C14 Structures de données linéaires

Définition : structure de données

- En informatique, une structure de données est une façon d'organiser, de gérer et de stocker des données permettant d'accéder et de modifier ces données de façon efficace.

C14 Structures de données linéaires

Définition : structure de données

- En informatique, une structure de données est une façon d'organiser, de gérer et de stocker des données permettant d'accéder et de modifier ces données de façon efficace.

  Les listes de Python sont une structure de données.

C14 Structures de données linéaires

Définition : structure de données

- En informatique, une structure de données est une façon d'organiser, de gérer et de stocker des données permettant d'accéder et de modifier ces données de façon efficace.

  Les listes de Python sont une structure de données.

- L'interface de la structure de données est l'ensemble des opérations accessibles à un utilisateur de la structure de données.

C14 Structures de données linéaires

Définition : structure de données

- En informatique, une structure de données est une façon d'organiser, de gérer et de stocker des données permettant d'accéder et de modifier ces données de façon efficace.

  Les listes de Python sont une structure de données.

- L'interface de la structure de données est l'ensemble des opérations accessibles à un utilisateur de la structure de données.

  `append` qui permet d'ajouter un élément à la fin d'une liste de Python fait partie de l'interface des listes

C14 Structures de données linéaires

Définition : structure de données

- En informatique, une structure de données est une façon d'organiser, de gérer et de stocker des données permettant d'accéder et de modifier ces données de façon efficace.

  Les listes de Python sont une structure de données.

- L'interface de la structure de données est l'ensemble des opérations accessibles à un utilisateur de la structure de données.

  `append` qui permet d'ajouter un élément à la fin d'une liste de Python fait partie de l'interface des listes

- L'implémentation de la structure de données est la façon dont elle est représentée et codée en mémoire et n'est pas forcément accessible à l'utilisateur. Cependant, la complexité des opérations de l'interface est généralement fournie.

C14 Structures de données linéaires

Définition : structure de données

- En informatique, une structure de données est une façon d'organiser, de gérer et de stocker des données permettant d'accéder et de modifier ces données de façon efficace.
  Les listes de Python sont une structure de données.
- L'interface de la structure de données est l'ensemble des opérations accessibles à un utilisateur de la structure de données.
  `append` qui permet d'ajouter un élément à la fin d'une liste de Python fait partie de l'interface des listes
- L'implémentation de la structure de données est la façon dont elle est représentée et codée en mémoire et n'est pas forcément accessible à l'utilisateur. Cependant, la complexité des opérations de l'interface est généralement fournie.
  Nous ne savons pas comment sont représentées ou codées les listes de Python, mais nous savons que par exemple la copie d'une liste est une opération en $O(n)$

Caractérisation par l'interface

- La différence entre interface et implémentation est fondatamentale et doit être bien comprise. En effet une même structure de données peut avoir plusieurs implémentations. C'est comme par exemple un téléviseur, on retrouve toujours la même interface (la télécommande) mais l'implémentation peut être différente (tube cathodique, écran plasma, lcd, led, ...).

C14 Structures de données linéaires

Caractérisation par l'interface

- La différence entre interface et implémentation est fondatamentale et doit être bien comprise. En effet une même structure de données peut avoir plusieurs implémentations. C'est comme par exemple un téléviseur, on retrouve toujours la même interface (la télécommande) mais l'implémentation peut être différente (tube cathodique, écran plasma, lcd, led, ...).
- Lorsqu'on définit un *cahier des charges* pour une structure de données (ensemble des données et opérations possibles), on définit ce qu'on appelle un type abstrait de données. Ainsi, une structure de données peut être vue comme une implémentation d'un type abstrait de données.

C14 Structures de données linéaires

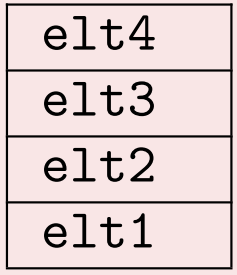
Piles

- Au niveau sémantique, une pile est semblable à une pile d'objet dans la vie de tous les jours.

C14 Structures de données linéaires

Piles

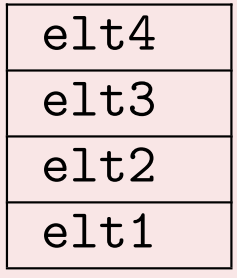
- Au niveau sémantique, une pile est semblable à une pile d'objet dans la vie de tous les jours.



