pour noms de variables,

fonctions, modules, classes...

Mémento Python 3

Version originale sous licence CC4 https://perso.limsi.fr/pointal/python:memento

```
Types de base
entier, flottant, booléen, chaîne
   int 783 0 -192
float 9.23 0.0 -1.7<u>e-6</u>
 bool True False
   str "Un\nDeux"
                                 d immutables
        retour à la ligne
```

Identificateurs

```
int("15") \rightarrow 15
float ("3.14") \rightarrow 3.14
int (15.56) \rightarrow 15
```

type (expression)

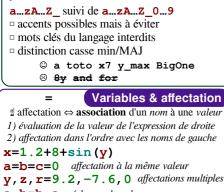
troncature de la partie décimale

Conversions

Séquences d'entiers range ([début,] fin [,pas])

₫ début défaut 0, fin non compris dans la séquence, pas signé et défaut 1 range (5) \rightarrow 0 1 2 3 4 range $(2, 12, 3) \rightarrow 25811$ range (3, 8) \rightarrow 3 4 5 6 7 range (20, 5, -5) \rightarrow 20 15 10

🖠 range fournit une séquence immutable d'entiers construits au besoin



```
pour les listes, chaînes de caractères, tuples ...
               -4 -3 -2
index négatif
           -5
                                 -1
                      2
                             3
           0
                 1
index positif
     lst=[10, 20, 30, 40, 50]
```

Sur les séquences modifiables (list), suppression avec del lst[3] et modification par affectation 1st [4] = 25

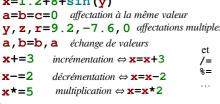
Indexation conteneurs séquences

Accès individuel aux **éléments** par **1st** [index]

 $lst[0] \rightarrow 10$ 1st[1]→20

Nombre d'éléments index à partir de 0 len (1st) $\rightarrow 5$ (de 0 à 4 ici)

1st.append (val) ajout d'un élément à la fin





tant que la condition est vraie **while** condition logique: → bloc d'instructions

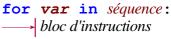
bloc d'instructions exécuté



= 0 } initialisations **avant** la boucle i = 1 condition avec au moins une valeur variable (ici i) while i <= 100: Algo: i=100

s = s + i**2 i = i + 1 print("somme:",s) 🖠 faire varier la variable de condition !







Parcours des valeurs d'un conteneur

s = "Du texte" | initialisations avant la boucle cpt = 0

variable de boucle, affectation gérée par l'instruction for for c in s: if c == "e": Algo: comptage cpt = cpt + 1
print("trouvé", cpt, "'e'") du nombre de e dans la chaîne.

bonne habitude : ne pas modifier la variable de boucle

Logique booléenne

Comparateurs: < > <= >= != $(résultats\ booléens) \le \ge = \ne$

a and b et logique les deux en même temps

a or **b** ou logique l'un ou l'autre ou les deux 🖠 piège : and et or retournent la valeur de

a ou de b (selon l'évaluation au plus court). ⇒ s'assurer que **a** et **b** sont booléens.

not a non logique True False

constantes Vrai/Faux

Blocs d'instructions

instruction parente: →bloc d'instructions 1... instruction parente: bloc d'instructions 2...

instruction suivante après bloc 1

½ régler l'éditeur pour insérer 4 espaces à la place d'une tabulation d'indentation.

Imports modules/noms module truc⇔fichier truc.py from monmod import nom1, nom2 as fct →accès direct aux noms, renommage avec as import monmod →accès via monmod.nom1 ... 🖆 modules et packages cherchés dans le python path (cf. sys.path)

un bloc d'instructions exécuté, Instruction conditionnelle uniquement si sa condition est vraie

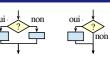
if condition logique: → bloc d'instructions

Combinable avec des sinon si, sinon si... et un seul sinon final. Seul le bloc de la

première condition trouvée vraie est exécuté

₫ avec une variable **x**:

if bool(x) == $True: \Leftrightarrow if x:$ if bool(x) ==False: ⇔ if not x:



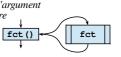
if age<=18: etat="Enfant" elif age>65: etat="Retraité' else: etat="Actif"

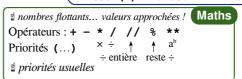
```
Définition de fonction
nom de la fonction (identificateur)
          paramètres nommés
def fct(x, y, z):
                                           fct
      """documentation"""
      # bloc instructions, calcul de res, etc.
    return res ← valeur résultat de l'appel, si rien
                        à retourner return None
```

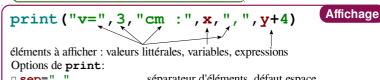
🛮 les paramètres et toutes les variables de ce bloc n'existent que dans le bloc et pendant l'appel à la fonction (penser "boîte noire")

Appel de fonction r = fct(3, "hey", a)stockage/utilisation une valeur d'argument de la valeur de retour par paramètre

de la fonction avec les parenthèses qui fait l'appel







□sep=" " séparateur d'éléments, défaut espace □ end="\n" fin d'affichage, défaut fin de ligne

Saisie s = input("Directives:") input retourne toujours une chaîne, la convertir vers le type désiré

(cf. encadré Conversions).