Documentation pour l'enseignant

Académie d'Amiens

version 2.1 du 4 avril 2012

Vincent MAILLE
Agnès BARAQUIN
François PREDINAS
Julien POLLET

Table des matières

Chapitre 1

Présentation

C'est avec un réel plaisir que je vous présente cette documentation d'AmiensPython, résultat de trois années de travail avec mes collègues Julien POLLET, Agnès BARAQUIN et François PREDINAS que je remercie ici de tout cœur pour leur collaboration. Cette documentation contient les instructions dont vous aurez besoin dans votre enseignement des mathématiques. De plus, elle regorge d'exemples réalisables pour la plupart avec des élèves. Pour des questions de taille, nous avons choisi de ne pas placer les commentaires dans les exemples de la documentation. Néanmoins, en cliquant sur le bouton Télécharger vous serez redirigé vers les pages du site officiel où figurent ces commentaires.

Merci aussi à tous ceux qui ont contribué à enrichir cette documentation par leurs échanges toujours enrichissants, en particulier je remercie Fatima ESTEVENS et Jean-François KENTZEL pour leurs relectures attentives.

I) Pourquoi AmiensPython ?

1) Le choix de Python

Le choix du langage Python avait déjà été fait par bon nombre de collègues et des formateurs de l'académie d'Amiens pour différentes raisons :

► Un langage simple et mathématiquement puissant pour lequel on trouve des bibliothèques mathématiques très évoluées dans de multiples domaines (traitement de l'image, analyse de Fourier, calcul numérique...).

► Le langage Python est libre et gratuit, et peut donc être installé dans les établissements et dans les foyers des élèves.

2) Les plus d'AmiensPython

Malgré les nombreux atouts du langage Python, ce dernier comporte quelques points faibles que nous voulions combler. AmiensPython est dérivé de portablepython basé sur le Python 2.6 auquel nous avons apporté quelques modifications, en particulier :

► Une interface traduite en français pour une plus grande convivialité.

► Une bibliothèque unique *lycee* qui regroupe les bibliothèques les plus courantes ainsi que de nouvelles fonctions pouvant servir au lycée.

► Une documentation en français à destination de l'enseignant contenant des exemples simples d'usage pédagogique et téléchargeables sur le site de l'académie d'Amiens.

► Un ensemble de fiches pour l'élève à distribuer tout au long de l'année.

AmiensPython est donc une version héritée du Python 2.6. Ainsi un programme réalisé sous Python 2.6 fonctionnera aussi avec AmiensPython. Mais l'inverse n'est pas forcément vrai.

Cependant cette version de Python propose une définition de la division qui ne nous convient pas, en effet `1/3` renvoie 0 (quotient de la division euclidienne) alors que les élèves pourraient attendre une valeur approchée décimale comme sur leur calculatrice. Il suffit d'ajouter en début de programme `from __future__ import division` pour que la division décimale soit activée. L'interface AmiensPython ajoute automatiquement cette ligne pour plus de convivialité.

II) Installation

Pour installer AmiensPython, rien de plus simple, rendez-vous sur le site officiel d'AmiensPython, téléchargez l'installateur et exécutez-le en renseignant l'emplacement où vous souhaitez installer le logiciel. Si vous ne savez pas où l'installer, lisez la section suivante.

Où installer AmiensPython ? Tout dépend de ce que vous voulez en faire...

► Installation sur une clé USB : dans ce cas choisissez le lecteur représentant votre clef, l'installation se fera alors dans un dossier nommé « AmiensPython ».

► Installation sur un ordinateur personnel : même principe que pour la clé USB, choisissez un emplacement qui vous convient.

► Installation sur réseau si vous êtes administrateur : installez AmiensPython dans un dossier en lecture seule pour éviter qu'une erreur de manipulation d'un élève ne se répercute sur tout le reste du lycée.

