LA

- break : interruption
- continue : court-circuit (sauter un élément) } boucle for

- while ... else / for ... else : while تتعدّى و لا يعاد كي ما تكبيرش حتّى مرّة في break
  break pour l'interruption

- Éviter les erreurs / exceptions, try : .... except : # fournir type d'exception
→ plusieurs exceptions : try : ... except : ... except : ... except : ...
→ lever volontairement une exception : raise

- **Structure : ordonnée :** _ liste / _ tuple / _ chaîne de caractères

- **liste** : ordonnée / modifiable / hétérogène : [, ,]
  - range(, ,) inclus exclu pas
  - sort()
  - append(.) ajout élément
  - extand(.) ajout liste d'éléments
  - reverse() liste تقلب
  - remove(.) supprimer élément
  - index(x) indice de l'élément x
  - pop(.) élmt آخر تنحّي
  - count(x) تحسب قدّاش ثمّة x
  - [a : b] : liste de a à b exclus ⚠
- liste [a : a] = liste [a] : insertion

- **tuple** : ordonnée / non modifiable / hétérogène : (, ,)

⇒ opérations : (in , not in) → bool / addition + / multiplication * / [i] : i^ème élément
[i : j : k] tranche de i à j avec pas k / longueur len(.) / max(.) / min(.) / index(.) / count(.)

- import copy : copy.copy(a) / copy.deepcopy(a) ou bien : c = a[:] (copie simple)

- **chaîne de caractères** :

- **Tableaux associatifs** :

- **dictionnaire** : modifiable et non ordonné ⇒ couple (clé , valeur)
  d_1 = { } ou d_1 = dict() (vide à remplir) ⇒ d_1 = {clé : "valeur", ...} ou d_1 = dict(clé : valeur , ...)
  ou bien : d_1 = dict([(clé, valeur), (clé, valeur)])
- d_1[clé] → valeur
- del d_1[clé] → suppression du (clé : valeur)
- d_1[clé] = valeur (ajout élément)
- d_1.keys() affichage des clés
- sorted(d_1.keys()) : clés ordonnées en alphabet

• d_1.values() : affichage des valeurs • sorted(d_1.values()) : valeurs ordonnées en alphabet

• len() : nombre de couples • clé in d_1 → bool : True / False

• **ensemble** : non ordonné

• x = set('ensemble') ⇒ {'e', 'n', 's', 'm', 'b', 'l'} pas de duplication de 'é' les تكرار محذوفة

• 'a' in x ⇒ False • y = set('semble'), x - y ⇒ 'n' (عناصر في x وليست في y)

• x ^ y ⇒ عناصر في x أو y وليست في كليهما • y - x ⇒ (عناصر في y وليست في x) • x | y ⇒ union • x & y : intersection

* **fichier texte** :

• **ouverture** : f1 = open("monfichier", "r", encoding="utf8") { "r" : lecture, "w" : écriture, "a" : ajout }

• **fermeture** : f1.close()

• **écriture séquentielle** : s = "chaîne", f.write(s), l = ['a', 'b', 'c'], f.writelines(l)

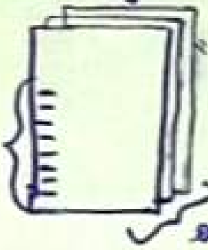
• **lecture séquentielle** : s = f.read(), s = f.readlines() (lire toutes les lignes), s = f.read(3) (lire au plus 3 octets), s = f.readline() (lire ligne suivante)

• **option (file)** : file = f1

• **itération sur les conteneurs** :

• POO •

nouveau type : objet → variables définient l'objet



les attributs

"notion de" و "les objets" اللي يفرق بين
"structure" loge des actions, les f^{ts} ∈ tâ l'objet → Méthodes

ensemble des objets → classe

Class C :

• self : référence d'instance.

```
* def __init__(self, *, *):
      self.0 = *
```
} constructeur : initialisation des attributs de la classe !

• Méthodes de surcharge : redéfinition des fonctions :

* __neg__ : (-*)   * __sub__ : (-)   * __div__ : (/)
* __add__ : (+)   * __mul__ : (x)   * __floordiv__ : (//)

→ Affichage :

```
def __str__(self):
    return ....
```

• Héritage : Class classe_fille(classe_mère):

• Polymorphisme : 2 méthodes ou plus ayant même nom mais ∈ 2 classes héritées différentes distinctes d'effectuer un travail différent.