TP de programmation en langage Python

9 octobre 2013

1 Pour apprendre le langage Python

Il existe bon nombre de sites qui présentent une initiation au langage Python. Nous en citons ici quelques-uns :

1. Les docs de référence : `http://docs.python.org/2/index.html`. Pour répondre aux exercices qui suivent, vous pourrez chercher les éléments utiles sur ce site.
2. Le site ioi-france : `http://www.france-ioi.org/`.
3. Un site permettant de tester votre code en ligne (on peut notamment modifier une partie du code proposé dans le cours et visualiser l'effet pour mieux comprendre) : `http://interactivepython.org/courselib/static/thinkcspy/index.html`.
4. Un autre site interactif : `http://cscircles.cemc.uwaterloo.ca/`.
5. Le site du zéro (openclassrooms) : `http://fr.openclassrooms.com/informatique/cours/apprenez-a-programmer-en-python`

2 Quelques exemples.

L'objectif de ce paragraphe est de vous donner l'essentiel des éléments de syntaxe du langage par le biais de quelques exemples.

2.1 Boucle for et test conditionnel if.

La population d'une ville est initialement, en l'an 2000, de 3 000 habitants. Chaque année, la population connaît une croissance de 1% ($p_n = 1{,}01 p_{n-1}$).
Écrire une fonction python :

Entrée l'année n

Sortie l'effectif de population de cette année n.

Programme 1 – Python 2.7

```
# -*- coding: utf-8 -*-

def p(n):
    """ n est une année >= 2000.
    p(n) est l'effectif de population cette année n.
    Cette version est itérative."""
    p=3000
    if n<2000:
        p="donnée non connue."
    else:
        for j in range(2000,n):
            p*=1.01
    return p


```

```
print p(2000)
print p(2001)
print p(1998)
```

Quelques remarques :

1. Il n'y a pas de begin et end ou d'accolades pour marquer le début et la fin d'un bloc : c'est l'indentation qui joue ce rôle.
2. On remarquera que la variable p peut désigner dans la même fonction une chaîne de caractères ou un flottant (ce qui serait interdit dans un langage comme java). Bien entendu, ce changement de nature de la valeur retournée est fortement déconseillée dans un programme un plus ambitieux (que se passerait-il si un utilisateur non averti voulait calculer à partir de la valeur de retour...)
3. Le descriptif de la fonction placé en début de fonction entre """ et """ sert à documenter cette fonction. L'instruction print p.__doc__ affichera ce texte. Vous pouvez de cette façon afficher une aide pour les fonctions python prédéfinies ou les fonctions de la plupart des modules.

2.2 Récursivité.

Écrire une version récursive de la fonction précédente en supposant que l'utilisateur entre un entier $n \geqslant 2000$.

Programme 2 – Python 2.7

```
def pp(n) :
    if n==2000 : return 3000
    return pp(n-1)*1.01


annee=raw_input("Entrez une année >= 2000 :")
print pp(int(annee))
```

Quelques remarques :

1. L'entier donné par l'utilisateur sera transformé en chaîne de caractères par raw_ input, ce qui explique qu'on le retransforme en int (type entier) lors de l'appel de la fonction.
2. L'affectation se fait avec le symbole =, le test d'égalité avec ==.
3. Nous n'avons pas utilisé else pour le cas $n > 2000$: la ligne return pp(n−1)*1.01 ne sera exécutée que si la première ligne ne l'est pas. L'instruction « return » met en effet fin à l'exécution de la fonction, tout ce qui se trouve après l'exécution d'un return n'est donc pas exécuté.

2.3 Boucle while

Écrire une fonction python :

Entrée un nombre $M > 0$.

Sortie l'année n pour laquelle l'effectif de la population dépasse pour la première fois M.

On utilise la fonction p précédemment définie :

Programme 3 – Python 2.7

```
def depasse(M) :
    """retourne l'année n pour laquelle, pour la
    première fois, on a p(n)>=M."""
    n=2000
    while p(n)<M:
        n+=1
    return n

print depasse(4000)
```

2.4 Calculs sur les entiers

Écrire une fonction de calcul du pgcd de deux entiers naturels (algorithme d'Euclide).

