Chapitre 1 : Parcours d'une structure séquentielle

1. Types et opérations

1.1. Types simples et opérations

Il s'agit essentiellement de types numériques :

- Les entiers (**int**)
- Les flottants (**float**)
- Les complexes (**complex**)
- Les booléens (**bool**) : un booléen peut être **True** ou **False**

Les tableaux ci-dessous résument les différents opérateurs sur les objets simples.

Opérations

a+b	somme des deux nombres (int,float,complex)
a - b	différence des deux nombres (int,float,complex)
a*b	produit des deux nombres (int,float,complex)
a/b	quotient de deux flottants
a//b	quotient de la division euclidienne d'entiers
a%b	reste de la division euclidienne d'entiers
divmod(a,b)	quotient et reste de la division euclidienne
a**b pow(a,b)	a élevé à la puissance b
abs(a)	valeur absolue ou module de a

Comparaisons et opérations logiques

a==b	test d'égalité, renvoie un booléen
a !=b	test de différence, renvoie un booléen
a>b, a<b, a>=b, a<=b	tous ces tests de comparaison, respectivement $a > b$, $a < b$, $a \geqslant b$, $a \leqslant b$ renvoient un booléen
P and Q	conjonction de deux booléens
P or Q	disjonction de deux booléens
not P	négation d'un booléen

Conversions

int(a)	convertit en entier un flottant ou une chaîne
float(a)	convertit en flottant un entier ou une chaîne
complex(a,b)	convertit en complexe un couple de flottants
int(car,b)	donne la valeur de l'entier donné par son écriture car (de type chaîne) en base b
bin(a)	renvoie l'écriture en binaire de l'entier a
oct(a)	renvoie l'écriture en octal de l'entier a
hex(a)	renvoie l'écriture en hexadecimal de l'entier a

1.2. Types composés

Il s'agit de collections d'objets. On parle aussi d'objet séquentiel (ou structure séquentielle).

- Un objet composé est dit *itérable* si et seulement si on peut déterminer si un élément appartient ou non à cet objet composé.
  Exemple : on peut effectuer un test d'appartenance de l'élément **a** dans l'objet composé **A** avec la commande **>>> a in A**
- Un objet composé est dit *indexable* si et seulement si les éléments qui le composent sont indexés, c'est-à-dire possède un rang (position).
  Exemple : la commande **>>> A[k]** renvoie l'élément de rang k de l'objet **A**
- Un objet composé est dit *mutable* si et seulement si il est possible de modifier les éléments qui le composent.

Parmi les types composés, nous distinguerons :

- Les chaînes de caractère (**str**) : une chaîne de caractère est une suite de caractères quelconques encadrée par une paire de simple ou double guillemets. Les chaînes de caractères sont itérables, indexables mais non-mutables.
- Les listes (**list**) : une liste est une suite d'éléments quelconques (pas forcément de même type), séparés par des virgules, encadrée par des crochets. Les listes sont itérables, indexables et mutables.
  Exemple : **['Pierre','Dupont',156034367865410,1956]** est une liste contenant les prénom, nom, numéro de sécurité social et année de naissance d'une personne.
- Les t-uplets (**tuple**) : un t-uplet est une suite d'éléments quelconques (pas forcément de même type), séparés par des virgules, encadrée par des parenthèses. Les t-uplets sont itérables, indexables mais non-mutables.
- Les ensembles (**set**) : un ensemble est une collection d'objets séparés par des virgules, encadrée par des accolades. Les ensembles sont itérables mais non-indexables et non-mutables.

Les objets séquentiels que sont les listes, les t-uplets et les chaînes, sont toujours indexés à partir de 0 . **A[0]** désigne le premier élément de la séquence, **A[1]** le deuxième, etc. On peut également effectuer une lecture à l'envers : **A[-1]** désigne le dernier élément de la séquence, **A[-2]** l'avant-dernier, etc

Les tableaux suivants résument les différents opérateurs sur les objets composés.