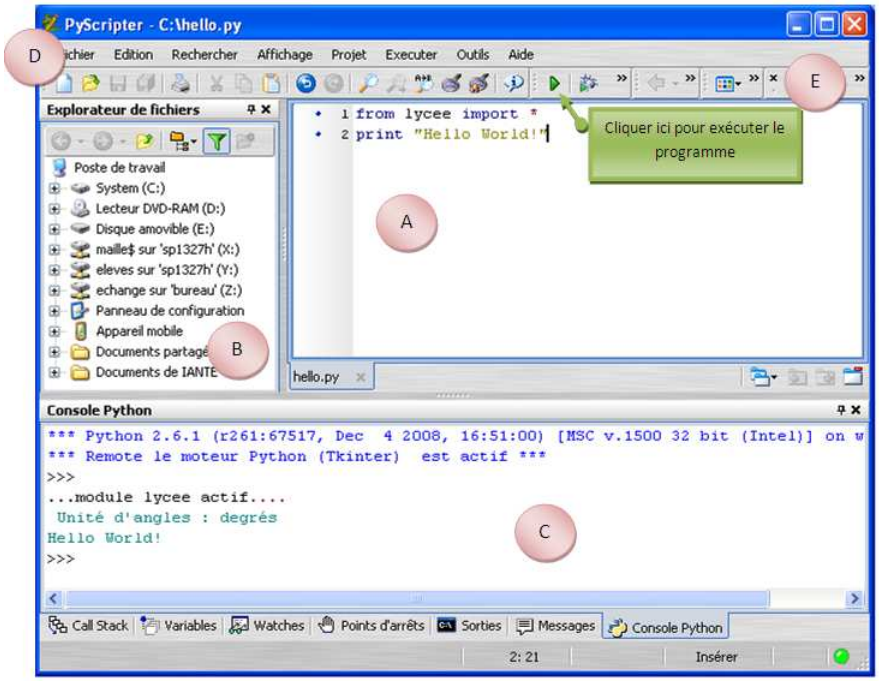
► Installation pour une classe si vous êtes professeur : installez AmiensPython dans le lecteur de la classe de vos élèves et de préférence dans un dossier en lecture seule (le dossier fiche par exemple qui se trouve dans le dossier _commun convient parfaitement).

► Installation sur votre compte personnel dans un établissement (si vous êtes élève ou enseignant) : installez alors Python sur votre lecteur personnel (celui qui porte votre nom en général).

Remarque : Du fait que nous utilisons une version portable de Python, votre ordinateur ne saura pas ouvrir de lui-même les programmes Python (.py). Pour exécuter un programme vous devrez donc le rechercher dans vos dossiers et vos fichiers à partir de l'interface AmiensPython déjà ouverte.

III) Hello world !

Comme la coutume l'exige, notre premier programme sera l'affichage du texte "Hello World". L'interface choisie pour AmiensPython est un dérivé de Pyscripter, c'est une interface conviviale que nous avons traduite en français. Voici comment elle se présente :



La fenêtre de l'éditeur est composée de plusieurs zones :

A : Zone de saisie du programme

B : Zone de l'explorateur windows pour aller chercher vos fichiers

C : Zone où le programme s'exécute, cette zone s'appelle aussi la *console Python*

D : La barre de menu

E : La barre d'outils

Nous ne détaillerons pas davantage ici la barre d'outils très riche qui nous semble relativement intuitive à utiliser.

Notez cependant que pour utiliser la bibliothèque *lycee* que nous avons réalisée, un programme Python devra commencer par :

`from lycee import *` *(Pas d'accent à* `lycee` *et un espace après* `import`*)*

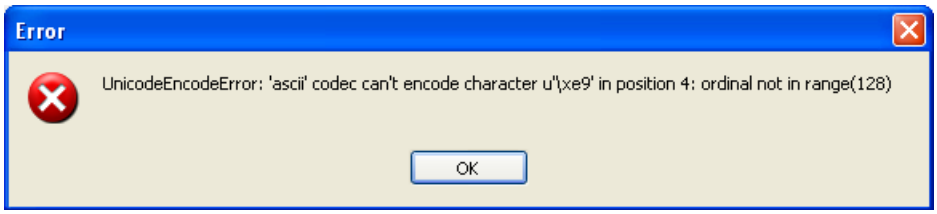
l'interface d'AmiensPython ajoute automatiquement cette ligne quand vous cliquez sur « nouveau ».

