EXEMPLE

```
>>>
```

test.txt

Mot1
Mot2
Mot3
Mot4



LES FICHIERS : EXEMPLE

```
>>> with open('test.txt', 'r', encoding='utf-8') as f:
```

test.txt

Mot1
Mot2
Mot3
Mot4



LES FICHIERS : EXEMPLE

test.txt

Mot1
Mot2
Mot3
Mot4

```
>>> with open('test.txt', 'r', encoding='utf-8') as f:  
    print(f.read(2))  
'Mo'
```

LES FICHIERS : EXEMPLE

test.txt

Mot1
Mot2
Mot3
Mot4



```
>>> with open('test.txt', 'r', encoding='utf-8') as f:  
    print(f.read(2))  
    print(f.readline())  
'Mo'  
't1\n'
```

LES FICHIERS : EXEMPLE

test.txt

Mot1

Mot2

Mot3

Mot4

```
>>> with open('test.txt', 'r', encoding='utf-8') as f:  
    print(f.read(2))  
    print(f.readline())  
    print(f.readlines())  
  
'Mo'  
't1\n'  
[ 'Mot2\n', 'Mot3\n', 'Mot4\n' ]
```

LES FICHIERS : EXEMPLE

test.txt

Mot1
Mot2
Mot3
Mot4



```
>>> with open('test.txt', 'r', encoding='utf-8') as f:  
    print(f.read(2))  
    print(f.readline())  
    print(f.readlines())  
  
'Mo'  
't1\n'  
['Mot2\n', 'Mot3\n', 'Mot4\n']  
>>> with open('test.txt', 'r', encoding='utf-8') as f:
```

LES FICHIERS : EXEMPLE

test.txt

Mot1

Mot2

Mot3

Mot4

```
>>> with open('test.txt', 'r', encoding='utf-8') as f:  
    print(f.read(2))  
    print(f.readline())  
    print(f.readlines())  
  
'Mo'  
't1\n'  
['Mot2\n', 'Mot3\n', 'Mot4\n']  
>>> with open('test.txt', 'r', encoding='utf-8') as f:  
    print(f.read())  
'Mot1\nMot2\nMot3\nMot4'
```

UN PEU PLUS SUR LES FICHIERS

- Lecture de la ligne courante depuis la position courante (retour à la ligne inclus) : `f.readline()`
- Lecture des lignes restantes d'un fichier (basée sur le retour à la ligne inclus) : `f.readlines()` ou `list(f)`
- Écriture à la position courante : `write()`, ne pas oublier le `'\n'` pour imposer un retour à la ligne
- Position courante exprimée en octets pour un fichier binaire (par un nombre obscur utile pour `f.seek` dans un fichier texte) : `f.tell()`
- Autres déplacements dans un fichier binaire : `f.seek(n, pos)` : +`n` octets à partir de `pos` 1 (début), 2 (courant) ou 3 (fin)
- Autres déplacements dans un fichier texte : `f.seek(0, 2)` pour aller à la fin du fichier ou `f.seek(val)` avec `val` obtenu par `f.tell()`

Bonjour à toi. Ne soit pas triste à la lecture de ce dernier slide du cours d'introduction à la programmation !

test.txt

```
>>> with open('test.txt', 'r', encoding='utf-8') as f:
>>>     texte = f.read()
>>>     mots = [mot.lower() for mot in texte.split() if mot[-1] == 'e']
>>>     print(mots)
['ne', 'triste', 'lecture', 'de', 'ce', 'slide']
```

ce
de
lecture
ne
slide
triste

res.txt

```
with open('res.txt', 'w', encoding='utf-8') as f:
    for mot in sorted(mots):
        f.write(mot + '\n')
```

YSINL2A1 - INTRODUCTION À LA POO EN PYTHON

2023-2024

Semaine 2/10 et 3/10 - 16/1/2024 et 24/1/2014
CM2 et CM3 - TP3 - TP4 - TP5 - TP6
dictionnaires/grilles/compréhensions/GIT



Ce pictogramme indique un exemple de code sur *ecampus*
dans un fichier commençant par NUMCHAP_NUMSLIDE

Fabrice Maurel - L1 INFO/MATH/MIASH - Semestre 2
fabrice.maurel@unicaen.fr - 02 31 56 73 97 - S3-364

DICTIONNAIRES (MUTABLES) : { DICT }

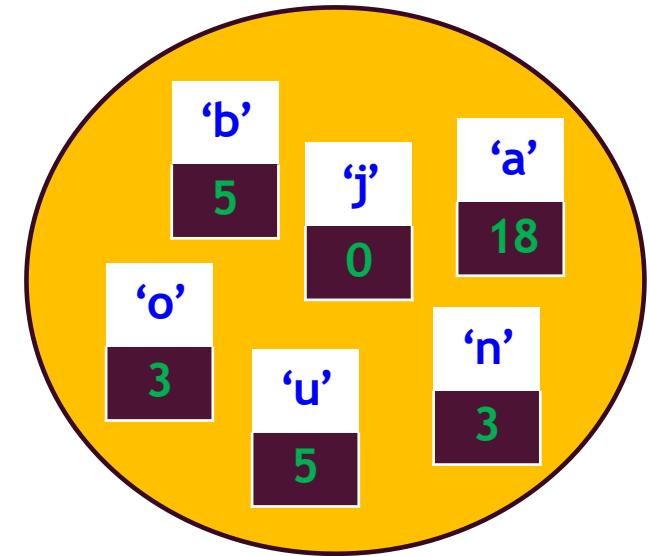
- Les dictionnaires sont de type **mutable**
- Ils combinent explicitement **deux listes** :
 - Une liste de **clés** pour les index (de n'importe quel type **non mutable**)
 - Une liste de **valeurs** (de n'importe quel type, associés aux **clés**)
- `d = {'b':5,'o':3,'n':3,'j':0,'u':5,'a':18}`
- `list(d.keys()) → ['a','o','n','b','u','j']`
- `list(d.values()) → [18,3,3,5,5,0]`
- `list(d.items()) → [('a', 18), ('o', 3), ('n', 3), ('b', 5), ('u', 5), ('j', 0)]`
- `d['j'] = 13 → {'b':5,'o':3,'n':3,'j':13,'u':5,'a':18}`
- `d['k'] = 13 → {'b': 5, 'o': 3, 'n': 3, 'j': 13, 'u': 5, 'k': 13, 'a': 18}`
- `print(d['m']) → KEY ERROR mais print(d.get('m',0)) → 0`
- Astuce < créer ou incrémenter > : `d['m'] = d.get('m',0) + 1`



`list` nécessaire pour afficher les listes de **clés**, de **valeurs** ou de paires (**clé**, **valeur**) ; mais pas pour les parcourir avec un `for` !



Un dictionnaire est par nature **non ordonné** : l'ordre de création des **clés** n'assure pas un parcours ou un affichage des **clés** dans le même ordre !



UN PEU PLUS SUR LES DICTIONNAIRES

Certaines **fonctions natives** sont très utiles pour gérer les dictionnaires :

```
>>> len({'a': 3, 'b': 0})
2
```

```
>>> dict([('a', 3), ('b', 0)])
{'a': 3, 'b': 0}
```

```
>>> dict(zip(['a', 'b'], [3, 0]))
{'a': 3, 'b': 0}
```

Astuce pour travailler sur un dictionnaire trié sur les clés



Astuce pour travailler sur un dictionnaire trié sur les valeurs

```
>>> d = {'b': 25, 'c': 10, 'a': 50}
>>> sorted(list(zip(d.keys(), d.values())))
[('a', 50), ('b', 25), ('c', 10)]
```

```
>>> d = {'b': 25, 'c': 10, 'a': 50}
>>> sorted(list(zip(d.values(), d.keys())))
[(10, 'c'), (25, 'b'), (50, 'a')]
```



Astuce : combiner la méthode `items` avec les **listes en compréhension** pour modifier clés et/ou valeurs d'un dictionnaire

```
>>> d = {'b': 25, 'c': 10, 'a': 50}
>>> dict([(cle, val*2) for cle, val in d.items() if cle not in 'aeiouy'])
>>> {'b': 50, 'c': 20}
```

COMPREHENSION DE LISTES : [MAP for element in sequence FILTER]

Soit n éléments e_i d'une liste $l = [e_0, e_1, e_2, \dots, e_{n-1}]$

$map(l, f) = [f(e_0), f(e_1), f(e_2), \dots, f(e_{n-1})]$

```
>>> words_length = []
>>> for word in words:
>>>     words_length.append(len(word))
[7, 1, 4]
```

```
>>> words_length = [len(word) for word in words]
[7, 1, 4]
```

$map_filter(l, f1, f2) = [tous les f1(e_i) tels que f2(e_i) == True]$

```
>>> long_words_length = []
>>> for word in words:
>>>     if len(word) > 1:
>>>         long_words_length.append(len(word))
[7, 4]
```

```
>>> words = ["Bonjour", "à", "tous"]
```

$filter(l, f) = [tous les e_i tels que f(e_i) == True]$

```
>>> long_words = []
>>> for word in words:
>>>     if len(word) > 1:
>>>         long_words.append(word)
["Bonjour", "tous"]
```

```
>>> long_words = [word for word in words if len(word) > 1]
["Bonjour", "tous"]
```

COMPREHENSION DE LISTES : [MAP for element in sequence FILTER]

❤️ **Astuce** : penser à utiliser la syntaxe des listes en compréhension dès que le nombre d'itération est connu et qu'il est utile de construire une liste **à partir d'une autre liste**

```
>>> any('o' in word for word in words)
True
>>> all('o' in word for word in words)
False
>>> all('o' in word for word in words if len(word) > 1)
True
>>> sum(len(word) for word in words) / len(words)
4.0
```

COMPREHENSION DE dictionnaires : {MAP for element in sequence FILTER}

```
>>> words_length = {}  
>>> for word in words:  
>>>     words_length[word] = len(word)  
{'bonjour': 7, 'à': 1, 'tous': 4}
```

```
>>> dict((word, len(word)) for word in words)  
{'bonjour': 7, 'à': 1, 'tous': 4}
```

```
>>> {word: len(word) for word in words}  
{'bonjour': 7, 'à': 1, 'tous': 4}
```

```
>>> long_words_length = {}  
>>> for word in words:  
>>>     if len(word) > 1:  
>>>         long_words_length[word] = len(word)  
{'bonjour': 7, 'tous': 4}
```

```
>>> dict((word, len(word)) for word in words if len(word) > 1)  
{'bonjour': 7, 'tous': 4}
```

```
>>> {word: len(word) for word in words if len(word) > 1}  
{'bonjour': 7, 'tous': 4}
```

NOTION DE GRILLES

| | 0 | 1 | 2 | 3 | 4 | 5 |
|---|----|----|----|----|----|----|
| 0 | 0 | 1 | 2 | 3 | 4 | 5 |
| 1 | 6 | 7 | 8 | 9 | 10 | 11 |
| 2 | 12 | 13 | 14 | 15 | 16 | 17 |
| 3 | 18 | 19 | 20 | 21 | 22 | 23 |
| 4 | 24 | 25 | 26 | 27 | 28 | 29 |

`nb_lig = 5`
`nb_col = 6`



Cases repérées par un entier de 0 à `nb_lig * nb_col - 1`

`(i, j)`

Cases repérées par un numéro de ligne de 0 à `nb_lig - 1` et un numéro de colonne de 0 à `nb_col - 1`

NOTION DE GRILLES

| | 0 | 1 | 2 | 3 | 4 | 5 |
|---|----|----|----|----|----|----|
| 0 | 0 | 1 | 2 | 3 | 4 | 5 |
| 1 | 6 | 7 | 8 | 9 | 10 | 11 |
| 2 | 12 | 13 | 14 | 15 | 16 | 17 |
| 3 | 18 | 19 | 20 | 21 | 22 | 23 |
| 4 | 24 | 25 | 26 | 27 | 28 | 29 |

`nb_lig = 5`
`nb_col = 6`

Case

Cases repérées par un entier de 0 à `nb_lig * nb_col - 1`

(i, j)

Cases repérées par un numéro de ligne de 0 à `nb_lig - 1` et un numéro de colonne de 0 à `nb_col - 1`