Programme 4 – Python 2.7

```
# -*- coding : utf-8 -*-

def pgcd(a,b) :
    if b==0 : return a
    return pgcd(b,a%b)

print pgcd(13*12,13*7*2)
```

1. a% b retourne le reste de la division euclidienne de a par b.
2. Si a et b sont des entiers, a/b renvoie le quotient entier de a par b en python 2 mais renvoie une approximation du nombre $\frac{2}{3}$ en python 3. En python 3, on obtiendra un quotient euclidien par a//b.

2.5 Liste.

Ci-dessous une fonction (peu efficace, et même pas très propre mais l'objectif est ici d'utiliser quelques mots du langage) de tri d'une liste :

Programme 5 – Python 2.7

```
# -*- coding : utf-8 -*-

def lePlusPetit(L) :
    """ retourne l'indice d'un plus petit élément de la liste L,
    on suppose que la liste L ne contient que des nombres."""
    m=L[0] # m=premier élément de la liste
    ind=0 # ind : variable qui contiendra l'indice du min
    for j,x in enumerate(L) : # j,x parcourt les couples (indice, L[indice])
        if x<m :
            ind,m=j,x # équivalent ici à ind=j suivi de m=x
    return ind

def triNaif(L) :
    """retourne une liste LL contenant les mêmes éléments que L mais
    en ordre croissant. A la fin de la fonction, la liste L sera vide."""
    LL=[] # LL : liste vide
    longueurListe=len(L)
    for j in range(longueurListe) :
        k=lePlusPetit(L)
        LL.append(L[k])# ajoute en fin de liste LL l'élément L[k]
        L.pop(k) # supprime l'élément d'indice k de L
    return LL

L=[3,4,5,1,2,1]
print  triNaif(L)
```

Quelques remarques :

1. L'instruction range(n) est équivalente à range(0,n) et peut être considérée comme retournant la liste des entiers de 0 à $n-1$.
2. Après exécution de la fonction, si vous demandez un affichage de la liste L, vous constaterez qu'elle a été vidée également au niveau global : une liste est passée par référence dans les paramètres de fonctions.

3 Exercices.

Les corrigés des exercices se trouvent en fin de fichier : cela vous permettra d'aller chercher quelques éléments de syntaxe qui vous manqueraient.

3.1 Prise en main

***Exercice* 1.** Écrire un programme qui affiche «Bonjour le monde ».

***Exercice* 2.** Écrire un programme qui demande la saisie du nom de l'utilisateur et qui renvoie «Bonjour », suivi de ce nom.

***Exercice* 3.** Écrire un programme qui demande à l'utilisateur la saisie de deux entiers positifs a et b (b non nul) et qui affiche le quotient et le reste de la division euclidienne de a par b sans utiliser les opérateurs // et % (utiliser soustraction, addition, boucle).

***Exercice* 4.** Écrire un programme qui demande à l'utilisateur son année de naissance et qui affiche son âge.

***Exercice* 5** (donné en exercice aux élèves)**.** Écrire un programme qui convertisse un nombre entier de secondes fourni au départ, en un nombre d'années, de mois, de jours, de minutes et de secondes. (Utilisez l'opérateur modulo : %).

Les exercices ci-dessous devront être implantés sous forme de fonctions

3.2 Itération

***Exercice* 6.** Écrire un programme qui affiche la table de multiplication d'un entier saisi au clavier.

***Exercice* 7.** Écrire un programme qui calcule la somme des n premiers entiers :

$$1+2+...+n=\sum_{i=1}^{n} i$$

***Exercice* 8.** Écrire un programme qui calcule $n! = 1 \times 2 \times 3 \times \cdots \times n$.

***Exercice* 9.** Un nombre parfait est un nombre qui est égal à la somme de tous ses diviseurs excepté lui-même. Par exemple 6 est un nombre parfait car $6 = 1 + 2 + 3$.

1. Écrire un programme qui décide si un entier donné est parfait ou non.
2. Modifier le programme précédent pour qu'il donne tous les entiers parfaits inférieurs à un entier m saisi par l'utilisateur.

