# 0.1 Cours

## 0.1.1 Définition et méthodes

On veut modéliser les notes des élèves en précisant une matière. On utilise pour cela un nouveau type de données appelé dictionnaire. Ceux-ci sont utiles dès que l'on veut enregistrer des données contenant plusieurs champs nommés. Ce type est par exemple utilisé dans les métadonnées EXIF des photographies Les dictionnaires sont des conteneurs mais pas des séquences: on accède aux éléments en utilisant une clé:

```
dict = {clé_1:valeur_1, clé_2:valeur_2, ...., clé_p:valeur_p}
```

Attention: les clés ne peuvent être modifiables (mutables): on ne pourra par exemple pas utiliser une liste comme clé. Voici un dictionnaire stockant des données avec plusieurs champs nommées: prenom, nom et age et un dictionnaire vide.

```
>>> perso = {"prenom":"Rick", "nom":"Sanchez", "age": 70}
>>> dict_vide = {}
```

Leur type est 'dict'

```
>>> perso["prenom"]
'Rick'
>>> type(perso)
<class 'dict'>
```

On peut créer un dictionnaire à partir d'une liste de liste (ou de tuple) à 2 éléments:

```
>>> liste = [('a', 1), ['b', 2], ['c', 3]]
>>> d = dict(liste)
>>> d
{'a': 1, 'b': 2, 'c': 3}
```

On peut respectivement itérer sur les clés, les valeurs et le couple (clé, valeur) avec `perso.keys()`, `perso.values()`, `perso.items()`.

```
>>> for val in perso.values():
...     print(val)
...
Rick
Sanchez
70
```

Les éléments d'un dictionnaire peuvent aussi être des dictionnaires. Voici un exemple :

```
series = {'Breaking Bad':  {'année':2008 , 'saisons': 5, 'durée':47},
          'Rick et Morty': {'année':2013 , 'saisons': 4, 'durée':22},
          'Stranger Things': {'année':2016 , 'saisons': 4, 'durée':42},
          'Mr Robot':{'année':2016 , 'saisons': 4, 'durée':50}}
           'Dark':{'année':2017 , 'saisons': 3, 'durée':50}
          }
```

## 0.1.2 Appartenance

Pour vérifier si `c` est une clé du dictionnaire `d`, on utilise `c in d`:

```
>>>'age' in perso
True
```

Il serait en effet maladroit d'utiliser `c in d.keys()` pour tester si la clef `c` est dans le dictionnaire `d`. En effet, ceci conduirait Python à calculer la liste de toutes les clefs de `d`, puis à voir si `c` est parmi elles, pour une complexité en $O(n)$ où $n$ est le nombre de clefs dans `d`, alors que précisément **un dictionnaire est optimisé pour rechercher efficacement une clef particulière** en utilisant une *fonction de hachage*.

## 0.2 Exercices

**Exercice 1**

1. Construire par compréhension un dictionnaire `liste_carre` dont les clés sont les entiers naturels de 1 à 10 et les valeurs le carré de ces clés.

2. Les photos prises par un smartphone ou un APN sont souvent accompagnées de métadonnées EXIF, stockées dans l'image.


Figure 1: Données EXIF

   (a) Représenter les données EXIF ci-dessus par un dictionnaire nommé 'exif'.
   (b) Quelle instruction Python permet d'afficher la largeur de l'image ?
   (c) Quelle instruction Python permet d'afficher toutes les données EXIF de l'image ?

3. Sur un réseau de machines, chaque machine peut être représentée par son adresse IP, selon la syntaxe clé:valeur du type `'machine_i':'238.135.45.26'`

   (a) Représenter un parc de 3 machines sur ce réseau avec un dictionnaire nommé `parc`.

   (b) Ecrire une instruction Python qui renvoie la liste de toutes les adresse IP des machines connectées au réseau.

   (c) Ecrire une instruction Python qui permet de savoir si la machine ayant pour IP `'238.135.45.38'` est bien connectée au réseau.