`case = i * nb_col + j`

`20 = 3 * 6 + 2`

`i = case // nb_col`

`3 = 20 // 6`

`j = case % nb_col`

`2 = 20 % 6`

UNE REPRÉSENTATION « NATURELLE » EN PYTHON

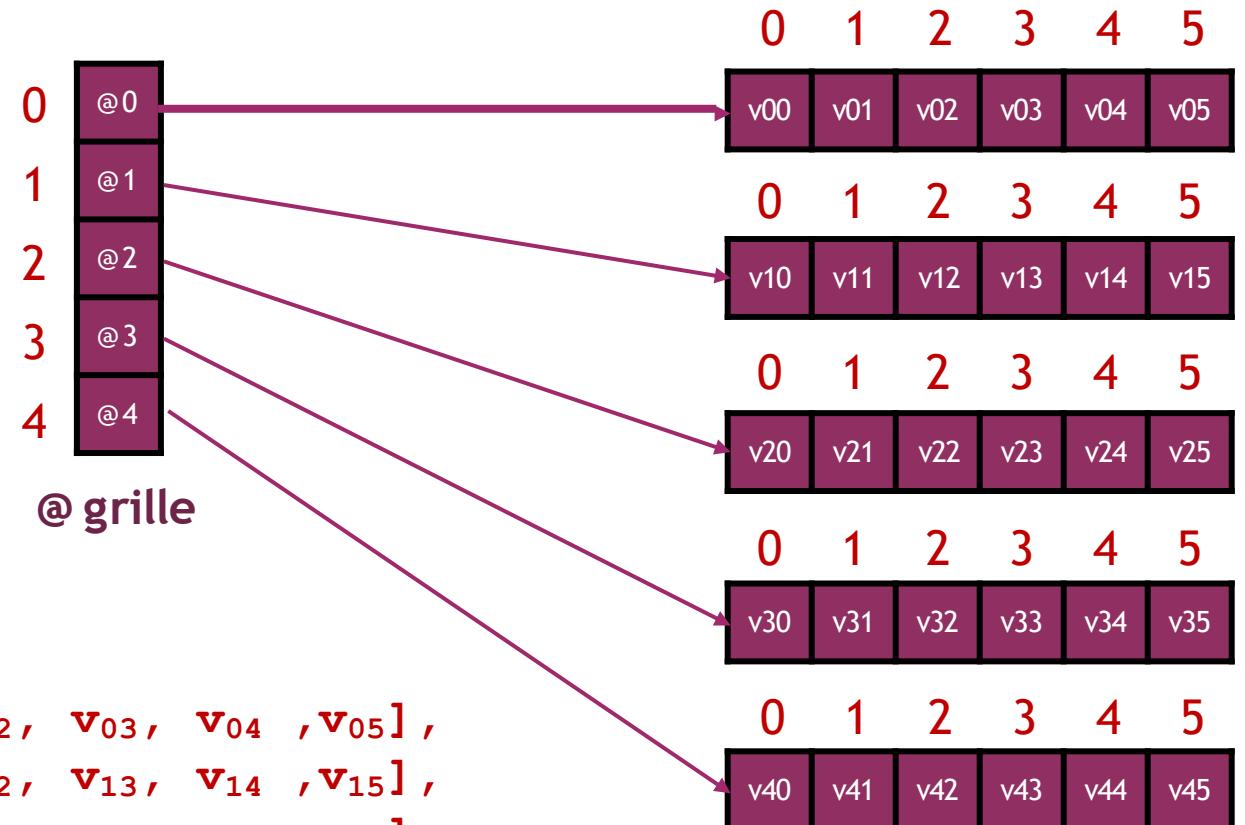
| | 0 | 1 | 2 | 3 | 4 | 5 |
|---|-----|-----|-----|-----|-----|-----|
| 0 | v00 | v01 | v02 | v03 | v04 | v05 |
| 1 | v10 | v11 | v12 | v13 | v14 | v15 |
| 2 | v20 | v21 | v22 | v23 | v24 | v25 |
| 3 | v30 | v31 | v32 | v33 | v34 | v35 |
| 4 | v40 | v41 | v42 | v43 | v44 | v45 |

SUD (2)

nb_lig = 5
nb_col = 6

v_{ij} : valeur ligne i
et colonne j

```
grille = [
    [v00, v01, v02, v03, v04, v05],
    [v10, v11, v12, v13, v14, v15],
    [v20, v21, v22, v23, v24, v25],
    [v30, v31, v32, v33, v34, v35],
    [v40, v41, v42, v43, v44, v45]]
```

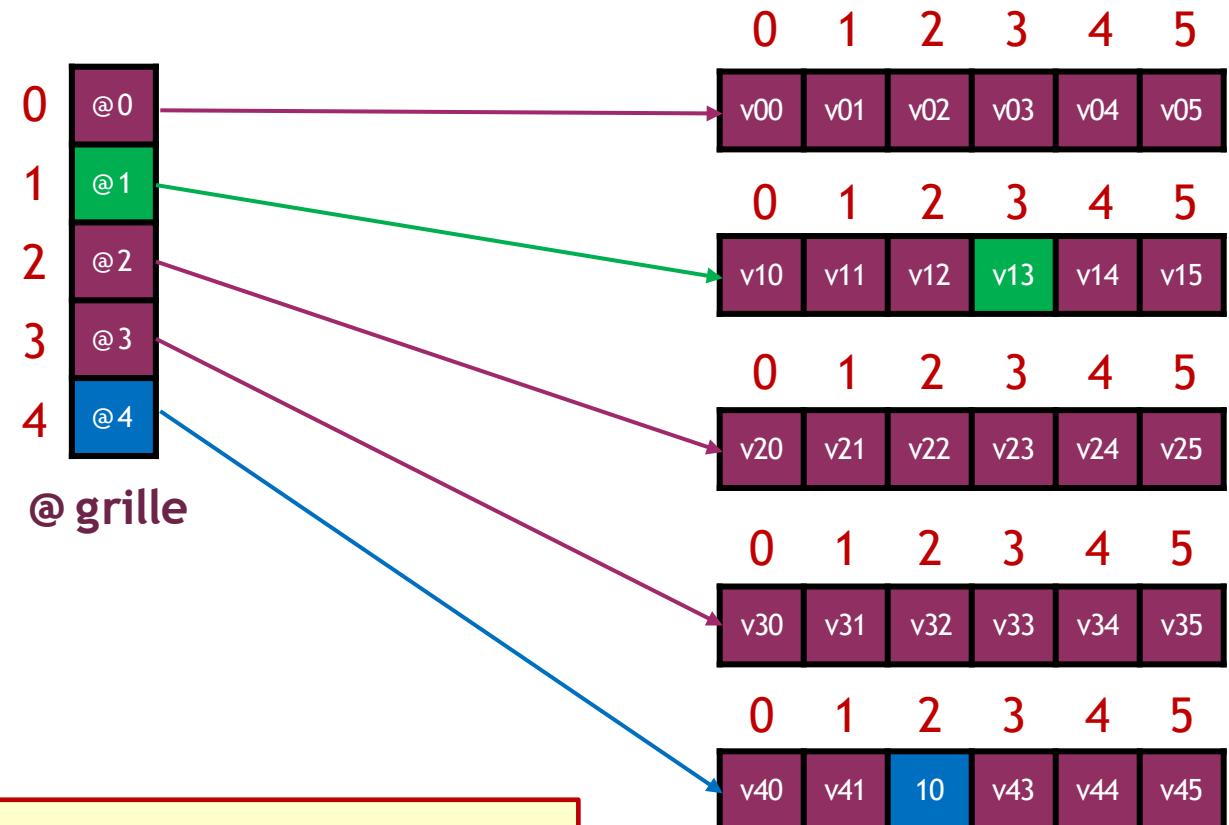


UNE REPRÉSENTATION « NATURELLE » EN PYTHON

```
grille = [
    [v00, v01, v02, v03, v04, v05],
    [v10, v11, v12, v13, v14, v15],
    [v20, v21, v22, v23, v24, v25],
    [v30, v31, v32, v33, v34, v35],
    [v40, v41, v42, v43, v44, v45]]
```

Gestion par numéros de ligne et de colonne

```
>>> print(grille[1][3])
v13
>>> grille[4][2] = 10
```



! Il serait malgré tout intéressant de pouvoir passer d'un système de représentation en ligne, colonne à un autre en fonction des objectifs de manipulation applicative.

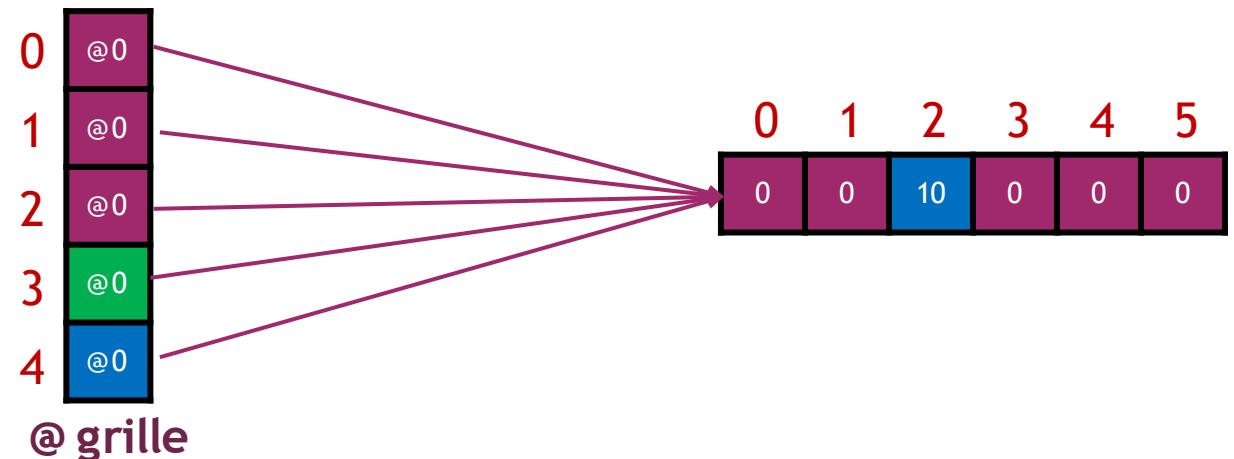
GÉNÉRATION INCORRECTE D'UNE GRILLE EN PYTHON



```
def creer(nb_lig, nb_col, init):
    return [[init] * nb_col] * nb_lig
```

```
>>> grille = creer(5, 6, 0)
>>> grille[4][2] = 10
>>> print(grille[3][2])
10
```

Attention à l'opérateur * pour dupliquer une liste...

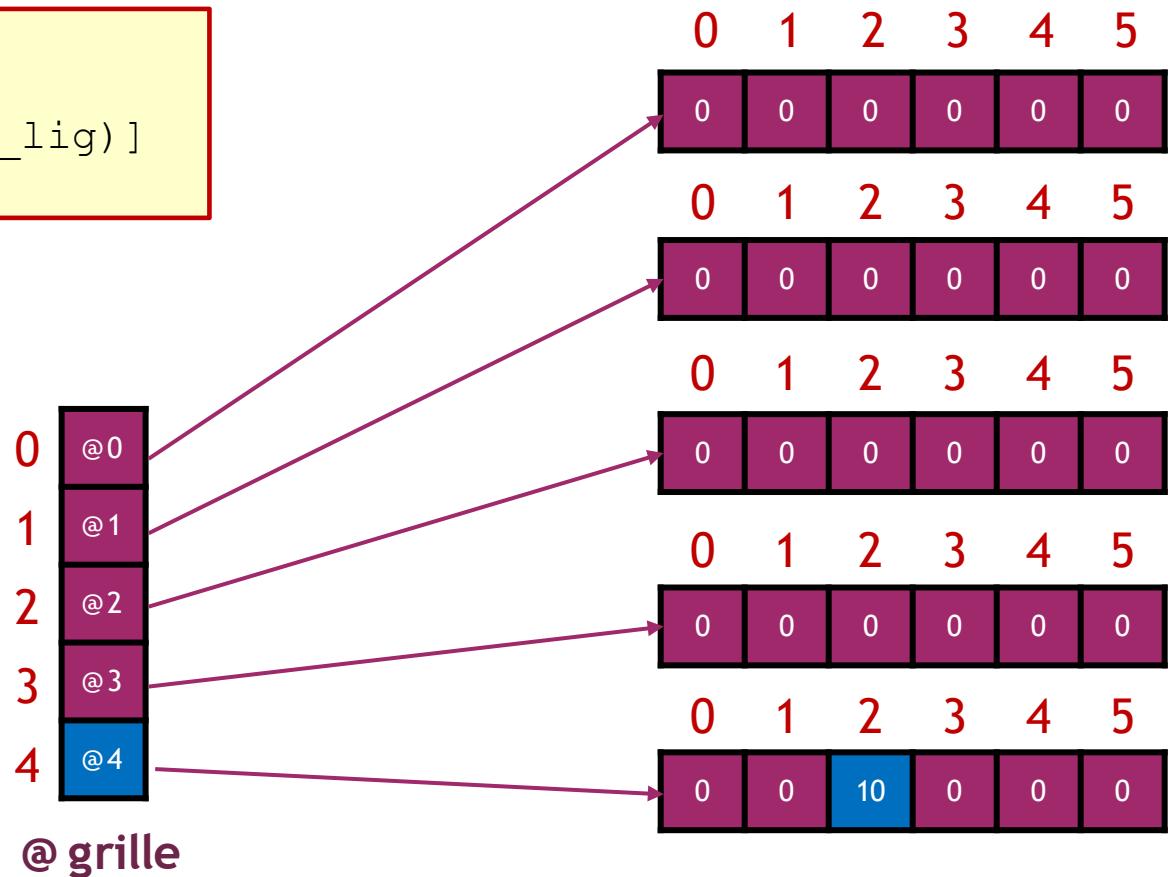


GÉNÉRATION CORRECTE D'UNE GRILLE EN PYTHON

```
def creer(nb_lig, nb_col, init):
    return [[init] * nb_col for _ in range(nb_lig)]
```

```
def creer(nb_lig, nb_col, init):
    res = []
    for i in range(nb_lig):
        ligne = []
        for j in range(nb_col):
            ligne.append(init)
        res.append(ligne)
    return res
```

```
>>> grille = creer(5, 6, 0)
>>> grille[4][2] = 10
```

