Données 11 - Le type abstrait Liste

Introduction : Type abstrait ?

Abstraction	→	Réalisation concrète
Algorithme	→	Programme
Type abstrait	→	Type (implémenté)

Le passage d'une idée abstraite en une réalisation concrète se nomme _____.

I – La Liste en tant que type abstrait de données (TAD)

1.1 1er partie de la définition de Liste : l'idée générale

Le **type abstrait Liste** a les propriétés suivantes :

- Une **`Liste`** une **structure LINÉAIRE** composée d'une séquence _____ **et** _____ de **`Places`** contenant chacune un **`Élément.`**
- On doit pouvoir insérer et supprimer au moins une **`Place`** (à choisir : au début, la fin, ...?)
- On doit pouvoir accéder à toutes les **`Places`**.
- On doit pouvoir lire un **`Élément`** connaissant sa **`Place.`**
- On doit pouvoir trouver le successeur unique de chaque **`Place.`**
- La **première `Place`** de la **`Liste`** est nommée _____ (_____ en anglais).
- L'**ensemble des autres `Places`** forme la _____ (_____ en anglais).
- La **dernière `Place`** est nommée _____. Attention, on trouve parfois le mot _____...
- L'**indice** d'une Place correspond au **nombre de sauts nécessaires pour l'atteindre à partir de la tête** : la tête est à l'indice 0, la suivante 1...
- La Liste est **homogène** si tous les éléments stockés sont de même type.

Exercices 01-02-03-04° Quel élément forme ici la tête ? la queue ? Que contient au total la fin ?

`Exo 1 :` **`12`** `→ 4 → 5 → -15` `Exo 3 :` `21 → 12 → 13 → 0`

`Exo 2 :` `23 ← 12 ← 460 ←` **`1`** `Exo 4 :` `20 ← 45 ← 10 ← 18`

Exercice 05° Seule la tête est indiquée en gras. Mêmes questions : `23 - 12 - 460 -` **`1`**

1.2 Représentations d'une liste

Liste 1 : **5** → 8 → 2 → 3

Liste 1 : **5** → liste2	**`liste1 = (5, liste2)`**	`liste1 =`
Liste 2 : **8** → liste3	**`liste2 = (8, liste3)`**	`liste2 =`
Liste 3 : **2** → liste 4	**`liste3 = (2, liste4)`**	`liste3 =`
Liste 4 : **3** → liste 5	**`liste4 = (3, liste5)`**	`liste4 =`
Liste 5 : vide !	**`liste5 = ()`**	`liste5 = ()`

Cette notation est assez pénible. On peut simplement utiliser une représentation **en séquence ordonnée** :

`liste1 =`

DEFINITION : Une Liste est → soit

→ soit

DEFINITION FORMELLE :

06-07° Donner les représentations imbrication et séquence : ***Exo 6*** : **12** → 4 ***Exo 7*** : **5** → 2 → 10

1.3 Définition de signature

1.4 Définition de spécification

1.5 Définition de primitives d'un type abstrait de données

1.6 2nd partie de la définition de `Liste` : spécifications des primitives

Les 8 opérations primitives

`Liste : « meuble »`
`Place : « dossier/case »`
`Elément : le truc rangé`

1. **`nLV()`** ou **`nouvelleListeVide`**`() → Liste  VIDE`
2. **`estLV()`** ou **`estListeVide`**`(`**`lst`**`:Liste) → Booléen`
3. **`ins()`** ou **`inserer`**`(`**`lst`**`:Liste, `**`elt`**`:Element, ... `**`i`**`:Entier Naturel VALIDE) → Liste NON VIDE`
4. **`sup()`** ou **`supprimer`**`(`**`lst`**`:Liste NON VIDE, ... `**`i`**`:Entier Naturel VALIDE) → Liste`
5. **`longueur`**`(`**`lst`**`:Liste) → Entier Naturel`
6. **`acc()`** ou **`acces`**`(`**`lst`**`:Liste NON VIDE, ... `**`i`**`:Entier Naturel VALIDE)) → Place`
7. **`contenu`**`(`**`plc`**`:Place) → Elément`
8. **`succ()`** ou **`successeur`**`(`**`plc`**`:Place) → Place|Vide`

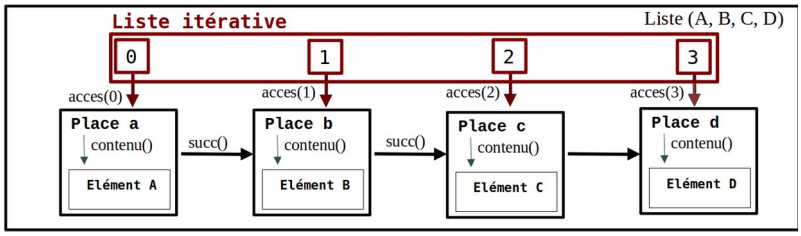
1.7 Muable/Mutable ou Immuable ?

`InsererTete``(`**`lst`**`:Liste, `**`elt`**`:Elément) → Vide` `Muable car on ne renvoie pas de nouvelle liste.`

`supprimerTete``(`**`lst`**`:Liste NON VIDE) → Element` `Muable car on ne renvoie pas de nouvelle liste.`

II – Deux Listes en tant que type abstrait de données !