Remarque:

- *Si vous oubliez l'espace entre le* `import` *et* `*`*, le programme fonctionnera mais vous ne bénéficierez pas de l'auto complétion et de l'aide par info bulle.*
- *D'autre part, la première exécution peut parfois prendre un peu de temps.*

IV) Les caractères accentués

Attention, **Python n'apprécie pas du tout les noms de fichiers contenant des accents**. Si vous enregistrez un fichier qui en comporte à quelque niveau que ce soit du chemin, vous ne pourrez pas exécuter le programme et le message d'erreur suivant apparaîtra :



V) Notes techniques sur les modules présents

Pour information, vous trouverez ici les références des bibliothèques Python présentes sur la distribution d'AmiensPython :

XTurtle : Version 0.95a0.01 : Module qui permet d'utiliser la tortue.

matplotlib : Version 0.99.1 : Module très puissant de sortie graphique.

numpy : Version 1.3.0 : Bibliothèque de calcul numérique.

Et, plus particulièrement pour l'ISN :

PIL : Version 1.1.6 : Bibliothèque de traitement d'image.

Pyserial : Version 2.6 : Module pour communiquer avec un port série.

sqlite3 : Bibliothèque pour utiliser des bases données avec Python sans utiliser de serveur.

Pour le divertissement et la curiosité enfin :

pygame : Version 1.8.1 : Bibliothèque pour réaliser des jeux.

Chapitre 2

Les instructions de base

Sommaire

I) Les bases du langage Python

1) La gestion des espaces

Dans le langage Python, on peut passer des lignes pour plus de clarté, ce qui n'est pas pris en compte lors de l'exécution du programme. Par contre, vous ne pouvez pas ajouter un espace en début de ligne comme bon vous semble, car cela a une signification. On appelle ***indentation*** ce décalage d'un cran d'une ou de plusieurs lignes d'un programme. Elle permet de délimiter un bloc d'instructions dans une boucle ou lors d'une exécution conditionnelle.

Voici ce qui pourrait être un extrait d'un programme qui résout $ax = b$:

Télécharger

Code: ***Extrait***

```
from lycee import *
a=2
b=6
if a<>0:
    print "Il y a une unique solution:"

    print b/a
```

Aperçu du résultat dans la console

```
Il y a une unique solution:
3
```

Indication:

On peut aussi appuyer sur la touche de tabulation ⇆ *, à gauche de la touche "A" pour gérer l'indentation.*

Remarque:

Afin que le programme fonctionne, nous avons attribué à a et b des valeurs arbitraires. Il est évidemment plus intéressant de demander des valeurs à l'utilisateur mais ceci sera traité plus loin.

2) Commentaires

#

(se lit « croisillon »pour les puristes ou « dièse »sinon) permet de faire figurer dans le corps du programme un commentaire qui ne sera pas pris en compte lors de son exécution. # porte sur le reste de la ligne.

Télécharger

Code: ***Résolution de l'équation*** $ax = b$

```
from lycee import *
a=2
b=6
if a<>0:  # cas où a est différent de zéro
    print "Il y a une unique solution:"
    print b/a
```

Aperçu du résultat dans la console

```
Il y a une unique solution:
3
```

3) Majuscules ou minuscules

Les instructions Python s'écrivent en minuscules.

On peut utiliser des majuscules dans les noms de variables. Par contre, il faudra l'écrire exactement de la même façon dans la suite du programme, en recopiant bien minuscules et majuscules : c'est ce qu'on appelle ***"respecter la casse"***.

Télécharger

Code: ***affichage d'une équation de droite***

```
from lycee import *
CoefficientDirecteur=2
ordonneeAlOrigine=6

print "y=",CoefficientDirecteur,"x +", ordonneeAlOrigine
```

Aperçu du résultat dans la console

```
y= 2 x + 6
```

4) Le rôle du symbole =, l'affectation

=

Le symbole = n'est pas celui de l'égalité mathématique, il n'est d'ailleurs pas symétrique.
Il s'agit d'affecter une valeur à une variable : on stocke une valeur numérique ou du texte dans une mémoire.
La syntaxe est $NomDeLaVariable = valeur$.

