Dauphine | PSL
UNIVERSITÉ PARIS

MIDO
MATHEMATIQUES ET INFORMATIQUE
DE LA DECISION ET DES ORGANISATIONS

Exemples de cartes mentales Algorithme et Programmation L1

Vous trouverez dans ce document des exemples de cartes mentales résumant certaines notions importantes du cours de Algorithme et Programmation 1

① Déclaration d'un tableau `T` en pseudo-code dont les éléments sont de type **`TypeElement`** (ex. `NOMBRE, BOOLEEN, CHAINE`)

② Il faut d'abord créer le tableau
`T` ← **`CRÉER_TABLEAU`**`(n)`
avec `n` un nombre entier

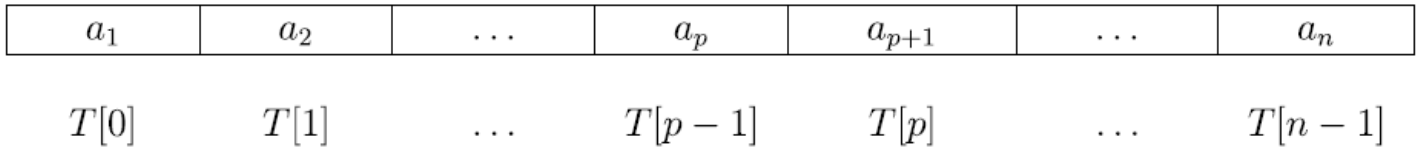
crée un tableau vide de `n` cases

Tableau
`T TABLEAU de TypeElement`

position = indice + 1

Pour obtenir la taille (i.e. le nombre de cases) :
`TAILLE(T)`

Eléments du tableau indicé de `0` à `(n-1)`



③ Il faut remplir le tableau :

```
i ← 0
TANT_QUE i < n FAIRE
  DEBUT
     LIRE T[i]
     i ← i + 1
FIN
```

ou

```
POUR i ALLANT DE 0 A n
     DEBUT
       LIRE T[i]
     FIN
```

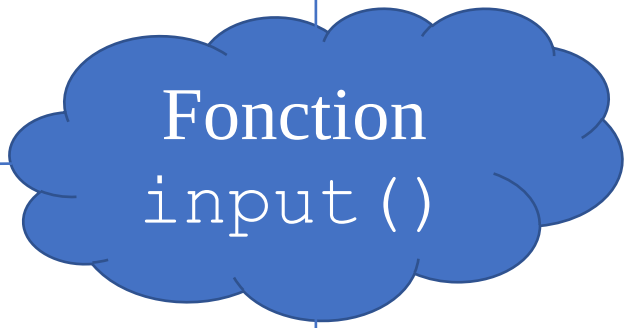
Faire une boucle pour afficher le tableau :

```
i ← 0
TANT_QUE i < n FAIRE
  DEBUT
     ECRIRE T[i]
     i ← i + 1
  FIN
ou
POUR i ALLANT DE 0 A n
    DEBUT
      ECRIRE T[i]
    FIN
```

renvoie une chaîne de caractères contenant tout ce que l'utilisateur a saisi au clavier jusqu'à la touche *Enter*

Il faut récupérer la valeur renvoyée :

```
ch = input() # ch est de type str
```



Conversion de la valeur renvoyée :

```
int(input())
float(input())
```

peut prendre une chaine de caractères en paramètre qui correspond à un message à afficher à l'utilisateur

```
ch = input(messageAfficher)
```

Accès à un élément : `L[i]` avec i ∈ **[**`0,len(L)-1`**]** ou **[**`-len(L), -1`**]**

indices négatifs	$-len(L)$	$-len(L)+1$		-2	-1
`L`	élément 1	élément 2	...	élément $n-1$	élément n
indices positifs	0	1		$len(L)-2$	$len(L)-1$

Valeur délimitée par **[ ]**

Type modifiable : contrairement au type `str` !
possibilité de faire `L[i]` = *valeur ou expression*

