THÈME 9:
DICTIONNAIRES

Dictionnaire

- ***Tout comme une liste***, un dictionnaire permet de sauvegarder en mémoire plusieurs valeurs de types quelconques.

- ***Contrairement à une liste***, les valeurs d'un dictionnaire ne sont pas stockées dans un ordre particulier.

Création et utilisation d'un dictionnaire

- Déclarer un dictionnaire D:

```
D = { cle1: valeur1, cle2: valeur2, …, cleN: valeurN }
```

- Exemple:

```
>>> notes = { 'quentin': 15.5, 'nathan': 12.0}
```

  - La variable `notes` est un dictionnaire contenant les notes de deux étudiants.
  - Les chaines de caractères `'nathan'` et `'quentin'` sont les **clés** du dictionnaire. `12.0` et `15.5` sont les **valeurs** du dictionnaire.
  - Les éléments du dictionnaire ne sont pas ordonnés:

```
>>> notes
{'nathan': 12.0, 'quentin': 15.5}
```

Création et utilisation d'un dictionnaire

- L'accès à une valeur du dictionnaire se fait non pas par sa position (indice), mais grâce à sa **clé**.
- Exemple:

```
>>> notes = {'nathan': 12.0, 'quentin': 15.5}
>>> notes['quentin']
15.5
```

  - Ici, la chaine de caractères `'quentin'` est la clé, et la valeur qui y est associée dans le dictionnaire *`notes`* est `15.5`

- Les dictionnaires sont aussi appelés « listes associatives », car ils permettent d'associer à chaque clé une valeur de type quelconque.

Ajouter une nouvelle entrée dans un dictionnaire

- Pour **rajouter** une nouvelle entrée (clé+valeur) dans un dictionnaire existant, il suffit d'utiliser l'opérateur = en spécifiant la clé comme suit:

```
>>> D = {} # crée un dictionnaire vide
>>> D
{}
>>> D['a'] = 1 # ajout de la nouvelle entrée
>>> D
{'a': 1}
```

- Si la clé existe déjà, elle prend la nouvelle valeur:

```
>>> D['a'] = 3
>>> D
{'a': 3}
```

Supprimer une entrée d'un dictionnaire

- L'opérateur **del** permet de **supprimer** une association d'un dictionnaire:

```
>>> D = {'a': 1, 'b': 2, 'c': 3}
>>> del D['a']
>>> D
{'b': 2, 'c': 3}
```

Vérifier l'existence d'une entrée dans un dictionnaire

- Pour vérifier s'il existe une valeur associée à une clé donnée, on utilise l'opérateur **in** comme dans le cas des listes:

```
>>> prix = {'asus': 450, 'alienware': 1200, 'lenovo': 680}
>>> 'asus' in prix
True
>>> 'toshiba' in prix
False
```

ATTENTION ! L'opérateur **`in`** vérifie l'existence d'une **clé**, et non pas d'une valeur. Exemple:

```
>>> 1200 in prix
False
```

KeyError: Clé introuvable

- Si on tente d'accéder à une **entrée qui n'existe pas** dans le dictionnaire, le programme renvoie une **erreur de clé (KeyError)**, exemple:

```
>>> lettres = {'a': 103, 'b': 8, 'e': 150}
>>> lettres['k']
KeyError: 'k'
>>> lettres['u'] = lettres['u'] + 1
KeyError: 'u'
>>> del lettres['j']
KeyError: 'j'
```

- Avant d'accéder à une valeur, on prendra l'habitude de toujours vérifier l'existence de la clé:

```
if 'u' in lettres:
    lettres['u'] = lettres['u'] + 1
else:
    lettres['u']=1
```

Parcourir un dictionnaire

- La boucle for peut être utilisée pour parcourir toutes les clés d'un dictionnaire:

```
for key in D:
    print('La clé', key, 'a pour valeur: ', D[key])
```

- Exemple:

```
dates_naissance=
    {'ingrid':[12,6,1995],'marc':[27,8,1996],'brice':
[11,10,1995]}
for nom in dates_naissance :
    date = dates_naissance[nom]
    print(nom, 'fetera son anniversaire le ',
        date[0], '/', date[1], '/2017')
```

→*Affiche:*

ingrid fetera son anniversaire le 12 / 6 /2017

marc fetera son anniversaire le 27 / 8 /2017

brice fetera son anniversaire le 11 / 10 /2017

Quels types pour les clés et valeurs ?

- Comme dans le cas des listes, les valeurs dans un dictionnaire peuvent être de n'importe quel type, y compris le type dictionnaire.

```
>>> mon_pc = {
    'ram': 16,
    'cpu': 3.5,
    'portable': False,
    'os': 'windows',
    'ports': ['usb3.0', 'jack', 'ethernet', 'hdmi'],
    'carte_graphique': {
        'vram': 4,
        'nom': 'gtx970',
        'bus': 256
      }
  }
```

- En revanche, seuls certains types peuvent être utilisés comme **clés**. Dans ce cours on se limitera aux **entiers** et aux **chaînes de caractères**.

Copie de dictionnaires

- Comme dans le cas des listes, l'affectation d'un dictionnaire vers une variable ne fait que référencer le même dictionnaire, exemple:

```
>>> D = {1: 10, 2: 20, 3: 30}
>>> E = D
>>> E[5] = 50
>>> E
{1: 10, 2: 20, 3: 30, 5: 50}
>>> D
{1: 10, 2: 20, 3: 30, 5: 50}
```

- Pour créer une copie d'un dictionnaire, on utilise `dict()` :

```
>>> F = dict(D)
>>> F[6] = 60
>>> F
{1: 10, 2: 20, 3: 30, 5: 50, 6: 60}
>>> D
{1: 10, 2: 20, 3: 30, 5: 50}
```

- Ces slides ont été réalisés par:
  - Amir Charif
  - Lydie Du Bousquet
  - Aurélie Lagoutte
  - Julie Peyre
  - Florence Thiard
- Leur contenu est placé sous les termes de la licence **Creative Commons CC BY-NC-SA**