Télécharger

Code: ***affectation 1***

```
from lycee import *
a=2      # a vaut 2
b=3.1    # b vaut 3,1
s=b+3    # s vaut 6,1
c,d=4.2,7   # c vaut 4,2 et d vaut 7
e= "bonjour"   # e vaut "bonjour" (chaine de caractères)
f="prêt à faire des mathématiques?"
     # f vaut "prêt à faire des mathématiques?" (chaine de caractères)
```

Remarque:

- *Évidemment si vous exécutez ce programme rien de visible ne se passe puisqu'aucun affichage n'est demandé.*
- *On peut utiliser l'affectation simultanée a,b=2,3.1 à la place des deux premières lignes du code précédent.*

Indication:

Quand on veut définir une chaine de caractères, on la délimite soit à l'aide de guillemets " " soit d'apostrophes ' '.
Si la chaine de caractères que l'on veut définir est **Vous avez "gagné".**, *on utilisera les apostrophes pour la délimiter puisqu'elle contient des guillemets. On écrira par exemple :*

```
reponse='Vous avez "gagné".'
```

De même quand la chaine de caractères contient une apostrophe, on la délimite par des guillemets.

Télécharger

Code: *affectation 2*

```
from lycee import *
a,b=3,2
a,b=a+b, a-b
print a,b
```

```
Aperçu du résultat dans la console
5 1
```

Télécharger

Code: *affectation 3*

```
from lycee import *
a,b=3,2
a=a+b
b=a-b
print a,b
```

```
Aperçu du résultat dans la console
5 3
```

Indication:

Dans l'exemple de gauche ci-dessus, les valeurs de a et b sont affectées simultanément en utilisant les valeurs des lignes précédentes. En revanche dans celui de droite, les affectations sont successives, ce qui explique les résultats différents. Ainsi `a,b = b,a` *échange les valeurs des deux variables* `a` *et* `b`.

Télécharger

Code: ***Que fait le programme suivant ?***

```
from lycee import *
x=demande('Entrez une valeur pour x')
y=demande('Entrez une valeur pour y')
x,y=x+y,x-y
x,y=x+y,x-y
print "maintenant, x=",x,"et y=",y
```

II) Entrées/Sorties

1) Affichage et calculs simples

print

print affiche la valeur numérique ou le texte qui suit.
print a,b affiche à la suite sans passer à la ligne les éléments a et b.

Télécharger

Code: ***affichages***

```
from lycee import *
print "Bonjour,"
print "ce programme calcule 2+3=",
print 2+3
print "et 3(-2)=",3*-2
```

Aperçu du résultat dans la console

```
Bonjour,
ce programme calcule 2+3= 5
et 3(-2)= -6
```

Indication:

- *Quand on veut afficher plusieurs choses à la suite qui ne sont pas dans la même ligne de programme, on indique de ne pas passer à la ligne suivante après l'affichage en finissant par une virgule.*
- *Quand un programme ne produit pas le résultat attendu, il est important de penser à ajouter l'affichage des valeurs prises par les variables à chaque étape. Cela permet de comprendre ce que fait réellement le programme, et facilite la recherche d'erreurs, quitte ensuite à supprimer cette ligne ou la transformer en commentaire.*

Remarque:

Pour un produit, le symbole * *est indispensable, même s'il y a des parenthèses.*

2) Les variables

Les noms de variables commencent par une lettre, puis on peut se faire succéder les caractères que l'on veut, minuscules, majuscules, chiffres, *etc.* sans espace. Il est toujours commode d'employer des noms de variables explicites.

Télécharger

Code: ***nombre de 6 obtenus en lançant un dé 10 fois***

```
from lycee import *
nb6=0
for i in range(10):
    if randint(1,6)==6:
        nb6=nb6+1
print "Sur les 10 lancers, on a obtenu", nb6, 'fois le numéro "6".'
```

Aperçu du résultat dans la console

```
Sur les 10 lancers, on a obtenu 3 fois le numéro "6"
```

Indication:

Pour dénombrer, on peut utiliser un compteur (variable nb6 ici) que l'on initialise à 0, avant la boucle. Puis dans la boucle, on l'incrémente de 1 pour chaque cas favorable à l'aide de la commande `nb6=nb6+1`.

