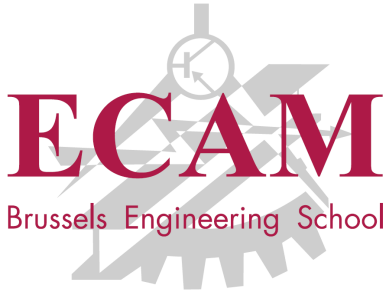



B201A Informatique appliquée

# Séance 1

# Ensemble, dictionnaire et base de données

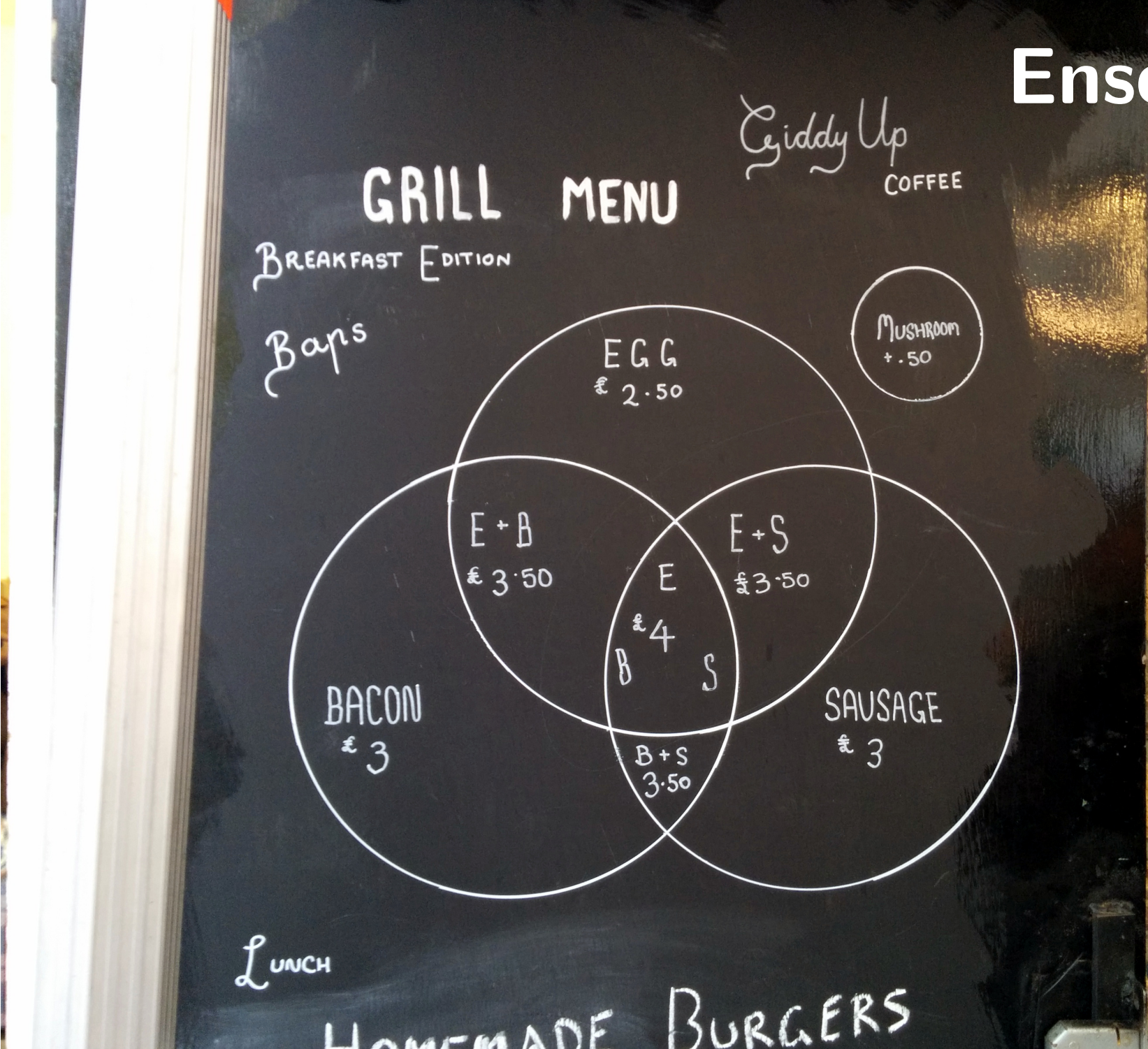
*Sébastien Combéfis, Quentin Lurkin*

*2017–2018*

# Objectifs

- Structures de données avancées
  - Ensemble et opérations ensemblistes
  - Dictionnaire et méthodes de parcours
  - Imbrication de données et structures complexes
- Introduction aux base de données
  - Format JSON et base de données orientée documents
  - Sérialisation et désérialisation de données