C14 Structures de données linéaires

Piles

- Au niveau sémantique, une pile est semblable à une pile d'objet dans la vie de tous les jours.

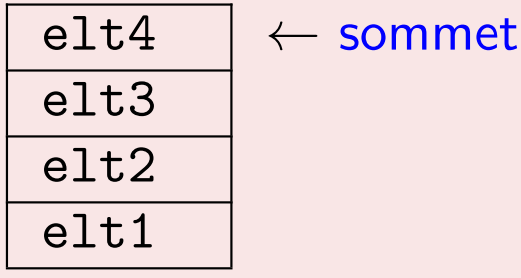


- L'élément situé en haut de la pile s'appelle le sommet.

C14 Structures de données linéaires

Piles

- Au niveau sémantique, une pile est semblable à une pile d'objet dans la vie de tous les jours.

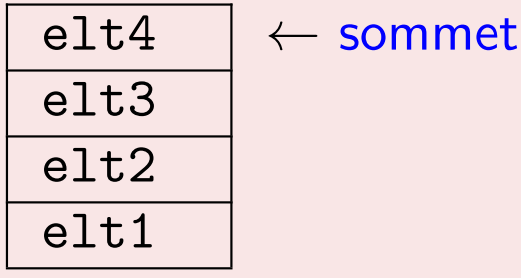


- L'élément situé en haut de la pile s'appelle le sommet.

C14 Structures de données linéaires

Piles

- Au niveau sémantique, une pile est semblable à une pile d'objet dans la vie de tous les jours.

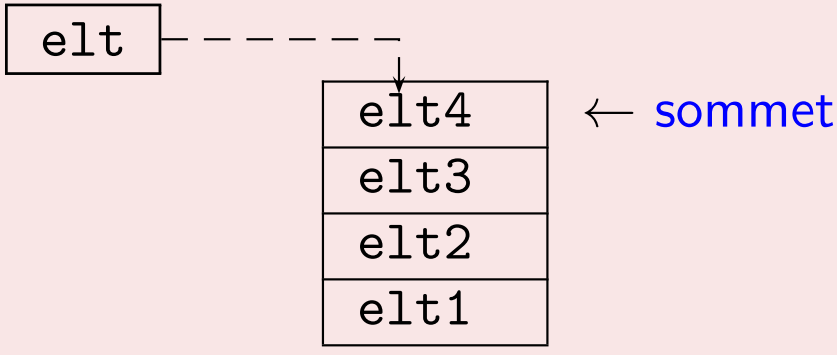


- L'élément situé en haut de la pile s'appelle le sommet.
- Empiler signifie ajouter un élément au sommet de la pile

C14 Structures de données linéaires

Piles

- Au niveau sémantique, une pile est semblable à une pile d'objet dans la vie de tous les jours.

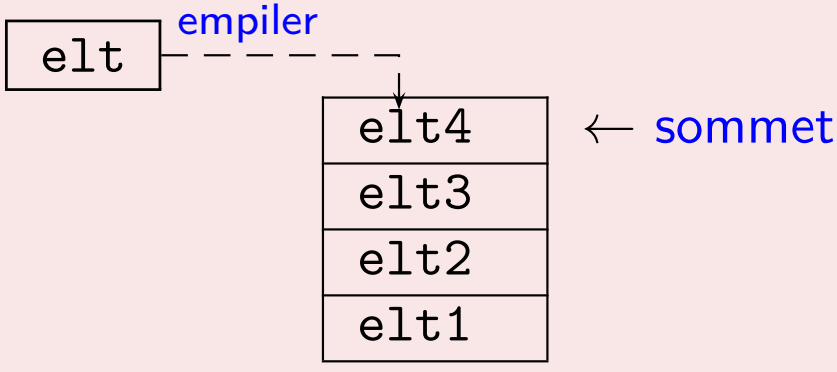


- L'élément situé en haut de la pile s'appelle le sommet.
- Empiler signifie ajouter un élément au sommet de la pile

C14 Structures de données linéaires

Piles

- Au niveau sémantique, une pile est semblable à une pile d'objet dans la vie de tous les jours.