3) Demande d'un nombre

var=**demande**(*question*) *lycee*

affiche une fenêtre où figure le texte *question* et un cadre blanc dans lequel on entrera ce qui est demandé. La réponse est alors affectée à la variable *var*.

Télécharger

Code: ***Calculer l'opposé d'un nombre***

```
from lycee import *
x=demande("Entrez un nombre pour obtenir son opposé")
print "L'opposé de ", x, "est", -x
```

Aperçu du résultat dans la console

```
l'opposé de  -8 est 8
```

Remarque:

Important *: Même si on veut demander l'entrée d'un nombre par l'utilisateur, il est conseillé d'écrire d'abord un code en affectant des valeurs bien choisies dans le corps du programme. Ainsi, on peut le tester sans avoir à entrer de valeur pour chaque essai : c'est plus efficace pour le débogage et l'optimisation.*

Télécharger

Code: ***Calculer l'IMC***

```
from __future__ import division
from lycee import *
m=demande("masse en kilo : ")
t=demande("taille en mètre : ")
IMC=m/(t*t)
print "L'IMC est de : ",IMC
```

On peut demander aux élèves de se renseigner sur les valeurs conseillées de l'IMC et connaissant sa taille, donner l'intervalle dans lequel le poids « idéal »devrait être.

4) Demande d'un texte

chaine=**texte __ demande**(*question*) *lycee*

Affiche une fenêtre où figure le texte *question* et un cadre blanc dans lequel on entrera la réponse, qui sera considérée comme une chaine de caractères. Cette valeur est ensuite affectée à la variable *chaine*.

Télécharger

Code: ***Programme du professeur Tournesol***

```
from lycee import *
nb=demande("Entrez 3+2")
tex=texte_demande("Pourriez-vous répéter,svp?")
print nb
print tex
print nb+1
print tex+"1"
```

Aperçu du résultat dans la console

```
5
3+2
6
3+21
```

Remarque:

- `nb + 1` *est une addition alors que* `tex + "1"` *est une concaténation.*
- *On peut aussi entrer du texte avec un simple* `demande` *ou* `input`*, en encadrant la réponse avec des guillemets, ce qui est difficile à gérer pour l'utilisateur du programme.*

Indication:

Les fonctions demande et texte_demande existent déjà dans la version 2.6 de Python sous la forme input et raw_input. Cependant ces deux dernières n'acceptent ni les accents, ni les caractères spéciaux dans la question. Merci à F. Mary pour l'aide au codage de ces deux fonctions.

III) Tests et conditions

1) Si ... alors ...

if *test* **:**

effectue (une fois) les instructions indentées qui suivent lorsque le test est vérifié.

Télécharger

Code: ***nombre de zéros d'un trinôme***

```
from lycee import *
a,b,c=demande("Entrez les coefficients du trinôme séparés par des virgules.")
delta=b*b-4*a*c
if delta>0:
    print "Il y a deux solutions."
    print'Je vous laisse les trouver.'
if delta==0:
    print "Il y a une solution unique."
    print "Je ne vous l'indiquerai pas."
if delta<0:
    print "Il n'y a aucune solution!"
```

Remarque:

Le "alors" n'apparaît pas en Python, c'est l'indentation qui délimite le bloc à exécuter.

Indication:

On peut imbriquer les boucles, à la manière d'un arbre qui se dessinerait avec les indentations.