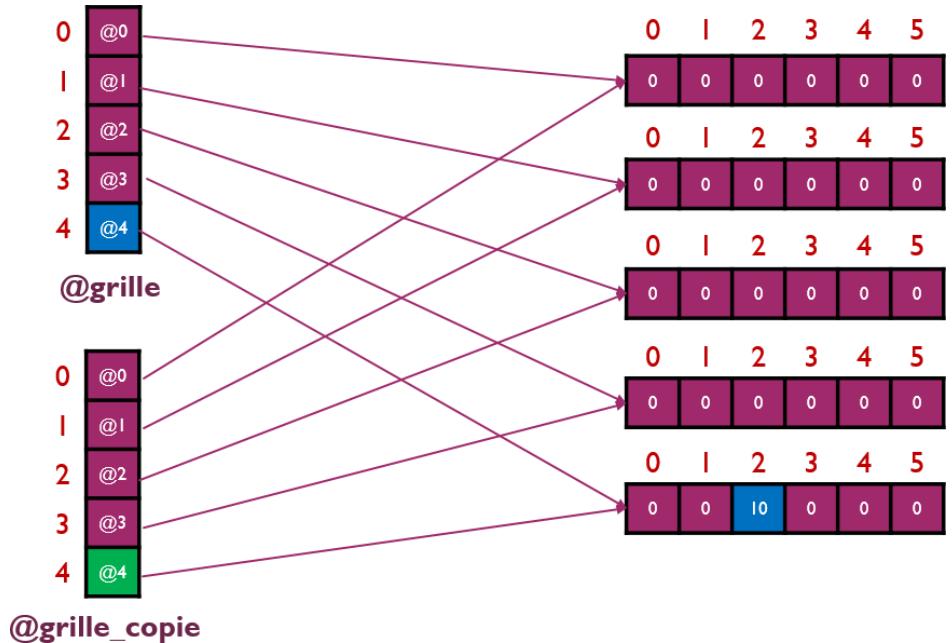


GÉNÉRATION INCORRECTE D'UNE GRILLE EN PYTHON

L'opération `[:]` pour faire une copie est inadaptée aux grilles



```
>>> grille = creer_correct(5, 6, 0)
>>> grille_copie = grille[:]
>>> grille[4][2] = 10
>>> print(grille_copie[4][2])
10
```



COPIE DE GRILLE

Une solution est de faire sa propre fonction comme `copy_iteratif`

- 1

```
def copy_iteratif(l):
    res = []
    for line in l:
        new_line = []
        for val in line:
            new_line.append(val)
        res.append(new_line)
    return res
```

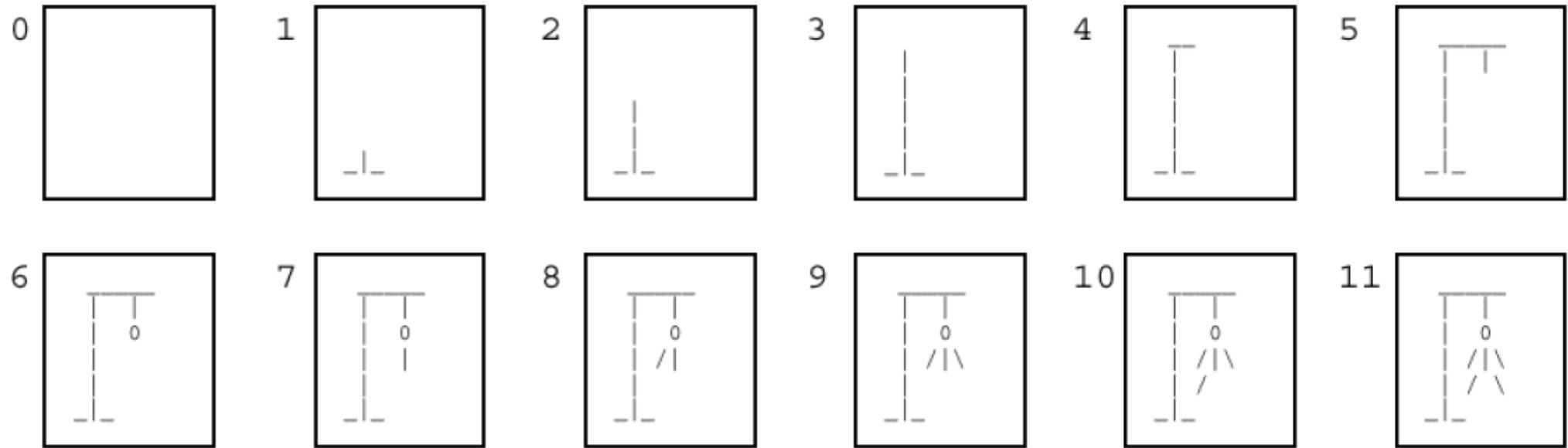
Deux autres solutions :

- 2 Utilisation des **listes en compréhension** (`line[:]` peut être remplacé par `list(line)`)
 - ! Sans copie explicite, chaque sous-liste a été copiée par référence !
- 3 Utilisation du module `deepcopy` (fonctionne quelque soit le niveau d'imbrication)

- 1

```
>>> grille = [[1, 2], [3, 4]]
>>> autre_grille1 = copy_iteratif(grille)
>>> autre_grille2bad = [line for line in grille] !
>>> autre_grille2 = [line[:] for line in grille]
>>> from copy import deepcopy
>>> autre_grille3 = deepcopy(grille)
>>> grille[1][0] = 100
>>> grille
[[1, 2], [100, 4]]
>>> autre_grille1
[[1, 2], [3, 4]]
>>> autre_grille2bad
[[1, 2], [100, 4]]
>>> autre_grille2
[[1, 2], [3, 4]]
>>> autre_grille3
[[1, 2], [3, 4]]
```

EXEMPLE : LE PENDU ALPHANUMÉRIQUE



Comment gérer un tel affichage en fonction du nombre d'erreurs dans les lettres proposées ?

EXEMPLE : DEUX TABLEAUX MULTIDIMENSIONNELS

| | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| 0 | - | - | - | - | - | - |
| 1 | | | | | | |
| 2 | | | | o | | |
| 3 | | / | | \ | | |
| 4 | | / | | \ | | |
| 5 | - | | - | | | |

② Numéro de l'erreur pour l'apparition du caractère en (i, j)

| | | | | | |
|---|---|---|----|---|----|
| 0 | 4 | 4 | 5 | 5 | 5 |
| 0 | 3 | 0 | 0 | 5 | 0 |
| 0 | 3 | 0 | 0 | 6 | 0 |
| 0 | 2 | 0 | 8 | 7 | 9 |
| 0 | 2 | 0 | 10 | 0 | 11 |
| 1 | 1 | 1 | 0 | 0 | 0 |

Fonctions

- Dessiner le pendu
- Incrémenter erreurs
- Test victoire
- Caractère en i, j en fonction du nombre d'erreurs
- Dessiner les vignettes en fonction du nombre d'erreurs

Variables globales

- tab_pendu et tab_limite
- nb_erreur

EXEMPLE : DESSINER LE PENDU EN ENTIER

```
tab_pendu = [
    [' _ ', '| ', '| ', '| ', '| ', '| ', '| '],
    [' _ ', '| ', '| ', '| ', '| ', '| ', '| '],
    [' _ ', '| ', '| ', '| ', '| ', '| ', '| '],
    [' _ ', '| ', '| ', '| ', '| ', '| ', '| '],
    [' _ ', '| ', '| ', '| ', '| ', '| ', '| '],
    [' _ ', '| ', '| ', '| ', '| ', '| ', '| ']
]
```

```
for i in range(6):
    for j in range(6):
        print(tab_pendu[i][j], end=' ')
print()
```

```
for ligne in tab_pendu:
    for car in ligne:
        print(car, end=' ')
    print()
```

Dépôt distant GIT pour les TPs (1)

The screenshot shows a web browser window with the following details:

- Address Bar:** https://forge.info.unicaen.fr/projects/poo_tps_2023?jump=welcome
- Toolbar:** Includes standard browser icons for back, forward, search, and refresh.
- Header:** Shows the Unicaen Forge navigation bar with links like Getting Started, selection web, tutos design, Planning Studio, photos, Google Docs - Home, Académie de Caen, cours, Prochaines retransmises..., Diffusions en direct, li..., Z-UP, test, Chaussures Reebok fr..., Maps, Actualités, Débuter avec Firefox, and Autres marque-pages.
- User Information:** Connecté en tant que maurelf, Mon compte, Déconnexion.
- Project Navigation:** Accueil, Ma page, Projets, redmine.org/Aide, POO_L1S2_2023 > POO_TPs_2023.
- Project Header:** POO_TPs_2023.
- Project Tabs:** Aperçu (selected), Activité, Supprimer projet, Demandes, Nouvelle demande, Gantt, Annonces, Documents, Wiki, Fichiers, Dépôt, Configuration.
- Aperçu Tab Content:**
 - Sous-projets: MAUREL_FABRICE, MAUREL_FABRICE, p2o malterre
 - Suivi des demandes:** Voir toutes les demandes | Gantt
- Membres Tab Content:** Manager: Fabrice Maurel. SubManager: Abdellah Ahbani, Abderrahim Berrabia, Abdoulaye Kane, Abdurrahim Ouahabi, Aboubacar Dao, Aboubacary Diallo, Achraf Zaghbani, Adam El Rhana, Adam Hilali, Adele Janis, Adele Orleach, Adrian Romanowicz, Adrien Renault, Agate Lefebvre, Ahmed Boukheir, Ahmet Erdem, Aissata Niang, Alan Grysant, Albane Germanicus--Corvellec, Alexandre Gaudin, Alexandre Marquilly, Alexandre Mehring, Alexis Anselme, Alexis Menager, Ali Atteib Atteib, Ali Zouzal, Alice Bertrand, Alice Mehue, Alicia Mechroub, Allan Bendjaballah, Amadou Siby, Ambre Thomine, Ambroise Mouget, Amelia Maugendre, Ameline Ribault, Amin Allaoui, Amine Ahjam, Amirdine Ibrahim, Amirmohammad Ezatabadipoor, Anatole Colin, Andrea Cousin, Andrea Gjoreska, Andrii Rogava, Anouar Abdoul-Bassit, Anthony Viallon, Antonin Coutable, Antonn Marquant, Ardivale Samba Bilenga, Arion Barua, Aristide Henry, Armel Bouanga, Arthur Lete, Audrey Le Basnier, Audrey Scordel, Aurelien Giguere, Axel Lailler--Lesage, Aya Jaber, Aya Nour Elimane Sahbi, Ayila Laleye, Ayman Cherif, Aymen Lahssini, Aymen Tarek Beddar, Aymeric Leboucher, Azya Tshilanda Mwamba, Babacar Mbaye, Babacar Ndiaye, Baptiste Charrue, Baptiste Gillio, Basile Patte, Basile Tellier, Bellah Bellah, Ben Sanou, Benjamin Demaret--Feillu, Benjamin Helou, Benjamin Lemore, Benjamin Lolamou, Benjamin Vrauko, Betul Isci, Bineta Sarr, Bleunig Guarnieri, Brandon Hu, Brice Ramette, Camille Bertaud, Camille Esnault, Camille Loiseau, Camille Suvigny, Carlos Basse, Cedric Briant, Charline Lapersonne, Charlotte Levallois, Chayman Borjon, Chloe Pereira Da Silva, Christophe Dugouset-Pasquet, Clairy Nibuturonsa, Clara Croissant, Clemence Alvin, Clemence Roussel, Clement
- Temps passé:** 0.00 heure. Saisir temps | Détails | Rapport.

- Connexion à la forge de Unicaen : <https://forge.info.unicaen.fr>
- Menu Aller à un projet...
- Choisir le projet **POO_TPs_2023**
- Choisir **Nouveau sous-projet**

Dépôt distant GIT pour les TPs (2)

https://forge.info.unicaen.fr/projects/new?parent_id=poo_tp_2023

Getting Started selection web tutos design Planning Studio photos Google Docs - Home Académie de Caen 1 cours Prochaines retransmises... Diffusions en direct, li... Z-UP test Chaussures Reebok fr... Maps Actualités Débuter avec Firefox Autres marque-pages

Accueil Ma page Projets redmine.org/Aide Connecté en tant que maurelf Mon compte Déconnexion

Recherche: Aller à un projet...

Nouveau projet

Nom * MAUREL_FABRICE

Description

Identifiant * maurel_fabrice

Longueur comprise entre 1 et 100 caractères. Seuls les lettres minuscules (a-z), chiffres, tirets et tirets bas sont autorisés, doit commencer par une minuscule.

Un fois sauvégarde, l'identifiant ne pourra plus être modifié.

Site web

Public

Sous-projet de » POO_TPs_2023

Hériter les membres

Modules

Suivi des demandes Suivi du temps passé Publication d'annonces Publication de documents Publication de fichiers Wiki
 Dépôt de sources Forums de discussion Calendrier Gantt Supprimer projet

Trackers

Anomalie Evolution Assistance Tâche

Créer **Créer et continuer**

- Saisir **NOM_PRENOM**
- Vérifier **paramétrage**
- Bouton **Créer**

Dépôt distant GIT pour les TPs (3)

The screenshot shows a Redmine interface for a project named "POO_L1S2_2023". The top navigation bar includes links like "Getting Started", "selection web", "tutos design", "Planning Studio", "photos", "Google Docs - Home", "Académie de Caen", "cours", "Prochaines retransmis...", "Diffusions en direct, li...", "Z-UP", "test", "Chaussures Reebok fr...", "Maps", "Actualités", "Débuter avec Firefox", and "Autres marque-pages". The user is connected as "maurelf".

The main content area shows the "Configuration" section under "Dépôt". The "Membres" tab is selected. A search bar at the top right contains "» MAUREL_FABRICE".

The "Utilisateur / Groupe" table lists "Fabrice Maurel" with the role "Manager". There are "Modifier" and "Supprimer" buttons for this entry.

The "Nouveau membre" sidebar shows a search for "lecarpe", with one result found: "Jean-marc Lecarpentier" (checked). The "Rôles:" section includes "Manager" (checked), "SubManager", "Développeur", and "Rapporteur". An "Ajouter" button is present.

