TDLOG : compléments sur Python & interfaces graphiques

Xavier Clerc - xavier.clerc@enseignants.enpc.fr
14 octobre 2015

Plan

- Assertions & exceptions
- Ressources
- Modules
- Collections
- Interfaces graphiques

Assertions & exceptions

- Assertions
  - erreurs irrécupérables
  - *p. ex.* violation de contrat
- Exceptions
  - erreurs récupérable
  - *p. ex.* connexion réseau impossible

Assertions

- Vérifier une propriété à l'exécution
- Formes **`assert cond`** et **`assert cond, message`**
  - ```
    while n > 0:
        assert (res >= 0), "res doit être positif"
        res *= n
        n -= 1
    ```
- Programmation par contrat
  - ```
    def f(n):
        assert (n >= 0)
        ...
        assert (res >= 0)
        return res
    ```

Assertions

- Aide à la mise au point
- Question du surcoût à l'exécution
- Désactivables (option `-o`)

Exceptions

- Définition
- Lancement
- Traitement
- Avantages & inconvénients

Quelques exceptions prédéfinies

- **`Exception`**
  - **`ArithmeticError`**
    - **`ZeroDivisionError`**
  - **`LookupError`**
    - **`IndexError`**
    - **`KeyError`**
  - **`IOError`**

Définition

- ```
  class MonException(Exception): pass
  ```

- ```
  class MonException(Exception):
    def __init__(self, val):
      self.valeur = val
    def get_valeur(self):
      return self.valeur
    def __str__(self):
      return "Mon exception ({})".format(
        self.valeur)
  ```

Lancement

- `raise MonException`

- `raise MonException(p1, ..., pn)`

- Appel du constructeur de la classe

- Lancement => recherche du bloc de traitement

Traitement

- ```
  try:
      ...
  except:
      ...
  ```
  bloc protégé

  bloc de traitement

- ```
  try:
      ...
  except E1:
      ...
  except E2 as e:
      ...
  ```
  discrimination de l'exception

  accès à l'instance

Traitement

- **`try:`**

```
...
...
```

bloc protégé

`except:`

```
...
...
```

bloc de traitement

`else:`

```
...
...
```

exécuté en l'absence d'exception

`finally:`

```
...
...
```

toujours exécuté

Dynamique

```
def f(x, y):
  return x // y

def g(msg, x, y):
  try:
    print(msg, f(x, y))
  except KeyError:
    print("key error")
  except:
    print("any error")

g("resultat :", 3, 0)
```

appel de **`f`** avec **`3`** et **`0`**

levé de **`ZeroDivisionError`**

propagation par l'exception à l'appelant

pas une **`KeyError`**

`except:` attrape toutes les exceptions

exécution de **`print(...)`**

exécution de **`g(...)`**

fin de l'exécution

Traceback

- Affiché lorsqu'un exception s'*échappe*

- Chemin jusqu'au point de levée

```
Traceback (most recent call last):
  File <stdin>, line 1 in <module>
  File <stdin>, line 3 in fonct1
  File <stdin>, line 8 in fonct2
__main__.MonException: message exception
```

Avantages & inconvénients

- Code d'erreur ou exception ?
  - code d'erreur nécessite **`if`** à chaque appel
  - exceptions plus flexibles & séparation calcul/erreurs
- Limites
  - traitements pour un bloc
  - *distance* entre levée et traitement
  - code interrompu dans un état incohérent