2.1 Liste itérative

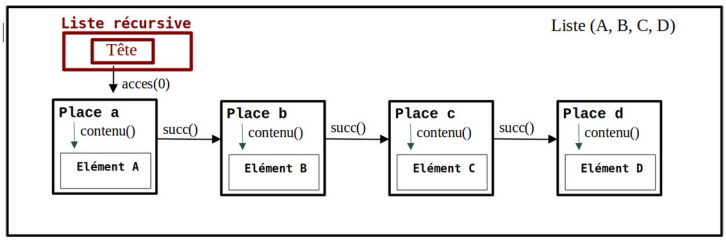


La Liste pointe toutes les Places.
Les liaisons Liste-Places sont fixées par leur indice.
Les éléments doivent changer de Place.

AVANTAGE :

DÉSAVANTAGE :

2.2 Liste récursive



La liste ne pointe que la tête.
Chaque Place connaît son successeur.
Les Places sont mobiles.

AVANTAGE :

DÉSAVANTAGE :

2.3 Utilisation usuelle du mot "Liste"

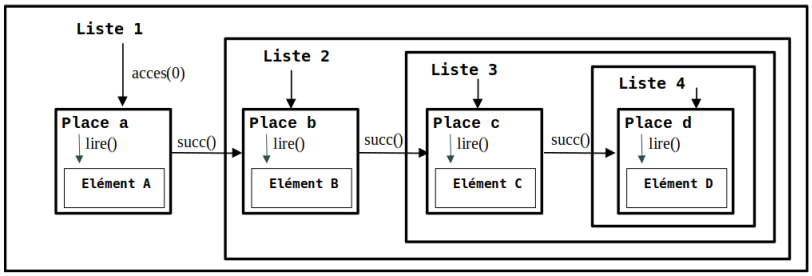
Normalement, Liste désigne soit une Liste Itérative, soit une Liste Récursive. En réalité, la plupart des gens utilisent

- le mot **tableau** lorsqu'ils parlent d'une implémentation d'une **liste itérative** et
- le mot **liste** pour désigner l'implémentation d'une **liste récursive**.

2.4 Le type `list` de Python ?

```
>>> t = [10, 100, 6]
>>> t.append(1000)
>>> t
[10, 100, 6, 1000]
>>> t.pop()
1000
>>> t
[10, 100, 6]
```

III – Liste récursive : couple (Tête, Queue)



On considère une liste IMMUABLE.

La Liste ne contient que les informations de la place en Tête.

Les primitives d'accès, d'insertion et suppression ne peuvent agir que sur la Tête.

7 opérations primitives (qu'on suppose pouvoir implémenter à coût constant) :

- **nLV()** **nouvelleListeVide()** → Liste VIDE
- **estLV()** **estListeVide**(lst:Liste) → Booléen
- **insT()** **insererTete**(lst:Liste, elt:Élément) → Liste NON VIDE
- **supT()** **supprimerTete**(lst:Liste NON VIDE) → Liste
- **tete() accT()** **accesTete**(lst:Liste NON VIDE) → Place
- **contenu**(plc:Place) → Élément
- **succ()** **successeur**(plc:Place) → Place|Vide

CONCLUSION :

A partir des 7 opérations primitives de la Liste récursive (Tête, Queue), on parvient à créer toutes les fonctions désirées sur la Liste.

Important : les algorithmes réalisés Q13 à Q20 sont à savoir refaire à partir de ces primitives, en itératif et en récursif.

premier(lst:Liste NON VIDE) → Élément
longueur(lst:Liste) → Entier Naturel
inverser(lst:Liste) → Liste
inserer(lst:Liste, elt:Élément, i:Entier Naturel VALIDE) → Liste NON VIDE
ieme(lst:Liste NON VIDE, i:Entier Naturel VALIDE) -> Élément
recherche(lst:Liste, x:Élément) -> Entier

IV – Implémentation de certains types abstraits en Python

4.1 Vocabulaire

Type abstrait (TAD)	Idée abstraite (mais définie formellement via les primitives) de données.
Type (implémenté)	Implémentation en machine d'un type abstrait. Cela regroupe donc les types simples et les types construits. D'où le nom de la fonction native Python **type()**.
Type simple	Implémentation d'une donnée unique, comme un entier, un flottant...
Type structuré **Type construit**	Implémentation d'un type abstrait qui est une structure contenant d'autres données. C'est une sorte de conteneur.
Structure de données	Type construit sur lequel on peut agir en utilisant des opérateurs et des fonctions prédéfinies.

4.2 Implémentations de type abstrait présentes de base : les types natifs

- Le type **booléen** est implémenté via le type natif **bool**.
- Le type **entier** est implémenté via le type natif **int**.
- Le type **nombre à virgule flottante** est implémenté via le type natif **float.**
- Le type **tableau statique** (array en anglais) via le type natif ... **list**.
- Le type **tableau dynamique** (vector en anglais) via le type natif **list**.
- Le type **tableau associatif** via le type natif **dict**.
- Le type **p-uplet nommé** via le type natif **dict**.
- Le type **p-uplet** via le type **tuple**.
- Le type **ensemble** via le type **set** *(hors programme en NSI), ou* ***dict*** *s'il le faut.*