

# PYTHON

## AGRO-VETO

### 2024

#### Listes

```

l.pop(k) -- Renvoie le  $k$ ème élément de la liste L et l'enlève de L
[a]*n ----- Crée une liste avec  $n$  fois l'élément a
L.append(a) Ajoute l'élément a à la fin de la liste L
L1 + L2 ---- Concatène les deux listes L1 et L2
len(L) ---- Renvoie le nombre d'éléments de la liste L

```

#### Numpy

```

import numpy as np
np.array() ----- Transforme une liste en matrice numpy
np.linspace(a,b,n) ----- Crée une matrice ligne de  $n$  valeurs
                                         uniformément réparties entre  $a$  et  $b$  (inclus)
np.zeros([n,m]) ----- Crée la matrice nulle de taille  $n \times m$ 
np.eye(n) ----- Crée la matrice identité de taille  $n$ 
np.diag(L) ----- Crée la matrice diagonale dont les termes
                                         diagonaux sont les éléments de la liste L
np.transpose(M) ----- Renvoie la transposée de M
np.dot(M,P) ----- Renvoie le produit matriciel  $MP$ 
np.sum(M) ----- Renvoie la somme de tous les éléments de M
np.prod(M) ----- Renvoie le produit de tous les éléments de M
np.max(M) ----- Renvoie le plus grand élément de M
np.min(M) ----- Renvoie le plus petit élément de M
np.shape(M) ----- Renvoie dans un couple le format de la matrice M
np.size(M) ----- Renvoie le nombre d'éléments de M
np.arange(a,b,eps) ----- Renvoie la liste des flottants de  $a$  à  $b$  de pas constant  $eps$ 

```

#### Logique

```

a == b ----- Teste l'égalité «  $a = b$  »
a != b ----- Teste «  $a \neq b$  »
a < b ----- Teste «  $a < b$  »
a <= b ----- Teste «  $a \leq b$  »
a > b ----- Teste «  $a > b$  »
a => b ----- Teste «  $a \geq b$  »
not A ----- Renvoie la négation de A
A and B --- Renvoie «  $A$  et  $B$  »
A or B ----- Renvoie «  $A$  ou  $B$  »
True ----- Constante booléenne « Vrai »
False ----- Constante booléenne « Faux »

```

Cette liste est non exhaustive. Les candidats sont libres d'utiliser les commandes de leur choix.

#### Matplotlib.pyplot

```

import matplotlib.pyplot as plt
plt.plot(X,Y, '+r') ----- Génère la courbe des points définis par les listes X et Y (abscisses et ordonnées) avec les options :
                                         • symbole : '·' point, 'o' rond, 'h' hexagone, '+' plus, 'x' croix, '*' étoile, ...
                                         • ligne : '-' trait plein, '--' pointillé, '-.' alterné, ...
                                         • couleur : 'b', bleu, 'r', rouge, 'g', vert, 'c', cyan, 'm', magenta, 'k' noir, ...
plt.bar(X,Y) ----- Génère l'histogramme des points définis par les listes X et Y (abscisses et ordonnées)
plt.axis('equal') ----- Rend le repère orthonomé
plt.xlim(xmin,xmax) ----- Fixe les bornes de l'axe des abscisses
plt.ylim( ymin,ymax) ----- Fixe les bornes de l'axe des ordonnées
plt.show() ----- Affiche le graphique

```

#### Random

```

import random as rd
rd.random() ----- Simule une réalisation d'une variable  $X \sim \mathcal{U}([0,1])$ 
rd.randint(a,b) ----- Simule une réalisation d'une variable  $X \sim \mathcal{U}([a,b])$ 
rd.gauss(0,1) ----- Simule une réalisation d'une variable  $X \sim \mathcal{N}(0,1)$ 
rd.choice(L) ----- Choisit aléatoirement un élément de la liste L

```

#### Math

```

import math as m
m.atan(x) ----- Renvoie  $\arctan(x)$ 
m.floor(x) ----- Renvoie  $\lfloor x \rfloor$ 
m.factorial(n) -- Renvoie  $n!$  si  $n \in \mathbb{N}$ 
                                         m.sqrt(x) -- Renvoie  $\sqrt{x}$  si  $x \geq 0$ 
                                         m.log(x) --- Renvoie  $\ln(x)$  si  $x > 0$ 
                                         m.exp(x) --- Renvoie  $e^x$ 

```