Liste vide : `L=[]`

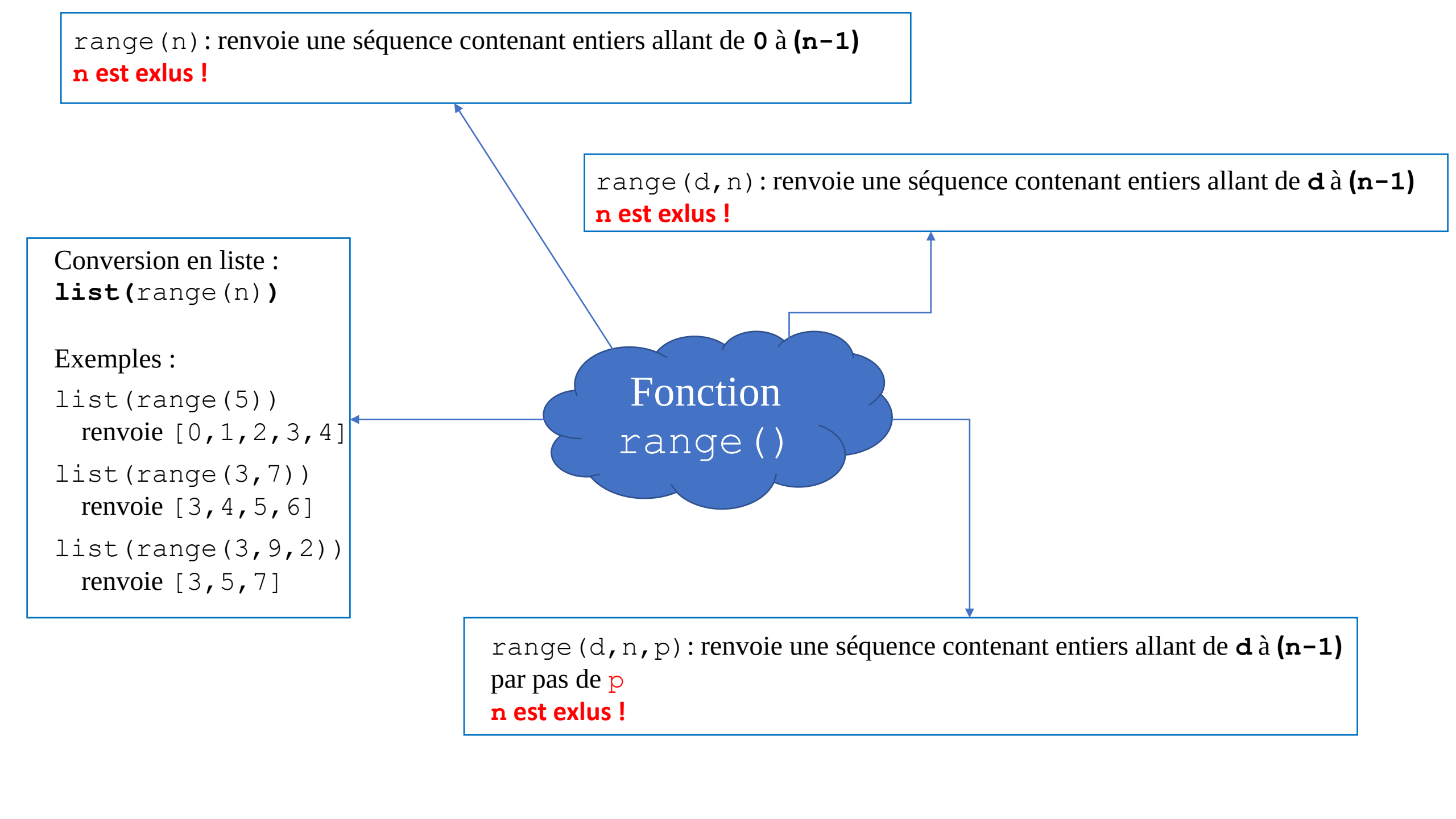
`L` de type `list`

Méthodes <u>renvoyant une valeur</u> :

- `L.pop()` : renvoie la valeur du dernier élément et le supprime de L (**L est modifiée**)
  `L.pop(i)` supprime l'élément d'indice `i`
  Attention, erreur si liste vide !
- `L.count(element)` : renvoie le nombre d'occurences de l'élément dans L
- `L.index(element)` : renvoie l'indice de la 1ere occurrence de l'élément dans L
  Attention, erreur si élément inexistant !