Ressources

- Nettoyage d'un traitement interrompu

- Alternative à **`try`**/**`except`**/**`finally`**

- Fondé sur la notion de *parenthèsage*

Utilisation

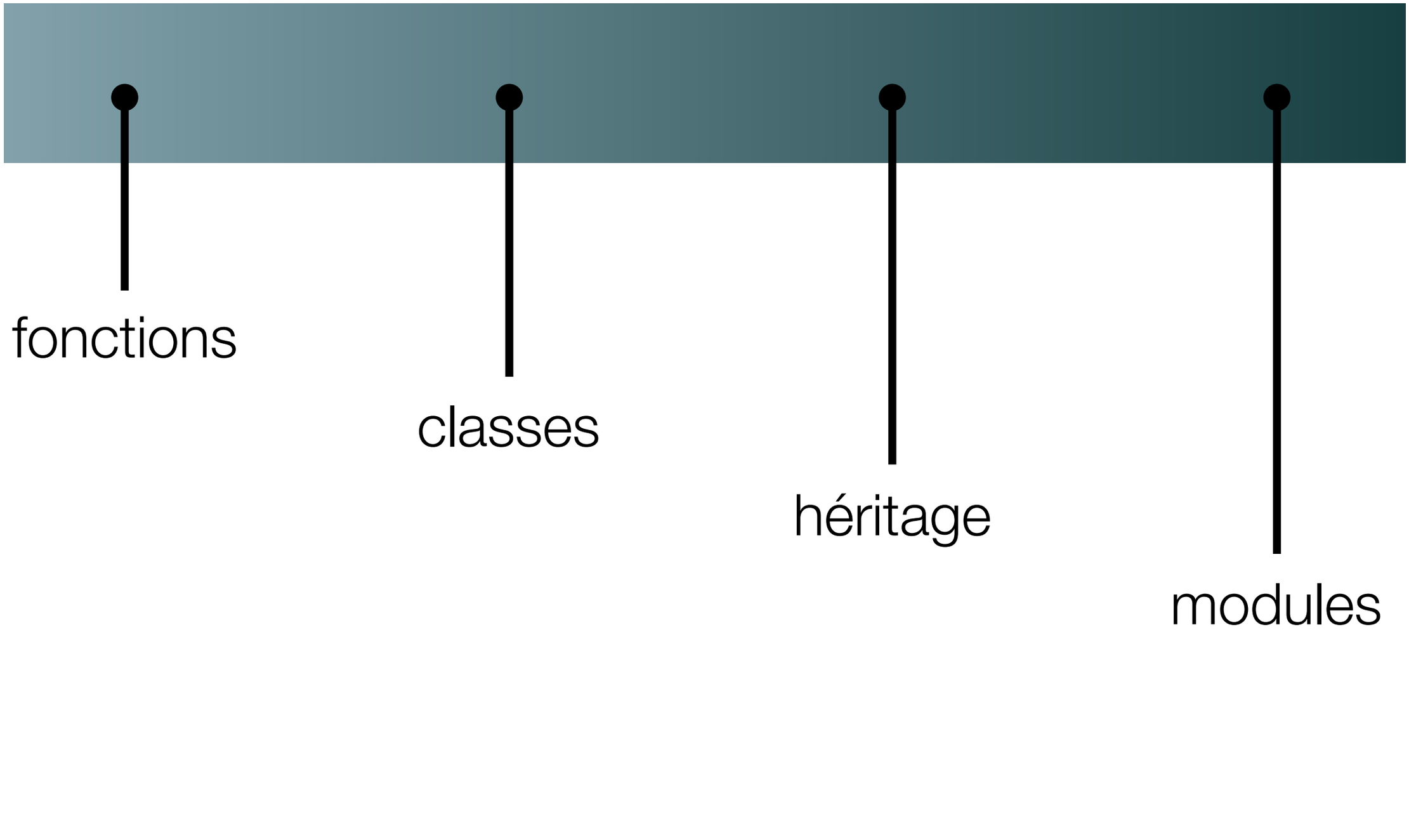
- **use e0 as id0, ..., en as idn:**
  ...

