# Fiche syntaxe Python : manipulation des types de données élémentaires

*Cette fiche résume les commandes essentielles à connaître pour manipuler les types de base de Python. Gardez-la pour les TP, mais par contre elle ne sera pas autorisée lors des DS car les commandes présentées ici sont élémentaires et doivent être connues par cœur.*

## 1 Entiers

| | | |
|---|---|---|
| opérations arithmétiques élémentaires | : | `+ - *` |
| puissance | : | `**` |
| comparaison | : | `<` `>` `<=` `>=` `==` `!=` |
| division euclidienne | : | `//` |
| reste de la division euclidienne | : | `%` |
| reste et quotient de la division euclidienne | : | `divmod` |
| convertir quelque chose en entier | : | `int` |

## 2 Flottants

- Les opérations arithmétiques élémentaires ont la même syntaxe que pour les entiers.
- On peut obtenir les fonctions mathématiques usuelles par exemple dans la bibliothèque `numpy`. Traditionnellement, on charge cette bibliothèque par `import numpy as np`, on peut alors utiliser `np.cos, np.sin, np.sqrt, np.exp, np.pi, np.sinh, np.cosh,...`
- Conversion en flottants : `float`.
- Le nombre $\pi$ : `np.pi`.

## 3 Chaînes de caractères

| | | |
|---|---|---|
| créer une chaîne | : | `maChaine = 'bonjour'` ou `maChaine= "bonjour"` |
| chaîne vide | : | `""` |
| concaténation | : | `+` |
| lire le caractère d'indice `i` d'une chaîne `c` | : | `c[i]` |
| afficher une chaîne | : | `print` |
| longueur d'une chaîne | : | `len` |
| convertir en chaîne | : | `str` |

- Les commandes <, >, etc. sont encore valides pour les chaînes de caractères. Il s'agit alors de l'ordre alphabétique.
- Plus délicat mais utile pour afficher un message dépendant d'un ou plusieurs paramètres : la méthode `.format`. Sur un exemple :

```
temps=30
distance=200
message="Vous avez parcouru {} mètres en {} secondes, félicitations.".format(distance
    ↪ ,temps)
print(message)
```

*Remarque :* Dans cet exemple, le contenu des variables `temps` et `distance` a été automatiquement converti en chaîne de caractères.

- Contrairement aux tableaux, les chaînes ne sont pas modifiables. La commande `maChaine[i] = 'b'` ne marche pas, et il n'y a pas non plus de méthode `append`.

# 4 Booléens

| | | |
|---|---|---|
| vrai, faux ..... | : | `True, False` |
| et, ou, non ..... | : | `and, or, not` |

# 5 Tableaux

| | | |
|---|---|---|
| tableau vide ..... | : | `[]` |
| créer un tableau ..... | : | `monTableau = [1,2,3]` |
| longueur d'un tableau ..... | : | `len` |
| accéder l'élément d'indice $i$ ..... | : | `monTableau[i]` |
| modifier l'élément d'indice $i$ ..... | : | `monTableau[i] = ...` |
| ajouter un élément à la fin ..... | : | `monTableau.append(...)` |
| supprimer le dernier élément et le renvoyer ..... | : | `monTableau.pop()` |
| concaténation ..... | : | `+` |
| rajouter une liste d'éléments ..... | : | `monTableau.extend( [...] )` |
| convertir en tableau ..... | : | `list` |

Les tableaux sont *modifiables*. Les méthodes `append`, `extend` et `pop`, et la commande `monTableau[i]=...` modifient le tableau.
**N.B.** Les tableaux Python sont conçus pour pouvoir facilement ajouter ou supprimer un éléments *à la fin* du tableau.

# 6 Exemples

Tapez les commandes suivantes dans une console Python, qu'obtenez-vous ?

- `5 //2`
- `5%2`
- `float(2)`
- `int(1.5)`
- `"bon" + "jour"`
- `"bonjour"[0]`
- `"bonjour"[3]`
- `not(True)`
- `not(False)`
- `True and False`
- `True or False`
- `t =[4,8,7]`
- `t[0]`
- `t[0]=12`
- `t`
- `t.append(47)`
- `t`
- `t.pop()`
- `t`
- `t + [4,8]`