***Exercice* 10** (donné en exercice aux élèves)**.** Écrire un programme, en utilisant le principe des divisions en cascade :
– input : un entier b au moins égal à 2 (et au plus égal à 9 pour simplifier la tâche) et un entier n écrit en base 10.
– output : une écriture de n en base b.

3.3 Chaines de caractères

***Exercice* 11** (donné en exercice aux élèves)**.** Écrire un script qui renvoie le nombre de caractères « b » présents dans une chaîne donnée par l'utilisateur.

***Exercice* 12.** Écrire une fonction compteLettres(ch) prenant en paramètre une chaîne de caractères ch et retournant le nombre de lettres 'a', de lettres 'b', ... utilisées dans la chaîne.

***Exercice* 13.** Écrire une fonction :

Entrées : une chaîne de caractères ch et un caractère carac.

Sortie : la chaîne de départ avec le caractère carac intercalé entre chaque caractère de ch.

Par exemple, insereCarac('ISN', '–') retournera 'I–S–N'.

***Exercice* 14.** Écrire un script qui recopie une chaîne (dans une nouvelle variable) en l'inversant. Ainsi par exemple, «zorglub »deviendra «bulgroz ».

***Exercice* 15** (donné en exercice aux élèves)**.** Écrire un script qui détermine si une chaîne de caractères donnée est un palindrome (c'est-à-dire une chaîne qui peut se lire indifféremment dans les deux sens), comme par exemple «radar »ou «s.o.s ».

3.4 Listes

***Exercice* 16.** Soient les listes suivantes :
t1 = [31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31]
t2 = ['Janvier', 'Février', 'Mars', 'Avril', 'Mai', 'Juin', 'Juillet', 'Août', 'Septembre', 'Octobre', 'Novembre', 'Décembre']
Écrire un programme qui crée une nouvelle liste t3. Celle-ci devra contenir tous les éléments des deux listes en les alternant, de telle manière que chaque nom de mois soit suivi du nombre de jours correspondant.

***Exercice* 17** (donné en exercice aux élèves)**.** Écrire un programme qui recherche et affiche le plus grand élément présent dans une liste donnée.

***Exercice* 18** (donné en exercice aux élèves)**.** Écrire un programme qui analyse un par un tous les éléments d'une liste de mots (par exemple :['longuement', 'court', 'bref', 'expansif', 'short', 'very long']) pour générer deux nouvelles listes. L'une contiendra les mots comportant moins de 6 caractères, l'autre les mots comportant au moins 6 caractères.

***Exercice* 19.** Une fonction python peut prendre une autre fonction en paramètre.
Écrire une fonction extremeListe qui prend en paramètre une liste L et une fonction f. Suivant la fonction f passée en paramètre, extremeListe pourra retourner le maximum ou le minimum de la liste L.

***Exercice* 20** (l'un des algorithmes du programme)**.** Écrire une procédure de recherche par dichotomie d'un élément dans une liste.

3.5 Le module tortue.

Un module est une collection de fonctions. Pour utiliser les fonctions d'un module appelé *module*, on pourra écrire la ligne suivante au début du script :

Programme 6 – Python 2.7

```
1 from module import *
```

Si le module 'module' définit une fonction 'dessine()', on pourra l'utiliser dans le fichier.
Un inconvénient majeur de cette méthode : si l'on charge deux modules définissant chacun une fonction de même nom, on ne pourra utiliser que la fonction du module chargé en dernier.
Pour pallier à ce problème, on chargera un module plutôt de la façon suivante :

Programme 7 – Python 2.7

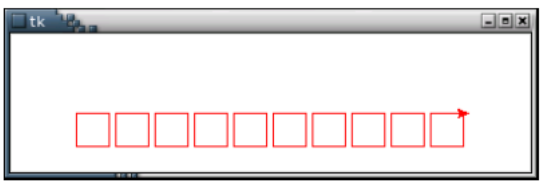
```
1 import module
```

Si la fonction 'dessine()' est définie dans 'module', on pourra maintenant l'utiliser par l'appel module.dessine().
Un inconvénient : lorsque le module a un nom long, on doit écrire ce nom... D'où la méthode des alias :

Programme 8 – Python 2.7

```
1 import module as md  # md  est un raccourci choisi par le programmeur
```