# Ensemble


Giddy Up
COFFEE
GRILL MENU
BREAKFAST EDITION
Baps
EGG
£2.50
MUSHROOM
+.50
E+B
£3.50
E
£4
B
S
E+S
£3.50
BACON
£3
B+S
3.50
SAUSAGE
£3
LUNCH
HOMEMADE BURGERS

# Ensemble

- Collection non ordonnée d'éléments distincts

  *Pas de doublons et pas d'ordre entre les éléments*

- Définition d'un ensemble non vide avec {}

  *Opérations des séquences (sauf modification) applicables*

```
1  numbers = {42, -2, 0, 7, 11}
2
3  print(numbers)              # {0, 42, 11, -2, 7}
4  print(len(numbers))         # 5
5  print(60 in numbers)        # False
6  print(type(numbers))        # <class 'set'>
7
8  for element in numbers:
9      print(element)
```

# Définition

- Par compréhension ou à partir d'une séquence

  *Élimination automatique des doublons*

- Ensemble vide créé avec `set()`

```
1  # {0, 42, 84, 21, 63}
2  S = {n for n in range(100) if n % 3 == 0 and n % 7 == 0}
3
4  # {0, 42, 12, -1}
5  A = set([12, 42, 0, 12, 0, -1, 0])
6
7  # {'!', 'C', 'i', 'c', 'o', 'r'}
8  B = set('Cocorico!')
```

$$S = \left\{ n \in \mathbb{N} \text{ avec } 0 \leq n < 100 \;\middle|\; n \text{ est divisible par } 3 \text{ et } 7 \right\}$$

# Modification d'un ensemble

- Modification d'un ensemble par ajout/suppression d'éléments

  *Utilisation des fonctions `add` et `remove`*

- Application de la fonction **sur** la variable contenant l'ensemble

  *Nom de la variable, suivi d'un point (.) puis*
  *du nom de la fonction à appeler, avec ses éventuels paramètres*

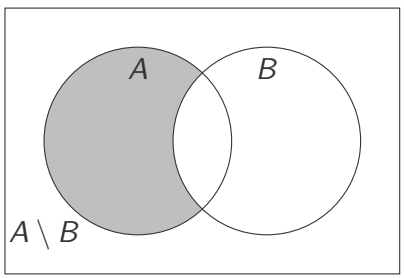
```
1  S = {1, 2, 3, 4}
2
3  S.remove(1)                  # {2, 3, 4}
4  S.remove(2)                  # {3, 4}
5  S.add(5)                     # {3, 4, 5}
```

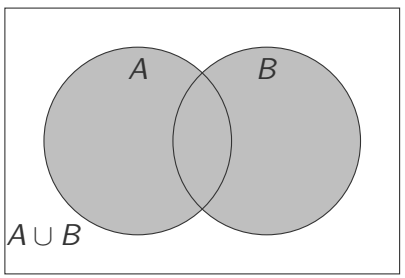
# Opération ensembliste

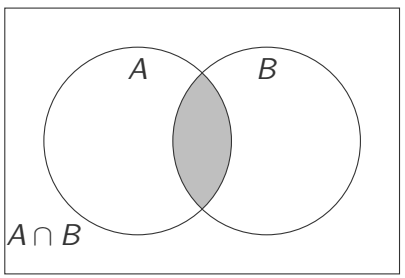
- Trois opérations ensemblistes de base

  *Différence, union et intersection d'ensembles*

- Ces opérateurs créent tous des nouveaux ensembles




```
1  A = {1, 2, 3, 4}
2  B = {3, 4, 5}
3
4  print(A - B)                # {1, 2}              (différence)
5  print(A & B)                # {3, 4}              (intersection)
6  print(A | B)                # {1, 2, 3, 4, 5}     (union)
```

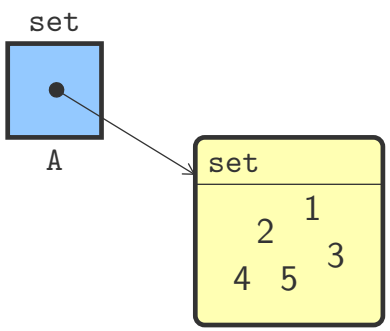
# Comparaison add et | (1)