- Choisir le dossier **NOM_PRENOM**
- Menu **configuration** – Sous-menu **Membres**
- Ajouter votre **encadrant de TP et moi-même** en **Manager**

Dépôt distant GIT pour les TPs (4)

The screenshot shows a Redmine project management interface. At the top, there is a navigation bar with links to various services like Getting Started, selection web, tutos design, Planning Studio, photos, Google Docs - Home, Académie de Caen, cours, Prochaines retransmis..., Diffusions en direct, li..., Z-UP, test, Chaussures Reebok fr..., Maps, Actualités, Débuter avec Firefox, and Autres marque-pages. Below the navigation bar, there is a breadcrumb trail: Accueil > Ma page > Projets > redmine.org/Aide > POO_L1S2_2023 > POO_TPs_2023 > MAUREL_FABRICE. On the right side of the header, there are links for Connecté en tant que maurelf, Mon compte, and Déconnexion. A search bar is also present.

The main content area has a green header bar with links: Aperçu, Activité, Supprimer projet, Demandes, Nouvelle demande, Gantt, Annonces, Documents, Wiki, Fichiers, Dépôt (which is highlighted), and Configuration.

The main content area displays a Git repository page for the project "MAUREL_FABRICE". It shows the command to clone the repository: `git clone https://maurelf@forge.info.unicaen.fr/git/maurel_fabrice`. Below this, there is a table showing the latest revision:

| Nom | Taille |
|-----|--------|
| | |

Below the table, there is a section titled "Dernières révisions" showing the following data:

| # | Date | Auteur | Commentaire |
|----------|------------------|----------|----------------|
| 2f04cf2b | 22/01/2024 00:15 | Sysadmin | Initial commit |

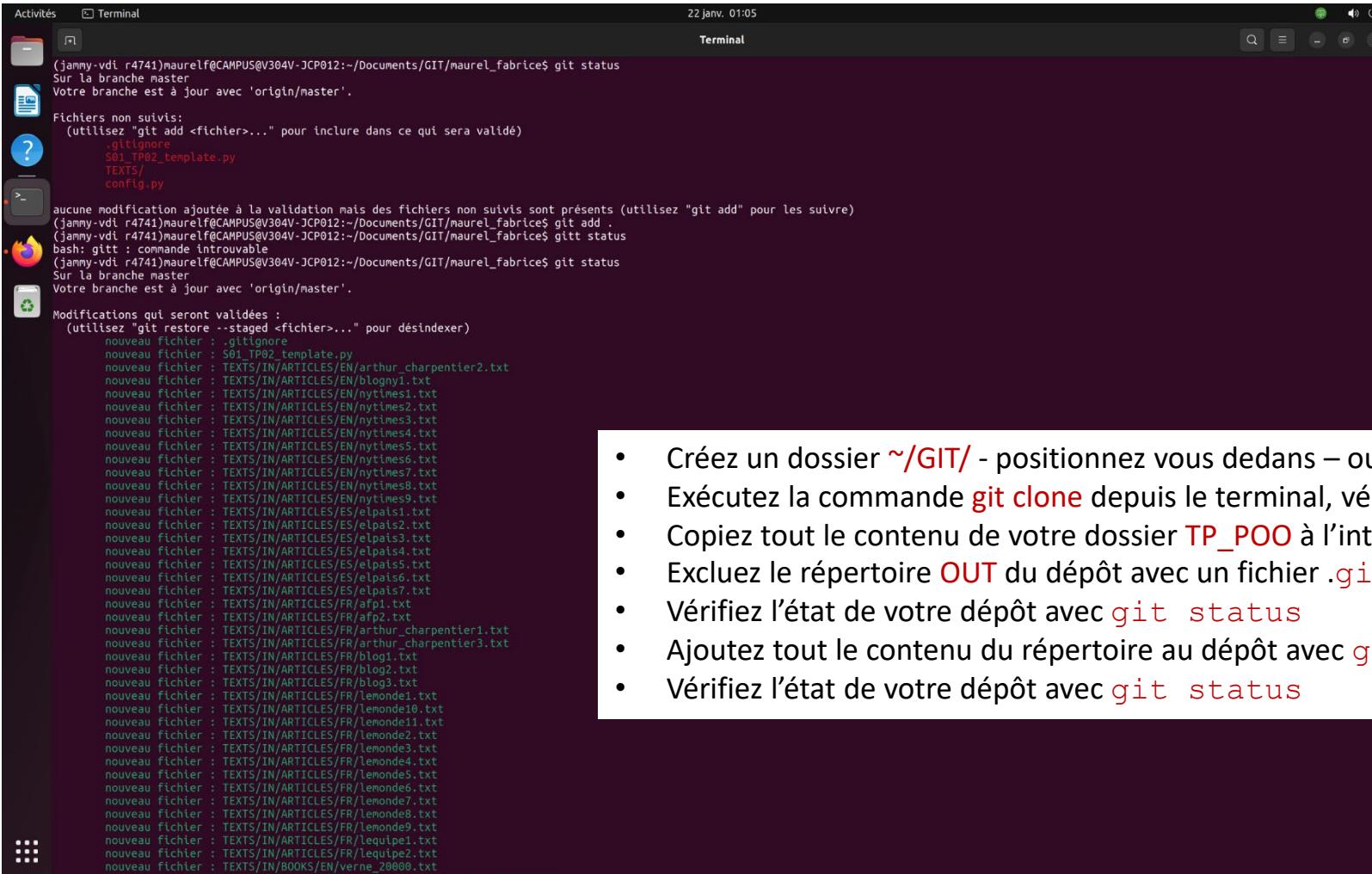
At the bottom left, there are links to "Voir toutes les révisions" and "Voir les révisions". At the bottom right, it says "Formats disponibles : Atom".

On the right side of the page, there is a sidebar titled "Dépôts" which lists "Dépôt principal" and "Subversion".

- Menu **Dépôt**
- Repérer la commande **GIT** de création du **dépôt local**

`git clone https://persopass@forge.info.unicaen.fr/git/nom_prenom`

Dépôt distant GIT pour les TPs (5)



```
(jammy-vdi r4741)maurelf@CAMPUS@V304V-JCP012:~/Documents/GIT/maurel_fabrice$ git status
Sur la branche master
Votre branche est à jour avec 'origin/master'.

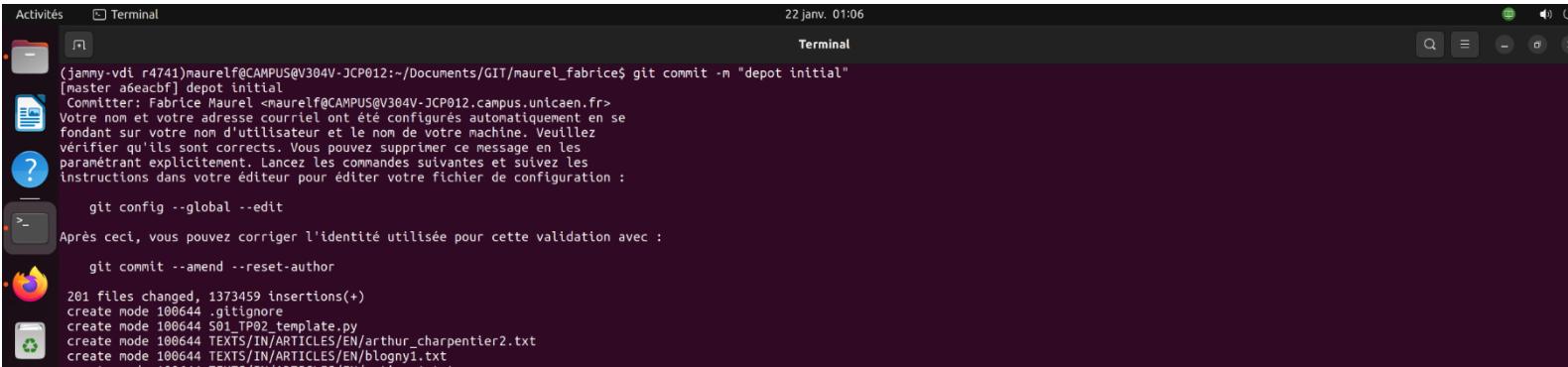
Fichiers non suivis:
  (utilisez "git add <fichier>..." pour inclure dans ce qui sera validé)
    .gitignore
    S01_TP02_template.py
    TEXTS/
    config.py

aucune modification ajoutée à la validation mais des fichiers non suivis sont présents (utilisez "git add" pour les suivre)
(jammy-vdi r4741)maurelf@CAMPUS@V304V-JCP012:~/Documents/GIT/maurel_fabrice$ git status
bash: gitt : commande introuvable
(jammy-vdi r4741)maurelf@CAMPUS@V304V-JCP012:~/Documents/GIT/maurel_fabrice$ git status
Sur la branche master
Votre branche est à jour avec 'origin/master'.

Modifications qui seront validées :
  (utilisez "git restore --staged <fichier>..." pour désindexer)
    nouveau fichier : .gitignore
    nouveau fichier : TEXTS/IN/ARTICLES/EN/arthur_charpentier2.txt
    nouveau fichier : TEXTS/IN/ARTICLES/EN/blogny1.txt
    nouveau fichier : TEXTS/IN/ARTICLES/EN/nytimes1.txt
    nouveau fichier : TEXTS/IN/ARTICLES/EN/nytimes2.txt
    nouveau fichier : TEXTS/IN/ARTICLES/EN/nytimes3.txt
    nouveau fichier : TEXTS/IN/ARTICLES/EN/nytimes4.txt
    nouveau fichier : TEXTS/IN/ARTICLES/EN/nytimes5.txt
    nouveau fichier : TEXTS/IN/ARTICLES/EN/nytimes6.txt
    nouveau fichier : TEXTS/IN/ARTICLES/EN/nytimes7.txt
    nouveau fichier : TEXTS/IN/ARTICLES/EN/nytimes8.txt
    nouveau fichier : TEXTS/IN/ARTICLES/EN/nytimes9.txt
    nouveau fichier : TEXTS/IN/ARTICLES/ES/elpais1.txt
    nouveau fichier : TEXTS/IN/ARTICLES/ES/elpais2.txt
    nouveau fichier : TEXTS/IN/ARTICLES/ES/elpais3.txt
    nouveau fichier : TEXTS/IN/ARTICLES/ES/elpais4.txt
    nouveau fichier : TEXTS/IN/ARTICLES/ES/elpais5.txt
    nouveau fichier : TEXTS/IN/ARTICLES/ES/elpais6.txt
    nouveau fichier : TEXTS/IN/ARTICLES/ES/elpais7.txt
    nouveau fichier : TEXTS/IN/ARTICLES/FR/afp1.txt
    nouveau fichier : TEXTS/IN/ARTICLES/FR/afp2.txt
    nouveau fichier : TEXTS/IN/ARTICLES/FR/arthur_charpentier1.txt
    nouveau fichier : TEXTS/IN/ARTICLES/FR/arthur_charpentier3.txt
    nouveau fichier : TEXTS/IN/ARTICLES/FR/blog1.txt
    nouveau fichier : TEXTS/IN/ARTICLES/FR/blog2.txt
    nouveau fichier : TEXTS/IN/ARTICLES/FR/blog3.txt
    nouveau fichier : TEXTS/IN/ARTICLES/FR/lemonde1.txt
    nouveau fichier : TEXTS/IN/ARTICLES/FR/lemonde10.txt
    nouveau fichier : TEXTS/IN/ARTICLES/FR/lemonde11.txt
    nouveau fichier : TEXTS/IN/ARTICLES/FR/lemonde2.txt
    nouveau fichier : TEXTS/IN/ARTICLES/FR/lemonde3.txt
    nouveau fichier : TEXTS/IN/ARTICLES/FR/lemonde4.txt
    nouveau fichier : TEXTS/IN/ARTICLES/FR/lemonde5.txt
    nouveau fichier : TEXTS/IN/ARTICLES/FR/lemonde6.txt
    nouveau fichier : TEXTS/IN/ARTICLES/FR/lemonde7.txt
    nouveau fichier : TEXTS/IN/ARTICLES/FR/lemonde8.txt
    nouveau fichier : TEXTS/IN/ARTICLES/FR/lemonde9.txt
    nouveau fichier : TEXTS/IN/ARTICLES/FR/lequipe1.txt
    nouveau fichier : TEXTS/IN/ARTICLES/FR/lequipe2.txt
    nouveau fichier : TEXTS/IN/BOOKS/EN/verne_20000.txt
```

- Créez un dossier **~/GIT/** - positionnez vous dedans – ouvrez un terminal depuis ce dossier
- Exécutez la commande **git clone** depuis le terminal, vérifiez la bonne création du dossier **~/GIT/nom_prenom**
- Copiez tout le contenu de votre dossier **TP_POO** à l'intérieur
- Excluez le répertoire **OUT** du dépôt avec un fichier **.gitignore** contenant **TEXT/OUT/**
- Vérifiez l'état de votre dépôt avec **git status**
- Ajoutez tout le contenu du répertoire au dépôt avec **git add .**
- Vérifiez l'état de votre dépôt avec **git status**

Dépôt distant GIT pour les TPs (6)