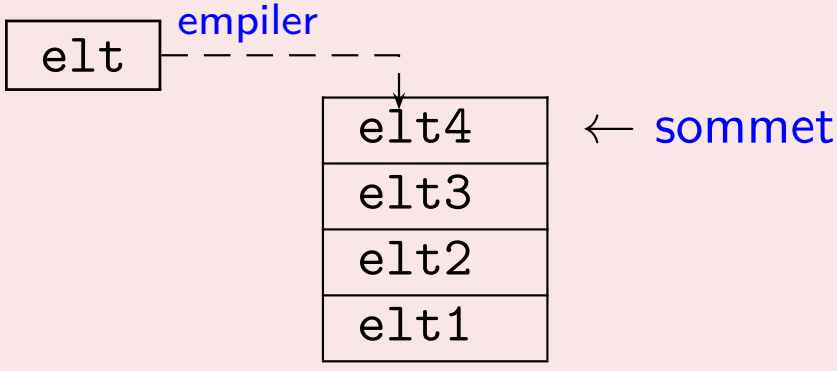


- L'élément situé en haut de la pile s'appelle le sommet.
- Empiler signifie ajouter un élément au sommet de la pile

C14 Structures de données linéaires

Piles

- Au niveau sémantique, une pile est semblable à une pile d'objet dans la vie de tous les jours.

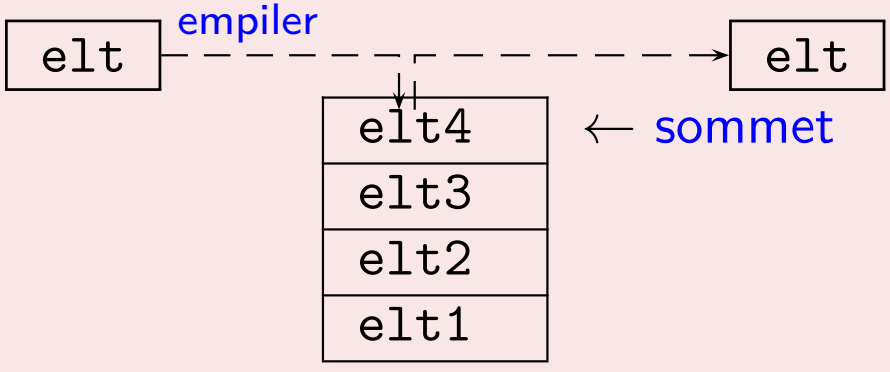


- L'élément situé en haut de la pile s'appelle le sommet.
- Empiler signifie ajouter un élément au sommet de la pile
- Dépiler signifie retirer l'élément situé au sommet de la pile

C14 Structures de données linéaires

Piles

- Au niveau sémantique, une pile est semblable à une pile d'objet dans la vie de tous les jours.

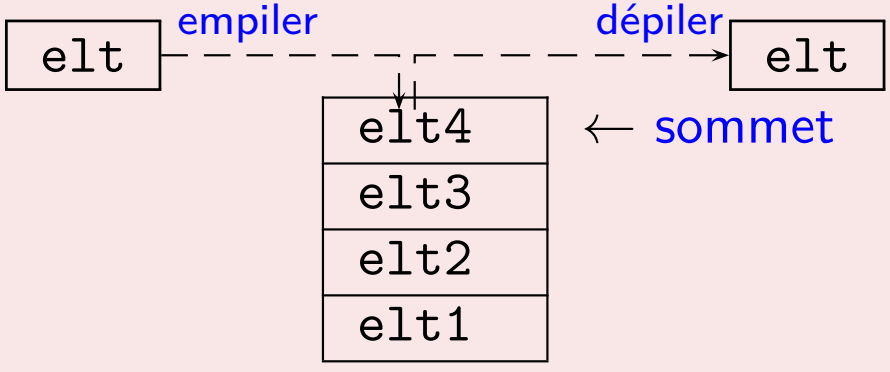


- L'élément situé en haut de la pile s'appelle le sommet.
- Empiler signifie ajouter un élément au sommet de la pile
- Dépiler signifie retirer l'élément situé au sommet de la pile

C14 Structures de données linéaires

Piles

- Au niveau sémantique, une pile est semblable à une pile d'objet dans la vie de tous les jours.