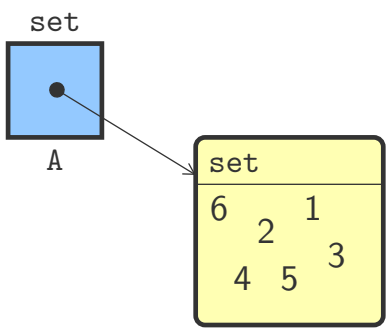
- La fonction `add` et l'opérateur `|` sont similaires

  *Tous les deux permettent d'ajouter un élément à un ensemble*

- La fonction `add` ajoute un élément à l'ensemble

```
1  A = {1, 2, 3, 4, 5}
2  A.add(6)                    # {1, 2, 3, 4, 5, 6}
```



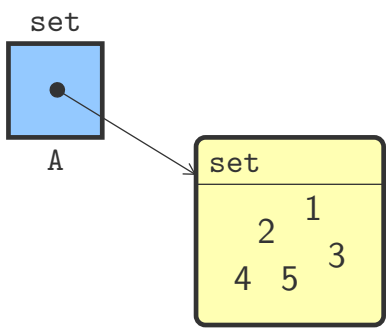
# Comparaison add et | (2)

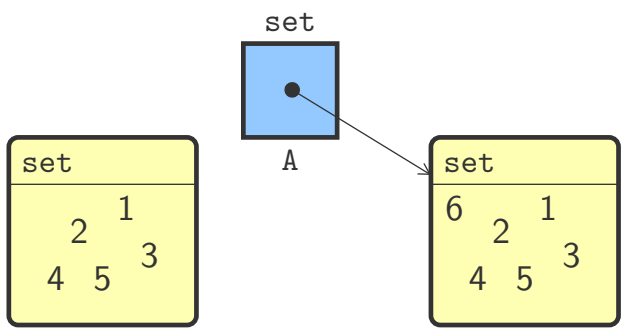
- La fonction `add` et l'opérateur `|` sont similaires

  *Tous les deux permettent d'ajouter un élément à un ensemble*

- L'opérateur `|` crée un nouvel ensemble avec un élément ajouté

```
1  A = {1, 2, 3, 4, 5}
2  A |= {6}                    # {1, 2, 3, 4, 5, 6}
```



# Éléments d'un ensemble

- Les éléments d'un ensemble doivent être uniques

  *Par conséquent, ils doivent être non modifiables*

- On ne peut pas créer un ensemble d'ensembles

```
1 # L'ensemble des sous-ensembles de {1, 2, 3}
2 A = {{}, {1}, {2}, {3}, {1, 2}, {1, 3}, {2, 3}, {1, 2, 3}}
```

```
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: unhashable type: 'set'
```

# Le type `frozenset`

- Le type `frozenset` représente un ensemble non modifiable

  *Les opérations des ensembles (sauf modification) sont applicables*

- On peut créer un ensemble d'ensembles non modifiables

  *Un `set` dont les éléments sont des `frozenset`*

```
1  # L'ensemble des sous-ensembles de {1, 2, 3}
2  A = {
3      frozenset(),
4      frozenset({1}), frozenset({2}), frozenset({3}),
5      frozenset({1, 2}), frozenset({1, 3}), frozenset({2, 3}),
6      frozenset({1, 2, 3})
7  }
```

# Somme des éléments d'un ensemble

- On déclare une variable `sum` pour la somme

  *Variable initialisée à zéro*

- On parcourt les éléments de l'ensemble en mettant `sum` à jour

  *On utilise une affectation composée avec une somme*

- On renvoie la somme

```
1  def sumElements(S):
2      sum = 0
3      for elem in S:
4          sum += elem
5      return sum
6
7  S = {1, 2, 3}
8  print(sumElements(S))                    # 6
```

# Plus grande valeur d'un ensemble

- On déclare une variable `max` pour la plus grande valeur trouvée

  *De base, on n'a rien trouvé, variable initialisée à* $-\infty$

- On parcourt les éléments de l'ensemble en mettant `max` à jour

  *Un `if` vérifie si on a un élément plus grand que l'actuel `max`*

- On renvoie la plus grande valeur trouvée

```
1  def maxElement(S):
2      max = float("-inf")
3      for elem in S:
4          if elem > max:
5              max = elem
6      return max
7
8  S = {1, 24, -2, 99, 16}
9  print(maxElement(S))                    # 99
```

# Dictionnaire

# Dictionnaire

- Ensemble de paires clé-valeur

  *Les clés sont uniques et non modifiables*

- Définition d'un dictionnaire avec {}

  *Opérations des séquences (sauf indices) applicables*

```
1  phone = {'Quentin': 8723, 'Cédric': 2837, 'Nathalie': 4872}
2
3  print(phone)                # {'Quentin': 8723, 'Cédric': 2837,
4                              #  'Nathalie': 4872}
5  print(len(phone))           # 3
6  print('Cédric' in phone)    # True
7  print(type(phone))          # <class 'dict'>
```