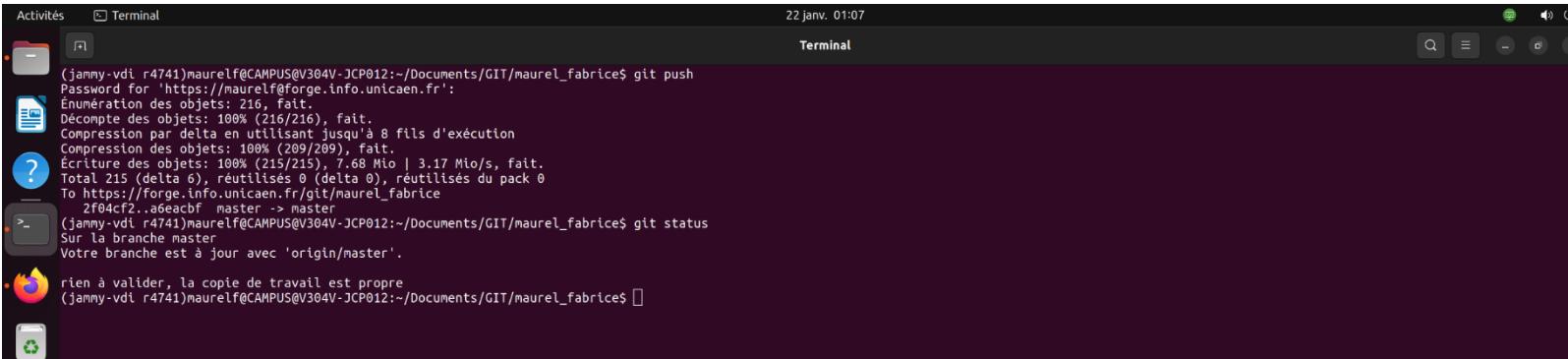
```
(jammy-vdi r4741)maurelf@CAMPUS@V304V-JCP012:~/Documents/GIT/maurel_fabrice$ git commit -m "depot initial"
[master a6eacbf] depot initial
  Committer: Fabrice Maurel <maurelf@CAMPUS@V304V-JCP012.campus.unicaen.fr>
  Votre nom et votre adresse courriel ont été configurés automatiquement en se
  fondant sur votre nom d'utilisateur et le nom de votre machine. Veuillez
  vérifier qu'ils sont corrects. Vous pouvez supprimer ce message en les
  paramétrant explicitement. Lancez les commandes suivantes et suivez les
  instructions dans votre éditeur pour éditer votre fichier de configuration :

    git config --global --edit
Après ceci, vous pouvez corriger l'identité utilisée pour cette validation avec :
  git commit --amend --reset-author

  201 files changed, 1373459 insertions(+)
 create mode 100644 .gitignore
 create mode 100644 S01_TP02_template.py
 create mode 100644 TEXTS/IN/ARTICLES/EN/arthur_charpentier2.txt
 create mode 100644 TEXTS/IN/ARTICLES/EN/blogny1.txt
 create mode 100644 TEXTS/IN/ARTICLES/EN/nytimes1.txt
 create mode 100644 TEXTS/IN/ARTICLES/EN/nytimes2.txt
 create mode 100644 TEXTS/IN/ARTICLES/EN/nytimes3.txt
 create mode 100644 TEXTS/IN/ARTICLES/EN/nytimes4.txt
 create mode 100644 TEXTS/IN/ARTICLES/EN/nytimes5.txt
 create mode 100644 TEXTS/IN/ARTICLES/EN/nytimes6.txt
 create mode 100644 TEXTS/IN/ARTICLES/EN/nytimes7.txt
 create mode 100644 TEXTS/IN/ARTICLES/EN/nytimes8.txt
 create mode 100644 TEXTS/IN/ARTICLES/EN/nytimes9.txt
 create mode 100644 TEXTS/IN/ARTICLES/ES/elpais1.txt
 create mode 100644 TEXTS/IN/ARTICLES/ES/elpais2.txt
 create mode 100644 TEXTS/IN/ARTICLES/ES/elpais3.txt
 create mode 100644 TEXTS/IN/ARTICLES/ES/elpais4.txt
 create mode 100644 TEXTS/IN/ARTICLES/ES/elpais5.txt
 create mode 100644 TEXTS/IN/ARTICLES/ES/elpais6.txt
 create mode 100644 TEXTS/IN/ARTICLES/FR/afp1.txt
 create mode 100644 TEXTS/IN/ARTICLES/FR/afp2.txt
 create mode 100644 TEXTS/IN/ARTICLES/FR/arthur_charpentier1.txt
 create mode 100644 TEXTS/IN/ARTICLES/FR/arthur_charpentier3.txt
 create mode 100644 TEXTS/IN/ARTICLES/FR/blog1.txt
 create mode 100644 TEXTS/IN/ARTICLES/FR/blog2.txt
 create mode 100644 TEXTS/IN/ARTICLES/FR/blog3.txt
 create mode 100644 TEXTS/IN/ARTICLES/FR/lemonde1.txt
 create mode 100644 TEXTS/IN/ARTICLES/FR/lemonde10.txt
 create mode 100644 TEXTS/IN/ARTICLES/FR/lemonde11.txt
 create mode 100644 TEXTS/IN/ARTICLES/FR/lemonde2.txt
 create mode 100644 TEXTS/IN/ARTICLES/FR/lemonde3.txt
 create mode 100644 TEXTS/IN/ARTICLES/FR/lemonde4.txt
 create mode 100644 TEXTS/IN/ARTICLES/FR/lemonde5.txt
 create mode 100644 TEXTS/IN/ARTICLES/FR/lemonde6.txt
 create mode 100644 TEXTS/IN/ARTICLES/FR/lemonde7.txt
 create mode 100644 TEXTS/IN/ARTICLES/FR/lemonde8.txt
 create mode 100644 TEXTS/IN/ARTICLES/FR/lemonde9.txt
 create mode 100644 TEXTS/IN/ARTICLES/FR/lequipe1.txt
 create mode 100644 TEXTS/IN/ARTICLES/FR/lequipe2.txt
 create mode 100644 TEXTS/IN/BOOKS/EN/verne_20000.txt
 create mode 100644 TEXTS/IN/BOOKS/FR/darwin_de_l_origine_des_especes.txt
 create mode 100644 TEXTS/IN/BOOKS/FR/declaration_droits_homme.txt
 create mode 100644 TEXTS/IN/BOOKS/FR/proust_du_cote_de_chez_swann.txt
 create mode 100644 TEXTS/IN/BOOKS/FR/proust_swann.txt
 create mode 100644 TEXTS/IN/BOOKS/FR/proust_un_amour_de_swann.txt
```

- Validez localement vos modifications avec un message informatif avec `git commit -m "init dépôt"`
- Vérifiez l'état de votre dépôt avec `git status`
- Poussez vos modifications vers le dépôt distant avec `git push`

Dépôt distant GIT pour les TPs (7)



A screenshot of a Linux desktop environment. In the top left corner, there's a dock with icons for Activities, Terminal, Dash, and others. A terminal window is open, showing a session of a user named 'maurelf' pushing changes to a remote repository. The terminal output is as follows:

```
(jammy-vdi r4741)maurelf@CAMPUS@V304V-JCP012:~/Documents/GIT/maurel_fabrice$ git push
Password for 'https://maurelf@forge.info.unicaen.fr':
Enumerating objects: 216, done.
Counting objects: 100% (216/216), done.
Compression par delta en utilisant jusqu'à 8 fils d'exécution
Compressing objects: 100% (209/209), done.
Writing objects: 100% (215/215), 7.68 MiB | 3.17 MiB/s, done.
Total 215 (delta 0), reused 0 (delta 0), reused 0 from pack.
To https://forge.info.unicaen.fr/git/maurel_fabrice
  2f04cf2..a6eacbf master -> master
(jammy-vdi r4741)maurelf@CAMPUS@V304V-JCP012:~/Documents/GIT/maurel_fabrice$ git status
Sur la branche master
  Votre branche est à jour avec 'origin/master'.
rien à valider, la copie de travail est propre
(jammy-vdi r4741)maurelf@CAMPUS@V304V-JCP012:~/Documents/GIT/maurel_fabrice$
```

Poussez vos modifications vers le dépôt distant avec `git push`

Dépôt distant GIT pour les TPs (8)

The screenshot shows a web browser window with the following details:

- Address Bar:** https://forge.info.unicaen.fr/projects/maurel_fabrice/repository
- Header:** Getting Started, selection web, tutos design, Planning Studio, photos, Google Docs - Home, Académie de Caen, 1 cours, Prochaines retransmis..., Diffusions en direct, li..., Z-UP, test, Chaussures Reebok fr..., Maps, Actualités, Débuter avec Firefox, Autres marque-pages.
- User Information:** Connecté en tant que maurelf, Mon compte, Déconnexion.
- Breadcrumbs:** Accueil, Ma page, Projets, redmine.org/Aide, POO_L1S2_2023 > POO_TPs_2023 > MAUREL_FABRICE
- Search Bar:** Recherche: MAUREL_FABRICE
- Menu Bar:** Aperçu, Activité, Supprimer projet, Demandes, Nouvelle demande, Gantt, Annonces, Documents, Wiki, Fichiers, Dépôt, Configuration.
- Current View:** Dépôt
- Clone URL:** Clone : git clone https://maurelf@forge.info.unicaen.fr/git/maurel_fabrice
- File List:** TEXTS, .gitignore, S01_TP02_template.py, config.py
- File Statistics:** Statistiques | Branche: master | Révision: ||
- File Details:** Nom, Taille

| Nom | Taille |
|----------------------|------------|
| .gitignore | 12 octets |
| S01_TP02_template.py | 3,707 ko |
| config.py | 332 octets |

- Recent Commits:** Dernières révisions

| # | Date | Auteur | Commentaire |
|----------|------------------|----------------|----------------|
| a6eacbfd | 22/01/2024 01:06 | Fabrice Maurel | depot initial |
| 2f04cf2b | 22/01/2024 00:15 | Sysadmin | Initial commit |

- Buttons:** Voir les différences, Voir toutes les révisions, Voir les révisions.
- Right Panel:** Dépôts, Dépôt principal, Subversion.

- Menu Dépôt
- Vérifier la mise à jour du dépôt distant

YSINL2A1 - INTRODUCTION À LA POO EN PYTHON

2023-2024

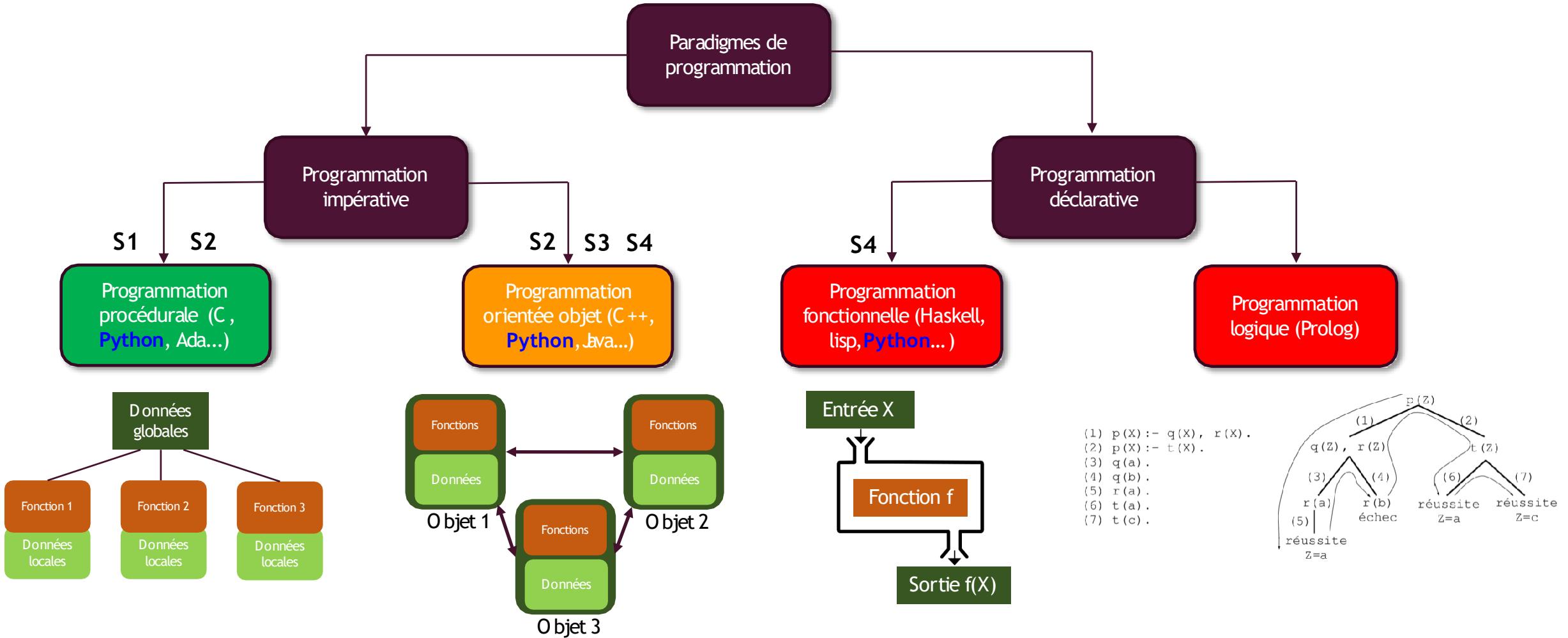
**Semaine 4/10 - 29/1/2024
CM4 - TP7 - TP8
Classes/objets**



Ce pictogramme indique un exemple de code sur *ecampus*
dans un fichier commençant par NUMCHAP_NUMSLIDE

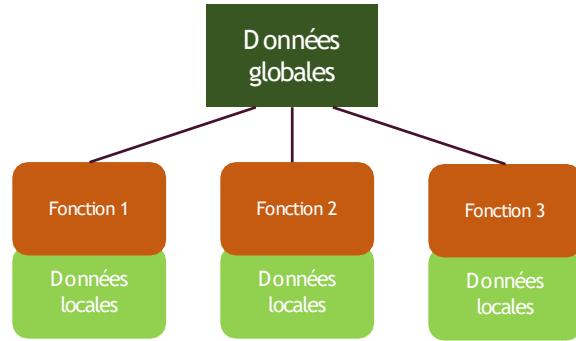
Fabrice Maurel - L1 INFO/MATH/MIASH - Semestre 2
fabrice.maurel@unicaen.fr - 02 31 56 73 97 - S3-364

