type (expression) Conversions

```
entier, flottant, booléen, chaîne Types de base
                        -192
   int 783
                  0
float 9.23
                  0.0
                           -1.7e-6
 bool True
                  False
   str "Un\nDeux"
                            'L\'.âme'
       retour à la ligne
                            ' échappé
                      """X\tY\tZ
             multiligne
                         \t2\\t,3"""
non modifiable,
séquence ordonnée de caractères
                            tabulation
```

```
Types Conteneurs

    séquences ordonnées, accès index rapide, valeurs répétables

                              ["x", 11, 8.9]
     list
              [1, 5, 9]
                                                     ["mot"]
                                                                       []
               (1, 5, 9)
                               11, "y", 7.4
                                                     ("mot",)
   tuple
                                                                       ()
non modifiable
                          expression juste avec des virgules
      *str en tant que séquence ordonnée de caractères
 ■ sans ordre a priori, clé unique, accès par clé rapide ; clés = types de base ou tuples
              {"clé": "valeur"}
     dict
                                                                       {}
dictionnaire couples clé/valeur {1: "un", 3: "trois", 2: "deux", 3.14: "\pi"}
 ensemble
              {"clé1", "clé2"}
                                            {1,9,3,0}
                                                                  set()
```

```
pour noms de variables, Identificateurs fonctions, modules, classes...

a..zA..Z_ suivi de a..zA..Z_0..9

accents possibles mais à éviter

mots clés du langage interdits

distinction casse min/MAJ

a toto x7 y_max BigOne

by and
```

```
on peut spécifier la base du nombre entier en 2<sup>nd</sup> paramètre
 int("15")
 int (15.56)
                  troncature de la partie décimale (round (15.56) pour entier arrondi)
 float ("-11.24e8")
 str (78.3)
                  et pour avoir la représentation littérale ----- repr ("Texte")
           voir au verso le formatage de chaînes, qui permet un contrôle fin
bool → utiliser des comparateurs (avec ==, !=, <, >, ...), résultat logique booléen
                        utilise chaque élément de
                                              _____['a','b','c']
 list("abc") __
                        la séquence en paramètre
 dict([(3, "trois"), (1, "un")])-
                                                   > {1:'un',3:'trois'}
                             utilise chaque élément de
 set (["un", "deux"])—
                                                        → { 'un', 'deux' }
                             la séquence en paramètre
 ":".join(['toto','12','pswd'])—
                                                   → 'toto:12:pswd'
chaîne de jointure
                       séquence de chaînes
 "des mots espacés".split()——→['des','mots','espacés']
 "1,4,8,2".split(",")
```

```
Affectation de variables

x = 1.2+8+sin(0)

valeur ou expression de calcul

nom de variable (identificateur)

y, z, r = 9.2, -7.6, "bad"

noms de conteneur de plusieurs

variables valeurs (ici un tuple)

x+=3 incrémentation

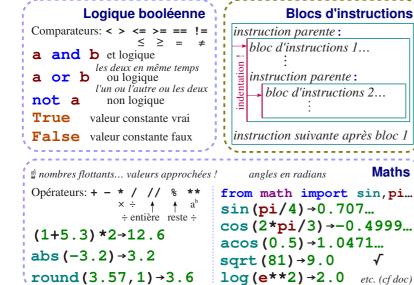
décrémentation

décrémentation

variables valeur constante « non défini »
```

```
pour les listes, tuples, chaînes de caractères,... Indexation des séquences
  index négatif
                 -6
                                 -4
                                              -3
                                                       -2
                                                               -1
                                                                          len(lst)—
   index positif
                         1
                                   2
                                              3
                                                       4
                                                                5
                                                                         accès individuel aux éléments par [index]
                                "abc"
                                                       42,
                                            3.14,
                                                              1968]
                                                                          lst[1] \rightarrow 67
                                                                                                      1st [0] \rightarrow 11 le premier
tranche positive 0
                                                           5
                                                                          1st[-2] \rightarrow 42
                                                                                                      1st [-1] → 1968 le dernier
tranche négative -6 -5
                                                   -2
                                                         -1
                            -4
                                        -¦3
                                                                         accès à des sous-séquences par [tranche début:tranche fin:pas]
        lst[:-1] \rightarrow [11, 67, "abc", 3.14, 42]
                                                                          lst[1:3] \rightarrow [67, "abc"]
        lst[1:-1] \rightarrow [67, "abc", 3.14, 42]
                                                                          lst[-3:-1] \rightarrow [3.14,42]
        lst[::2] \rightarrow [11, "abc", 42]
                                                                          lst[:3] \rightarrow [11, 67, "abc"]
        lst[:] \rightarrow [11, 67, "abc", 3.14, 42, 1968]
                                                                          lst[4:] \rightarrow [42, 1968]
                                 Indication de tranche manquante \rightarrow à partir du début / jusqu'à la fin.
    Sur les séquences modifiables, utilisable pour suppression del lst[3:5] et modification par affectation lst[1:4]=['hop', 9]
```