# Définition

- Par compréhension ou à partir d'une liste de paires clé-valeur

  *Élimination automatique des doublons*

- Dictionnaire vide avec {}

  *Attention à ne pas confondre avec l'ensemble vide*

```
1  # {1: 1, 3: 9, 9: 81, 5: 25, 7: 49}
2  square = {n : n ** 2 for n in range(1,10) if n % 2 != 0}
3
4  # {'A': 65, 'C': 67, 'B': 66, 'E': 69, 'D': 68, 'F': 70}
5  mapping = {chr(i): i for i in range(65, 71)}
```

# Accès et modification

- Accès à une valeur à l'aide de la clé entre crochets

  *Permet également la modification d'une valeur*

- Suppression paire clé-valeur avec la fonction `del`

- Deux situations si la clé n'existe pas

  - À droite de = : provoque une erreur
  - À gauche de = : ajoute une paire clé-valeur au dictionnaire

```
1  price = {"lemon": 0.85, "pear": 1}
2
3  price['lemon'] = 0.90      # {"lemon": 0.90, "pear": 1}
4  price['apple'] = 1.00      # {"lemon": 0.90, "pear": 1, "apple": 1}
5  del(price['pear'])         # {"lemon": 0.90, "apple": 1}
```

# Parcours d'un dictionnaire

- Accès aux clés avec la fonction `keys` et aux paires avec `items`

*Renvoient des séquences que l'on peut convertir en liste*

```
1  # ['lemon', 'pear', 'apple']
2  print(list(price.keys()))
3
4  # [('lemon', 0.9), ('pear', 1.0), ('apple', 1.0)]
5  print(list(price.items()))
6
7  # Parcours avec les clés
8  for fruit in price.keys():
9      print(fruit, price[fruit], sep=' : ')
10
11 # Parcours direct
12 for key, value in price.items():
13     print(key, value, sep=' : ')
```

# Somme des durées

- Liste de musiques dont il faut calculer la durée totale
- Chaque musique est stockée avec un dictionnaire

*Passer en revue chaque dictionnaire et extraire la durée*

```
1  def totalDuration(playlist):
2      duration = 0
3      for music in playlist:
4          duration += music['duration']
5      return duration
6
7  myPlaylist = [
8     {'artist': 'Axwell /\ Ingrosso', 'title': 'Sun Is Shining', '
       duration': 250},
9     {'artist': 'Black M', 'title': 'Sur ma route', 'duration': 251},
10    {'artist': 'AronChupa', 'title': "I'm an Albatraoz", 'duration':
       167}
11 ]
12 print(totalDuration(myPlaylist))              # 668
```

# Liste des titres interprétés par un artiste

- Parcours du dictionnaire, à la recherche de l'artiste
- Construction d'une liste des titres

```
1  def findTitles(playlist, artist):
2      titles = []
3      for music in playlist:
4          if music['artist'] == artist:
5              titles.append(music['title'])
6      return titles
7
8  myPlaylist = [
9     {'artist': 'Axwell /\ Ingrosso', 'title': 'Sun Is Shining', '
       duration': 250},
10    {'artist': 'Black M', 'title': 'Sur ma route', 'duration': 251},
11    {'artist': 'AronChupa', 'title': "I'm an Albatraoz", 'duration':
       167}
12 ]
13 print(findTitles(myPlaylist, "AronChupa")) # ["I'm an Albatraoz"]
```