Opérations

len(A)	retourne la longueur de A (liste, t-uplet, chaîne, ensemble)
A[k]	accède à l'élément d'indice k de la séquence A (liste, t-uplet, chaîne)
A + B	opération de concaténation, juxtapose les deux séquences (liste,t-uplet,chaîne)
n * A	opération de duplication, juxtapose n (entier) fois la séquence A (liste,t-uplet,chaîne)
max(A)	retourne l'objet maximal de la collection A (liste, t-uplet, chaîne, ensemble)
min(A)	retourne l'objet minimal de la collection A (liste, t-uplet, chaîne, ensemble)
sum(A)	retourne la somme des éléments d'une collection A de nombres (liste, t-uplet, chaîne, ensemble)

Comparaisons entre objets

a in A	teste l'appartenance d'un objet à A (liste, t-uplet, chaîne, ensemble) et retourne un booléen
a not in A	teste la non appartenance d'un objet à A (liste, t-uplet, chaîne, ensemble) et retourne un booléen
A==B	teste l'égalité de A et B (listes, t-uplets, chaînes, ensembles), retourne un booléen
A != B	teste la non égalité de A et B (listes, t-uplets, chaînes, ensembles), retourne un booléen
A < B	test d'inégalité dans l'ordre lexicographique, lorsque A et B sont des séquences (liste, t-uplet, chaîne), retourne un booléen
A < B	test d'inclusion lorsque A et B sont des ensembles, retourne un booléen

Constructions et conversions

[a,b,c]	crée la liste d'objets a, b, c
[] ou list()	crée la liste vide
(a,b,c)	crée le t-uplet d'objets a, b, c
a,b,c	crée l'ensemble d'éléments a, b, c
set()	crée l'ensemble vide
"abc" ou 'abc'	crée la chaîne de caractères a, b, c
"" ou '' ou str()	crée la chaîne de caractères vide
list(A)	convertit en liste tout type d'objet composé (liste, t-uplet, chaîne, ensemble)
set(A)	convertit en ensemble. Élimine les répétitions et range les éléments par ordre croissant, si possible
str(A)	convertit en chaîne de caractères tout type d'objet composé (liste, t-uplet, chaîne, ensemble)

2. Boucles et autres outils

2.1.Les boucles

Il existe deux types de boucle :

- Les boucles conditionnelles (boucle While)

```
while condition:
    instructions
```

- Les boucles non-conditionnelles (boucle For)

```
for elt in sequence:
    instructions
```

Contrairement à une boucle conditionnelle, pour une boucle non-conditionnelle, le nombre d'itérations (passages dans la boucle pour lesquels le bloc d'instruction est effectué) est fixé et correspond à la longueur de la séquence. La variable **elt** prend successivement la valeur d'un élément de la séquence. Cette variable **elt** peut être utilisée dans le bloc d'instructions ou pas.

La séquence peut correspondre à une liste, un t-uplet ou une chaîne de caractères.

Exemple : le script suivant permet d'afficher les différents termes d'une liste

```
for e in L:
        print(e)
```

Il est également possible de remplacer une boucle non-conditionnelle par une conditionnelle de la manière suivante :

```
for i in range (0,n):
        instructions
```

Équivaut à

```
i=0
while i<n:
        instructions
        i=i+1
```

Dans ce cas, la variable de boucle n'est pas détruite après l'exécution de la boucle (ce qui n'est pas le cas pour la boucle non-conditionnelle) et peut donc être réutilisée.

2.2. Compteurs

D'une manière générale, on peut avoir besoin de connaître le nombre d'apparitions d'un certain fait dans le déroulement d'un programme (par exemple, le nombre d'itérations dans une boucle conditionnelle). Pour cela, on a recours à une variable appelée compteur, initialiée à 0 et qui sera incrémentée pour chaque fait apparu.

Exemple : la fonction suivante permet d'établir le nombre de chiffres intervenant dans l'écriture d'un nombre en base 10.

```
def nb_chiffres(n):
        cpt=0
        while n>0:
                n=n//2
                cpt+=1
        return(cpt)
```

2.3. Accumulateurs

Un accumulateur, tout comme un compteur, est une variable mais qui, à la différence du compteur, peut être incrémenté d'une valeur autre que 1 ou décrémenté. Là encore, il sera nécessaire d'initiatiliser la variable à 0.

Exemple : la fonction suivante permet de calculer la moyenne des éléments d'une liste.

```
def moyenne(L):
        acc=0
        for x in L:
                acc+=x
        return(acc/len(L))
```

3. Recherche de valeurs dans une structure séquentielle

3.1. Définition

Une occurence est l'apparition d'un fait, par exemple la présence d'un mot dans un texte. Lorsqu'on cherche une occurence d'un élément dans un objet séquentiel, on cherche si une place est occupée par cet élément et le rang associé.

