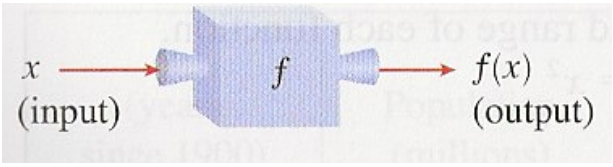
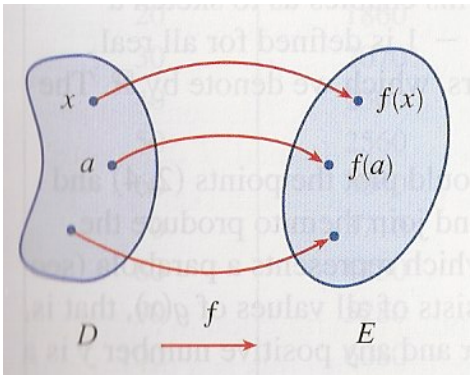
Les dictionnaires, petite introduction

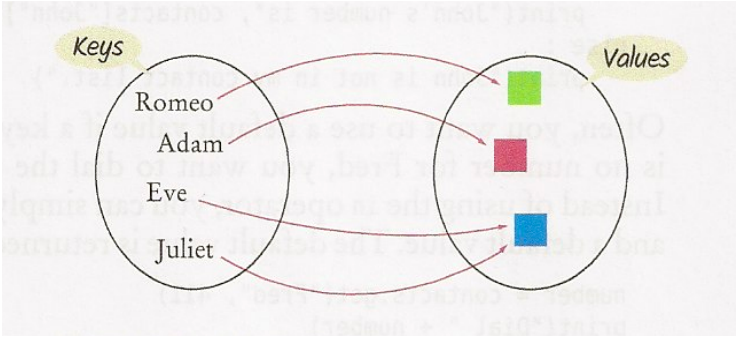
Pour stocker des données, nous avons vu à présent deux types de « ***conteneur*** » : les listes (`list`) et les tableaux (`array`). Il existe d'autres façons dans Python de stocker des données comme les `tuples`, les ensembles (`set`). Nous allons découvrir à présent les dictionnaires (`dictionary` (`ies`) pluriel).

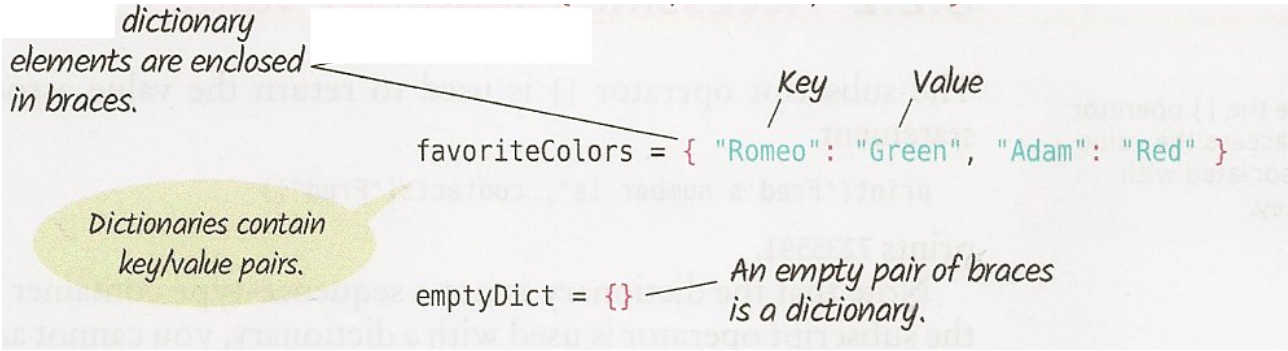
En mathématiques, une fonction $f:\begin{cases} D\subset\Re\to E\subset\Re \\ x\mapsto y \end{cases}$ est une « règle » qui associe à chaque $x\in D$ ***exactement un seul élément*** $y=f\left(x\right)\in E$.





De façon similaire un dictionnaire associe à ***une clé*** (key) ***une unique valeur*** (value). ***La clé est associée à une unique valeur mais une valeur donnée peut-être associée à plusieurs clés.***





Contrairement aux éléments d'une liste, les paires (clé, valeur) d'un dictionnaire ***ne sont pas stockées dans un ordre particulier*** et ne peuvent donc pas être accessibles par leur position. La commande `nom_dictionnaire[i]` n'a pas de sens pour un dictionnaire.

Un `dictionnaire` est un ***conteneur*** (de données) ***mutable*** (que l'on peut modifier) et non ordonné. Une `liste` est un ***conteneur mutable ordonné***. Une `chaîne de caractère` est un ***conteneur non mutable*** (non modifiable).