PARADIGMES DE LA PROGRAMMATION : STRUCTURATION

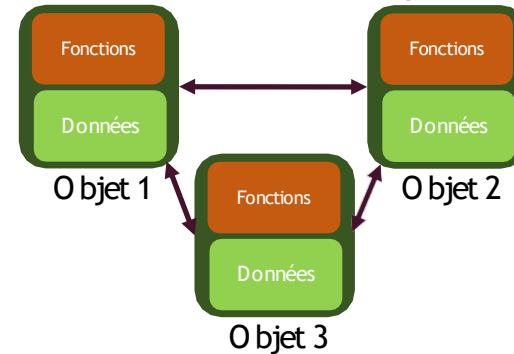


PROCÉDURAL VS. OBJET

Procédural



Orienté objet

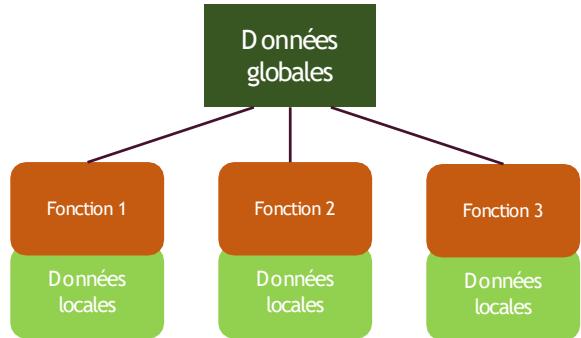


- Orienté traitements : « que doit faire le programme ? »
- Division en sous-programmes (fonctions, procédures)
- Traitements hiérarchisés

- Orienté données : « sur quoi porte le programme ? »
- Modélisation des données comme des objets
- Données et traitements encapsulés

PROCÉDURAL VS. OBJET

Procédural

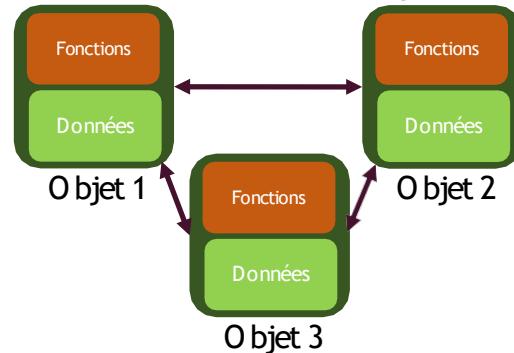


```

def envoi(mess,dest):
    res="Cher " + dest + ",\n\n" + mess
    return res

message1 = envoi("Machin","je vous salue.")
message2 = envoi("Truc","salut à toi !")
print(message1)
print(message2)
  
```

Orienté objet



```

class Message(object):

    def __init__(self,texte):
        self.mess=texte

    def envoi(self,dest):
        res="Cher " + dest + ",\n\n" + self.mess
        return res

message1= Message("je vous salue")
message2 = Message("salut à toi !")
message1.envoi("Machin")
message2.envoi("Truc")
print(message1, message2)
  
```

CLASSE VS. OBJETS

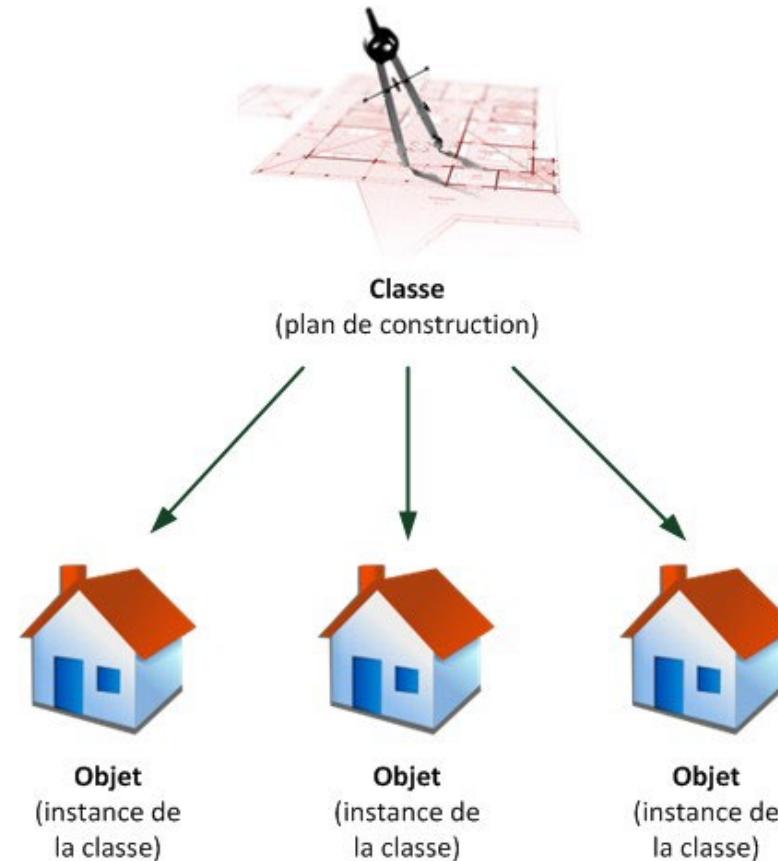
- Une classe est un concept abstrait représentant des éléments variés : concrets, abstraits, comportementaux, composants d'une application, structures informatiques...
- Un objet ou instance est une concrétisation d'un concept abstrait décrit par une classe
 - Une *ferrari Enzo* est une instance du concept *Automobile*
 - *marie* ou *pierre* sont deux instances du concept d'*Humain*
 - *la jalouse de Pierre pour la ferrari Enzo de Marie* est une instance du concept de *Jalousie*

CLASSE VS. OBJETS (INSTANCES)

Logique du comportement



Pratique de l'utilisation



Un logiciel est une collection d'**objets** définis par des **propriétés**

L'**identité** de l'objet permet de le distinguer des autres objets indépendamment de son état

ATTRIBUTS VS. MÉTHODES

Encapsulation

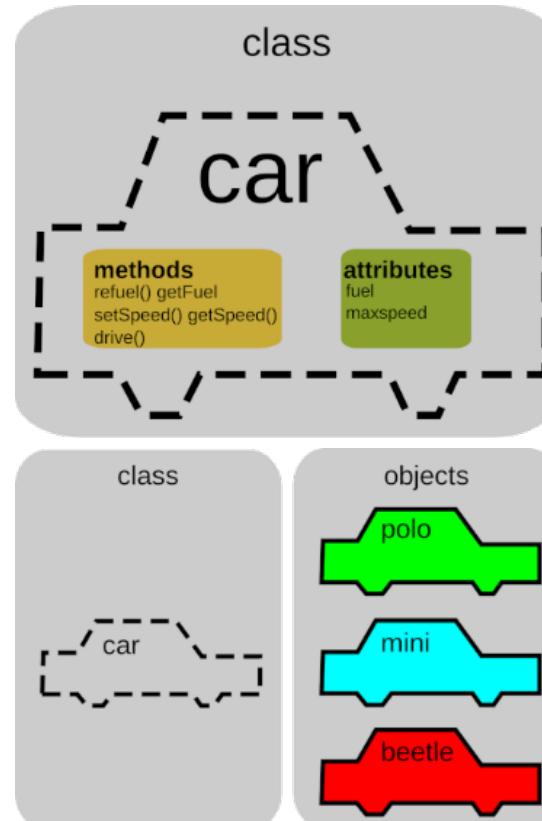


Instantiation

Une propriété est

Soit un **attribut** : donnée caractérisant l'état de l'objet

Soit une **méthode** : opération que l'objet est à même de réaliser



CONSTRUCTEUR, IDENTITÉ SELF ET OPÉRATEUR .

```
class Personne(object):
    1. def __init__(self,nom, age=0):
        self.nom = nom
    2. self.age = age
    3. def vieillir(self,annees=1):
        self.age = self.age+annees
```

```
class Message(object):

    def __init__(self,tex):
        self.mess = tex

    def envoi(self,dest):
        res = "Cher " + dest + ",\n\n" + self.mess
        return res

P1 = Personne("Machin",40)
p1.vieillir()
Texte = "bon anniversaire pour tes " + str(p1.age) +
"ans"
print(Message(texte).envoi(p1.nom))
```



Ne pas confondre paramètres du constructeur et attributs de la classe

CONCEPTS :

- ENCAPSULATION
- INSTANCIATION
- INTERFACE D'UN OBJET



1. Appel de la méthode constructeur :
Personne.__init__(p1,"Machin", 40)

2. Crédation des attributs :

p1.nom = "Machin"

p1.age = 40

3. Crédation de la méthode :

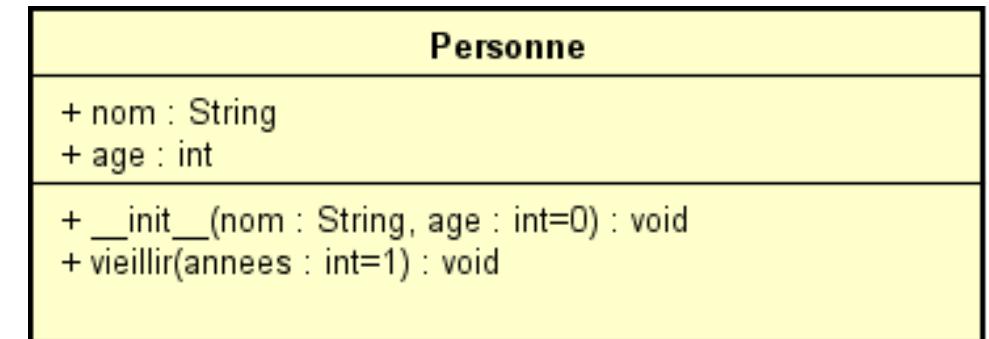
```
def p1.vieillir(annees=1):
    p1.age = p1.age + annees
```

REPRÉSENTATION UML D'UNE CLASSE

```
class Personne:

    def __init__(self, nom, age=0):
        self.nom = nom
        self.age = age

    def vieillir(self, annees=1):
        self.age = self.age + annees
```



EXERCICE COMPLEXE ET MATRICE

Complexe

```
+ re: float
+ im: float

+ __init__(re:float=0, im:float=1): void
+ complexe_str(): str
+ add(c: Complexe): Complexe
+ sub(c: Complexe): Complexe
+ mul(c: Complexe): Complexe
+ conjugue(): Complexe
+ module() : float
+ argument(): float
```

```
>>> from math import atan2, pi
>>> c = Complexe(-3, 4)
>>> print(c.conjugue())
-3-i4
>>> print(c.mul(c))
-7-i24
>>> print(c.module())
5.0
>>> print(c.argument())
>>> 126.86989764584402
```

Matrice

```
+ matrice: list[list[float]]
+ nb_lig: int
+ nb_col: int

+ __init__(tab: list[list[float]]): void
+ est_valide(): bool
+ ligne_str(i: int): str
+ matrice_str(): str
+ somme(m: Matrice): Matrice
```

```
>>> matrice_test = [[1, 2, 3], [4, 3, 2]]
>>> m = Matrice(matrice_test)
>>> print(m.nb_lig, m.nb_col)
2, 3
>>> print(m.matrice_str())
| 1 2 3 |
| 4 3 2 |
>>> print(m.matrice_str(m.somme(m)))
| 2 4 6 |
| 8 6 4 |
```

STRUCTURE GÉNÉRALE D'UN PROGRAMME ET IMPORTATIONS

```
class Human:

    def __init__(self, name, age=0):
        self.name = name
        self.age = age

    def ageing(self, years=1):
        self.age = self.age + years

if __name__ == "__main__":
    NAME_TEST = "Toto"
    AGE_TEST = 21
    p = Personne(NAME_TEST, AGE_TEST)
    p.ageing()
    print(f"Je m'appelle {p.name} et j'ai {p.age} ans.")
```

python person.py

>>> Je m'appelle Toto et j'ai 22 ans.

```
if __name__ == "__main__":
```

Concentre les tests d'un module au même endroit dans le fichier et empêche leur exécution en cas d'importation.

```
from person import *
if __name__ == "__main__":
    p1 = Human("Tata", 40)
    p2 = Human("Titi", 37)
    print(p1.name, p2.name)
```

Python main.py

>>> Tata, Titi

FONCTIONNEMENT DE LA FONCTION « *PRINT* »

```
class Human:

    def __init__(self, name, age=0):
        self.name = name
        self.age = age

    def ageing(self, years=1):
        self.age = self.age + years

    def to_string(self):
        return f"Je m'appelle {self.name} et j'ai {self.age} ans."

if __name__ == "__main__":
    p1 = Human("Toto", 25)
    p2 = Human("Tata", 20)
    print(p1.to_string())
    print(p2.to_string())
```

Je m'appelle **Toto** et j'ai **25** ans.
Je m'appelle **Tata** et j'ai **20** ans.

FONCTIONNEMENT DE LA FONCTION « *PRINT* »

```
class Human:

    def __init__(self, name, age=0):
        self.name = name
        self.age = age

    def ageing(self, years=1):
        self.age = self.age + years

    def to_string(self):
        return f"Je m'appelle {self.name} et j'ai {self.age} ans."

if __name__ == "__main__":
    p1 = Human("Toto", 25)
    p2 = Human("Tata", 20)
    print(p1.to_string())
    print(p2.to_string())
    print(p1)
    print(p2)
```



Je m'appelle Toto et j'ai 25 ans.
Je m'appelle Tata et j'ai 20 ans.
<__main__.Human object at 0x0000015227C37070>
<__main__.Human object at 0x0000015227C36A10>

FONCTIONNEMENT DE LA FONCTION « *PRINT* »

```
class Human:

    def __init__(self, name, age=0):
        self.name = name
        self.age = age

    def ageing(self, years=1):
        self.age = self.age + years

    def __repr__(self):
        return f"Je m'appelle {self.name} et j'ai {self.age} ans."

if __name__ == "__main__":
    p1 = Human("Toto", 25)
    p2 = Human("Tata", 20)
    print(p1.__repr__())
    print(p2.__repr__())
    print(p1)
    print(p2)
```

En Python,
derrière toute
fonction *built-in*
se cache une
méthode
« spéciale »
de la classe
impliquée

Je m'appelle **Toto** et j'ai **25** ans.
Je m'appelle **Tata** et j'ai **20** ans.
Je m'appelle **Toto** et j'ai **25** ans.
Je m'appelle **Tata** et j'ai **20** ans.