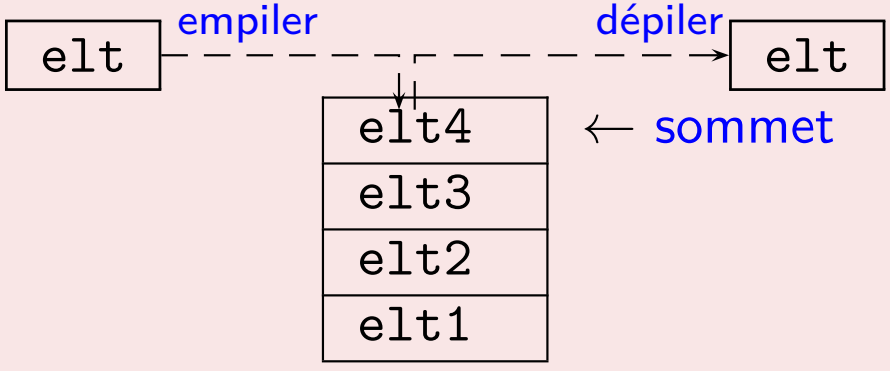


- L'élément situé en haut de la pile s'appelle le sommet.
- Empiler signifie ajouter un élément au sommet de la pile
- Dépiler signifie retirer l'élément situé au sommet de la pile

C14 Structures de données linéaires

Piles

- Au niveau sémantique, une pile est semblable à une pile d'objet dans la vie de tous les jours.



- L'élément situé en haut de la pile s'appelle le sommet.
- Empiler signifie ajouter un élément au sommet de la pile
- Dépiler signifie retirer l'élément situé au sommet de la pile
- Ainsi le premier élément entré dans la pile sera aussi le dernier à en sortir, on dit qu'une pile est une structure LIFO *Last In First Out*

C14 Structures de données linéaires

Piles comme structures de données

L'interface d'une structure de données Pile se limite donc aux opérations suivantes :

Piles comme structures de données

L'interface d'une structure de données Pile se limite donc aux opérations suivantes :

- `est_vide()` qui renvoie un booléen indiquant si la pile est vide ou non.

Piles comme structures de données

L'interface d'une structure de données Pile se limite donc aux opérations suivantes :

- `est_vide()` qui renvoie un booléen indiquant si la pile est vide ou non.
- `empiler()` qui ajoute un élément au sommet de la pile.

Piles comme structures de données

L'interface d'une structure de données Pile se limite donc aux opérations suivantes :

- `est_vide()` qui renvoie un booléen indiquant si la pile est vide ou non.
- `empiler()` qui ajoute un élément au sommet de la pile.
- `depiler()` qui retire l'élément situé au sommet (cela n'est possible que si la pile n'est pas vide).

C14 Structures de données linéaires

Piles comme structures de données

L'interface d'une structure de données Pile se limite donc aux opérations suivantes :

- `est_vide()` qui renvoie un booléen indiquant si la pile est vide ou non.
- `empiler()` qui ajoute un élément au sommet de la pile.
- `depiler()` qui retire l'élément situé au sommet (cela n'est possible que si la pile n'est pas vide).

Utilisation

En dépit de sa simplicité, cette structure de données a de nombreuses applications en informatique : pile d'appel récursif, pile d'évaluation d'une expression, ...

C14 Structures de données linéaires

Files

- Au niveau sémantique, une file est semblable à une file d'attente dans la vie de tous les jours.

C14 Structures de données linéaires

Files

- Au niveau sémantique, une file est semblable à une file d'attente dans la vie de tous les jours.

C14 Structures de données linéaires

Files

- Au niveau sémantique, une file est semblable à une file d'attente dans la vie de tous les jours.

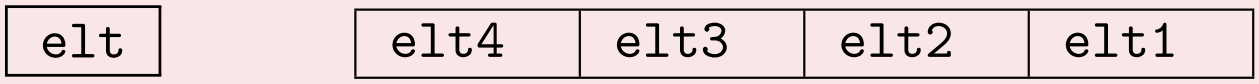
elt4	elt3	elt2	elt1

- Enfiler signifie ajouter un élément en fin de file

C14 Structures de données linéaires

Files

- Au niveau sémantique, une file est semblable à une file d'attente dans la vie de tous les jours.

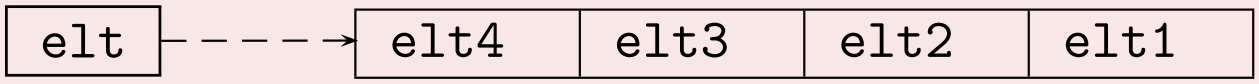


- Enfiler signifie ajouter un élément en fin de file

C14 Structures de données linéaires

Files

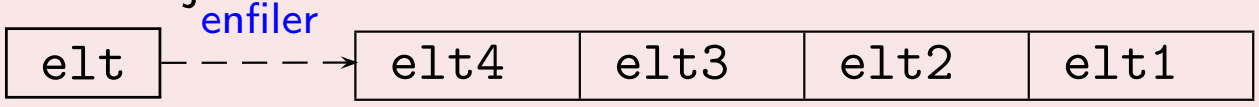
- Au niveau sémantique, une file est semblable à une file d'attente dans la vie de tous les jours.



