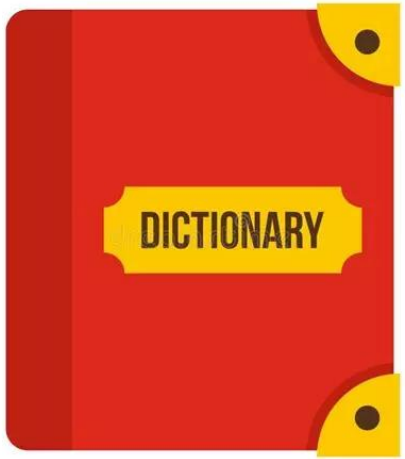
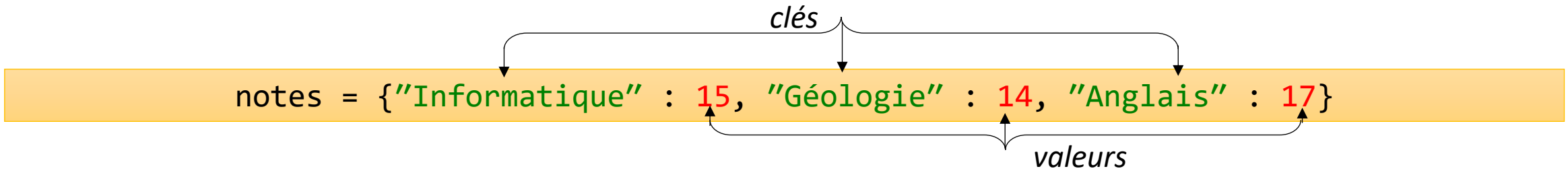
Les dictionnaires en Python



Qu'est-ce qu'un dictionnaire ?

- Structure de données qui associe des clés à des valeurs



- Non ordonné (i.e. il n'y a pas de 'positions')
- Les clés sont uniques et de type immuable (str, int, tuple, etc.)
- Les valeurs peuvent être de n'importe quel type.

Exemple 1

```
region_metropolitaine = {”ARA” : ”Lyon”, ”BFC” : ”Dijon”, BRE” : ”Rennes” , ”CVL” :
”Orléans”, ”20R” : ”Ajaccio”, ”GES” : ”Strasbourg”, ”HDF” : ”Lille”, ”IDF” : ”Paris”,
”NOR” : ”Rouen”, ”NAQ” : ”Bordeaux”, ”OCC” : ”Toulouse”, ”PDL” : ”Nantes”, ”PAC” :
”Marseille”}
```

```
region_metropolitaine = {”ARA” : [”Lyon”, 69711], ”BFC” : [”Dijon”,47784], BRE” :
[”Rennes”,27208] , ”CVL” : [”Orléans”, 39151], ”20R” : [”Ajaccio”,8680], ”GES” :
[”Strasbourg”,57441], ”HDF” : [”Lille”,31806], ”IDF” : [”Paris”,12012], ”NOR” :
[”Rouen”,29907], ”NAQ” : [”Bordeaux”, 84036], ”OCC” : [”Toulouse”, 72724], ”PDL” :
[”Nantes”, 32082], ”PAC” : [”Marseille”, 31400]}
```

```
notes = {['informatique', 'anglais'] : 15}  ➡  TypeError: unhashable type: 'list'
```

```
notes = {15 : 'informatique', 14 : 'geologie', 15 : 'anglais'}
                                        ➡ notes[15]  ⟵ 'anglais
```

Créer un dictionnaire

- Utilisation de {…}

```
{cle1 : valeur1, cle2 : valeur2, …}
```

```
{'informatique':15, 'geologie' : 14}
```

Dictionnaire vide `{}`

- Utilisation de dict(…)

```
dict([(cle1, valeur1), (cle2, valeur2), …])
```

```
dict([('informatique', 15), ('geologie', 14)]
```

Si les clés sont des **chaînes de caractères** on peut utiliser :

```
dict(chaine1 = valeur1, chaine2 = valeur2, …)
```

```
dict(informatique = 15, geologie = 14)
```