chaîne de séparation



uniquement si une condition est vraie **if** expression logique: → bloc d'instructions combinable avec des sinon si, sinon si... et un seul sinon final, exemple: if x==42: # bloc si expression logique x==42 vraie print("vérité vraie") elif x>0: # bloc sinon si expression logique x>0 vraie print("positivons") elif bTermine: # bloc sinon si variable booléenne bTermine vraie print("ah, c'est fini") else: # bloc sinon des autres cas restants

print("ça veut pas")

bloc d'instructions exécuté Instruction conditionnelle

```
bloc d'instructions exécuté Instruction boucle conditionnelle
                                                                      bloc d'instructions exécuté pour Instruction boucle itérative
tant que la condition est vraie
                                                                      chaque élément d'un conteneur ou d'un itérateur
             while expression logique:
                                                                                         for variable in séquence:
                   bloc d'instructions
                                                        Contrôle de boucle !
                                                                                               bloc d'instructions
 i = 1 } initialisations avant la boucle
                                                       'break
                                                                                  Parcours des valeurs de la séquence
                                                                sortie immédiate
                                                                                  s = "Du texte"
  condition avec au moins une valeur variable (ici 1)
                                                                                                         hinitialisations avant la boucle
                                                        continue
                                                                                  cpt = 0
                                                               itération suivante
 while i <= 100:
                                                                                    variable de boucle, valeur gérée par l'instruction for
       # bloc exécuté tant que i \leq 100
                                                                                  for c in s:
                                                                                                                     Comptage du nombre
       s = s + i**2
                                                                                                    "e":
                                                                                                                     de e dans la chaîne.
       \mathbf{i} = \mathbf{i} + \mathbf{1}  afaire varier la variable
                                                                                              cpt = cpt + 1
                        de condition!
                                                                                  print("trouvé", cpt, "'e'")
 print ("somme:", s) \résultat de calcul après la boucle
                                                                        boucle sur dict/set = boucle sur séquence des clés
                                                                        utilisation des tranches pour parcourir un sous-ensemble de la séquence
                    dattention aux boucles sans fin!
                                                                        Parcours des index de la séquence
                                                                        □ changement de l'élément à la position
                                             Affichage / Saisie
                                                                        □ accès aux éléments autour de la position (avant/après)
                                                                        lst = [11, 18, 9, 12, 23, 4, 17]
                                                                        perdu = []
 éléments à afficher : valeurs littérales, variables, expressions
                                                                                                                     Bornage des valeurs
                                                                        for idx in range(len(lst)):
    Options de print:
                                                                                                                     supérieures à 15.
                                                                              val = lst[idx]
    □ sep=" " (séparateur d'éléments, défaut espace)
                                                                                                                     mémorisation des
                                                                              if val> 15:
                                                                                                                     valeurs perdues.
    □ end="\n" (fin d'affichage, défaut fin de ligne)
                                                                                    perdu.append(val)
    □ file=f (print vers fichier, défaut sortie standard)
                                                                                    lst[idx] = 15
 s = input("Directives:")
                                                                        print("modif:",lst,"-modif:",perdu)
    input retourne toujours une chaîne, la convertir vers le type
                                                                        Parcours simultané index et valeur de la séquence:
        désiré (cf encadré Conversions au recto).
                                                                        for idx, val in enumerate(lst):
                                       Opérations sur conteneurs
                                                                           très utilisé pour les Génération de séquences d'entiers
 len (c) → nb d'éléments
                                                                           boucles itératives for par défaut 0 \
                         sum (c)
min(c) max(c)
                                       Note: Pour dictionnaires et ensembles,
                                                                                               range ([début,] fin [,pas])
 sorted(c) → copie triée
                                       ces opérations travaillent sur les clés.
val in c → booléen, opérateur in de test de présence (not in d'absence)
                                                                                                                     → 0 1
                                                                           range (5)
                                                                                                                              2 3 4
enumerate (c) → itérateur sur (index,valeur)
                                                                           range (3, 8)
                                                                                                                       3
                                                                                                                               5
                                                                                                                                  6 7
Spécifique aux conteneurs de séquences (listes, tuples, chaînes):
                                                                           range (2, 12, 3)
                                                                                                                         2 5 8
reversed (c) \rightarrow itérateur inversé c*5 \rightarrow duplication c+c2 \rightarrow concaténation
                                 c.count (val) → nb d'occurences
c.index(val) → position
                                                                                range retourne un « générateur », faire une conversion