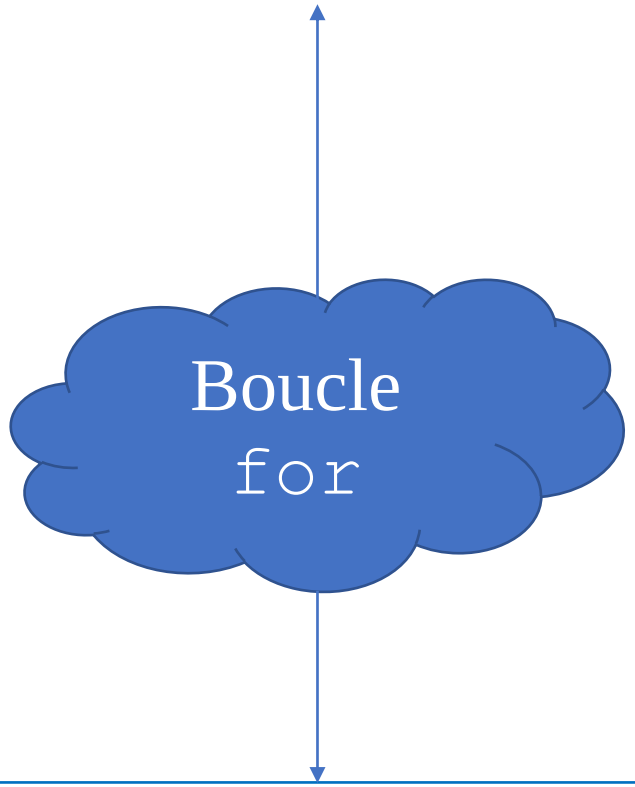
Méthodes <u>ne renvoyant rien</u> mais modifiant la liste :

- `L.append(`*`elementAAjouterALaFin`*`)`
- `L.remove(`*`elementASupprimer`*`)` : supprime la 1ere occurrence
  Attention : si élément inexistant erreur !!
- `L.insert(`*`indice, elementAAjouter`*`)`
- `L.reverse()` : inverse l'ordre des éléments
- `L.sort()` : trie les éléments
- `L.extend(L2)` : ajoute les éléments de `L2` à la fin de `L`


range(n) : renvoie une séquence contenant entiers allant de 0 à (n-1)
n est exlus !
range(d,n) : renvoie une séquence contenant entiers allant de d à (n-1)
n est exlus !
Conversion en liste :
list(range(n))
Exemples :
list(range(5))
renvoie [0,1,2,3,4]
list(range(3,7))
renvoie [3,4,5,6]
list(range(3,9,2))
renvoie [3,5,7]
Fonction
range()
range(d,n,p) : renvoie une séquence contenant entiers allant de d à (n-1)
par pas de p
n est exlus !

Avec `S` de type `str` ou `list`

```
for i in range(len(S)) : # pour itérer sur les indices
    blocInstructionsDuFor
```



Avec `S` de type `str` ou `list`

```
for i in S : # pour itérer sur les éléments de S
    blocInstructionsDuFor
```

Pour connaître le répertoire courant :

```
from os import getcwd()
getcwd()
```

Pour changer de répertoire courant :

```
from os import chdir
chdir("cheminAbsoluOuRelatif")
```

① Ouverture du fichier

```
f = open(nomFichier,modeOuverture)
```

`f` : descripteur de fichier (adresse en mémoire)

`nomFichier str` contenant le chemin absolu ou relatif ou juste le nom du fichier (s'il est dans le répertoire courant)

`modeOuverture` : `"r"` lecture, `"w"` écriture (écrase le fichier si existant), ou `"a"` (ajout à la fin, qui doit exister)

Ecrire dans un fichier : **: ouvert avec `"w"` ou `"a"`**

```
f.write(1ChaineDeCaractères)
f.writelines(listeDeChaines)
```

Mettre des `'\n'` dans les chaines si on veut des sauts de ligne dans le fichier

Lire dans un fichier **: ouvert avec `"r"`**

- `s = f.read()` avec `s : str` qui contient tout le contenu du fichier
- `s = f.readline()` avec `s : str` qui contient 1 ligne (courante) du fichier – vide si on est à la fin du fichier

  Il faut une boucle :

```
s = f.readline()
while s !="" :
        s = f.readline()
  …
```

- `L = f.readlines()` avec `L : list` dont chaque élément est une ligne du fichier

② Fermer le fichier : sinon rien n'est copié sur le disque!

```
f.close()
```