# Définition de tuples ( elts )

# Accéder aux valeurs d'un tuple

>>> data[1] # Le second

20

## Vérifier la présence ou non d'une données dans le tuple

>>> is\_present = 30 in l

## Parcourir un tuple

Via indice ou index c-a-d len et while

FOR value in data

## La notion de « tuple unpacking »

Le « tuple unpacking » (déballage de tuple, en français) consiste à sortir les valeurs stockées dans un tuple dans un ensemble de variables nommées.

ATTENTION nombre égaux de valeurs à prendre en considération

>>> data = (10, 20, 30)

>>> a, b, c = data

>>> a

10

>>> b

20

>>> c

30

>>> a, b = data

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

ValueError: too many values to unpack (expected 2)

>>>

# La notion de « slicing » data[indiceDebut  : indiceFin +1 ]

>>> data = (10, 20, 30, 40, 50)

**>>> data[1:3]**

(20, 30)

>>> data[1:4]

(20, 30, 40)

>>> data[1:5]

(20, 30, 40, 50)

# Utilisation conjointe d'un tuple et d'un générateur

Compléments sur la manipulation de vos tuples

Quelques fonctions applicables aux tuples : count and index

Quelques méthodes de la classe tuple

Quelques opérateurs de la classe tuple

**Travaux pratiques : Les énoncés**

**Exercice 1**

A partir du tuple suivant comme base de travail, Il faut constitué de triplets d'informations associés à des villes. La première information est le nom de la ville à visiter (dans l'exemple, des villages du sud de la France ), la seconde information est la latitude du lieu et enfin on trouve la longitude. Il peut y avoir un nombre quelconque de triplet.

data = (

"Andon", 43.77348, 6.78596,

"Caille", 43.77887, 6.72979,

"Séranon", 43.77435, 6.70415,

"Valderoure", 43.79629, 6.70749

# Vous ne devez pas présager du nombre de triplets.

)

**A Faire**

Extraire les données du tuple et produire l'affichage suivant :

+---------------+---------------+---------------+

| Ville | Latitude | Longitude |

+---------------+---------------+---------------+

| Andon | 43.77348 | 6.78596 |

| Caille | 43.77887 | 6.72979 |

| Séranon | 43.77435 | 6.70415 |

| Valderoure | 43.79629 | 6.70749 |

+---------------+---------------+---------------+

le chapitre sur le formatage des chaînes de caractères pourrait aussi vous être utile !

**Exercice 2 :**

vous devez produire le même affichage que dans l'exercice précédent, mais en utilisant le tuple de tuples suivant comme données d'entrée.

data = (

("Andon", 43.77348, 6.78596),

("Caille", 43.77887, 6.72979),

("Séranon", 43.77435, 6.70415),

("Valderoure", 43.79629, 6.70749)

# Vous ne devez pas présager du nombre de triplets.

)

Un tuple de tuple

# Présentation du type set ou les ensembles mathématiques

**dataSet = {liste des éléments}** ensemble au sens math

Opérations possibles sur vos sets : ajout de données, suppression de données, recherche de données.

Opérations mathématiques : unions | , les intersections **(l'opérateur &)** , les différences et différences symétriques d'ensembles **(opérateur ^.).**

Ce qui est remarquable, lors des affichages des sets, c'est que les données ne se retrouvent pas dans l'ordre initial. Cela est dû au fait, que le type [set](https://koor.fr/Python/API/python/builtins/set/Index.wp) organisme à sa manière les données afin de pouvoir très rapidement savoir si une donnée est déjà présente ou non dans le set

opérateur ^.

## Instanciation d'un set

## Vérifier la présence ou non d'une donnée dans un set

not in or in

## Ajouter des valeurs dans un set (méthode add)

dataSet.add(newValue)

## Supprimer des valeurs d'un set (methode remove et clear() )

DataSet.remove(existingValue)

dataSet.clear() efface tous les éléments de l’ensemble

## Parcourir un set de données

# Fonctions ou méthodes applicables aux sets

### Quelques méthodes de la classe set

Vous vous en doutez, la classe [set](https://koor.fr/Python/API/python/builtins/set/Index.wp) expose aussi de nombreuses méthodes bien utiles. En voici quelques-unes : vous pouvez cliquer sur le nom de la méthode pour en obtenir un descriptif plus détaillé.

