pour noms de variables,

not a

True

False

non logique

constantes Vrai/Faux

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
```

troncature de la partie décimale

Conversions type (expression)

```
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

```
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
        ⊗ 8y and for
                  Variables & affectation
 affectation ⇔ association d'un nom à une valeur
 1) évaluation de la valeur de l'expression de droite
 2) affectation dans l'ordre avec les noms de gauche
x=1.2+8+sin(y)
a=b=c=0 affectation à la même valeur
y, z, r=9.2, -7.6, 0 affectations multiples
a, b=b, a échange de valeurs
x+=3 incrémentation \Leftrightarrow x=x+3
                                             /=
                                             ·
응=
           d\acute{e}cr\acute{e}mentation \Leftrightarrow x=x-2
\mathbf{x} = 2
           multiplication \Leftrightarrow x=x*2
x*=5
```

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

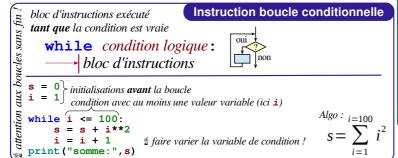
Sur les séquences modifiables (list), suppression avec del lst[3] et modification par affectation 1st [4] = 25

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

```
lst[0] \rightarrow 10
                                     1st[1]→20
```

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

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



Instruction boucle itérative bloc d'instructions exécuté **pour** chaque élément d'un conteneur ou d'un itérateur for var in séquence: fini → bloc d'instructions

Parcours des valeurs d'un conteneur s = "Du texte" | initialisations avant la boucle cpt = 0variable 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 Blocs d'instructions Comparateurs: < > <= >= != $(résultats\ booléens) \le \ge = \ne$ instruction parente: **a and b** et logique les deux en même temps →bloc d'instructions 1... **a** or **b** ou logique l'un ou l'autre ou les deux instruction parente: 🖠 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.

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

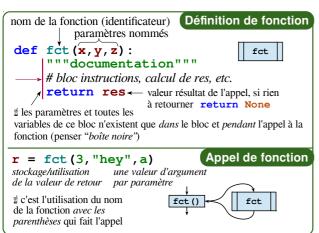
if condition logique: bloc d'instructions 2... → bloc d'instructions instruction suivante après bloc 1

½ régler l'éditeur pour insérer 4 espaces à

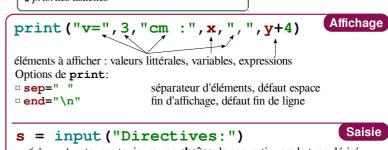
la place d'une tabulation d'indentation.

Combinable avec des sinon si, sinon si... et un seul sinon final. Seul le bloc de la première condition trouvée vraie est exécuté

if age<=18: etat="Enfant" elif age>65: etat="Retraité' ₫ avec une variable **x**: else: if bool(x) == $True: \Leftrightarrow if x:$ etat="Actif" if bool(x) ==False: ⇔ if not x:



Opérateurs : + - * / // % ** × ÷ ♠ ♠ a^b Priorités (...) ÷ entière reste ÷ d priorités usuelles



input retourne toujours une chaîne, la convertir vers le type désiré (cf. encadré Conversions).