- Enfiler signifie ajouter un élément en fin de file

Files

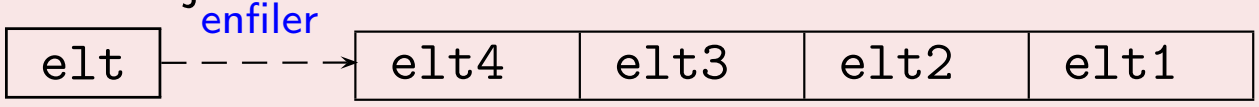
- Au niveau sémantique, une file est semblable à une file d'attente dans la vie de tous les jours.



- Enfiler signifie ajouter un élément en fin de file

Files

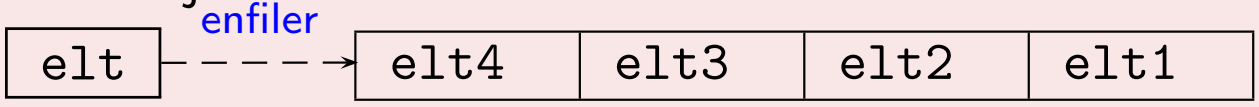
- Au niveau sémantique, une file est semblable à une file d'attente dans la vie de tous les jours.



- Enfiler signifie ajouter un élément en fin de file
- Défiler signifie retirer l'élément situé au début de la file.

Files

- Au niveau sémantique, une file est semblable à une file d'attente dans la vie de tous les jours.

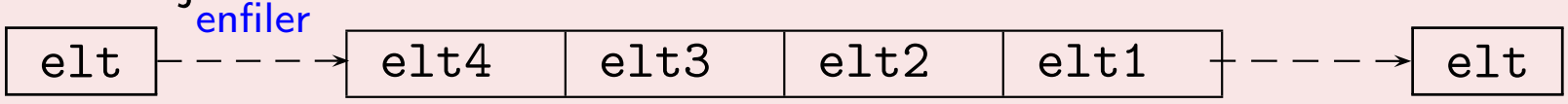


- Enfiler signifie ajouter un élément en fin de file
- Défiler signifie retirer l'élément situé au début de la file.

C14 Structures de données linéaires

Files

- Au niveau sémantique, une file est semblable à une file d'attente dans la vie de tous les jours.

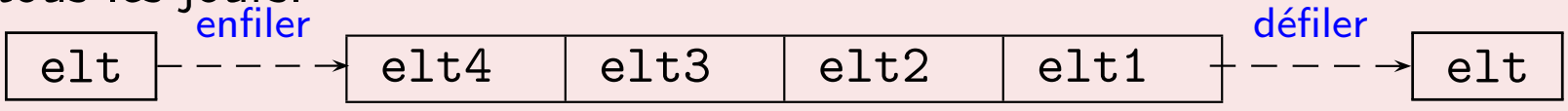


- Enfiler signifie ajouter un élément en fin de file
- Défiler signifie retirer l'élément situé au début de la file.

C14 Structures de données linéaires

Files

- Au niveau sémantique, une file est semblable à une file d'attente dans la vie de tous les jours.

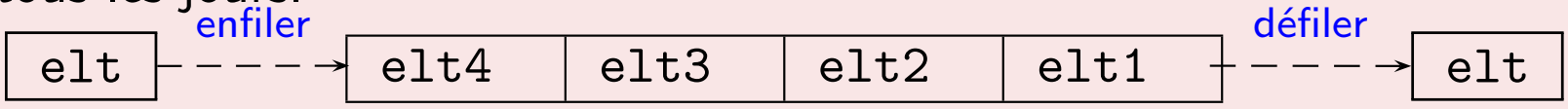


- Enfiler signifie ajouter un élément en fin de file
- Défiler signifie retirer l'élément situé au début de la file.

C14 Structures de données linéaires

Files

- Au niveau sémantique, une file est semblable à une file d'attente dans la vie de tous les jours.