YSINL2A1 - INTRODUCTION À LA POO EN PYTHON

2023-2024

Semaine 5/10 - 5/2/2024
CM5 - TP9 - TP10

Privé/public, d'instance/de classe, getter/setter



Ce pictogramme indique un exemple de code sur *ecampus*
dans un fichier commençant par NUMCHAP_NUMSLIDE

Fabrice Maurel - L1 INFO/MATH/MIASH - Semestre 2
fabrice.maurel@unicaen.fr - 02 31 56 73 97 - S3-364

ATTRIBUTS PUBLICS VS. PRIVÉS

```
class Human:

    def __init__(self, name, age=0):
        self.name = name
        self.age = age

    def ageing(self, years=1):
        self.age = self.age + years

    def __repr__(self):
        return f"Je m'appelle {self.name} et j'ai {self.age} ans."

if __name__ == "__main__":
    p = Human("Toto", 25)
    p.ageing(-100)
    print(p)
```



Je m'appelle Toto et j'ai -75 ans.

ATTRIBUTS PUBLICS VS. PRIVÉS

```
class Human:

    def __init__(self, name, age=0):
        self.name = name
        self.age = age

    def ageing(self, years=1):
        if years > 0:
            self.age = self.age + years

    def __repr__(self):
        return f"Je m'appelle {self.name} et j'ai {self.age} ans."

if __name__ == "__main__":
    p = Human("Toto", 25)
    p.ageing(-100)
    print(p)
```

Solution

Je m'appelle Toto et j'ai 25 ans.

ATTRIBUTS PUBLICS VS. PRIVÉS

```
class Human:

    def __init__(self, name, age=0):
        self.name = name
        self.age = age

    def ageing(self, years=1):
        if years > 0:
            self.age = self.age + years

    def __repr__(self):
        return f"Je m'appelle {self.name} et j'ai {self.age} ans."

if __name__ == "__main__":
    p = Human("Toto", 25)
    p.age = -100
    print(p)
```

Solution partielle !

Je m'appelle Toto et j'ai -75 ans.

ATTRIBUTS PUBLICS VS. PRIVÉS

```
class Human:

    def __init__(self, name, age=0):
        self.name = name
        self.__age = age

    def ageing(self, years=1):
        if years > 0:
            self.__age = self.__age + years

    def __repr__(self):
        return f"Je m'appelle {self.name} et j'ai {self.__age} ans."

if __name__ == "__main__":
    p = Human("Toto", 25)
    print(p.__age)
    print(p)
```

Solution complète

`AttributeError: 'Human' object has no attribute '_age'`

Un attribut privé est **inaccessible directement** hors de la classe. **Conséquence** : il est **obligatoire de passer par une méthode** prévue à cet effet pour le manipuler correctement (ici **en lecture**, génère une erreur).

ATTRIBUTS PUBLICS VS. PRIVÉS

```
class Human:

    def __init__(self, name, age=0):
        self.name = name
        self.__age = age

    def ageing(self, years=1):
        if years > 0:
            self.__age = self.__age + years

    def __repr__(self):
        return f"Je m'appelle {self.name} et j'ai {self.__age} ans."

if __name__ == "__main__":
    p = Human("Toto", 25)
    p.__age = -100
    print(p)
```

Solution complète

Je m'appelle Toto et j'ai 25 ans.

Un attribut privé est **inaccessible directement** hors de la classe. **Conséquence** : il est **obligatoire de passer par une méthode** prévue à cet effet pour le manipuler correctement (ici **en écriture**, l'affectation n'a pas d'effet).

ATTRIBUTS PUBLICS VS. PRIVÉS

```
class Human:

    def __init__(self, name, age=0):
        self.name = name
        self.__age = age

    def ageing(self, years=1):
        if years > 0:
            self.__age = self.__age + years

    def __repr__(self):
        return f"Je m'appelle {self.name} et j'ai {self.__age} ans."

if __name__ == "__main__":
    p = Human("Toto", 25)
    p.ageing()
    print(p)
```

Solution complète

Je m'appelle Toto et j'ai 26 ans.

Un attribut privé est **inaccessible directement** hors de la classe. **Conséquence** : il est **obligatoire de passer par une méthode** prévue à cet effet pour le manipuler correctement (ici **en écriture**, seule la méthode *ageing* fonctionne).

ATTRIBUTS PUBLICS VS. PRIVÉS

```
class Human:

    def __init__(self, name, age=0):
        self.name = name
        self.__age = age

    def ageing(self, years=1):
        if years > 0:
            self.__age = self.__age + years

    def __repr__(self):
        return f"Je m'appelle {self.name} et j'ai {self.__age} ans."

if __name__ == "__main__":
    p = Human("Toto", 25)
    p.ageing()
    print(p)
```

| Human |
|--|
| + name:str - age:float |
| __init__(name:str, age:float=0):void __repr__():str |
| ageing(years :int=1):void |

Je m'appelle Toto et j'ai 26 ans.

Un attribut privé est **inaccessible directement** hors de la classe. **Conséquence** : il est **obligatoire de passer par une méthode** prévue à cet effet pour le manipuler correctement (ici **en écriture**, seule la méthode *ageing* fonctionne).

ACCESSEUR ET MUTATEUR D'UN ATTRIBUT PRIVÉ

```
class Human:

    def __init__(self, name, age=0):
        self.name = name
        self.__age = age

    def ageing(self, years=1):
        if years > 0:
            self.__age = self.__age + years

    def get_age(self):
        return self.__age

    def __repr__(self):
        return f"Je m'appelle {self.name} et j'ai {self.__age} ans."

if __name__ == "__main__":
    p = Human("Toto", 25)
    p.ageing()
    print(p.get_age())
```

Mutateur (ou setter) : gère l'accès d'un attribut privé en écriture. Souvent par convention `set_nom_attribut_privé`

Accesseur (ou getter) : gère l'accès d'un attribut privé en lecture. Souvent par convention `get_nom_attribut_privé`

ACCESSEUR ET MUTATEUR D'UN ATTRIBUT PRIVÉ

```

class Human:

    def __init__(self, name, age=0):
        self.name = name
        self.__age = age

    def ageing(self, years=1):
        if years > 0:
            self.__age = self.__age + years

    def get_age(self):
        return self.__age

    def __repr__ (self):
        return f"Je m'appelle {self.name} et j'ai {self.__age} ans."

if __name__ == "__main__":
    p = Human("Toto", 25)
    p.ageing()
    print(p.get_age())

```

| Human |
|--------------------------------------|
| + name:str |
| - age:float |
| __init__(name:str, age:float=0):void |
| __repr__():str |
| ageing(years :int=1):void |
| get_age():float |

ATTRIBUT D'INSTANCE VS. ATTRIBUT DE CLASSE

```

class Human:

    human_counts = 0

    def __init__(self, name, age=0):
        self.name = name
        self.__age = age
        Human.human_counts += 1

    def ageing(self, years=1):
        if years > 0:
            self.__age = self.__age + years

    def get_age(self):
        return self.__age

    def __repr__ (self):
        return f"Je m'appelle {self.name} et j'ai {self.__age} ans."

if __name == "__main__":
    print(Human.human_counts)
    p1 = Human("Toto", 25)
    p2 = Human("Tata", 20)
    print(Human.human_counts)

```

Attribut de classe : lié à la classe et partagé par tous les objets issus de cette classe

Accès via le nom de la classe `Human` plutôt que via l'objet `self`

| Human |
|--------------------------------------|
| human_counts:int |
| + name:str |
| - age:float |
| |
| __init__(name:str, age:float=0):void |
| __repr__():str |
| ageing(years :int=1):void |
| get_age():float |

- 0 Nul besoin de l'existence d'un objet pour l'appeler.
Ici, modification par incrémentation de 1 à chaque appel du constructeur.
- 2

ATTRIBUT DE CLASSE PRIVÉ ET MÉTHODE DE CLASSE

```

class Human:

    __human_counts = 0

    @classmethod
    def get_human_counts(cls):
        return cls.__human_counts

    def __init__(self, name, age=0):
        self.name = name
        self.__age = age
        Human.__human_counts += 1

    def ageing(self, years=1):
        if years > 0:
            self.__age = self.__age + years

    def get_age(self):
        return self.__age

    def __repr__(self):
        return f"Je m'appelle {self.name} et j'ai {self.__age} ans."

if __name__ == "__main__":
    print(Human.get_human_counts())
    p1 = Human("Toto", 25)
    p2 = Human("Tata", 20)
    print(Human.get_human_counts())

```

Attribut de classe privé : lié à la classe et partagé par tous les objets issus de cette classe mais inaccessible directement hors de la classe

Méthode de classe : manipule uniquement des attributs de classe. Prend en paramètre la classe `cls` plutôt que l'instance `self`

| Human |
|--|
| - <code>human_counts</code> :int |
| + <code>name</code> :str |
| - <code>age</code> :float |
| <code>get_human_counts()</code> :int |
| <code>__init__(name:str, age:float=0)</code> :void |
| <code>__repr__()</code> :str |
| <code>ageing(years :int=1)</code> :void |
| <code>get_age()</code> :float |

- 0 Nul besoin de l'existence d'un objet pour l'appeler.
Ici, modification par incrémentation de 1 à chaque appel du constructeur.
- 2

YSINL2A1 - INTRODUCTION À LA POO EN PYTHON

2022-2023

Semaine 6/10 - 12/2/2023
CM6 - TP11 - TP12

A grégation, composition forte et héritage simple



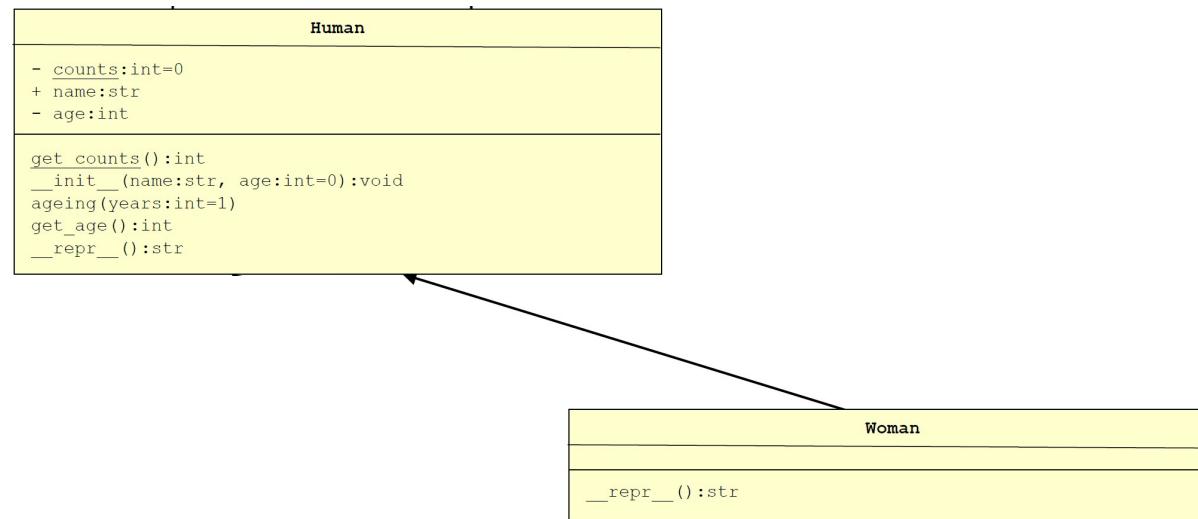
Ce pictogramme indique un exemple de code sur *ecampus*
dans un fichier commençant par NUMCHAP_NUMSLIDE

Fabrice Maurel - L1 INFO/MATH/MIASH - Semestre 2
fabrice.maurel@unicaen.fr - 02 31 56 73 97 - S3-364