- **with open("nom_fichier", "r") as fichier:**
  for ligne in fichier:
  print(ligne.rstrip("\n").upper())

- en Python 2.*x* :
  from __future__ import with_statement

programming in the small

programming in the large



fonctions

classes

héritage

modules

Modules

- Regrouper
  - variables
  - fonctions
  - classes
- en unités de programmation

Modules

- Lisibilité
- Maintenance
- Performances

Utilisation

- `import module`
  - `module.element`
- `import module as m`
  - `m.element`
- `import module1, ..., module2`
- `parent.sousmodule`

Utilisation

- `from module import element`
  - `element`
- `from module import element as e`
  - `e`
- `from module import *`

Définition

- Module : fichier **`.py`**

- Hiérarchie : système de fichiers

- Variable **`sys.path`** (appeler **`.append("path")`**)

  - contient initialement **`"."`**

Quelques modules usuels

- **`sys`** : contenu de la ligne de commande, infos système
- **`os`** : appels systèmes
- **`math`** & **`cmath`** : fonctions mathématiques
- **`random`** : générateurs pseudo-aléatoires
- **`io`** : entrées/sorties

Collections

Caractéristiques	Listes	Ensembles
Ordre ?	ordre des indices	non spécifié
Unicité ?	non	oui
Mutabilité ?	oui	oui si **`set`** non si **`frozenset`**
Accès ?	par indice	par test de présence
Parcours ?	indice parcours séquentiel	parcours (ordre non spécifié)