3.2.Recherche d'une occurence

Il s'agit d'écrire un algorithme capable de rechercher de manière séquentielle un élément dans un tableau (liste, t-uplet ou chaîne de caractères). Pour cela, nous utiliserons la méthode dite de balayage (ou recherche linéaire) :

- L'élément cherché est comparé successivement à ceux du tableau
- L'algorithme doit s'arrêter dès que l'élément est trouvé ou si la fin du tableau est atteinte

Il existe différentes possibilités afin d'effectuer cette tâche à partir de boucles conditionnelles ou non-conditionnelles.

Considérons un tableau (liste, chaîne de caractères ou t-uplet) **tab** et un élément **val** dont on cherche la présence ou non.

✓ Écrire à l'aide d'une boucle **while** une fonction **Recherche1** permettant de savoir si l'élément **val** est présent dans **tab**.

✓ Écrire à l'aide d'une boucle **for** et d'une sortie anticipée une fonction **Recherche2** permettant de savoir si l'élément **val** est présent dans **tab**.

✓ Écrire une fonction **Recherche3 où Python fait tout le travail**.

3.3.Recherche de première occurence

On cherche maintenant à écrire une fonction renvoyant l'indice pour lequel la première occurence a lieu. Dans le cas où l'élément est absent, on souhaite renvoyer **len(tab)**.

✓ Écrire à l'aide d'une boucle **while** une fonction **Recherche4** renvoyant la première occurence de l'élément **val** dans **tab**.

✓ Écrire à l'aide d'une boucle **for** et une sortie de boucle anticipée une fonction **Recherche5** renvoyant la première occurence de l'élément **val** dans **tab**.

3.4.Recherche de dernière occurence

On cherche désormais à écrire une fonction renvoyant l'indice pour lequel la dernière occurence a lieu. Dans le cas où l'élément est absent, on souhaite renvoyer **len(tab)**.

✓ Écrire à l'aide d'une boucle **for** une fonction **Recherche6** renvoyant la dernière occurence de l'élément **val** dans **tab** (on pourra parcourir la séquence en partant de la fin).

3.5.Recherche d'extremum

✓ Écrire à l'aide d'une boucle **for** une fonction **Recherche7** renvoyant le maximum de **tab** (indication : si la liste est non vide, on pourra considérer le premier élément comme maximum provisoire puis parcourir la séquence en remplaçant ce dernier chaque fois qu'une valeur plus grande est rencontrée).

✓ Comment modifier la fonction précédente pour qu'elle renvoie également l'indice pour lequel le maximum est rencontré ?

4. Dictionnaires

Dans une liste, un t-uplet ou une chaîne de caractère, les termes sont ordonnés. On les repère par leur indice. Si n est la longueur de la structure séquentielle, à chaque valeur entière de 0 à $n-1$ correspond un élément.

Nous allons voir maintenant un nouveau type de structure séquentielle pour lequel l'indexation se fait par un système de clé.

4.1.Définition

Un dictionnaire (objet de type **dict**) permet d'associer des clés à des valeurs. Les clés sont des objets non-mutables et les valeurs des objets quelconques. Les clés ne sont pas ordonnées.

Dans un dictionnaire, les couples **clé : valeur** sont séparés par des virgules et encadrés par des accolades.

Exemple : **{'Dupont' : 3.6, 'Martin' : 6.8, 'Durand' : 7.4 }** est un dictionnaire contenant les notes de trois élèves, les noms correspondant aux clés.

4.2.Opérations sur les dictionnaires

Considérons le dictionnaire suivant :
day={'lundi' : 'monday', 'mardi' : 'tuesday', 'mercredi':'wednesday'}

- L'appel d'une valeur se fait en indiquant la clé (et non plus le rang comme pour les listes)

Exemple : **day['lundi']** renvoie **'monday'**

- L'ajout d'une paire **clé : valeur** s'effectue de la manière suivante : **day['jeudi']='thursday'**
- La fonction **len()** renvoie la longueur d'un dictionnaire qui est à la fois le nombre de clés et le nombre de valeurs.
- L'accès aux clés s'effectue par la fonction **keys**

Exemple : **day.keys()** renvoie **['lundi', 'mardi', 'mercredi', 'jeudi']**

- L'accès à l'ensemble des couples s'effectue par la fonction **items**
- Attention, le test d'appartenance ne porte que sur les clés et non sur les valeurs.

Exemple :

- **'lundi' in day** renvoie **True**
- **'Monday' in day** renvoie **False**