                                                                                en liste pour voir les valeurs, par exemple:
                                                                                print(list(range(4)))
🙎 modification de la liste originale
                                              Opérations sur listes
lst.append(item)
                                  ajout d'un élément à la fin
lst.extend(seq)
                                  ajout d'une séquence d'éléments à la fin
                                                                           nom de la fonction (identificateur) Définition de fonction
lst.insert(idx, val)
                                 insertion d'un élément à une position
                                                                                                paramètres nommés
lst.remove(val)
                                 suppression d'un élément à partir de sa valeur
lst.pop(idx)
                     suppression de l'élément à une position et retour de la valeur
                                                                           def nomfct(p_x,p_y,p_z):
lst.sort()
                  lst.reverse()
                                            tri / inversion de la liste sur place
                                                                                   """documentation"""
                                                                                   # bloc instructions, calcul de res, etc.
Opérations sur dictionnaires
                                       Opérations sur ensembles
                                                                                   return res ← valeur résultat de l'appel.
                                      Opérateurs:
d[clé]=valeur
                     d.clear()
                                                                                                           si pas de résultat calculé à

    | → union (caractère barre verticale)

d[clé] →valeur
                     del d[clé]
                                                                           les paramètres et toutes les
                                      & → intersection
                                                                                                           retourner: return None
                                                                           variables de ce bloc n'existent
d.update (d2)√ mise à jour/ajout
                                      - \stackrel{\wedge}{\longrightarrow} différence/diff symétrique
                                                                            que dans le bloc et pendant l'appel à la fonction (« boite noire »)
d.keys()
                   des couples
                                       < <= > >= → relations d'inclusion
d. values () vues sur les clés,
                                                                                                                  Appel de fonction
                                      s.update(s2)
                                                                                  nomfct(3, i+2, 2*i)
d.items() ∫ valeurs, couples
                                     s.add(cl\acute{e}) s.remove(cl\acute{e})
                                                                                                un argument par paramètre
d.pop(clé)
                                      s.discard(clé)
                                                                            récupération du résultat retourné (si nécessaire)
 stockage de données sur disque, et relecture
                                                              Fichiers
                                                                                                             Formatage de chaînes`
f = open("fic.txt", "w", encoding="utf8")
                                                                                                           valeurs à formater
                                                                             directives de formatage
                                                                            "modele{} {} {}".format(x,y,r) =
               nom du fichier
variable
                               mode d'ouverture
                                                       encodage des
                                                                            " { sélection : formatage ! conversion } "
fichier pour
               sur le disque
                               □ 'r' lecture (read)
                                                       caractères pour les
                                                                             Sélection :
les opérations
              (+chemin...)
                               □ 'w' écriture (write)
                                                      fichiers textes:
                                                                                                   "{:+2.3f}".format(45.7273)
                               □ 'a' ajout (append)... utf8
                                                              ascii
                                                                                                   → '+45.727 '
cf fonctions des modules os et os.path
                                                       latin1
                                                                                                  "{1:>10s}".format(8, "toto")
                                                                              0.nom
                                                                                                              toto'
                                  chaîne vide si fin de fichier
    en écriture
                                                            en lecture
                                                                                                   "{!r}".format("L'ame")
                                                                              0[2]
                                   = f.read(4)<sub>si nb de caractères</sub>
f.write("coucou")
                                                                           □ Formatage :
                                                                                                   →'"L\'ame"'
                                                         pas précisé, lit tout
                                       lecture ligne

    fichier texte → lecture / écriture

                                                                           car-rempl. alignement signe larg.mini.précision~larg.max type
 de chaînes uniquement, convertir
                                       suivante
                                                         le fichier
 de/vers le type désiré
                                 s = f.readline()
                                                                                                 o au début pour remplissage avec des 0
                                                                                     + - espacé
 f.close() de ne pas oublier de refermer le fichier après son utilisation!
                                                                           entiers: b binaire, c caractère, d décimal (défaut), o octal, x ou X hexa.
                  Fermeture automatique Pythonesque: with open (...) as f:
                                                                           flottant: e ou E exponentielle, f ou F point fixe, g ou G approprié (défaut),
 très courant : boucle itérative de lecture des lignes d'un fichier texte :
                                                                                   % pourcentage
 for ligne in f :
                                                                           chaîne: s.
                                                                           \hfill\Box Conversion : {\tt s} (texte lisible) ou {\tt r} (représentation littérale)
     🕇 bloc de traitement de la ligne
```