Imbrication de données

# Imbrication de données

- Imbriquer des données

  *Insertion d'une structure de données comme élément d'une autre*

- Contraintes selon la structure principale

  *Éléments d'un ensemble et clés d'un dictionnaire non modifiables*

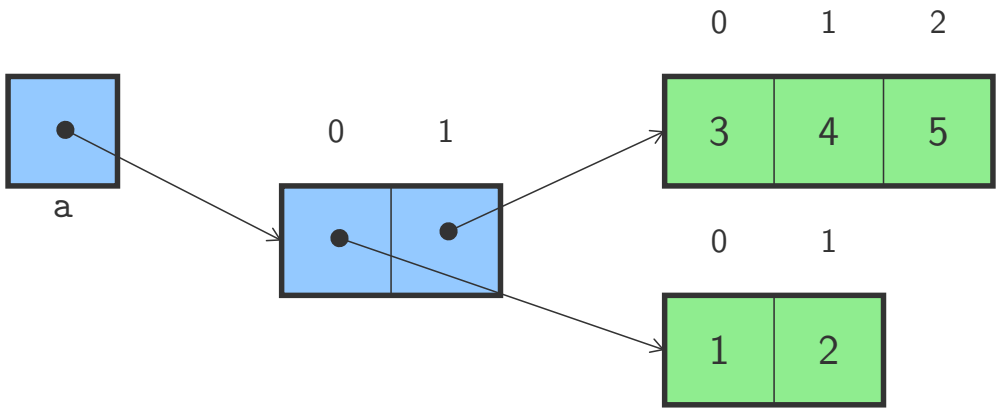
- Représentation de données complexes

  *Bien organiser et définir la structure de données*

# Liste à deux dimensions

- On peut construire une liste dont les éléments sont des listes
- Accès multiples à faire pour accéder aux éléments imbriqués

```
1  A = [1, 2]
2  B = [3, 4, 5]
3  L = [A, B]                    # Équivalent à L = [[1, 2], [3, 4, 5]]
4
5  print(L[0][1])                # 2
6  print(L[1][2])                # 5
```



# Parcours d'une liste à deux dimensions

- Parcours à l'aide d'une double boucle

  *Imbrication d'une boucle dans une autre, `while` ou `for`*

- La première boucle passe en revue les listes imbriquées

  *La seconde boucle parcourt les éléments de chaque liste imbriquée*

```
1  L = [[1, 2], [3, 4, 5]]
2
3  for elem in L:                  # elem est une liste
4      for data in elem:           # data est un nombre entier
5          print(data, end=' ')
6      print('| ', end='')
```

```
 1 2 | 3 4 5 |
```

# Représentation d'une matrice

- Matrice représentée par une liste à deux dimensions

  *Toutes les listes imbriquées ont le même nombre d'éléments*

- Stockage d'une liste des lignes de la matrice
  - La première dimension représente les lignes
  - Colonnes représentées par la deuxième dimension

```
1 M = [[1, 2, 3], [4, 5, 6]]
```

$$M = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix}$$

Par exemple, `M[1][0]` vaut 4

# Parcours d'une matrice

- Dimension de la matrice obtenue avec la fonction `len`

  *La matrice a `len(M)` lignes et `len(M[0])` colonnes*

- Parcours avec boucle `while` et indices ou avec boucle `for`

```
1  # Avec une boucle while
2  i = 0
3  while i < len(M):
4      j = 0
5      while j < len(M[0]):
6          print(M[i][j])
7          j += 1
8      i += 1
9
10 # Avec une boucle for
11 for line in M:
12     for elem in line:
13         print(elem)
```

# Structures imbriquées (1)

- Imbrication d'autres structures dans des listes

  *Liste de tuples, ensembles et dictionnaires*

```
1  # Liste de tuples
2  coords = [(0,0), (7,-2), (4,5), (-3,-9)]
3
4  # Liste d'ensembles
5  lunches = [
6      {'apple', 'banana', 'grape'},
7      {'yogurt', 'cereals'},
8      {'bread', 'cheese', 'ham'},
9      {'sausage'}
10 ]
11
12 # Liste de dictionnaires
13 contacts = [
14     {'firstname': 'Alexis', 'lastname': 'King'},
15     {'firstname': 'Brice', 'lastname': 'Monster'},
16     {'firstname': 'Sébastien', 'lastname': 'Adams'}
17 ]
```

# Structures imbriquées (2)

- Structures imbriquées en clés et valeurs de dictionnaires

  *Les clés d'un dictionnaire doivent être non modifiables*

```
1  # Tuples en clés d'un dictionnaire
2  config = {
3      (0, 0): 'Arnaud',
4      (2, 1): 'Louis',
5      (-1, 3): 'Marie',
6      (3, -1): 'Dan'
7  }
8
9  # Listes en valeurs d'un dictionnaire
10 config = {
11     (0, 0): ['Arnaud', 'Pierre'],
12     (2, 1): ['Louis'],
13     (-1, 3): ['Marie', 'Éric', 'Tom'],
14     (3, -1): ['Dan']
15 }
```

# Imbrication complexes

- On peut imbriquer des structures à plusieurs niveaux

  *Des séquences, ensembles et dictionnaires*

```
1  address = {'street': "Promenade de l'Alma", 'number': 50, 'zip':
   1200, 'city': "Woluwé-Saint-Lambert"}
2  marchand = {'firstname': "Cédric", 'lastname': "Marchand", 'address
   ': address}
3
4  # Équivalent à
5  # marchand = {'firstname': "Cédric", 'lastname': "Marchand", '
   address': {'street': "Promenade de l'Alma", 'number': 50, 'zip':
   1200, 'city': "Woluwé-Saint-Lambert"}}
6
7  print(marchand['firstname'])              # Cédric
8  print(marchand['address']['city'])        # Woluwé-Saint-Lambert
```

