Langage Python

Cours 2/5 : structures de données

Hubert Godfroy

05 novembre 2015

La dernière fois…

- Environnement Python
- Syntaxe du langage
- Exemple d'algorithme sur les entiers (somme, multiplication, pgcd, …) et les réels (racines d'un polynôme du second degrés)

La dernière fois…

- Environnement Python
- Syntaxe du langage
- Exemple d'algorithme sur les entiers (somme, multiplication, pgcd, …) et les réels (racines d'un polynôme du second degrés)

$\Rightarrow$ Comment utiliser des données plus complexes (tableaux, listes, arbres, …)

Plan

Plan

Quand deux objets sont-ils égaux ? (Exemples)

On dispose de l'opérateur ==.

- deux entiers ?

Quand deux objets sont-ils égaux ? (Exemples)

On dispose de l'opérateur ==.

- deux entiers ?
- deux tableaux ?

Quand deux objets sont-ils égaux ? (Exemples)

On dispose de l'opérateur ==.

- deux entiers ?
- deux tableaux ?
- deux arbres ?

Quand deux objets sont-ils égaux ? (Exemples)

On dispose de l'opérateur ==.

- deux entiers ?
- deux tableaux ?
- deux arbres ?

L'opérateur == essaye de "descendre" dans les structures pour tester l'égalité de chaque sous-terme.

L'opérateur == est une égalité structurelle

Quand deux objets sont-ils égaux ? (Cas général ?)

Comment doit se comporter l'égalité structurelle pour deux objets quelconques ?

Quand deux objets sont-ils égaux ? (Cas général ?)

Comment doit se comporter l'égalité structurelle pour deux objets quelconques ?

Impossible

- Qu'est-ce qu'un sous-terme ?
- L'opérateur == ne peut pas connaître toutes les structures à l'avance.

Quand deux objets sont-ils égaux ? (Cas général ?)

Comment doit se comporter l'égalité structurelle pour deux objets quelconques ?

Impossible

- Qu'est-ce qu'un sous-terme ?
- L'opérateur == ne peut pas connaître toutes les structures à l'avance.

$\rightsquigarrow$ Doit retourner quelque chose (philosophie Python)

Quand deux objets sont-ils égaux ? (Cas général ?)

Comment doit se comporter l'égalité structurelle pour deux objets quelconques ?

Impossible

- Qu'est-ce qu'un sous-terme ?
- L'opérateur == ne peut pas connaître toutes les structures à l'avance.

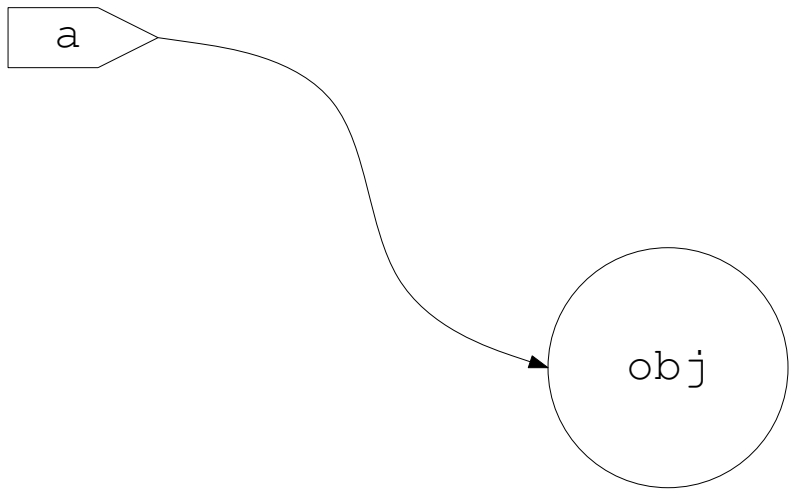
$\rightsquigarrow$ Doit retourner quelque chose (philosophie Python)

Quel autre sens donner à l'égalité `a == b` ?

Rappel et égalité triviale

Un variable est une étiquette.

