AGRO CAMPUS OUEST

# Python 3
# –
# Structures de données

AGRO CAMPUS OUEST

# Plan

*1) Introduction*
*2) Listes*
*3) Ensembles*
*4) Dictionnaires*

# Structures de données

- Actuellement, on ne dispose que de variables « simples »
  - Permet de résoudre beaucoup de problèmes, mais …
  - Semble compliqué dans certains cas !
    - Lesquels ?
      - On dispose d'une collection de valeurs du même « genre »
      - On souhaite traiter ces valeurs ensemble
    - Exemple : situation de la permutation circulaire de 100 variables
- Les structures de données sont des outils qui permettent de manipuler des collections de valeurs
  - On parle de « conteneur » (ils contiennent des valeurs)
  - Ce sont des « types complexes »

# Brève revue des conteneurs classiques de Python

- Les listes
  - Structure itérable et *indexée*
  - Structure ordonnée
  - Taille potentiellement
- Les tuples : taille fixe, indexé
- Les ensembles (set)
  - Structure non-ordonnées
  - Structure itérable
  - Pas de redondance
- Les dictionnaires
  - Structure associative : associe une valeur à une autre
  - Généralisation des listes :
    - utilise d'un index dont le type n'est pas nécessairement un entier
    - Version « creuses »
- En fonction des besoins, il faut savoir identifier la structure de données qui facilitera l'algorithme

AGRO CAMPUS OUEST

# Plan

*1) Introduction*
*2) Listes*
*3) Ensembles*
*4) Dictionnaires*

# Les listes

- Création d'une liste
  - Création à l'aide du « constructeur »

```
liste = list()
tl=type(liste)
print(tl)
```

  - Création à l'aide des crochets

```
liste = []
tl=type(liste)
print(tl)
```

```
liste = [34, 10, 2, 33]
print( liste )
tl=type(liste)
print(tl)
```

# Les listes

- Une liste Python peut contenir des éléments de types divers

```
liste = [34, 'coucou', True, 45]
print( liste )
```

- Voire même une autre liste … (!)

```
liste = [34, 'coucou', [True, False], 45]
print( liste )
```

- Connaitre la taille de la liste : `len`

```
liste = [34, 12, True, 45]
Taille = len(liste)
print( Taille )
```

# Les listes : accéder aux éléments de la liste

- Les listes sont indexées
  - Index par des entiers commençant par 0
  - Il est possible d'accéder aux éléments de la liste avec une notation crochetée

```
Liste = [45,67,78,89]
Val = Liste[2]
print(Val) ## 78
Liste[3]=-56
Liste[5]=2
```

- Exemple : recopier le programme et indiquer les valeurs correspondants (et expliquer!)

```
Liste = [23,34,45,67,78,89]
l1 = Liste[2:4]
l2 = Liste[:4]
l3 = Liste[4:]
l4 = Liste[-1]
l5 = Liste[-4:-2]
l6 = Liste[1:-2]
```

# Les listes : quelques fonctions usuelles

- Manipulation des éléments de la liste
  - Ajout d'éléments en fin de liste : `append`
  - Suppression d'éléments : `remove`
  - Construction d'une sous-liste : utilisation des notations crochetées
  - Ordonnées : `sort()`
  - Vider la liste : `clear`
  - Autres manips : ordonner une liste, concaténer des listes, copier, etc.
- Exemple

```
Liste = [34,23,45]
Liste.sort()
Liste.append( 'coucou' )
Liste.append( 34 )
Liste.remove( 34 )
Liste.clear()

help( list )
```

# Les listes : parcours d'une liste

- Parcours d'une liste par ses indices

```
Liste = [45,67,78,89]
for i in range( len(Liste) ) :
    Val = Liste[i]
    print( Val )
```

- Itération sur les éléments de la liste

```
Liste = [45,67,78,89]
for elem in Liste :
    print( elem )
```

# Définition intentionnelle des listes

- Définition « intentionnelle »
  - Syntaxe assez spécifique
  - Ecriture très rapide d'une liste construite

```
#liste des carres
Liste = [i*i for i in range(45)]

#Mettre une liste de textes en capitales
Liste = ['tot', 'fOO', 'Bar']
ListeCap = [ el.upper() for el in Liste ]
```

# Variance des listes : les tuples

- Un tuple est un conteneur
  - Indexé (utilisation de la notation crochetée)
  - De taille fixe
  - Mais … on ne peut pas changer les valeurs d'un élément

```
T = (23,'coucou',45)
print( T[1] )
T[1] = 45
```

# Plan

*1) Introduction*
*2) Listes*
*3) Ensembles*
*4) Dictionnaires*

# Les ensembles

- Conteneur basé sur la notion mathématique d'un ensemble
  - Pas de doublons (il gère tout seul)
  - Non ordonnés
  - pas indexé (car par ordonné)
- Fonctionnalités
  - Itérable
  - Union, différence, intersection
  - Tests de sous-ensembles

# Les ensembles : exemples

```
ens = set()
ens.add( 45 )
ens.add( 'coucou' )
ens.add( 45 )

print( len(ens) )

for Elem in ens :
    print(elem)

ens.remove( 34 )
```

```
Liste = [45,56,34,45,78,45,34]

ens = set( Liste )

Liste = list(ens)

print(Liste)
```

AGRO CAMPUS OUEST

# Plan

*1) Introduction*
*2) Listes*
*3) Ensembles*
*4) Dictionnaires*

# Les dictionnaires

- Conteneur associatif (aussi appelé « map »)
  - Associe une clé à une valeur
  - Les valeurs sont accédées par les clés
  - Les valeurs sont modifiables
  - Itérable
- Un premier exemple de création d'un dictionnaire

```
D = dict()
D = {}

D['coucou'] = 45
print(D)
```

# Les dictionnaires : exemples

- Création de dictionnaire
  - Création à partir d'une dictionnaire vide en le remplissant

```
Couleur = {}
Couleur[1] = "Rouge"
Couleur[2] = "Vert"
Couleur[3] = "Bleu"
```

  - Création initiale du dictionnaire

```
Couleur = {1: "Rouge", 2: "Vert", 3: "Bleu"}
```

# Dictionnaire : exemple d'histogramme

```
texte = "Les dictionnaires constituent un outil très élégant
pour construire des histogrammes."
caracteres = {}
for c in texte:
    caracteres[c] = caracteres.get(c, 0) + 1

print(caracteres)
```

# Les dictionnaires : les parcours

- Trois types de parcours
  - Par valeur
  - Par clés
  - Par couple clé/valeur
    - Items() : convertis le dictionnair en une liste de tuples

```
D = {"a" : 3, "z" : 7, "b" : 2}
for val in D.values():
    print(val)

for k in D.keys():
    print(k)

for k,val in D.items():
    Print( str(k)+ ` : ` +str(val) )
```

# Les dictionnaires

- Fonctionnalités
  - Suppression d'un élément `del tel['sape']`
  - Tester l'existence d'un élément de clé

```
if val in D:
    Print( D[val] )
```

  - Vider un dictionnaire : `clear`

# Combinaison des structures de données

- Il est possible de combiner les structures de données à volonté
  - Liste de listes …
    - Pas de contrainte sur les tailles des listes
  - Dictionnaire de listes
  - Liste de dictionnaires d'ensembles …
- Attention
  - les conteneurs pouvant mélanger les types, il faut les manipuler de manière rigoureuse