

④ Fichiers

① Ouvrir un fichier:

$f = \text{open}(\text{nom}, \text{mode})$

② Fermer un fichier:

$f.\text{close}()$

③ Modes d'ouverture:

a) Mode w:

- Elle efface tout le contenu.
- Si le fichier n'existe pas, elle le crée.
- Méthodes d'écriture:

→ $f.\text{write}(\text{"..."})$

→ $f.\text{writelines}([\text{ch1}, \text{ch2}, \dots])$

b) Mode a:

- Ajoute à la fin.
- Si le fichier n'existe pas, elle le crée.

c) Mode r:

- Lecture
- Si le fichier n'existe pas, message d'erreur.

Méthodes de Lecture:

→ $f.\text{read}()$ → renvoie une chaîne

→ $f.\text{readline}()$ → renvoie une chaîne d'une seule ligne.

→ $f.\text{readlines}()$ → renvoie une liste de chaînes.

④ Mix à son:

$f = \text{open}(\dots, \text{"a"})$

$L = f.\text{readlines}()$

$f.\text{close}()$

for i in L:

...
 $f.\text{write}(\dots)$

import os:

os.chdir ("\" (: \ - - \ - - ")

I/ Définition de classe:

• classe, attributs d'instance:

class A:

```
def __init__(self, paramètres):
    self.attributs = paramètres
```

• méthodes:

```
def méthode(self, paramètres):
    bloc d'instruction
```

• Créer instance de classe A:

* $X = A()$

Exp:

```
ch = str()
L = list()
S = set()
```

• Appel aux attributs, attributs + méthodes de A:

* $X.attribut$

Exp:

```
P = Pointe(x, y)
P.x
P.y
```

* $X.méthode()$

Exp:

```
P.coordonnées()
```

• Méthode spéciale:

* $def __méthode__(self, attributs):$

Exp:

```
def __add__(self, other) → +
def __mul__(self, other) → *
def __sub__(self, other) → -
def __eq__(self, other) → ==
def __len__(self) → len
def __str__(self) → print
def __repr__(self) → instance
```

Exp:

$L3 = L1 + L2$ on $L3 = L1 + L2$

• attribut de classe:

class A:

attribut de classe = --

```
def __init__(self, paramètres):
```

attribut d'instance = --

• Appel aux attributs, différences:

attribut de classe A

A.attribut

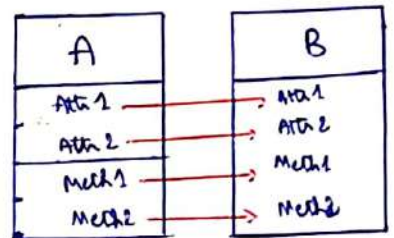
attribut d'instance X

X.attribut

II/ Héritage:

① Héritage total:

class B(A):



② Héritage partiel:

class B(A):

```
def __init__(self, paramètres):
```

```
    self.attr1 = ...
```

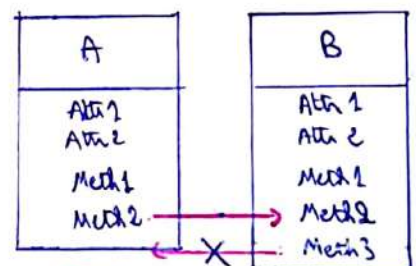
```
    self.attr2 = ...
```

```
def meth1(self, paramètres):
```

```
def meth3(self, paramètres):
```

pas reclassé les attributs

reclassé.



x.meth3() → Even

y.meth1() ✓
→ hérité de A

③ Appel à une méthode de la classe mère A:

A.méthode(self, ...) ou super().méthode(...)

④ En $def __init__$:

si A acher attr1 et attr2.

B acher attr1 et attr2 et attr3.

class B(A):

```
def __init__(self, a, b, c):
```

```
    super().__init__(a, b)
```

```
    self.attr3 = c
```

③ Base de données

I/AR:

① Opérations de base:

- Selection: σ
- Projection: π

② Opérations ensemblistes:

- Union: $\cup$
- Intersection: $\cap$
- Différence: $-$
- Produit Cartésien: $\times$
- Jointure: $\bowtie$

II/SQL:

① Création de Table:

create table Nom (
colonne 1 type,
colonne 2 type,
-----),

② Suppression de table:

drop table Nom,

③ Modification de table:

a) Ajout de colonne:

alter table relation
add colonne type,

b) Insertion de données dans une colonne:

insert into relation (att1, att2, ...)
values (val1, val2, ...)

c) Changement de valeur:

update relation
set C1 = ..., C2 = ...
where condition,

④ Types de données:

- char / text: chaîne de caractères.
- varchar: chaîne de taille variable
- int / integer: entier
- numeric: nombre
- real: nombre à virgule
- date: date
- bool: booléen

b) Suppression de colonne:

alter table relation
drop colonne,

d) Suppression des lignes:

delete from relation
where condition,

⑤ Règles d'intégrité:

- primary key.
- foreign key --- référentiel
- not null
- unique
- auto increment
- default.

III/ Manipulation des données:

AR

A - Opérations de base:

① Projection:

$\pi_{c_1, c_2, \dots} (R)$

② Selection:

$\sigma_{condition} (R)$

B - Opérations Ensemblistes:

① Union:

$R \cup S$

② Intersection:

$R \cap S$

③ Différence:

$R - S$

④ Produit cartésien:

$R \times S$

⑤ Jointure:

$R \bowtie_{R.C1=S.C1} S$

C - Fonctions d'agrégation:

- * $avg()$; $count()$;
- * $group by()$; $max()$; $min()$
- * $order by()$; $order by()$ desc
- * $having()$
- * $distinct()$

SQL

A - Opérations de base:

① Projection:

select C1, C2, ...
from R

② Selection:

select *
from R
where condition

B - Opérations Ensemblistes:

① Union:

select A1, A2 from R
union
select A1, A2 from S

② Intersection:

select A1, A2 from R
intersect
select A1, A2 from S

③ Différence:

select A1, A2 from R
minus
select A1, A2 from S

④ Produit Cartésien:

select * from R, S

⑤ Jointure:

select *
from R, S
where R.C1 = S.C1

D - Commandes de sélection:

- * and ; or ; not
- * in ; $not in$
- * $between$
- * is
- * $like$
- * $exists$

IV / SQLite:

import sqlite3

BD = sqlite3.connect("mon_base.db")

cur = BD.cursor()

1) Exécuter requête SQL:

• cur.execute(req) → 1 seule fois.

ou bien

• cur.executemany(req, L) → exécuter len(L) fois.

2) Récupérer résultat de req:

cur.fetchall() → retour liste de tuples

cur.fetchone() → retour 1^{ère} ligne de req : tuple.

3) Valider et fermer:

BD.commit()

BD.close()