# Mémento de Python

## 1 Variables

— opérateur d'affectation `=` ; par exemple `a=2`
— changement de type : `int(a)` transforme $a$ en entier ; `float(a)` en réel, `str(a)` en chaîne de caractères
— opérateurs : `+`, `-`, `*`, `/`, `**` (puissance), `//` (division entière) et `%` (modulo)
— saisie au clavier : `input("message")` ; à convertir en entier ou en réel : `int(input("message"))` ou `float(input("message"))`
— affichage : `print(variable)`

## 2 Listes

— déclaration : `L=[1,3,2,5]` ; liste vide : `L=[]`
— longueur de la liste : `len(L)`
— accès au contenu de la $i^e$ case : `L[i]` ; les cases sont numérotées de 0 à $len(L)-1$
— ajout d'un élément à la fin de la liste : `L.append(element)`. Attention à l'erreur classique : `L=L.append(element)` ne fonctionne pas !

## 3 Chaînes de caractères

— déclaration : `chaine="une chaine est entre guillemets"`
— nombre de caractères de la chaîne : `len(chaine)`
— accès au $i^e$ caractère : `chaine[i]` ; les caractères sont numérotés de 0 à $len(chaine)-1$
— accès à la sous-chaîne allant des caractères $i$ (inclus) à $j$ (exclu) : `chaine[i:j]`
— concaténation de chaînes avec l'opérateur `+`

## 4 Tests

— Structure d'un test :

```
if condition:
    bloc d instructions
elif condition2:
    bloc d instructions
else:
    bloc d instructions
```

Les `elif` et le `else` sont optionnels.
— opérateurs de test : `==`, `<`, `<=`, `>`, `>=`, `!=` (différent)

## 5 Boucles

— boucle quand on connaît le nombre d'itérations (attention, `fin` est exclu) :

```
for var in range(debut,fin,pas):
    bloc d instructions
```

— boucle quand on ne connaît pas le nombre d'itérations :

```
while condition:
    bloc d instructions
```

## 6 Fonctions

— déclaration :

```
def fonction(var1,var2,...):
    """ commentaire sur ce que fait la fonction"""
    bloc d instructions
    return valeur
```

— une fonction se termine par `return`, pas par `print` !
— les variables `var1`,...sont des paramètres qui sont donnés à la fonction lorsqu'on l'appelle ; il ne faut pas les changer dans la fonction !
— une fonction n'est pas exécutée tant qu'on ne l'appelle pas avec l'instruction `fonction(1,2,...)` en donnant des valeurs aux paramètres
— toute variable définie dans la fonction est *locale*, elle est détruite une fois le `return` atteint, donc sa valeur n'est pas utilisable en-dehors de la fonction

## 7 Fonctions mathématiques

— importer la bibliothèque avec `import math`
— utiliser les fonctions : `math.sqrt`, `math.cos`, `math.sin`, `math.tan`, `math.log` ($ln$), `math.log10` ($log$ en base 10),...