Collections

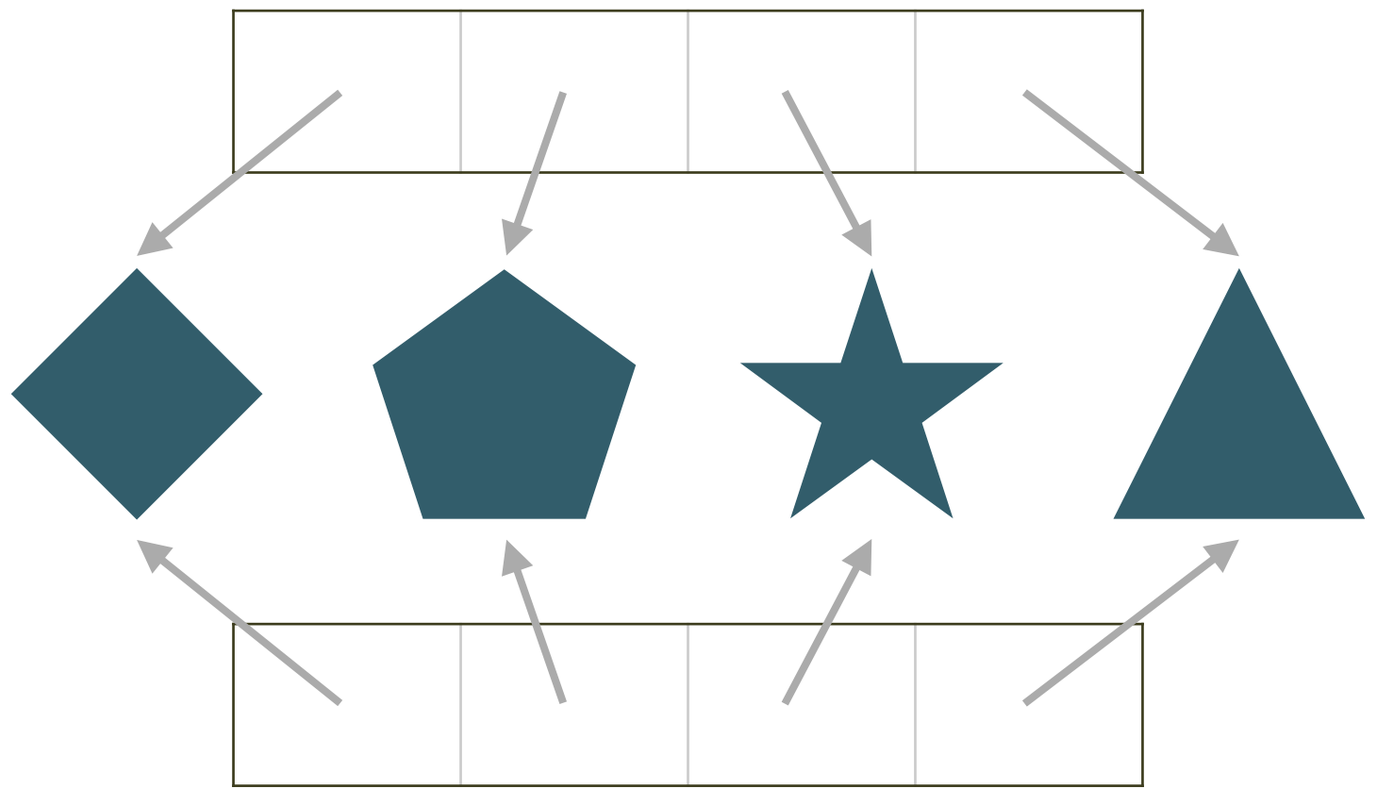
- Copies superficielles

- Copies profondes

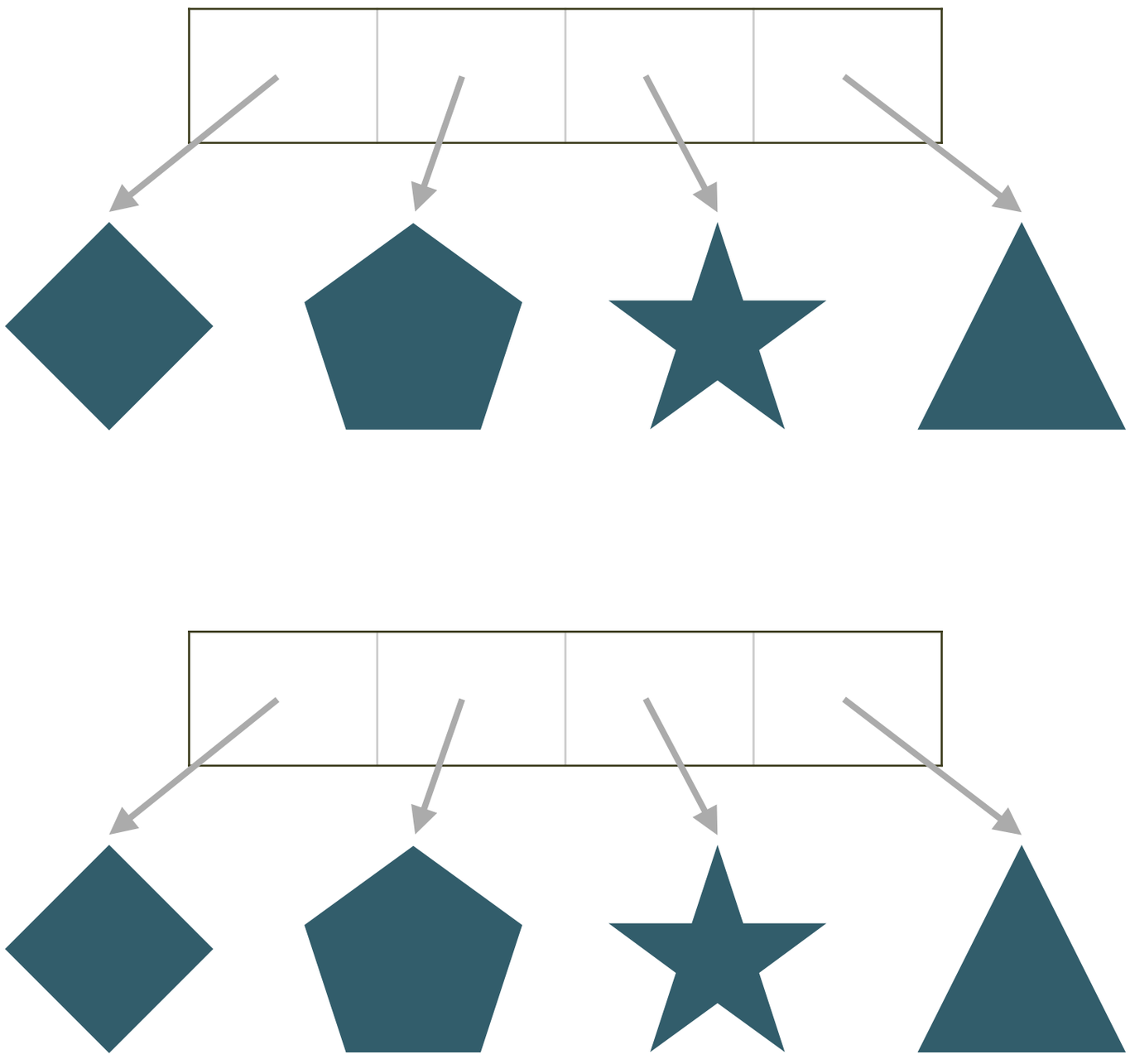
- Égalité structurelle `==`

- Égalité physique `is`

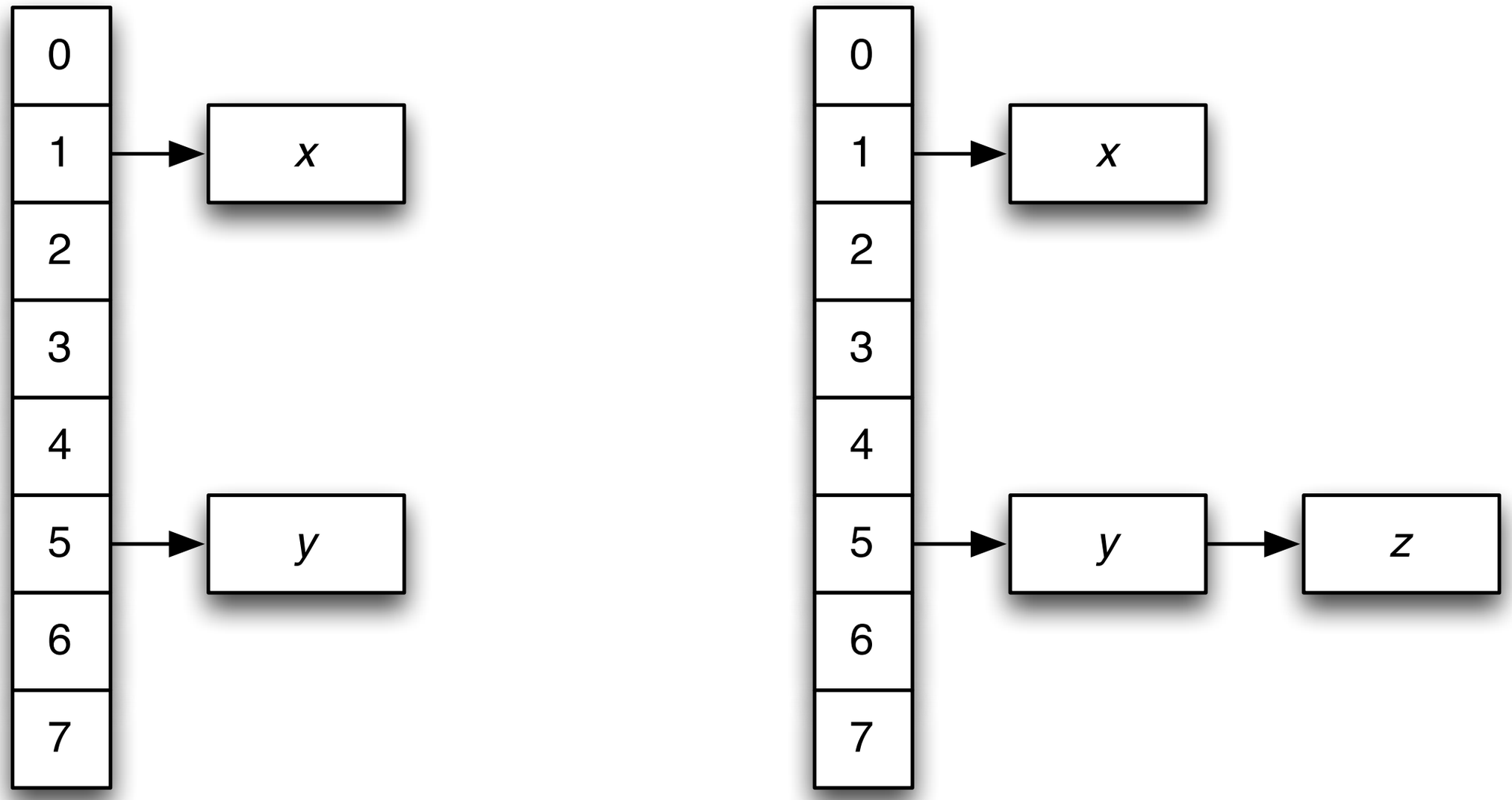
Copie superficielle



Copie profonde



Ensembles en Python



Fonction de hachage

- **Correcte :**

$$\forall x \forall y, x = y \Rightarrow hash(x) = hash(y)$$

$$\forall x \forall y, \texttt{x == y} \Rightarrow \texttt{x.\_\_hash\_\_() == y.\_\_hash\_\_()}$$

- **Efficace :**
  - minimise les collisions
  - répartit sur les entiers
- `def __hash__(self): return 0`

Interfaces graphiques

- Depuis les années 70...
- Modèle *WIMP* (*Windows*, *Icon*, *Menus*, *Pointer*)