Télécharger

Code: ***signe d'un trinôme***

```
from lycee import *
a,b,c=demande("Entrez les coefficients du trinôme séparés par des virgules.")
delta=b*b-4*a*c
if delta<0:
    if a>0:
        print "P(x)>0 pour tout réel x"
    if a<0:
        print "P(x)<0 pour tout réel x"
if delta==0:
    if a>0:
        print "P(x) est positif pour tout réel x"
    if a<0:
        print "P(x)est négatif pour tout réel x"
if delta>0:
    if a>0:
        print "P(x)>0 à l'extérieur des racines"
    if a<0:
        print "P(x)<0 à l'extérieur des racines"
```

2) Si ... alors ... sinon ...

if *test* **: ... else : ...**

effectue les instructions indentées lorsque le test est vérifié, sinon effectue les instructions alternatives indentées.

Télécharger

Code: ***Solutions de*** $ax+b=0$

```
from __future__ import division
from lycee import *
a=demande("Entrez a de l'équation ax+b=0")
b=demande("Entrez maintenant b de l'équation ax+b=0")
if a<>0:
    print "Il y a une unique solution:",
    print -b/a
else:
    if b==0:
        print "Il y a une infinité de solutions."
        print "Tous les réels sont solutions."
    else:
        print "Il n'y a aucune solution."
```

Remarque:

- *Le "else" est aligné avec le "if" qui lui correspond.*
- *Taper " :" puis appuyer sur la touche "entrée" pour passer à la ligne, provoque l'indentation automatique.*

3) Hormis les cas précédents, si...

if *test*1 **:** *instructions*1 **elif** *test*2 **:** *instructions*2

Effectue les *instructions*1 indentées lorsque le *test*1 est vérifié, sinon effectue le *test*2 et, si celui-ci est vérifié, effectue les *instructions*2 indentées.

Télécharger

Code: ***Image par une fonction définie par morceaux***

```
#M comme mathématiques
from lycee import *
x=demande('valeur de x')
if x<-4:
    print 'f(',x,')=',8*x+36
elif x<0: #on est dans le cas où x>=-4 et x<0
    print 'f(',x,')=',-x
elif x<4: #on est dans le cas où x>=-4 et x>=0 et x<4
    print 'f(',x,')=',x
else: #on est dans le cas où tous les tests précédents étaient négatifs
    print 'f(',x,')=',-8*x+36
```

Bonus : Représenter la fonction sur [-5 ;5] (Voir chapitre sur les graphiques)

Remarque:

On peut enchaîner autant de "elif" que nécessaire.
Il peut être intéressant de terminer une série de "elif" par un "else" afin d'être sûr de traiter tous les cas.

Indication:

elif est la contraction de « else if »Le "elif" remplace parfois avantageusement des boucles imbriquées.

4) Comparateurs

==

Ce symbole désigne l'égalité dans un test

Télécharger

Code: ***Recherche d'un triplet pythagoricien d'entiers consécutifs***

```
from lycee import *
for i in range(100000):
    a=i*i+(i+1)*(i+1)
    r=(i+2)*(i+2)
    if r==a:
        print"les nombres",i,",",i+1,"et",i+2,"forment un triplet pythagoricien"
```

Aperçu du résultat dans la console

```
les nombres 3 , 4 et 5 forment un triplet pythagoricien
```

< et >

Ces symboles désignent les inégalités strictes habituelles.

Télécharger

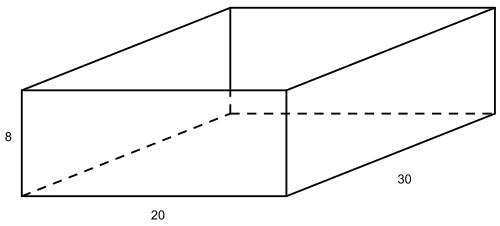
Code: ***Distance sur un axe gradué***

```
from lycee import *
xA=demande("Entrez l'abscisse du point A")
xB=demande("Entrez l'abscisse du point B")
if xA>xB :
    dist=xA-xB
else :
    dist=xB-xA
print "La distance AB est",dist
```

<= et >=

ces symboles désignent les inégalités larges $\leqslant$ et $\geqslant$ habituelles.

Exemple : On s'intéresse au volume d'eau contenu dans une cuve parallélépipédique de base rectangulaire (2m par 3m) et de 80 cm de hauteur. Ecrire un programme qui calcule le volume d'eau en fonction de la hauteur.