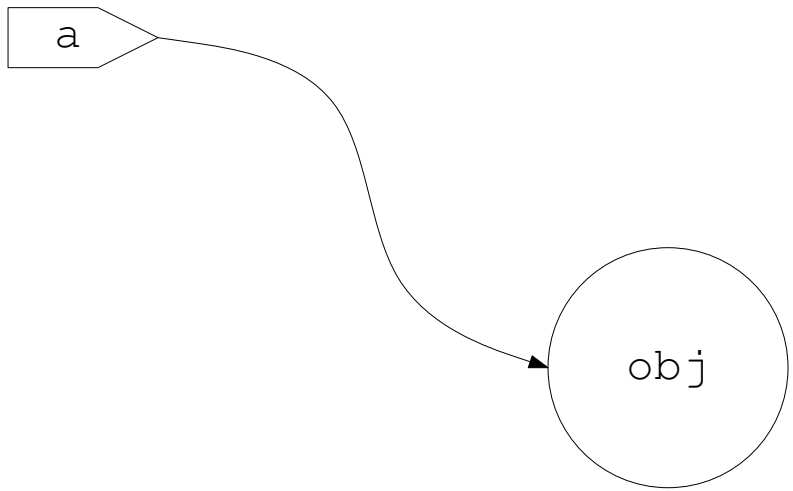
```
a = obj
```



Rappel et égalité triviale

Un variable est une étiquette.

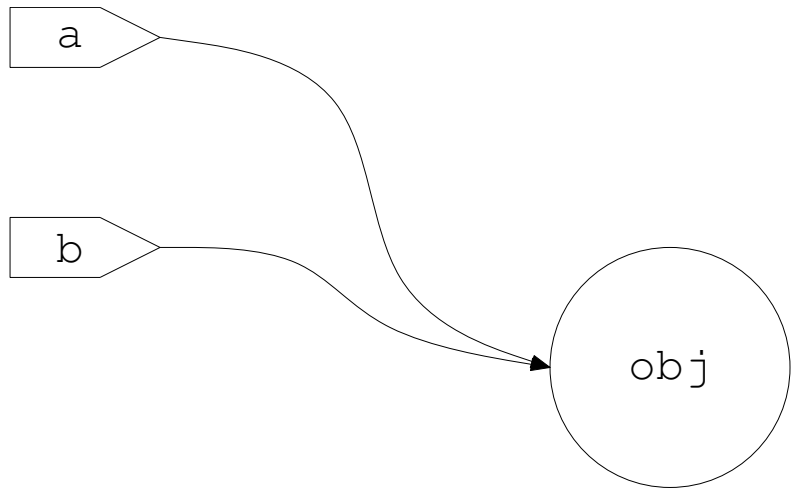
```
a = obj
```



L'égalité triviale de deux objets est l'égalité de leurs positions dans la mémoire.

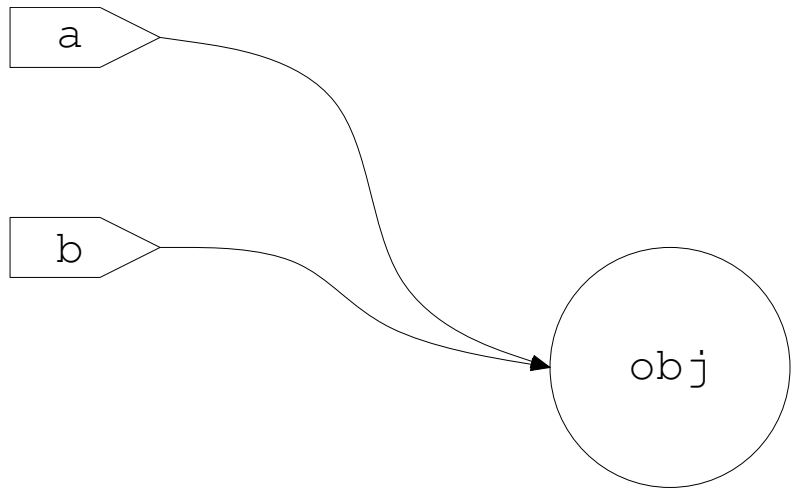
Schéma

La variable `a` est trivialement égale à ma variable `b` si et seulement si



Schéma

La variable `a` est trivialement égale à ma variable `b` si et seulement si



- Dans le cas général, l'opérateur == teste l'égalité triviale.
- Dans tous les cas, l'opérateur `is` teste l'égalité triviale.

