InitProg Python par l'exemple

Martin Monperrus

26 octobre 2012

Ce document illustre le polycopié du cours *Initiation à la programmation (InitProg)* de l'Université Lille 1 avec quelques programmes Python simples. Il est destiné à la compréhension des bases.

Table des matières

1 init_tas et deplacer_sommet

```
# {{init_tas et deplacer_sommet}}
# coding: utf-8
# Utilisation:
# - init_tas
# - deplacer_sommet

# Ce programme initialise le tas 1 avec une alternance de Pique et de Coeur
# sur le tas 1 et rien sur les autres tas, puis trie toutes les cartes du tas 1 par couleur
# sur les tas 2 (qui reçoit les Piques) et 3 (qui reçoit les Coeurs).


# Rappel: ceci est un commentaire, que ocaml et ocamlcartes ignorent.
# Les commentaires vous aident à structurer et comprendre votre propre programme,
# et, pour les professeurs, à corriger vos travaux.

# Nous utilisons le module Cartes
from Cartes import *

# Initialisation du tas 1 avec 4 cartes. Il y a une alternance de Pique et Coeur.
# La fonction init_tas prend deux paramètres:
#  - le premier est un numéro de tas entre 1 et 4
#  - le deuxième est une chaîne de caractères décrivant le tas

# C -> Coeur
# K -> Carreau
# T -> Trèfle
# P -> Pique
# La carte du dessous correspond au premier caractère:
# init_tas(1,"CPT")
#            |_La carte du dessous est un coeur
# Les numéros de carte sont tirés au hasard.

init_tas(1,"CPCP")

# Les autres tas sont vides. Un tas vide est représenté par une chaîne de caractères vide notée en Ocaml ""
init_tas(2,"")
init_tas(3,"")
init_tas(4,"")

# Nous savons que la carte du dessus est un Pique, donc va sur le tas 2
deplacer_sommet(1,2)

# La suivante est un coeur
deplacer_sommet(1,3)

# La suivante est un pique
deplacer_sommet(1,2)

# La dernière est un coeur
deplacer_sommet(1,3)

# Toutes les cartes sont triées, le programme est terminé!
pause("fini")
```

2 Le if/then et if/then/else

```
#  {{Le if/then et if/then/else}}
# coding: utf-8
# Utilisation:
# - le + et () dans init_tas
# - if/then/else

# Ce programme initialise les tas 1 avec des valeurs aléatoires.  Les autres tas sont vides.
# Ensuite, dépendamment de l'initialisation du tas 1, les coeurs partent
# sur le tas 2, les trèfles sur le tas 3 et les piques sur le tas 4

from Cartes import *

#  Initialisation du tas 1:
#- le + veut dire "ou": (C+P) donne un coeur ou un pique aléatoirement
#- les parenthèses permettent de délimiter des groupes
# Ici, les tas possibles sont:
#  T
#  CP
#  PP
#
init_tas(1,"((C+P)P)+T")

# Les autres tas sont vides. Un tas vide est représenté par
# une chaîne de caractères vide notée en Ocaml ""
init_tas(2,"")
init_tas(3,"")
init_tas(4,"")

#  Déplacements des cartes

if sommet_trefle(1):
  deplacer_sommet(1,3)
else:

  #  nous déplacons le pique du sommet
  deplacer_sommet(1,4)

  #  sous le pique, avons-nous un coeur ou un autre pique ?
  if sommet_coeur(1):
    deplacer_sommet(1,2)
  else:
    #  c'est nécessairement un pique (d'après le init_tas)
    deplacer_sommet(1,4)


pause("fini")
```

3 Algèbre booléenne

```
# coding: utf-8
#  {{Algèbre booléenne}}
# Utilisation:
#  - la fonction tas_vide, tas_non_vide
#  - l'algèbre booléenne not && ||

# Ce programme initialise les tas 1 avec des valeurs aléatoires.
# Les autres tas sont vides.

# Ensuite, dépendamment de l'initialisation du tas 1, les rouges partent
# sur le tas 2, les noires sur le tas

from Cartes import *

#  Initialisation du tas 1
#  Le tas peut être vide à cause du ()
init_tas(1,"((P+T)+(C+K))+()")

#  Les autres tas sont vides. Un tas vide est représenté par
#  une chaîne de caractères vide notée en Ocaml ""
init_tas(2,"")
init_tas(3,"")
init_tas(4,"")


#  Si le tas 1 n'est pas vide
#  NB: faire un sommet_coeur, sommet_carreau, sommet_pique
#  ou sommet_trefle sur un tas vide casse l'automate
if not(tas_vide(1)):

  #  si carte est rouge
  if sommet_coeur(1) or sommet_carreau(1):
    deplacer_sommet(1,2)

  #  si la carte est noire
  #  nous avons peut-être déplacé une carte, donc on rajoute un test
  #  notez que tas_vide(x) = not(tas_non_vide(x))
  if tas_non_vide(1) and (sommet_pique(1) or sommet_trefle(1)):
    deplacer_sommet(1,3)

pause("fini")
```