- Enfiler signifie ajouter un élément en fin de file
- Défiler signifie retirer l'élément situé au début de la file.
- Ainsi le premier élément entré dans la file sera aussi le premier à en sortir, on dit qu'une file est une structure FIFO *First In First Out*

Files comme structures de données

L'interface d'une structure de données File se limite donc aux opérations suivantes :

Files comme structures de données

L'interface d'une structure de données File se limite donc aux opérations suivantes :

- `est_vide()` qui renvoie un booléen indiquant si la file est vide ou non.

Files comme structures de données

L'interface d'une structure de données File se limite donc aux opérations suivantes :

- `est_vide()` qui renvoie un booléen indiquant si la file est vide ou non.
- `enfiler(element)` qui ajoute un élément à la fin de la file.

Files comme structures de données

L'interface d'une structure de données File se limite donc aux opérations suivantes :

- `est_vide()` qui renvoie un booléen indiquant si la file est vide ou non.
- `enfiler(element)` qui ajoute un élément à la fin de la file.
- `defiler()` qui retire l'élément situé au début de la file (cela n'est possible que si la file n'est pas vide).

C14 Structures de données linéaires

Files comme structures de données

L'interface d'une structure de données File se limite donc aux opérations suivantes :

- `est_vide()` qui renvoie un booléen indiquant si la file est vide ou non.
- `enfiler(element)` qui ajoute un élément à la fin de la file.
- `defiler()` qui retire l'élément situé au début de la file (cela n'est possible que si la file n'est pas vide).

Utilisation

Comme pour les piles, cette structure de données a de nombreuses applications en informatique : file d'attente d'une imprimante, simulation de files d'attentes réelles, ...

C14 Structures de données linéaires

Listes

Au niveau sémantique, une liste est suite finie d'éléments (avec répétition éventuelle) on peut noter une liste : $e_0, e_1, \ldots, e_n$.

C14 Structures de données linéaires

Listes

Au niveau sémantique, une liste est suite finie d'éléments (avec répétition éventuelle) on peut noter une liste : $e_0, e_1, \ldots, e_n$.

L'interface d'une liste contient généralement les éléments suivants :

C14 Structures de données linéaires

Listes

Au niveau sémantique, une liste est suite finie d'éléments (avec répétition éventuelle) on peut noter une liste : $e_0, e_1, \ldots, e_n$.

L'interface d'une liste contient généralement les éléments suivants :

- Un constructeur permettant de créer la liste vide.

C14 Structures de données linéaires

Listes

Au niveau sémantique, une liste est suite finie d'éléments (avec répétition éventuelle) on peut noter une liste : $e_0, e_1, \ldots, e_n$.

L'interface d'une liste contient généralement les éléments suivants :

- Un constructeur permettant de créer la liste vide.
- Une opération renvoyant un booléen suivant que la liste soit vide ou non.

C14 Structures de données linéaires

Listes

Au niveau sémantique, une liste est suite finie d'éléments (avec répétition éventuelle) on peut noter une liste : $e_0, e_1, \ldots, e_n$.

L'interface d'une liste contient généralement les éléments suivants :

- Un constructeur permettant de créer la liste vide.
- Une opération renvoyant un booléen suivant que la liste soit vide ou non.
- Une opération permettant d'ajouter un élément en début (ou en fin) de liste.

C14 Structures de données linéaires

Listes

Au niveau sémantique, une liste est suite finie d'éléments (avec répétition éventuelle) on peut noter une liste : $e_0, e_1, \ldots, e_n$.

L'interface d'une liste contient généralement les éléments suivants :

- Un constructeur permettant de créer la liste vide.
- Une opération renvoyant un booléen suivant que la liste soit vide ou non.
- Une opération permettant d'ajouter un élément en début (ou en fin) de liste.
- Une opération renvoyant le premier élément de la liste (on dit la tête de la liste).

C14 Structures de données linéaires

Listes

Au niveau sémantique, une liste est suite finie d'éléments (avec répétition éventuelle) on peut noter une liste : $e_0, e_1, \ldots, e_n$.

L'interface d'une liste contient généralement les éléments suivants :

- Un constructeur permettant de créer la liste vide.
- Une opération renvoyant un booléen suivant que la liste soit vide ou non.
- Une opération permettant d'ajouter un élément en début (ou en fin) de liste.
- Une opération renvoyant le premier élément de la liste (on dit la tête de la liste).
- Une opération renvoyant la queue de la liste, c'est à dire la liste privée de sa tête.