On peut alors faire l'appel de la fonction 'dessine()' du module 'module' par 'md.dessine()'.
On souhaite réaliser la série de dessins ci-dessous, à l'aide du module turtle (voir l'ouvrage disponible en ligne de G.Swinnen `http://inforef.be/swi/python.htm` pour quelques uns des exemples qui suivent) :

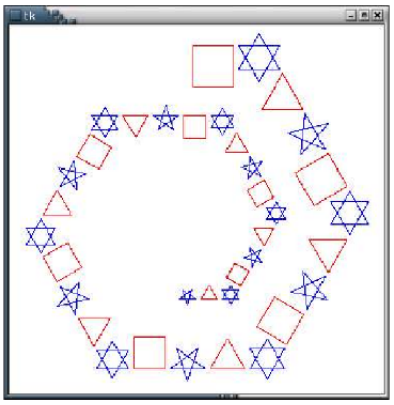


Écrire les lignes de code suivantes, et sauvegardez-les dans un fichier auquel vous donnerez le nom dessinstortue.py :

Programme 9 – Python 2.7

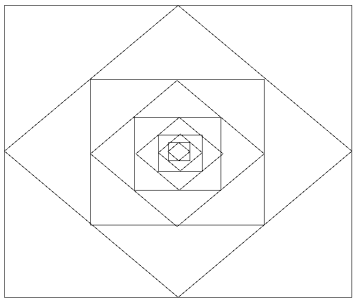
```
# -*- coding: utf-8 -*-

import turtle as tl

def carre(taille,couleur):
    """dessine un carré de longueur de côté 'taille'
    et de couleur 'couleur'. """
    tl.color(couleur)
    for j in range(4):
        tl.forward(taille)
        tl.right(90)

carre(10,'red')
tl.mainloop() #pour éviter fermeture fenêtre après exécution
```

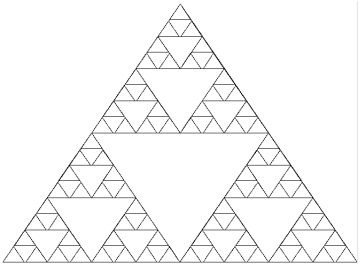
***Exercice* 21** (donné en exercice aux élèves)**.** Écrire un script python qui produise le dessin ci-dessous (9 séries de figures).



On écrira les fonctions suivantes : carre, triangle, etoile5, etoile6, decalage (fonction qui gère le décalage entre deux figures).
Quelques instructions :
- up() : lève le crayon (pas de dessin).
- down() : baisse le crayon.
- forward(n) :avance de n.
- left(r) : tourne à gauche de r degrés.
- right(r) : tourne à droite de r degrés.
- goto(x,y) : positionne le crayon en (x,y), (0,0) est le centre de l'écran.
- circle(r) : dessine un cercle de rayon r
- write('texte') : écrit le texte là où se trouve le curseur
- reset() : efface le dessin
- width(n) : définit la largeur du trait.
- color('blue') : définit la couleur du dessin, ici bleu.
- bgcolor('red') : définit la couleur du fond.

***Exercice* 22.** Réaliser les figures suivantes par une procédure récursive.





3.6 Sur les images

Les formats pbm ascii et ppm ascii utilisés dans les deux premiers exercices sont peu utilisés dans la pratique (formats peu efficaces) mais sont très utiles pour un usage pédagogique : le code couleur de chaque pixel peut être lu dans l'ordre d'inscription dans le fichier texte.
Pour manipuler des formats d'image plus complexes (comme jpg par exemple), on utilisera le module PIL (python image library).