- Modèle post-*WIMP*
- Par bibliothèque, *framework*, *toolkit*
- Programmation événementielle

Concepts de base

- Composant, *component*, *control*, *widget*
- Organisation arborescente
- *Layout*
- Événement
- Boucle d'événement
- *Callback*

Librairie Qt

- *cute* ou *cutie*

- *Cross-platform* (Android, iOS, Linux, OS X, Windows, *etc*.)

- Développé en/pour C++, nombreux *bindings*

- Notions de *slots* & *signaux*

- Sous licence GPL / LGPL

Bindings PyQt

- Module PyQt4
- Boucle d'événements explicite

```
import sys
from PyQt4 import QtGui
application = QtGui.QApplication(sys.argv)
...
sys.exit(application.exec_())
```

Widgets de base

- **`QMainWindow`**
- **`QPushButton`**
- **`QListWidget`**
- **`QTextEdit`**
- **`QFileDialog`**
- …

Exemple

```
import sys
from PyQt4 import QtGui
```

```
class Fenetre(QTGui.QMainWindow):
    def __init__(self):
        super().__init__(None)
        ...
        self.setGeometry(200, 100, 600, 400)
```

```
application = QtGui.QApplication(sys.argv)
fenetre = Fenetre()
fenetre.show()
fenetre.raise_()
sys.exit(application.exec_())
```

Signaux

- **`self.bouton.`***signal*`.connect(`***callback***`)`**
- **`self.bouton.clicked.connect(self.meth)`**
- **`def meth(self):`**

  `...`

- Boutons : **`clicked`**, **`pressed`**, **`released`**, *etc*.
- Zone de texte : **`textChanged`**, *etc*.
- Cases à cocher : **`stateChanged`**, *etc*.