# Copie (1)

- Affecter une même liste à deux variables crée un alias

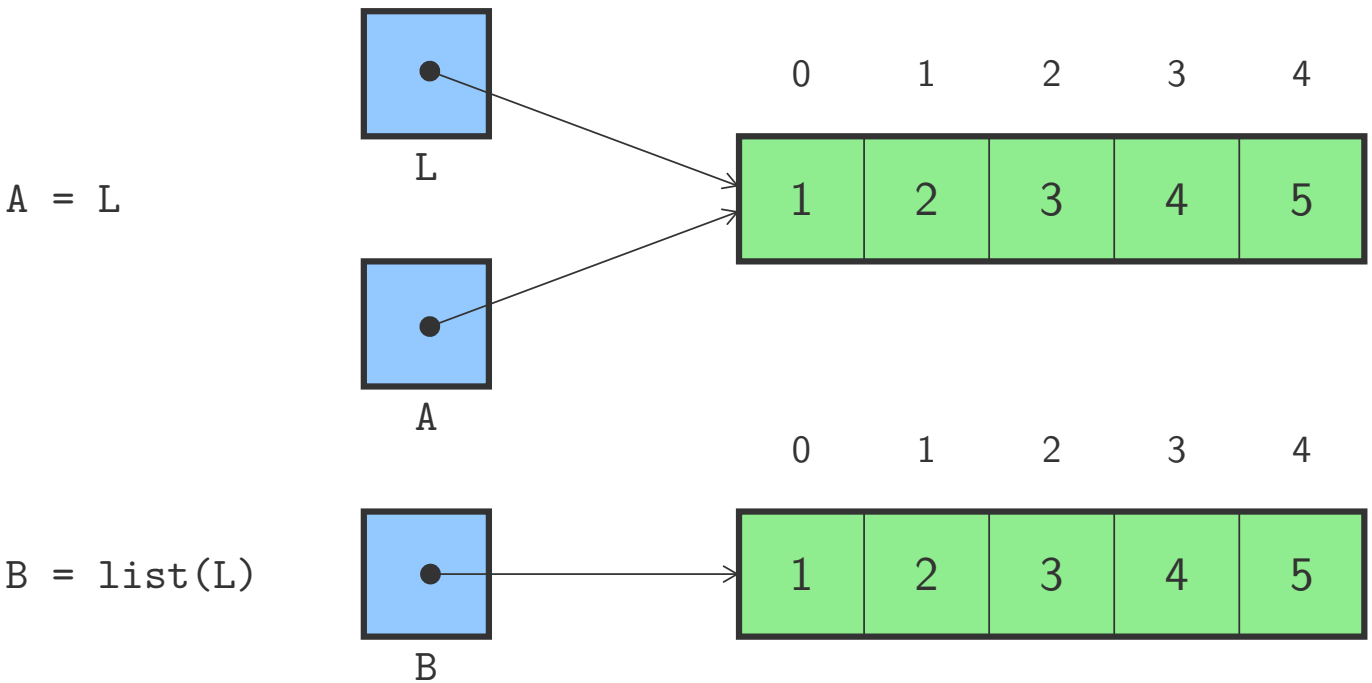
  *Même chose pour les séquences, les ensembles et les dictionnaires*

- On crée une véritable copie de liste avec la fonction `list`

  *Ou avec les fonctions `set`, `dict`...*

```
1  L = [1, 2, 3, 4, 5]
2  A = L                                  # A est un alias de L
3  A[0] = 42
4  print(L)                               # [42, 2, 3, 4, 5]
5
6  L = [1, 2, 3, 4, 5]
7  B = list(L)                            # B est une copie de L
8  B[0] = 42
9  print(L)                               # [1, 2, 3, 4, 5]
```