Mémento numérique Python 3

```
import matplotlib.pyplot as plt → charge le module pyplot sous le nom plt

plt.figure('titre') → crée une fenêtre de tracé vide

plt.plot(LX, LY, 'o-b') → trace le graphique défini par les listes LX et LY (abscisses et ordonnées)

| couleur: 'b' (blue), 'g' (green), 'r' (red), 'c' (cyan), 'm' (magenta), 'y' (yellow), 'k' (black)
| type de ligne: '-' (trait plein), '--' (pointillé), '-.' (alterné)...
| marque: 'o' (rond), 'h' (hexagone), '+' (plus), 'r' (croix), '*' (étoile)...

plt.xlim(xmin, xmax) → fixe les bornes de l'axe x

plt.ylim(ymin, ymax) → fixe les bornes de l'axe y

plt.axis('equal') → change les limites des axes x et y pour un affichage avec des axes orthonormés (le tracé d'un cercle plt.show() → affichage de la fenêtre donne un cercle)

plt.savefig(fichier)→ sauve le tracé dans un fichier (le suffixe du nom fichier peut donner le format; par exemple, 'image.png')
```

import numpy as np -> charge le module numpy sous le nom np

```
Construction de tableaux (de type ndarray)
```

 $np.zeros(n) \rightarrow crée$ un vecteur dont les n composantes sont nulles

np.zeros ((n, m)) \rightarrow crée une matrice $n \times m$, dont les éléments sont nuls

 $np.eye(n) \rightarrow crée la matrice identité d'ordre n$

np.linspace $(a,b,n) \rightarrow$ crée un vecteur de n valeurs régulièrement espacées de a à b np.arange $(a,b,dx) \rightarrow$ crée un vecteur de

valeurs de a incluse à b exclue avec un pas dx

M.shape \rightarrow tuple donnant les dimensions de M **M.size** \rightarrow le nombre d'éléments de M **M.ndim** \rightarrow le nombre de dimensions de M