| **Méthode** | **Description** |
| --- | --- |
| [add](https://koor.fr/Python/API/python/builtins/set/add.wp) | Ajoute une nouvelle valeur au set courant. |
| [clear](https://koor.fr/Python/API/python/builtins/set/clear.wp) | Supprime toutes les valeurs stockées dans le set. |
| [copy](https://koor.fr/Python/API/python/builtins/set/copy.wp) | Renvoie une copie « peu profonde » (shallow copy, en anglais) du set courant. |
| [difference](https://koor.fr/Python/API/python/builtins/set/difference.wp) | Calcule la différence de deux ensembles. |
| [difference\_update](https://koor.fr/Python/API/python/builtins/set/difference_update.wp) | Supprime tous les éléments d'une autre collection dans le set courant. |
| [discard](https://koor.fr/Python/API/python/builtins/set/discard.wp) | Retire la valeur spécifiée du set courant. |
| [intersection](https://koor.fr/Python/API/python/builtins/set/intersection.wp) | Calcule l'intersection de deux ensembles. |
| [intersection\_update](https://koor.fr/Python/API/python/builtins/set/intersection_update.wp) | Permet de conserver dans un premier ensemble uniquement les données qui sont présentes dans un second ensemble. |
| [isdisjoint](https://koor.fr/Python/API/python/builtins/set/isdisjoint.wp) | Indique si deux ensembles sont disjoints. |
| [issubset](https://koor.fr/Python/API/python/builtins/set/issubset.wp) | Indique si un ensemble est disjoint d'un autre. |
| [issuperset](https://koor.fr/Python/API/python/builtins/set/issuperset.wp) | Indique si un ensemble contient un autre ensemble. |
| [pop](https://koor.fr/Python/API/python/builtins/set/pop.wp) | Extrait la première valeur du set et la retourne. |
| [remove](https://koor.fr/Python/API/python/builtins/list/remove.wp) | Retire la valeur spécifiée du set courant ou déclenche une erreur si cette valeur n'existe pas. |
| [symmetric\_difference](https://koor.fr/Python/API/python/builtins/set/symmetric_difference.wp) | Calcule la différence symétrique de deux ensembles. |
| [symmetric\_difference\_update](https://koor.fr/Python/API/python/builtins/set/symmetric_difference_update.wp) | Calcule la différence symétrique de deux ensembles et stocke ce résultat dans le set courant. |
| [union](https://koor.fr/Python/API/python/builtins/set/union.wp) | Réalise l'union (la fusion) de deux sets. |
| [update](https://koor.fr/Python/API/python/builtins/set/update.wp) | Ajoute au set courant les valeurs du set passé en paramètre. |

### Quelques opérateurs de la classe set

De nombreux autres opérateurs sont aussi proposés par la classe [set](https://koor.fr/Python/API/python/builtins/set/Index.wp). On retrouve bien entendu les opérateurs d'union |, d'intersection &, de différence - et de différence symétrique ^, l'opérateur in et l'opérateur not in.

# **Travaux pratiques: énoncés**

**Exercice 1 :**

Considérons la liste de chaînes de caractères suivante :

names = ["Jean", "Alphonse", "Gilbert", "Bruno", "Jacques", "Jean", "Kevin", "Guido", "Bruno", "Frank"]

Construire une seconde liste sans doublons et triée par ordre alphabétique.

**Exercice 2 :**

**Considérons la structure de données ci-dessous :**

**names = [**

**("Anakin", "Skywalker"),**

**("Luke", "Skywalker"),**

**("Leia", "Skywalker"),**

**("Obi-Wan", "Kenobi"),**

**("Han", "Solo"),**

**("Ben", "Solo")**

**]**

**Afficher la liste ordonnée des noms de famille présents, sans doublons, bien entendu.**

# Manipulation de dictionnaires

Principe : (key : value)

## Présentation du type dict

### Instanciation d'un dictionnaire

**empty\_dict = {} comme un ensemble à la place des valeurs, il faut définir des paires de valeurs**

**(key,value) key :value**

NOTE : dans certains autres langages de programmation, C++ ou Java par exemple, on parle de « hashtable » ou encore de « map

Instanciation d'un dictionnaire

***>>> months = {"janvier": 31, "février": 28, "mars": 31, "avril": 30}***

*>>> len(months)*

*4*

*>>> months["mars"]*

*31*

*>>> months["truc"]*

*Traceback (most recent call last):*

*File "<stdin>", line 1, in <module>*

***KeyError****: 'truc'*

*>>>*

NOTE :

Une autre possibilité pour accéder à une donnée d'un dictionnaire consiste à utiliser la méthode [**a\_dict.get(key)**](https://koor.fr/Python/API/python/builtins/dict/get.wp). La différence avec l'opérateur [], vu dans l'exemple précédent, c'est qu'en cas de clé non existante, la valeur None vous sera retournée plutôt qu'une exception. En voici un exemple.

**Vérifier la présence ou non d'une association dans un dictionnaire**

Utilisation de l'opérateur not in *ou bien*

In

**dico.values()**

>>> months = {"janvier": 31, "février": 28, "mars": 31, "avril": 30}

# Ici on teste une clé et non une valeur.

s>>> 31 in months

False

>>> 31 in months**.values()**

True

>>> 32 in months.values()

False

>>>

**Ajouter des valeurs dans un dictionnaire**

from datetime import date

tasks = {

... date(2022, 3, 26): ["Apprendre Python", "Manger", "Dormir"],

... date(2022, 3, 27): ["Poursuivre l'apprentissage de Python", "..."]

... }

# Ajout d'une nouvelle liste dans le dictionnaire

**tasks[date(2022, 3, 28)] = ["Se reposer un peu", "Dormir"]**

# On accède à cette nouvelle donnée

for task in tasks[date(2022, 3, 28)]:

... print(task)

...

Se reposer un peu

Dormir

>>>

**TP**

**Exercice 1** : écrire un programme qui permet de saisir (dans une boucle) des noms de mois en anglais. A chaque saisie, il vous est demandé de retrouver dans un dictionnaire le nombre de jours associé au mois et de l'afficher (pour le mois de février, on reverra 28). Si l'utilisateur saisi exit, le programme doit s'arrêter.

**Exercice 2** : reprendre le programme précédent et permettre aussi la saisir des noms de mois en français. Pour ce faire, vous devrez utiliser un second dictionnaire pour gérer la traduction des mois en français vers les mois en anglais.