# Copie (2)

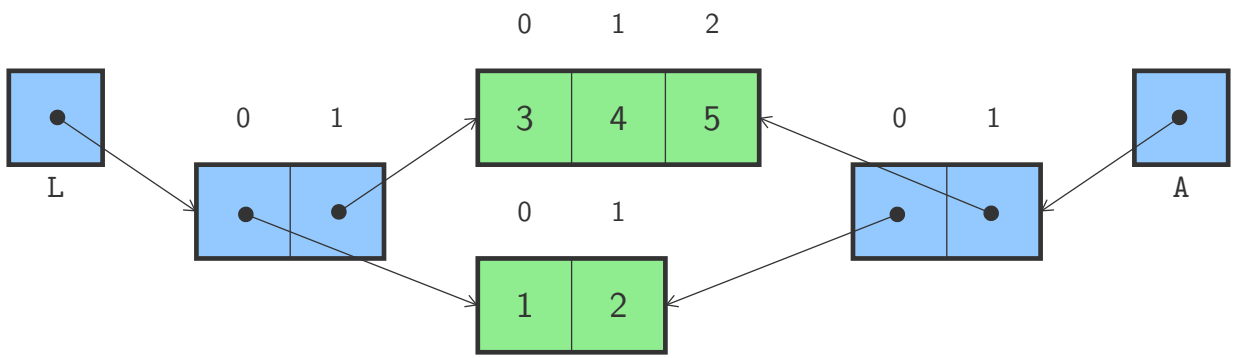


# Copie de structures imbriquées

- Pas de soucis de copies pour les collections non modifiables
- La copie ne se fait pas en profondeur

  *Seuls les éléments de « premier niveau » sont copiés*

```
1  L = [[1, 2], [3, 4, 5]]
2  A = list(L)
3
4  A[1][0] = 42
5  print(L)                          # [[1, 2], [42, 4, 5]]
```



# Module copy

- Deux fonctions proposées par le module copy
  - `copy` pour une copie « *shallow* »
  - `deepcopy` pour une copie « *deep* »
- Une copie en profondeur peut prendre du temps

  *Et aussi consommer beaucoup d'espace mémoire*

```
1  import copy
2
3  L = [[1], [2, 3], [4, 5, 6]]
4  A = copy.copy(L)                    # A est une copie shallow de L
5  B = copy.deepcopy(L)                # A est une copie deep de L
```


# Base de données

# Base de données

- Collection de données organisées selon un format choisi

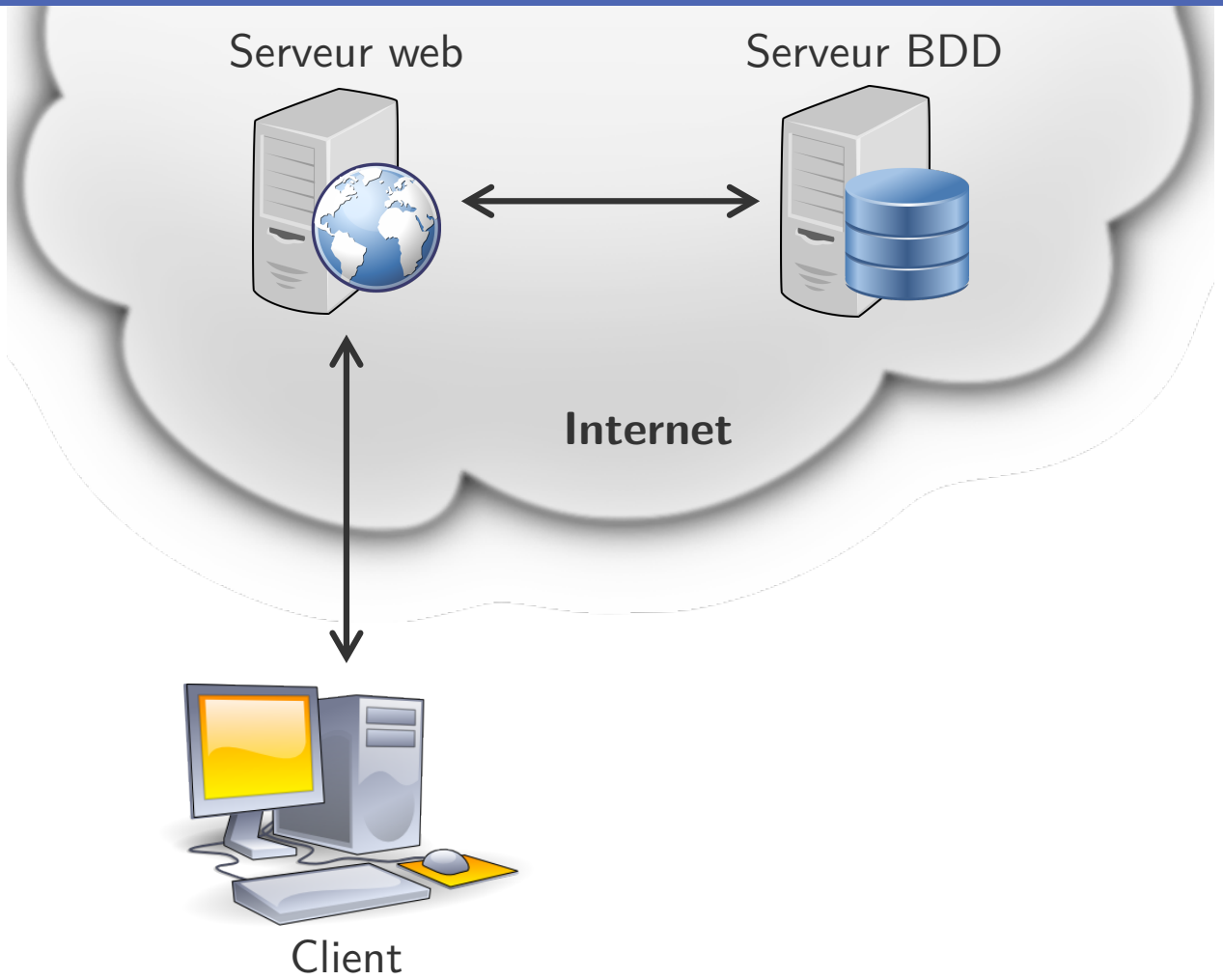
  *Similitudes avec les dictionnaires Python*