**Exercice 2** Vous avez sûrement déjà croisé certaines des appellations suivantes: ASCII, ANSI, Latin1, ISO-8859-1, MacRoman, Windows-1252, etc. Elles désignent toutes des codages de caractères: à une valeur décimale correspond un caractère.

Le premier d'entre eux à avoir standardisé ce codage fut le codage ASCII pour *American Standard Code for Information Interchange* au début des années 60. C'est un codage sur un octet qui contient 128 caractères: il ne permet pas d'écrire les langues européennes qui utilisent des lettres avec accent ou cédille. Vous avez peut-être déjà ouvert un fichier avec des symboles érronnés en lieu et place des accents. Cela est est dû à une ouverture du fichier avec un mauvais encodage.

L'encodage ISO-8859-1 ou Latin-1 est une extension d'ASCII qui comporte 191 caractères imprimables. Elle permet d'écrire la plupart des langues d'Europe de l'Ouest. Presque tous les caractères du français y sont (manque le œ).

| Decimal | Hex | Char | Decimal | Hex | Char |
|---|---|---|---|---|---|
| 64 | 40 | @ | 96 | 60 | ` |
| 65 | 41 | A | 97 | 61 | a |
| 66 | 42 | B | 98 | 62 | b |
| 67 | 43 | C | 99 | 63 | c |

Figure 2: Table ASCII (extrait)

1. Construire par compréhension un dictionnaire `ASCII` dont les clés sont les lettres de l'alphabet en majuscule et les valeurs le code décimal correspondant dans la table ascii.

   *Exemple pour trois lettres*:

   ```
   >>> ascii = {"A": 65, "B": 66, "C": 67 }
   ```

2. Améliorer ce dictionnaire en remplaçant la valeur décimale par un tuple contenant la valeur décimale, hexadécimale, binaire. *Exemple pour trois lettres*:

   ```
   >>> ascii = {'A': (65, '41', '1000001'), 'B': (66, '42', '1000010'),
   ...              'C': (67, '43', '1000011')}
   ```

**Indication**: la fonction `chr(v)` de python retourne le caractère ASCII correspondant au code d'une valeur $v$ telle que $0 \leqslant v \leqslant 127$. Des fonctions utiles:

```
>>> chr(80)
'P'
>>> ord('T')
84
```

```
>>> bin(80)
'0b1010000'
>>> hex(80)
'0x50'
```

**Exercice 2** Dans un groupe d'élève, Mila et Nathan ont obtenu respectivement 16 et 9 en Maths. Manon a eu 12 en NSI et Axel 10 en Physique.

1. Ecrire une variable `notes_du_groupe` de type dictionnaire pour modéliser cette situation.
2. Ajouter Liam qui a obtenu 15 en Physique.
3. Ecrire une fonction `note_min` qui prend une telle structure en paramètre et retourne le nom de l'élève qui a eu la moins bonne note, toutes matières confondues.

**Exercice 3** Ecrire une fonction `frequence_lettre` prenant en parametre une chaîne de caractère et retournant un dictionnaire:

- les clés sont les caractères de la chaîne.
- les valeurs sont les effectifs de chaque lettre dans la chaîne.

**Exercice 4** Écrire une fonction `RechercheMinMax` qui prend en paramètre un tableau de nombres non triés tab, et qui renvoie la plus petite et la plus grande valeur du tableau sous la forme d'un dictionnaire à deux clés 'min' et 'max'. Les tableaux seront représentés sous forme de liste Python.

**Exemples :**

```
>>> tableau = [0, 1, 4, 2, -2, 9, 3, 1, 7, 1]

>>> resultat = rechercheMinMax(tableau)

>>> resultat

{'min': -2, 'max': 9}

>>> tableau = []

>>> resultat = rechercheMinMax(tableau)

>>> resultat

{'min': None, 'max': None}
```