M. sum () \rightarrow somme de tous les éléments de M

M.min() \rightarrow plus petit élément de M **M.max()** \rightarrow plus grand élément de M

argument **axis** optionnel : $0 \rightarrow \text{lignes}$, $1 \rightarrow \text{colonnes}$:

M. sum (0) \rightarrow somme des lignes

 $M.min(0) \rightarrow plus petits éléments, sur chaque colonne$

M. max (1) → plus grands éléments, sur chaque ligne

import numpy.linalg as la

la.det (M) \rightarrow déterminant de la matrice carrée M

la.inv (M) \rightarrow inverse de M

la.eig (M) \rightarrow valeurs propres de M

la.matrix_rank (M) \rightarrow rang de M

la.matrix_power $(M, n) \rightarrow M^n$ (n entier)

la.solve (A, B) \rightarrow renvoie X tel que A X = B

import scipy.integrate as spi

spi.odeint(F,Y0,LT)

 \rightarrow renvoie une solution numérique du problème de Cauchy $\mathbf{Y}'(t) = \mathbf{F}(\mathbf{Y}(t),t)$, où \mathbf{Y} est un vecteur d'ordre n, avec la condition initiale $\mathbf{Y}(t_0) = \text{YO}$, pour les valeurs de t dans la liste LT de longueur k commençant par t_0 , sous forme d'une matrice $n \times k$

spi.quad (f, a, b) \rightarrow renvoie une évaluation numérique de l'intégrale : $\int_{a}^{b} f(t)dt$

Conversion ndarray <-> liste

 $V = np.array([1, 2, 3]) \rightarrow V : vecteur(1 2 3)$

 $L = V.tolist() \rightarrow L: liste [1, 2, 3]$

M = np.array([[1,2],[3,4]]) $\rightarrow M$: matrice $\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}$

L = $M.tolist() \rightarrow L : liste [[1, 2], [3,4]]$

Extraction d'une partie de matrice

M[i], $M[i,:] \rightarrow ligne de M d'index i$

M[:, \dot{j}] \rightarrow colonne de M d'index j

 $M[i:i+h, j:j+1] \rightarrow sous-matrice h \times l$

Copier un tableau avec la méthode copy :

M2 = M1.copy()

```
^{\prime} M1+M2, M1*M2, M**2 \rightarrow opérations « terme-à-terme »
```

 $\mathbf{C}^{\bigstar}\mathbf{M} \longrightarrow \text{multiplication de la matrice } M \text{ par le scalaire } c$

M+c \rightarrow matrice obtenue en ajoutant le scalaire c à chaque terme de M

V1 . dot (V2) → renvoie le produit scalaire de deux vecteurs

np.dot (V1, V2)

M.dot (V)
np.dot (M, V) → renvoie le produit d'une matrice par un vecteur

M1.dot (M2)
np.dot (M1, M2) → renvoie le produit de deux matrices

M.transpose() \rightarrow renvoie une copie de M transposée np.transpose(M) \rightarrow (ne modifie pas M)

M.trace()

m.trace() np.trace(M) \rightarrow renvoie la trace de M

Fonctions mathématiques usuelles

np.exp, np.sin, np.cos, np.sqrt etc.

→ fonctions qui s'appliquent sur des réels ou des complexes, mais aussi sur des vecteurs et des matrices (s'appliquent à chaque terme), qui sont optimisées en durée de calcul.

<u>Rappel</u>: ce mémento est fourni à titre indicatif. Il ne faut le considérer ni comme exhaustif, ni comme exclusif, ni comme un minimum à connaître absolument (l'examinateur n'attend pas du candidat qu'il connaisse parfaitement toutes ces fonctions et ces commandes).