***Exercice* 23** (exercice donné en classe)**.**

1. Exécuter le programme suivant. Il crée un fichier image au format pbm.

Programme 10 – Python 2.7

```
# -*- coding: utf-8 -*-

largeur=70
hauteur=70
n=largeur*hauteur

#ouverture d'un fichier texte en mode écriture :
f=open("coucou.pbm","w")

#on écrit P1 sur la ligne 1 du fichier :
f.write("P1")
# on passe à la ligne :
f.write("\n")
# on écrit la valeur de largeur, espace, la valeur de hauteur :
f.write(str(largeur)+" "+str(hauteur))


# pour chaque ligne :
for j in range(hauteur):
        # passage à la ligne :
    f.write("\n")
    # pour chaque colonne :
    for i in range(largeur):
        # si j est pair
        # on inscrit 0
        if j%2==0:f.write("0")
        # sinon on inscrit 1
        else: f.write("1")

# on ferme le fichier:
f.close()
```

2. Modifier le fichier précédent pour créer une image aux traits verticaux plutôt qu'horizontaux.

***Exercice* 24** (donné en classe)**.** Pour comprendre le format ppm, analyser le script suivant. Prévoir l'image produite avant de l'exécuter.

Programme 11 – Python 2.7

```
# -*- coding : utf-8 -*-
largeur=100
hauteur=600
n=largeur*hauteur
f=open("polychrome.ppm","w")
f.write("P3")
f.write("\n")
f.write(str(largeur)+" "+str(hauteur))
f.write("\n")
f.write("255")# valeur max pour l'intensité des couleurs
f.write("\n")


for j in range(hauteur//6):
    for i in range(largeur):
        f.write("255 0  0")
        f.write("\n")

for j in range(hauteur//6):
    for i in range(largeur):
        f.write("0 255  0")
        f.write("\n")

for j in range(hauteur//6):
    for i in range(largeur):
        f.write("0 0 255")
        f.write("\n")

for j in range(hauteur//6):
    for i in range(largeur):
        f.write("0 255 255")
        f.write("\n")

for j in range(hauteur//6):
    for i in range(largeur):
        f.write("255 0 255")
        f.write("\n")

for j in range(hauteur//6):
    for i in range(largeur):
        f.write("255 255 0")
        f.write("\n")

f.close()
```

***Exercice* 25** (donné en classe)**.** Dans la rubrique programmation sur pairformance, vous trouverez une image de rose rouge au format ppm ascii.
Écrire un programme qui aura pour principal effet de créer une rose verte à partir de cette image.

***Exercice* 26** (donné en classe)**.** Pour se familiariser avec le module PIL, analyser le fichier suivant (utiliser n'importe quelle image jpg en couleur pour tester ce script, en la renommant fruits ou en modifiant dans le code le nom de l'image).

Programme 12 – Python 2.7

```
# -*- coding : utf-8 -*-

from PIL import Image

```

```
# ouverture d'une image au format jpg :
imageSource=Image.open("fruits.jpg")
#   largeur et   hauteur en pixels de l'image
largeur,hauteur=imageSource.size

imageBut=Image.new("RGB",(largeur,hauteur))



# pour chaque ligne :
for y in range(hauteur):
    #pour chaque colonne :
    for x in range(largeur):
        # code du pixel (niveau de gris)
        p=imageSource.getpixel((x,y))
        m=min(p)
        M=max(p)
        s=(m+M)//2
        p=(s,s,s)
        # création du pixel correspondant dans la nv image :
        imageBut.putpixel((x,y),p)



# sauvegarde de l'image créée :
imageBut.save("fruitgris.jpg")
# on montre l'image :
imageBut.show()
```

Exercice 27 (donné en classe)**.** Transformer l'image en une image symétrique comme ci-dessous :

Vous trouverez l'image originale sur pairformance (rubrique programmation).

3.7 Tkinter

Dans l'exemple qui suit, nous allons créer une fenêtre très simple, et y ajouter deux widgets typiques : un bout de texte (ou label) et un bouton (ou button).

Programme 13 – Python 2.7

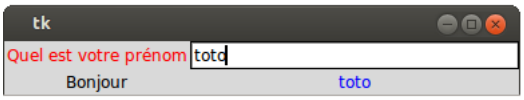
```
from Tkinter import *
fen1 = Tk()
tex1 = Label(fen1, text='Bonjour tout le monde !', fg='red')
tex1.pack()
bou1 = Button(fen1, text='Quitter', command = fen1.destroy)
bou1.pack()
fen1.mainloop()
```

3.7.1 Premier exemple

Examinons à présent plus en détail chacune des lignes de commandes exécutées :