- Système de gestion de bases de données (SGBD)

  *Ensemble de logiciels qui gère la base de données*

- Plusieurs opérations effectuées par le SGBD
  - Interrogation de la base de données
  - Gestion du stockage des données sur le disque
  - ...

# Accès à un site web



# JavaScript Object Notation (JSON) (1)

- Permet de représenter des objets

  *Notation issue de la notation des objets Javascript*

- Ensemble de paires (étiquette, valeur)

  *Étiquette entre guillemets, deux-points et valeur*

- Valeurs sont soit une simple valeur, soit une liste de valeurs

  *Liste de valeurs délimitée par des crochets []*

# JavaScript Object Notation (JSON) (2)

```
{
    "name": "Carnet d'adresses de Sébastien Combéfis",
    "contacts": [
        {
            "firstname": "Cédric",
            "lastname": "Marchand",
            "phone": 2837
        },
        {
            "firstname": "Jonathan",
            "lastname": "Verlant-Chenet",
            "phone": 4872
        },
        {
            "firstname": "Quentin",
            "lastname": "Lurkin",
            "phone": 8723
        }
    ]
}
```

# Base de données orientée document

- Stocke des documents JSON

  *Souplesse par l'absence de schéma prédéfini*

- Données répartissables sur plusieurs machines

  *Robustesse, duplication, efficacité...*

- Sérialisation et désérialisation des documents JSON

  *Vers des objets du langage cible, pour être utilisés*

# Sérialisation

- Sérialisation d'un dictionnaire en document JSON

  *Dictionnaire Python (`dict`) → document JSON (`str`)*

- Fonction `dumps` du module `json`

  *Renvoie une chaine de caractères*

```
1  import json
2
3  bb = {'seasons': 5, 'genre': ['crime drama', 'thriller']}
4  skins = {'seasons': 7, 'genre': ['teen drama', 'comedy drama']}
5  tvshows = {'Breaking Bad': bb, 'Skins': skins}
6
7  document = json.dumps(tvshows, indent=4)
```

# Désérialisation

- Désérialisation d'un document JSON en dictionnaire

  *Document JSON (`str`) → dictionnaire Python (`dict`)*

- Fonction `loads` du module `json`

  *Renvoie un dictionnaire*

```
1  import json
2
3  document = '{"Belgium":{"capital":"Brussels","languages":["french
   ","dutch","german"]},"China":{"capital":"Beijing","languages":["
   mandarin chinese"]}}'
4
5  countries = json.loads(document)
```

# Crédits

- https://www.flickr.com/photos/doctorow/11082104723
- https://www.flickr.com/photos/greeblie/3338710223
- https://www.flickr.com/photos/ken-ichi/4577032677
- https://www.flickr.com/photos/shindotv/3835365695
- https://openclipart.org/detail/25319/cartoon-cloud
- https://openclipart.org/detail/17924/computer
- https://openclipart.org/detail/163741/web-server
- https://openclipart.org/detail/163711/database-server
- https://en.wikipedia.org/wiki/File:MongoDB-Logo.svg
- https://en.wikipedia.org/wiki/File:CouchDB.svg
- https://en.wikipedia.org/wiki/File:OrientdbLogo.png
- https://en.wikipedia.org/wiki/File:CouchbaseLogo.svg