C14 Structures de données linéaires

Listes

Au niveau sémantique, une liste est suite finie d'éléments (avec répétition éventuelle) on peut noter une liste : $e_0, e_1, \ldots, e_n$.

L'interface d'une liste contient généralement les éléments suivants :

- Un constructeur permettant de créer la liste vide.
- Une opération renvoyant un booléen suivant que la liste soit vide ou non.
- Une opération permettant d'ajouter un élément en début (ou en fin) de liste.
- Une opération renvoyant le premier élément de la liste (on dit la tête de la liste).
- Une opération renvoyant la queue de la liste, c'est à dire la liste privée de sa tête.
- Une opération permettant d'accéder à un élément de la liste par son index.

C14 Structures de données linéaires

Listes

Au niveau sémantique, une liste est suite finie d'éléments (avec répétition éventuelle) on peut noter une liste : $e_0, e_1, \ldots, e_n$.

L'interface d'une liste contient généralement les éléments suivants :

- Un constructeur permettant de créer la liste vide.
- Une opération renvoyant un booléen suivant que la liste soit vide ou non.
- Une opération permettant d'ajouter un élément en début (ou en fin) de liste.
- Une opération renvoyant le premier élément de la liste (on dit la tête de la liste).
- Une opération renvoyant la queue de la liste, c'est à dire la liste privée de sa tête.
- Une opération permettant d'accéder à un élément de la liste par son index.

Deux implémentations des listes sont généralement utilisées :

C14 Structures de données linéaires

Listes

Au niveau sémantique, une liste est suite finie d'éléments (avec répétition éventuelle) on peut noter une liste : $e_0, e_1, \ldots, e_n$.

L'interface d'une liste contient généralement les éléments suivants :

- Un constructeur permettant de créer la liste vide.
- Une opération renvoyant un booléen suivant que la liste soit vide ou non.
- Une opération permettant d'ajouter un élément en début (ou en fin) de liste.
- Une opération renvoyant le premier élément de la liste (on dit la tête de la liste).
- Une opération renvoyant la queue de la liste, c'est à dire la liste privée de sa tête.
- Une opération permettant d'accéder à un élément de la liste par son index.

Deux implémentations des listes sont généralement utilisées :

Par un tableau (éléments contigues en mémoire)

C14 Structures de données linéaires

Listes

Au niveau sémantique, une liste est suite finie d'éléments (avec répétition éventuelle) on peut noter une liste : $e_0, e_1, \ldots, e_n$.

L'interface d'une liste contient généralement les éléments suivants :

- Un constructeur permettant de créer la liste vide.
- Une opération renvoyant un booléen suivant que la liste soit vide ou non.
- Une opération permettant d'ajouter un élément en début (ou en fin) de liste.
- Une opération renvoyant le premier élément de la liste (on dit la tête de la liste).
- Une opération renvoyant la queue de la liste, c'est à dire la liste privée de sa tête.
- Une opération permettant d'accéder à un élément de la liste par son index.

Deux implémentations des listes sont généralement utilisées :

Par un tableau (éléments contigues en mémoire)
Par une liste chainée (chaque élément contient alors un lien vers le suivant)

Ambiguité

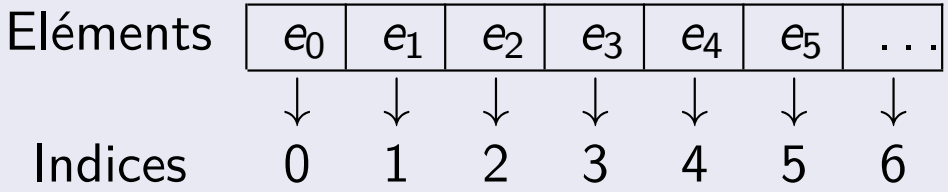
- Le terme liste devient ambigüe, en effet c'est à la fois le nom du type abstrait de données que nous venons de définir mais aussi celui d'une structure de données existante en Python.

⚠ Ambiguité

- Le terme liste devient ambigüe, en effet c'est à la fois le nom du type abstrait de données que nous venons de définir mais aussi celui d'une structure de données existante en Python.
- D'autres part, pour amplifier encore cette ambiguité, le terme *liste chaînée*, est parfois abrégé en liste, alors qu'il s'agît d'une implémentation possible et pas du type abstrait de données défini plus haut.