Layouts

- `QHBoxLayout`, `QVBoxLayout`, `QGridLayout`

- ```
  central = QWidget()
  horizontal = QHBoxLayout()
  ```

- ```
  horizontal.addWidget(...)
  ...
  horizontal.addWidget(...)
  ```

- ```
  central.setLayout(horizontal)
  self.setCentralWidget(central)
  ```
  ```

QtDesigner

- Création graphique de l'interface
- Enregistrement sous un fichier **`.ui`**
- Génération de code depuis fichier **`.ui`**

Boîte de widget

Filtre

Layouts
- Vertical Layout
- Horizontal Layout
- Grid Layout
- Form Layout

Spacers
- Horizontal Spacer
- Vertical Spacer

Buttons
- Push Button
- Tool Button
- Radio Button
- Check Box
- Command Link Button
- Button Box

Item Views (Model-Based)
- List View
- Tree View
- Table View
- Column View

Item Widgets (Item-Based)
- List Widget
- Tree Widget
- Table Widget

Containers
- Group Box
- Scroll Area

Éditeur de signaux et slots

Émetteur	Signal	Receveur	Slot

MainWindow – test.ui

Menu1 Menu2 Taper ici

RadioButton
RadioButton
RadioButton
CheckBox
0
01/01/2000
PushButton

Éditeur d'actions

Filtre

Nom	Utilisé	Texte	Raccourci

Éditeur de propriétés

Filtre

MainWindow : QMainWindow

Propriété	Valeur
QObject	
objectName	MainWindow
QWidget	
enabled	☑
geometry	[(0, 0), 619 x 505]
sizePolicy	[Preferred, Preferred, 0, 0]
minimumSize	0 x 0
maximumSize	16777215 x 16777215
sizeIncrement	0 x 0
baseSize	0 x 0
palette	Héritée
font	A [.Lucida Grande UI, 13]
cursor	Flèche
mouseTracking	☐
focusPolicy	NoFocus
contextMenuPolicy	DefaultContextMenu
acceptDrops	☐
windowTitle	MainWindow
windowIcon	
windowOpacity	1.000000
toolTip	
statusTip	
whatsThis	
accessibleName	
accessibleDescri	

Explorateur de ressources

Filtre

<resource root>

```
class Ui_MainWindow(object):
    def setupUi(self, MainWindow):
        MainWindow.setObjectName(_fromUtf8("MainWindow"))
        MainWindow.resize(619, 505)
        self.centralwidget = QtGui.QWidget(MainWindow)
        self.centralwidget.setObjectName(_fromUtf8("centralwidget"))
        self.radioButton = QtGui.QRadioButton(self.centralwidget)
        self.radioButton.setGeometry(QtCore.QRect(20, 20, 102, 20))
        self.radioButton.setObjectName(_fromUtf8("radioButton"))
        self.radioButton_2 = QtGui.QRadioButton(self.centralwidget)
        self.radioButton_2.setGeometry(QtCore.QRect(20, 50, 102, 20))
        self.radioButton_2.setObjectName(_fromUtf8("radioButton_2"))
        self.radioButton_3 = QtGui.QRadioButton(self.centralwidget)
        self.radioButton_3.setGeometry(QtCore.QRect(20, 80, 102, 20))
        self.radioButton_3.setObjectName(_fromUtf8("radioButton_3"))
        self.checkBox = QtGui.QCheckBox(self.centralwidget)
        self.checkBox.setGeometry(QtCore.QRect(20, 120, 87, 20))
        self.checkBox.setObjectName(_fromUtf8("checkBox"))
        self.lineEdit = QtGui.QLineEdit(self.centralwidget)
        self.lineEdit.setGeometry(QtCore.QRect(230, 20, 113, 21))
        self.lineEdit.setObjectName(_fromUtf8("lineEdit"))
        ...
        self.pushButton = QtGui.QPushButton(self.centralwidget)
        self.pushButton.setGeometry(QtCore.QRect(230, 140, 114, 32))
        self.pushButton.setObjectName(_fromUtf8("pushButton"))
        MainWindow.setCentralWidget(self.centralwidget)
        self.menubar = QtGui.QMenuBar(MainWindow)
        self.menubar.setGeometry(QtCore.QRect(0, 0, 619, 22))
        self.menubar.setObjectName(_fromUtf8("menubar"))
        self.menuMenu1 = QtGui.QMenu(self.menubar)
        self.menuMenu1.setObjectName(_fromUtf8("menuMenu1"))
        self.menuMenu2 = QtGui.QMenu(self.menubar)
        self.menuMenu2.setObjectName(_fromUtf8("menuMenu2"))
        MainWindow.setMenuBar(self.menubar)
        self.statusbar = QtGui.QStatusBar(MainWindow)
        self.statusbar.setObjectName(_fromUtf8("statusbar"))
        MainWindow.setStatusBar(self.statusbar)
        self.menubar.addAction(self.menuMenu1.menuAction())
        self.menubar.addAction(self.menuMenu2.menuAction())
```

Questions ?