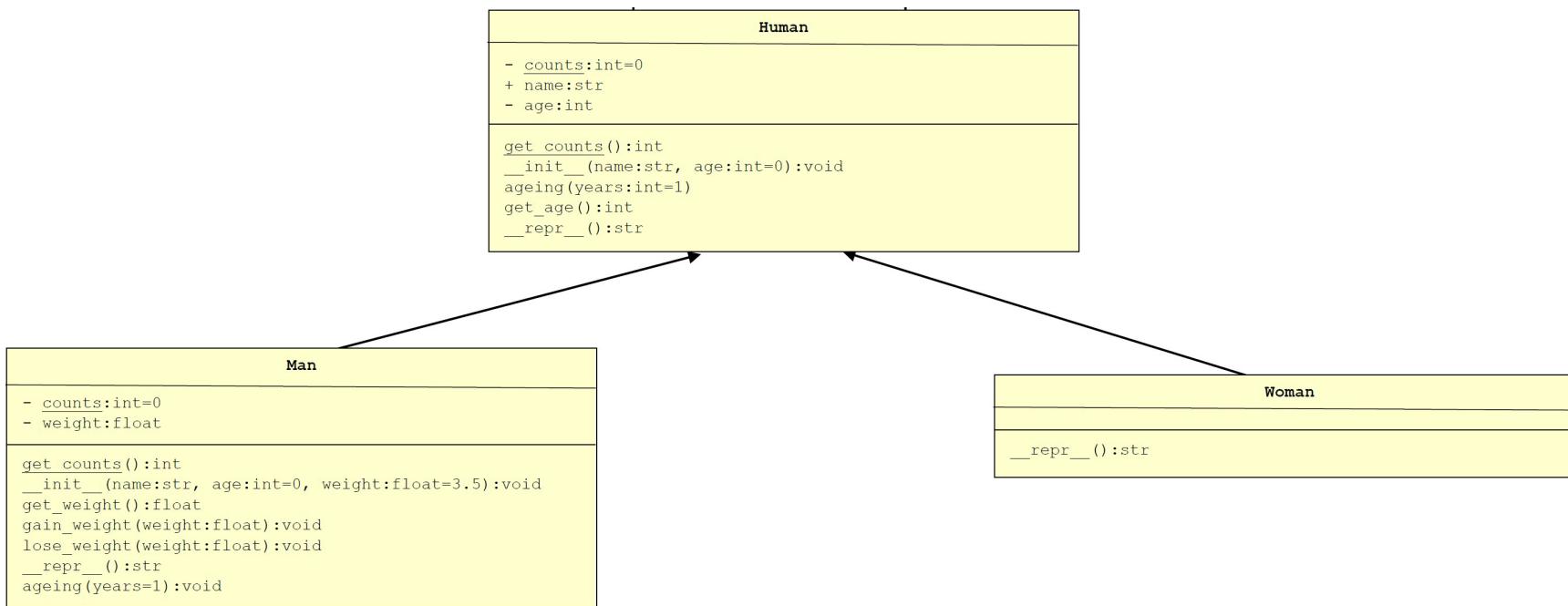
MÉTHODES SPÉCIALES - PUBLIC/PRIVÉ - DE CLASSE/D'INSTANCE

| Human |
|--|
| - <u>counts</u> :int=0 + name:str - age:int |
| get_counts():int <u>__init__</u> (name:str, age:int=0):void ageing(years:int=1) get_age():int <u>__repr__</u> ():str |

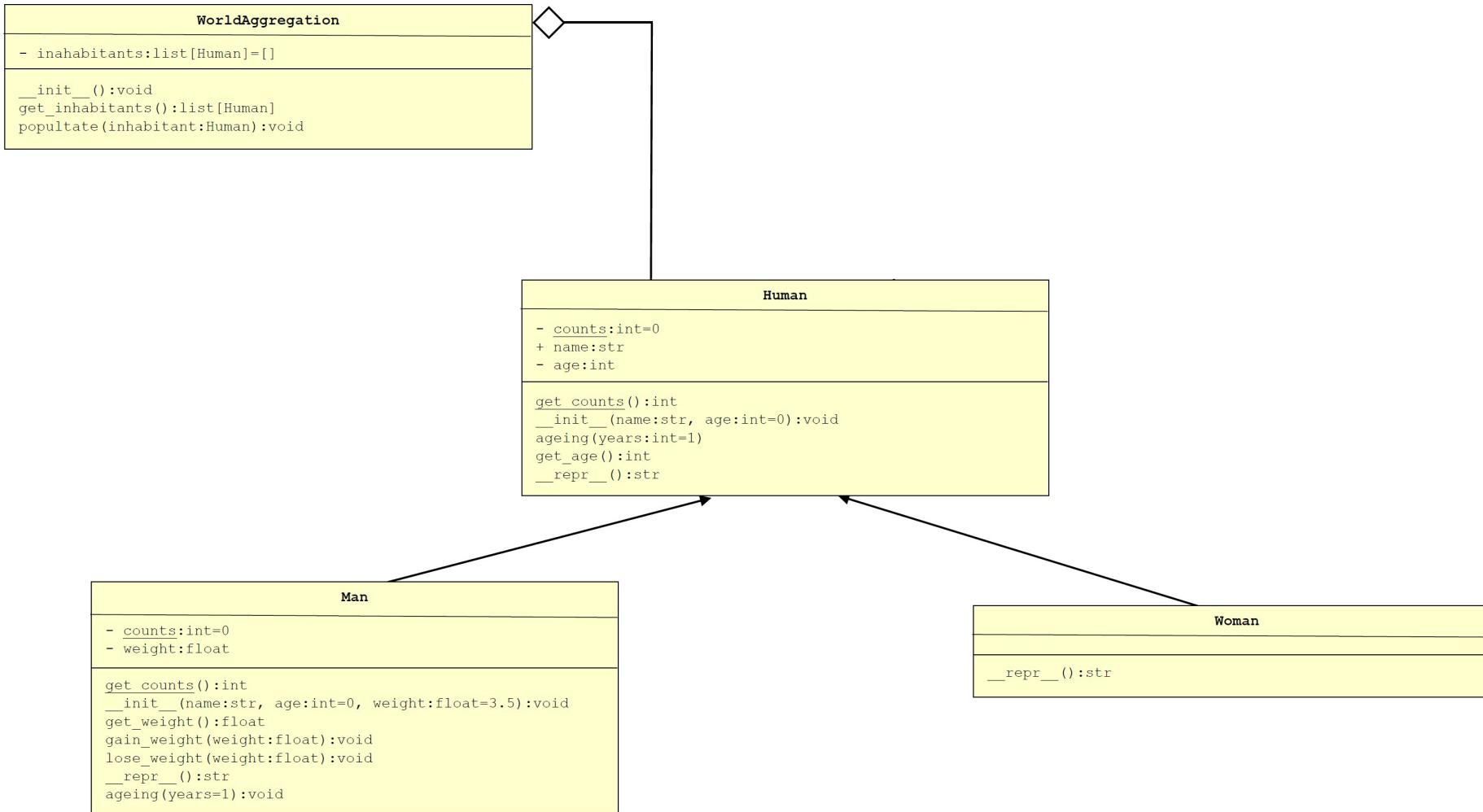
HÉRITAGE SIMPLE SANS RÉÉCRITURE DU CONSTRUCTEUR



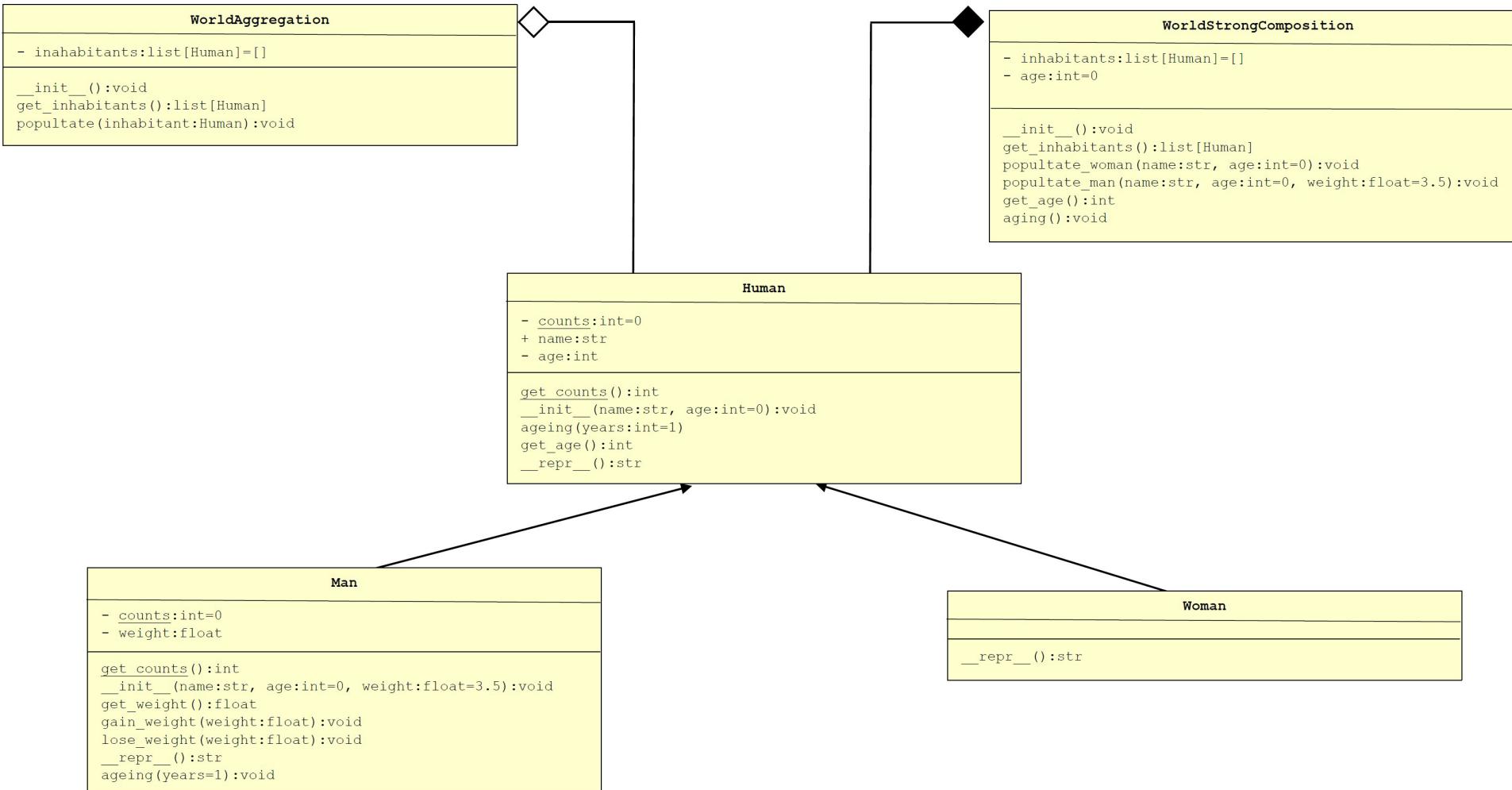
HÉRITAGE SIMPLE AVEC RÉÉCRITURE DU CONSTRUCTEUR



RELATION D'AGRÉGATION



RELATION DE COMPOSITION FORTE



CODE : HERITAGE SIMPLE

```
class Human:
    __counts = 0

    @classmethod
    def get_counts(cls):
        return cls.__counts

    def __init__(self, name, age=0):
        self.name = name
        self.__age = age
        Human.__counts += 1

    def ageing(self, years=1):
        if years > 0:
            self.__age = self.__age + years

    def get_age(self):
        return self.__age

    def __repr__(self):
        return f"Je m'appelle {self.name} et j'ai {self.__age} ans."
```

```
class Woman(Human):
    def __repr__(self):
        if self.get_age() > 40:
            age = round(self.get_age() * 0.9)
            return f"Je m'appelle {self.name} et j'ai {age} ans."
        return Human.__repr__(self)
```

```
class Man(Human):
    __counts = 0

    @classmethod
    def get_counts(cls):
        return cls.__counts

    def __init__(self, name, age=0, weight=3.5):
        Human.__init__(self, name, age)
        Man.__counts += 1
        self.__weight = weight

    def get_weight(self):
        return self.__weight

    def gain_weight(self, weight):
        if weight > 0:
            self.__weight += weight

    def lose_weight(self, weight):
        if weight < self.__weight:
            self.__weight -= weight

    def __repr__(self):
        return f"{Human.__repr__(self)} Je pèse {self.get_weight()} Kg."

    def ageing(self, years=1):
        Human.ageing(self, years)
        if self.get_age() > 40:
            self.gain_weight(years)
```

CODE :AGRÉGATION - COMPOSITION

```
class WorldStrongComposition:

    def __init__(self):
        self.__inhabitants = []
        self.__age = 0

    def get_inhabitants(self):
        return self.__inhabitants

    def populate_woman(self, name, age=0):
        self.__inhabitants.append(Woman(name, age))

    def populate_man(self, name, age=0, weight=3.5):
        self.__inhabitants.append(Man(name, age, weight))

    def get_age(self):
        return self.__age

    def ageing(self):
        for inhabitant in self.get_inhabitants():
            inhabitant.ageing(1)
```

```
class WorldAggregation:

    def __init__(self):
        self.__inhabitants = []

    def get_inhabitants(self):
        return self.__inhabitants

    def populate(self, human):
        self.__inhabitants.append(human)
```

CODE :TESTS

```
if __name__ == "__main__":
    print(Human.get_counts())
    print(Man.get_counts())
    f = Woman("Tata", 35)
    h = Man("Toto", 35, 80)
    print(f.get_age())
    print(f)
    print(h)
    f.ageing(20)
    h.ageing(20)
    print(f.get_age())
    print(f)
    print(h)
    world_aggregation = WorldAggregation()
    world_strong_composition = WorldStrongComposition()
    world_aggregation.populate(f)
    world_aggregation.populate(h)
    print(world_aggregation.get_inhabitants())
    world_strong_composition.populate_woman("Tata", 35)
    world_strong_composition.populate_man("Toto", 35, 80)
    print(world_strong_composition.get_inhabitants())
    world_strong_composition.ageing()
    print(world_strong_composition.get_inhabitants())
    print(Human.get_counts())
    print(Man.get_counts())
```

0
0

35
Je m'appelle Tata et j'ai 35 ans.
Je m'appelle Toto et j'ai 35 ans. Je pèse 80 Kg.

55
Je m'appelle Tata et j'ai 50 ans.
Je m'appelle Toto et j'ai 55 ans. Je pèse 100 Kg.

[Je m'appelle Tata et j'ai 50 ans., Je m'appelle Toto et j'ai 55 ans. Je pèse 100 Kg.]

[Je m'appelle Tata et j'ai 35 ans., Je m'appelle Toto et j'ai 35 ans. Je pèse 80 Kg.]

[Je m'appelle Tata et j'ai 36 ans., Je m'appelle Toto et j'ai 36 ans. Je pèse 80 Kg.]

4
2