Remarques

- L'égalité trivial implique l'égalité structurelle.
- On peut connaître la position d'un objet grâce à l'opérateur `id`

Mutabilité

Un objet est mutable si son contenu peut être modifié sans changer de position dans la mémoire.

Intérêts

- Écriture naturelle des algorithmes
- Permet de modifier les paramètres d'une fonction

Mutabilité

Un objet est mutable si son contenu peut être modifié sans changer de position dans la mémoire.

Intérêts

- Écriture naturelle des algorithmes
- Permet de modifier les paramètres d'une fonction

Désavantage

Moins de contrôle sur les objets (augmente le risque de bugs)

Exemple

Dans le programme suivant, si `obj` est mutable

```
a = obj
f(a)
```

on ne peut pas savoir si `a` a structurellement changé pendant l'appel de `f`.

Plan

Données non structurées

- Il s'agit des données de type entier et réel
- Ces données sont immuables (cf. exercices 1 à 3 du TD1).

Plan

Structures tabulaires

Type str (string)

- De la forme `s = 'bonjour'`.
- Immuable
- On accède au *i*-ème élément avec `s[i-1]`

Structures tabulaires

Type str (string)

- De la forme `s = 'bonjour'`.
- Immuable
- On accède au *i*-ème élément avec `s[i-1]`

Type list

- De la forme `s = [1, 42, 'abc', ...]`
- Mutable
- On accède au *i*-ème élément avec `s[i-1]`

Structures tabulaires

Type str (string)

- De la forme `s = 'bonjour'`.
- Immuable
- On accède au *i*-ème élément avec `s[i-1]`

Type list

- De la forme `s = [1, 42, 'abc', ...]`
- Mutable
- On accède au *i*-ème élément avec `s[i-1]`

Type tuple

- De la forme `s = (1, 42, 'abc', ...)`
- Immuable
- On accède au *i*-ème élément avec `s[i-1]`

Autres Structures

Type set

- De la forme `{obj1, obj2, obj3, ...}`.
- Mutable

Autres Structures

Type set

- De la forme `{obj1, obj2, obj3, ...}`.
- Mutable

Type dict (dictionnaire)

- De la forme `s = {key1 : value1, key2 : value2, ...}`.
- Mutable
- On accède à $\texttt{value}_i$ avec $\texttt{s}[\texttt{key}_i]$

Fonctions utiles

Fonctions génériques

- `len` donne le nombre d'élément de la structure
- `+` concatène deux structures de même type.
- `in` test l'appartenance d'un objet à la structure

Fonctions utiles

Fonctions génériques

- `len` donne le nombre d'élément de la structure
- `+` concatène deux structures de même type.
- `in` test l'appartenance d'un objet à la structure

Pour des fonctions plus particulières, on se reportera à
`https://docs.python.org/2/library/index.html`

Plan

Persistance

Comment pérenniser les données une fois que le programme termine ?

Persistance

Comment pérenniser les données une fois que le programme termine ?

Ouverture des fichiers

- Ouverture du fichier en lecture :

```
fichier = open('nomfichier', 'rb')
```

- Ouverture du fichier en écriture (efface un éventuel fichier déjà présent) :

```
fichier = open('nomfichier', 'wb')
```

Persistance

Opérations sur les fichiers

- Lecture sur fichier ouvert en lecture :

```
txt = fichier.read()
txt = fichier.read(42)
txt = fichier.readline()
liste_txt = fichier.readlines()
```

- Écriture sur un fichier ouvert en écriture

```
fichier.write(txt)
```