La table ci-dessous donne les principales opérations sur les tableaux que nous allons découvrir.

Table 2 Common Dictionary Operations

Operation	Description
`d = dict()` `d = dict(c)`	Creates a new empty dictionary or a duplicate copy of dictionary *c*.
`d = {}` `d = {k1: v1, k2: v2, ..., kn: vn}`	Creates a new empty dictionary or a dictionary that contains the initial items provided. Each item consists of a key (*k*) and a value (*v*) separated by a colon.
`len(d)`	Returns the number of items in dictionary *d*.
key `in` *d* *key* `not in` *d*	Determines if the key is in the dictionary.
`d[key] = value`	Adds a new *key*/*value* item to the dictionary if the *key* does not exist. If the key does exist, it modifies the value associated with the key.
`x = d[key]`	Returns the value associated with the given key. The key must exist or an exception is raised.
`d.get(key, default)`	Returns the value associated with the given key, or the default value if the key is not present.
`d.pop(key)`	Removes the key and its associated value from the dictionary and returns the value. Raises an exception if the key is not present.
`d.values()`	Returns a sequence containing all values in the dictionary.

Les dictionnaires

1. Création d'un dictionnaire

Le dictionnaire ci-dessous recense les quatre femmes lauréates du prix Nobel de Physique depuis sa création.

```
Nobel_physique_femme={"Curie":1903,"Goeppert-Mayer":1963,"Strickland":2018,"Ghez":2020}
```

La ***clé***, ici de type chaîne de caractère, correspond au nom de la récipiendaire et l'année de la récompense correspond à la ***valeur,*** ici un entier.
Pour l'histoire, il y a eu six femmes prix Nobel de Chimie dont trois Françaises : Marie Curie en 1911, sa fille Irène Curie en 1935 et Emmanuelle Charpentier en 2020. Seules quatre personnes ont obtenu deux prix Nobel, dont une seule femme : Marie Curie. La Canadienne Donna Strickland, pionnière dans le domaine des Lasers, a partagé le prix Nobel de Physique avec Gérard Mourou, son directeur de thèse,

La commande `dict` permet de dupliquer un dictionnaire :

```
New_Nobel_physique_femme=dict(Nobel_physique_femme)
```

2. Accès aux valeurs d'un dictionnaire

L'opérateur `[]` est utilisé pour retourner la valeur associée à une clé. On ne peut pas accéder à une valeur par un indice mais seulement par une clé.

```
print("La physicienne Strickland a eu le prix Nobel de Physique en",Nobel_physique_femme["Strickland"])
```

```
La physicienne Strickland a eu le prix Nobel de Physique en 2018
```

Pour vérifier si une clé est présente dans le dictionnaire, on peut utiliser les opérateurs `in` ou `not in`:

```
if "Alice" in Nobel_physique_femme:
    print("Alice a eu le prix nobel en",Nobel_physique_femme["Alice"] )
else:
    print("Alice n'a pas encore obtenu le prix Nobel de Physique mais c'est pour bientôt :-)")
print()
```

```
Alice n'a pas encore obtenu le prix Nobel de Physique mais c'est pour bientôt :-)
```

3. Manipuler un dictionnaire

✔ Ajouts de paires (clé, valeur)

```
Nobel_physique_femme["Annabelle"]=2040
Nobel_physique_femme["Alice"]=2041
Nobel_physique_femme["Ilona"]=2042
Nobel_physique_femme["Lili"]=2043

if "Ilona" in Nobel_physique_femme:
    print("Ilona a eu le prix Nobel en",Nobel_physique_femme["Ilona"] )
else:
    print("Ilona n'a pas encore obtenu le prix Nobel de Physique mais c'est pour bientôt :-)")
```

```
Ilona a eu le prix Nobel en 2042
```

Les dictionnaires

✔ Changer la valeur associée à une clé

```
Nobel_physique_femme["Annabelle"]=2040
Nobel_physique_femme["Alice"]=2041
Nobel_physique_femme["Ilona"]=2042
Nobel_physique_femme["Lili"]=2043

print("Lili a eu le prix Nobel en",Nobel_physique_femme["Lili"] )

Nobel_physique_femme["Lili"]=2053

print("Lili a eu le prix Nobel en",Nobel_physique_femme["Lili"] )
```

```
Lili a eu le prix Nobel en 2043
Lili a eu le prix Nobel en 2053
```

✔ Enlever une paire (clé, valeur) avec la méthode .pop

```
Nobel_physique_femme.pop("Annabelle")

if "Annabelle" in Nobel_physique_femme:
    print("Annabelle a eu le prix Nobel en",Nobel_physique_femme["Annabelle"] )
else:
    print("Annabelle n'a pas encore obtenu le prix Nobel de Physique mais c'est pour bientôt :-)")
```

```
Annabelle n'a pas encore obtenu le prix Nobel de Physique mais c'est pour bientôt :-)
```

La méthode `.pop` ***élimine à la fois la clé et sa valeur du dictionnaire***.