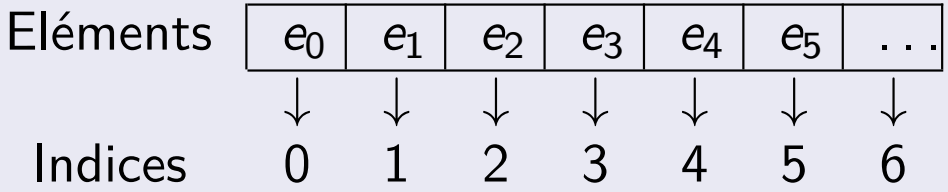
Python et le TAD liste

- En Python, le TAD liste est implémenté sous forme de tableau, c'est à dire que les éléments d'une liste de Python sont contigues en mémoire :

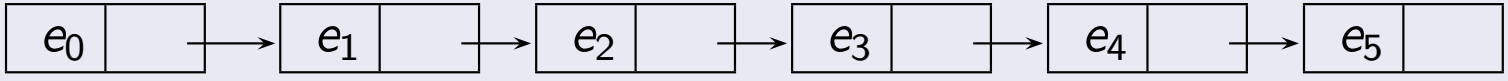


Python et le TAD liste

- En Python, le TAD liste est implémenté sous forme de tableau, c'est à dire que les éléments d'une liste de Python sont contigues en mémoire :



- L'autre implémentation que nous avons évoquée (celle par les listes chaînées) peut être importé à partir du module `collections` ou écrite par exemple à l'aide de la POO. Dans ce cas, les éléments d'une liste ne sont plus contigues en mémoire mais chaque élément doit aussi contenir un lien vers son suivant dans la liste :



Avantages et inconvénients de ces implémentations

Ces deux implémentations ont des conséquences importantes en terme de complexité des opérations mais aussi d'occupation mémoire :

- Sous forme de liste chaînée, l'occupation mémoire est plus importante (on stocke en plus de l'élément le lien vers l'élément suivant).

Avantages et inconvénients de ces implémentations

Ces deux implémentations ont des conséquences importantes en terme de complexité des opérations mais aussi d'occupation mémoire :

- Sous forme de liste chaînée, l'occupation mémoire est plus importante (on stocke en plus de l'élément le lien vers l'élément suivant).
- Accéder à un élément sous forme de tableau est direct alors que pour une liste chaînée il faut avoir parcouru au préalable tout ceux qui le précèdent depuis la tête.

Avantages et inconvénients de ces implémentations

Ces deux implémentations ont des conséquences importantes en terme de complexité des opérations mais aussi d'occupation mémoire :

- Sous forme de liste chaînée, l'occupation mémoire est plus importante (on stocke en plus de l'élément le lien vers l'élément suivant).
- Accéder à un élément sous forme de tableau est direct alors que pour une liste chaînée il faut avoir parcouru au préalable tout ceux qui le précèdent depuis la tête.
- L'insertion d'un élément dans un tableau oblige à décaler ceux qui le suivent alors que l'opération est directe en liste chaînée.

Avantages et inconvénients de ces implémentations

Ces deux implémentations ont des conséquences importantes en terme de complexité des opérations mais aussi d'occupation mémoire :

- Sous forme de liste chaînée, l'occupation mémoire est plus importante (on stocke en plus de l'élément le lien vers l'élément suivant).
- Accéder à un élément sous forme de tableau est direct alors que pour une liste chaînée il faut avoir parcouru au préalable tout ceux qui le précèdent depuis la tête.
- L'insertion d'un élément dans un tableau oblige à décaler ceux qui le suivent alors que l'opération est directe en liste chaînée.
- L'ajout d'un élément en fin de liste oblige à agrandir le tableau si la taille initialement prévue est dépassée alors qu'avec une liste cela ne pose pas de problèmes.

Choix d'une structure de données

En conclusion : on utilisera

Choix d'une structure de données

En conclusion : on utilisera

Un tableau (et donc une liste de Python) lorsque la taille des données est connu par avance et/ou lorsqu'on a besoin d'accéder aux éléments par leur indice rapidement.

Choix d'une structure de données

En conclusion : on utilisera

- Un tableau (et donc une liste de Python) lorsque la taille des données est connu par avance et/ou lorsqu'on a besoin d'accéder aux éléments par leur indice rapidement.
- Une liste chainée (module `deque`) pour les structures à taille variable et/ou lorsqu'on a besoin d'insérer ou de supprimer les éléments situés au debut ou à la fin.