Accéder aux éléments

- On accède aux valeurs par les clés

```
dico[key]
```

Les dictionnaires **sont indexables** (ou subscriptables) => on peut utiliser des [ ]

Les dictionnaires **ne sont pas ordonnés** => il n'y a pas de position.

```
notes = {'informatique' : 15, 'geologie' : 14, 'anglais' : 15}
print(notes[0])                    KeyError: 0
```

Modifier un dictionnaire

- Ajouter/modifier des éléments :

```
dico[key] = valeur
```

Si la clé **n'existe pas** dans le dictionnaire : **Ajoute** la clé et la valeur associée

Si la clé **existe** dans le dictionnaire : **Modifie** la valeur associée

```
dico.update(autre_dico)
```

Met à jour le dictionnaire en intégrant le contenu d'un autre dictionnaire.

- Supprimer des éléments

```
del dico[key]
```

```
dico.pop(key)
```

Supprime la clé et renvoie la valeur associée

```
dico.popitem()
```

Supprime et renvoie la **dernière paire** (clé, valeur)

Exemple 2

```
dates = {1789 : ”Révolution française”, 1515 : ”Marignan”, 800 : ”sacre Charlemagne”}
```

```
# Accéder à un élément
print(dates[1515])
```

➡ ”Marignan”

```
# Modifier un élément
dates[800] = ”renovatio imperii”
```

➡

```
dates = {1789 : ”Révolution française”, 1515 : ”Marignan”, 800 : ”renovatio imperii”}
```

```
# Ajouter un élément
dates[-52] = ”bataille de Gergovie”
```

➡

```
dates = {1789 : ”Révolution française”, 1515 : ”Marignan”, 800 : ”renovatio imperii”, -52 : ”bataille de Gergovie”}
```

```
# Supprimer un élément
del dates[1515]
```

➡

```
dates = {1789 : ”Révolution française”, 800 : ”renovatio imperii”, -52 : ”bataille de Gergovie”}
```

Parcourir un dictionnaire

- Parcourir le dictionnaire:

```
for key in dico :
    ...
```

- Parcourir les clés et les valeurs du dictionnaire

```
for key, valeur in dico.items() :
    ...
```

- Parcourir les valeurs du dictionnaires

```
for valeur in dico.values() :
    ...
```

Conditions et dictionnaire

- Tester si une clé est définie dans le dictionnaire :

```
if key in dico :
    ...
```

- Tester si une valeur est dans le dictionnaire :

```
if valeur in dico.values() :
    ...
```

Dictionnaire en compréhension

```
dico = {cle : valeur for ... in ... }
```

```
liste = ["hello", "the", "world"]
dico = {x : len(x) for x in liste}
```

```
{'hello': 5, 'the': 3, 'world': 5}
```

Comment choisir la bonne structure de données ?

Il existe 4 **structures de données** natives dans Python :

		Données contenues	Ordonné	Mutable
Les tuples	`('janvier', 'février', 'mars')`	Tout types	☑	☒
Les listes	`[1, 3, 5, 7, 9]`	Tout types	☑	☑
Les dictionnaires	`{'informatique' : 15, 'anglais' : 14}`	Immuables (clés), Tout types (valeurs)	☒	☑
Les sets	`{12, 'hello', 'world'}`	Immuables & uniques	☒	☑

Il existe plusieurs **types de données** immuables en Python parmi lesquels :

- Nombres (`int`, `float`)
- Chaînes de caractères (`str`)
- Booléens (`True` et `False`)

Note : Pour les TDs, on n'utilisera pas les sets.

Comment choisir la bonne structure de données ?

1. Donnée ou structure de données ?

Rappel bonnes pratiques : Principe YAGNI (You Ain't Gonna Need It) : Pas de structures inutiles.

Donnée ➡ Choix du bon type : `int`, `float`, `str`, `bool`, etc.

Structure de données ⬇

2. Les données sont-elles nommées (étiquettes, etc.) ?

oui ➡ **Dictionnaire**

non ➡ **3. Les données doivent-elles pouvoir être modifiées ?**

oui ➡ **Liste**

non ➡ **Tuple**

➔ **Les listes ne sont pas une béquille sur lesquelles on se repose !**