✔ Parcourir un dictionnaire

```
Nobel_physique_femme={"Curie":1903,"Goeppert-Mayer":1963,"Strickland":2018,"Ghez":2020}

print("Dates des prix Nobel de Physique féminins:")
for cle in Nobel_physique_femme:
     print(cle,Nobel_physique_femme[cle])
```

```
Dates des prix Nobel de Physique féminins:
Curie 1903
Goeppert-Mayer 1963
Strickland 2018
Ghez 2020
```

On peut vouloir parcourir le dictionnaire sur les ***valeurs*** plutôt que sur les ***clés*** en utilisant la méthode `.values`. Cela peut être intéressant pour créer une liste avec uniquement les valeurs des dates du dictionnaire dans notre exemple.

```
Date_Nobel_physique_femme=[] #creation d'une liste vide

for date in Nobel_physique_femme.values():
    Date_Nobel_physique_femme.append(date)

print(Date_Nobel_physique_femme)
```

```
[1903, 1963, 2018, 2020]
```

4. Exemple : les polynômes en tant que dictionnaires

Il est facile et commode de stocker et de manipuler un polynôme à l'aide d'un dictionnaire. Considérons par exemple le polynôme suivant $P(x) = -1 + x^2 + 3x^7$.

On peut utiliser le dictionnaire suivant pour relier les clés, les puissances, avec les valeurs, les coefficients :

```
P={0:-1, 2:1, 7:3}
```

Il est possible d'utiliser une liste mais dans ce cas il faut noter tous les zéros puisque les indices de la liste sont reliés à la puissance :

```
P=[-1,0,1,0,0,0,0,3]
```

L'avantage du dictionnaire est évidente ici, il suffit de stocker uniquement les coefficients non nuls. Pour le polynôme $1 + x^{100}$, le dictionnaire nécessite deux valeurs contre 101 éléments avec une liste !!

La fonction suivante permet d'évaluer le polynôme, représenté par un dictionnaire, pour différentes valeurs de x.

```
def eval_poly_dic(poly,x):
    sum=0.0
    for power in poly :
        sum += poly[power]*x**power
    return sum

P={0:-1, 2:1, 7:3}
x=4
evaluation= eval_poly_dic(P,x)
print("P(",x,")=",evaluation)
```

```
P( 4 )= 49167.0
```

Avec un dictionnaire, il est facile de représenter un polynôme avec des puissances négatives tel que $P(x) = \frac{1}{2}x^{-3} + 2x^4$ (Cela est plus délicat avec des listes).

```
P={-3:0.5, 4:2}
```

Si l'on essaie de taper pour ce dernier dictionnaire `P[1],` nous allons obtenir un message d'erreur `KeyError` puisque 1 n'est pas une clé pour `P`. Pour éviter cela, nous pouvons utiliser la méthode `.get` qui retourne ***une valeur par défaut***.

```
P={-3:0.5, 4:2}
valeur=P.get(5, 0.0)
print(valeur)
```

```
0.0
```

Dans cet exemple, la valeur retournée par défaut est 0.0 puisque la clé 5 n'existe pas dans le dictionnaire.

5. Exercices d'application

Exercice 1 : Créer un dictionnaire à partir d'une table.

A l'adresse https://github.com/hplgit/scipro-primer/blob/master/src/dictstring/constants.txt, vous pouvez récupérer le fichier texte suivant qui contient une table des valeurs et des unités des constantes fondamentales.

```
name of constant          value             dimension
------------------------------------------------------------
speed of light            299792458.0       m/s
gravitational constant    6.67259e-11       m**3/kg/s**2
Planck constant           6.6260755e-34     J*s
elementary charge         1.60217733e-19    C
Avogadro number           6.0221367e23      1/mol
Boltzmann constant        1.380658e-23      J/K
electron mass             9.1093897e-31     kg
proton mass               1.6726231e-27     kg
```

⌨ Ecrire une fonction qui lit et interprète le texte et qui finalement retourne un dictionnaire. Ce dernier aura pour clés le nom des constantes fondamentales auxquelles seront associées les valeurs des constantes correspondantes. Vous pouvez travailler avec les noms en anglais ☺.

Exercice 2 : Dériver un polynôme.

Nous avons vu qu'un polynôme peut être représenté par un dictionnaire. Il est donc aussi possible de représenter sa dérivée par un autre dictionnaire.

⌨ Ecrire une fonction qui aura pour paramètre d'entrée le dictionnaire caractérisant le polynôme et qui retournera le dictionnaire représentant sa dérivée.