1. from Tkinter import * consiste à importer toutes les classes contenues dans le module Tkinter.
2. fen1 = Tk(), nous utilisons l'une des classes du module Tkinter, la classe Tk(), et nous en créons une instance (autre terme désignant un objet spécifique), à savoir la fenêtre fen1. Ce processus d'instanciation d'un objet à partir d'une classe est une opération fondamentale dans les techniques actuelles de programmation. Celles-ci font en effet de plus en plus souvent appel à une méthodologie que l'on appelle programmation orientée objet. La classe Tk() , qui est l'une des classes les plus fondamentales de la bibliothèque Tkinter, contient tout ce qu'il faut pour engendrer différents types de fenêtres d'application, de tailles ou de couleurs diverses, avec ou sans barre de menus, etc. Nous nous en servons ici pour créer notre objet graphique de base, à savoir la fenêtre qui contiendra tout le reste. Dans les parenthèses de Tk(), nous pourrions préciser différentes options.
3. tex1 = Label(fen1, text='Bonjour tout le monde !', fg='red'), nous créons un autre objet (un widget), cette fois à partir de la classe Label().
4. tex1.pack() , nous activons une méthode associée à l'objet tex1 : la méthode pack(). Une méthode est une fonction intégrée à un objet (on dira aussi qu'elle est encapsulée dans l'objet).
5. Un certain nombre de procédures ou de fonctions (qui sont donc des algorithmes) : on les appelle les méthodes de l'objet. La méthode pack() fait partie d'un ensemble de méthodes qui sont applicables non seulement aux widgets de la classe Label(), mais aussi à la plupart des autres widgets Tkinter, et qui agissent sur leur disposition géométrique dans la fenêtre. Comme vous pouvez le constater par vous-même si vous entrez les commandes de notre exemple une par une, la méthode pack() réduit automatiquement la taille de la fenêtre maître afin qu'elle soit juste assez grande pour contenir les widgets esclaves définis au préalable.
6. bou1 = Button(fen1, text='Quitter', command = fen1.destroy), nous créons notre second widget esclave : un bouton.
7. la méthode pack() pour adapter la géométrie de la fenêtre au nouvel objet que nous venons d'y intégrer.
8. fen1.mainloop() est très importante, parce que c'est elle qui provoque le démarrage du réceptionnaire d'événements associé à la fenêtre. Cette instruction est nécessaire pour que votre application soit à l'affût des clics de souris, des pressions exercées sur les touches du clavier, etc. C'est donc cette instruction qui la met en marche, en quelque sorte.Comme son nom l'indique (mainloop), il s'agit d'une méthode de l'objet fen1, qui active une boucle de programme, laquelle tournera en permanence en tâche de fond, dans l'attente de messages émis par le système d'exploitation de l'ordinateur. Celui-ci interroge en effet sans cesse son environnement, notamment au niveau des périphériques d'entrée (souris, clavier, etc.).

***Exercice* 28.** Écrire un programme python comportant trois zones de texte (label) et une zone de saisie (Entry). L'utilisateur entre son nom et le script lui dit bonjour.



***Exercice* 29.** Le but de cet exercice est de créer une fenêtre contenant un canevas et trois boutons :
– Le canevas sera la zone de dessin
– le bouton [tracer un cercle] affichera un cercle dans le canevas
– le bouton [changer de couleur] modifiera la couleur des tracés suivants
– le bouton [quitter] fermera la fenêtre et terminera l'application

3.7.2 Détecter un clic de souris

***Exercice* 30.** Écrire un script de manière à faire apparaître un cercle rouge de rayon 10 pixels à l'endroit ou l'utilisateur a effectué son clic. Attention, le cercle doit être centré sur la souris...

***Exercice* 31** (la pêche au canard)**.** Le principe du jeu est simple : un cercle (le canard) apparaît de manière aléatoire dans un canevas et l'utilisateur doit cliquer dessus pour l'attraper avant qu'il ne disparaisse (pour réapparaitre ailleurs). L'interface de départ sera minimale, avec un canevas (grand) dans lequel on tracera le cercle (petit) à attraper, un label (en dessous), qui indiquera si le cercle a bien été attrapé et un bouton [Quitter].