Affectation, affichage

- `a, b = 10, 20`
  `print("la somme a + b = ", a + b)`
- Formatage des réels
  `x, y = 2.7172, 3.14159`
  `print("e =", x, "pi =", round(y, 2))`

Opérations, fonctions

- `a / b` le quotient de la division réelle
  `a // b` le quotient de la division euclidienne
  `a % b` le reste de la division euclidienne
  `a**2` fonction puissance
  `a += 1` raccourci pour `a = a + 1`
  On a aussi : `-=` `*=` `/=`
- Conversions
  `int()` troncature à l'unité (résultat `int`)
  `round(x, n)` arrondi de `x` à `n` chiffres après la virgule
- Pour utiliser des fonctions mathématiques :
  `import numpy as np`
  fonctions usuelles : `np.sin`, `np.sqrt`, ...
  `np.log`, `np.log10` logarithmes (de base e, de base 10)
  constantes: `np.pi`, `np.e`

Listes

- `L = [7,5,8,3]`
  `len(L)` ↪ la longueur (=4)
  `L[0]` ↪ le premier élément (=7)
  `L[1:3]` ↪ renvoie [5,8]
  `L[1:]` équivaut à `L[1:len(L)]`; ici: [5,8,3]
  `L[::-1]` ↪ renvoie `[3,8,5,7]`
  `L[-1]` ↪ le dernier élément (ici : 3)
  `L.append(1)` ajoute 1 à la liste : L vaut [7,5,8,3,1]
  `L = L + [3,4,5]` ↪ L vaut [7,5,8,3,1,3,4,5]
  `L.sort()` trie la liste `L` ("sur place")
  `L.sort(reverse=True)` trie dans l'ordre descendant
- Construction :
  `L = [ ]` liste vide (initialisation)
  `L = [1] * 5` ↪ renvoie `[1,1,1,1,1]`
  `[n**2 for n in range(3,7)]` ↪ renvoie [9,16,25,36]
- Opérations spéciales
  `list(map(f,L))` applique la fonction `f` à chaque élément de la liste `L`
  `sum(L)` somme d'une liste
  `A = L` crée un nouveau nom pour `L`,
  non pas une nouvelle liste !
  `A = L[:]` crée une nouvelle liste nommée `A`
  (une copie séparée de `L`)

Chaînes de caractères

- `c = "ceci est une chaine"`
  `c[0:6]` ↪ "ceci e"
  Boucle sur les lettres :

```
for ltr in chaine:
    print(ltr)
```

  rajouter une lettre : `chaine += lettre`
  `ord("a")` ↪ 97 (code ASCII)
  `chr(97)` ↪ `"a"`
  `c.split(" ")` ↪ la liste des mots d'une chaine `c`
  `" ".join(L)` ↪ réunion des mots dans la liste `L`

Programmation

Attention au décalage et aux deux-points ":"

- ```
  for n in range(3, 15, 4):
      print(n, n**2)
  print("ceci n'est pas dans la boucle")
  ```
- ```
  n = 3
  while n < 10:
       print(n, n**2)
       n += 1
  ```
- tests `if`, exemple fondamental :

```
if a >= b:
    print ("a est plus grand")
else:
    print ("b est plus grand")
```

  `a == b` test d'égalité
  `a != b` test de non égalité
  `x in y` : teste si `x` appartient à (une liste) `y`
  `and`, `or`, `not` : opérations logiques
- Fonctions

```
def carre_plus(n):
    b = n*n + 2
    return b
    print("Ceci ne s'affichera pas!")
```

Graphisme

Charger les modules nécessaires :

```
import numpy as np
import matplotlib.pyplot as plt
```

- Une suite de valeurs :
  ```
  L = [1,2,5,-1,4]
  plt.plot(L)
  ```
- Une suite de points dans le plan
  ```
  x = [3, 4, -1, 2]
  y = [1, 2, 3, 2]
  plt.plot(x, y)
  ```
- Couleur
  `plt.plot(x, y, '.', color='red')`
  Epaisseur du trait
  `plt.plot(x, y, linewidth=2.0)`
  Points séparés
  `plt.plot(x, y, '.', markersize=3)`
- Graphe d'une fonction :
  ```
  x = np.linspace(0, 2, 400)
  y = np.sin(2*np.pi*x)
  plt.plot(x, y)
  ```
- Orthonormalisation `plt.axis("equal")`
- Superposition et séparation des graphiques
  ```
  plt.plot(x, y, linewidth=2.5)
  plt.plot(x, np.sin(2*np.pi*(x+0.1)), color='r')
  plt.figure()
  plt.plot(x, np.cos(2*np.pi*(x+0.1)), color='g')
  ```