4 La boucle while

```
#  {{La boucle while}}
# coding: utf-8
# Utilisation:
# - le [] dans init_tas
# - la boucle while

# Ce programme initialise les tas 1 et 2 avec n coeurs ou trèfles aléatoirement.
# Puis déplace tous les coeurs du tas 1 sur le tas 4 et tous les coeurs du sommet du tas 2 sur le tas 4.
# Le tas 3 est utilisé comme entrepôt

from Cartes import *

#  les crochets veulent dire n au hasard dans [0,52]
# NB: l'expression entre crochet est exécutée n fois, donc les tas 1 et 2
# contiennent une alternance aléatoire de coeur et trèfle
init_tas(1,"[C+T]")
init_tas(2,"[C+T]")

#  Les autres tas sont vides.
init_tas(3,"")
init_tas(4,"")


#  déplace tous les coeurs du tas 1 sur le tas 4
while tas_non_vide(1):
  if sommet_coeur(1):
    deplacer_sommet(1,4)
  else:
    deplacer_sommet(1,3) #  dans l'entrepôt


#  tous les coeurs du sommet du tas 2 sur le tas 4.
#  Le tas 2 peut être vide si [] donne n=0 ou si il n'y a que des coeurs sur le tas 2
# et qu'ils ont tous été déplacés
while tas_non_vide(2) and sommet_coeur(2):
  deplacer_sommet(2,4)

pause("fini")
```

5 Définition d'une fonction

```
# coding: utf-8
#  {{Définition d'une fonction}}
# Utilisation:
#  - définition de fonctions

# Ce programme déplace toutes les cartes des tas 1, 2 et 3 sur le tas 4

from Cartes import *

# Initialisation des tas: tous les tas possibles
init_tas(1,"[P+C+K+T]")
init_tas(2,"[P+C+K+T]")
init_tas(3,"[P+C+K+T]")
#  Le tas 4 est vide
init_tas(4,"")

# Déplace toutes les cartes du tas x vers le tas y.
# Types: x et y sont des numéros de tas
# Contrainte d'utilisaton: x et y dans {1,2,3,4}
def deplacer_tas(x,y):
  while tas_non_vide(x):
    deplacer_sommet(x,y)

#  déplacement des tas en utilisant la fonction deplacer_tas
deplacer_tas(1,4)
deplacer_tas(2,4)
deplacer_tas(3,4)

pause("fini")
```

6 Appels de fonction

```
# coding: utf-8
#  {{Appels de fonction}}
# Ce programme trie toutes les cartes de tous les tas. Les rouges finissent sur le tas 3 et les noires sur le tas 4.

# La fonction "trie_tous_tas" appelle la fonction "trie_tas_par_couleur"
# et la fonction "trie_tas_par_couleur" appelle la fonction "bouge_noir_sur"
# et la fonction "bouge_rouge_sur"
# Comme au légo, on commence par les plus petites briques:
# l'ordre de définition des fonctions commence par les "petites" fonctions qui sont appelées par d'autres .

from Cartes import *

# Initialisation des tas: tous les tas possibles
init_tas(1,"[P+C+K+T]")
init_tas(2,"[P+C+K+T]")
init_tas(3,"[P+C+K+T]")
#  Le tas 4 est vide
init_tas(4,"")

# déplace la carte du sommet du tas x sur le tas y
# si et seulement si c'est une carte rouge
def bouge_rouge_sur(x,y):
  if tas_non_vide(x):
    if sommet_coeur(x) or sommet_carreau(x):
      deplacer_sommet(x,y)

# déplace la carte du sommet du tas x sur le tas y
# si et seulement si c'est une carte noire
def bouge_noir_sur(x,y):
  if tas_non_vide(x):
    if sommet_pique(x) or sommet_trefle(x):
      deplacer_sommet(x,y)

# trie les cartes du tas x en mettant toutes les cartes rouges sur y et les cartes noires sur z
def trie_tas_par_couleur(x,y,z):
  while tas_non_vide(x):
    bouge_rouge_sur(x,y)
    bouge_noir_sur(x,z)

# trie toutes les cartes de tous les tas en mettant toutes les cartes rouges sur y et les cartes noires sur z
def trie_tous_tas(y,z):
  #  on libère les tas de destination
  trie_tas_par_couleur(y,1,2)
  trie_tas_par_couleur(z,1,2);
  #  et on finit le tri
  trie_tas_par_couleur(1,y,z)
  trie_tas_par_couleur(2,y,z)

trie_tous_tas(3,4)
pause("fini")
```