Télécharger

Code: ***Volume d'eau dans une cuve***

```
h= demande("valeur de h en décimètres?")
if 0>h:
    print"h est impossible"
if h>=0 and h<=8:
    V= 600*h
    print "le volume contenu dans la cuve est",V,"litres pour h=",h
if h>8:
    print"la cuve déborde!!!"
```

Aperçu du résultat dans la console

```
le volume contenu dans la cuve est 3600 litres pour h= 6
```

!= ou <>

Ces symboles signifient tous deux « différent de »... A vous de choisir !

Télécharger

Code: ***Le jeu du nombre mystère***

```
from lycee import *
a=randint(1,100)
n=demande('Proposer une valeur')
if n<>a:
    print "vous avez perdu, le nombre mystère n'est pas",n
else:
    print 'gagné, vous devriez jouer au LOTO !'
```

Aperçu du résultat dans la console

```
vous avez perdu, le nombre mystère n'est pas 54
```

5) Opérateurs logiques

and

Permet d'effectuer une instruction si deux tests sont vérifiés simultanément.

or

Permet d'effectuer une instruction si au moins un test sur deux est vérifié.

On choisit un nombre entier au hasard entre 1 et 666. On note S l'évènement « Obtenir un multiple de 7 »et T : « Le nombre se termine par 3 ». Ecrire un algorithme permettant de calculer p(T), p(S), p($T \cap S$) et p($T \cup S$).

Télécharger

Code: ***Calcul de probabilités***

```
from lycee import *
p1,p2,p3,p4=0,0,0,0
for i in range(1,667):
    if reste(i,7)==0:
        p1=p1+1
    if reste(i,10)==3:
        p2=p2+1
    if reste(i,7)==0 and reste(i,10)==3:
        p3=p3+1
    if reste(i,7)==0 or reste(i,10)==3:
        p4=p4+1
print "p(S)=",p1,"/666   p(T)=",p2,"/666"
print "p(SnT)=",p3,"/666   p(SuT)=",p4,"/666"
```

Aperçu du résultat dans la console

```
p(S)= 95 /666   p(T)= 67 /666
p(SnT)= 9 /666   p(SuT)= 153 /666
```

Indication:

On peut écrire plusieurs "and" ou plusieurs "or" dans la même instruction, il suffit alors de mettre des parenthèses pour indiquer les priorités. Le mélange de "and" et "or" dans la même instruction peut faire travailler la logique pure.

IV) Boucles

Comme dans la plupart des langages, il existe en Python principalement deux manières de réaliser une boucle, c'est à dire une répétition d'un bloc d'instructions. Comme pour la commande **si**, la partie à répéter sera indentée vers la droite, ce qui permet en plus une bonne visibilité de l'algorithme.

1) Boucle dont on connait le nombre d'itérations

for *var* **in** *list* **:**

Réalise une boucle en faisant parcourir à la variable *var* toute la liste *list (Voir le chapitre sur les listes pour plus de détails).*

Remarque:

Notez que l'instruction se termine par deux points et que les instructions à répéter doivent être décalées, exactement comme pour le `if`*.*

Nous n'allons pas trop entrer dans les détails de l'utilisation des listes ici, cela fait partie d'un autre chapitre de la documentation. Mais voici quelques exemples d'utilisation d'une boucle `for` :

- On peut donner la liste de manière explicite :
  `for jour in ['Lundi', 'Mardi','Mercredi','Jeudi', 'Vendredi', 'Samedi','Dimanche']`
- Ou aussi utiliser l'instruction range pour créer la liste d'entiers.

range($debut, fin, pas$)

(Les paramètres *debut* et *pas* sont optionnels.)
Renvoie une liste d'entiers :

- Dans l'intervalle $[0; fin[$ si un seul paramètre est renseigné.
- Dans l'intervalle $[debut; fin[$ si 2 paramètres sont renseignés.
- Dans l'intervalle $[debut; fin[$ mais en réalisant une suite arithmétique de raison *pas* si les 3 paramètres sont renseignés.

Voici quelques exemples immédiats :

Télécharger

Code: ***Quelques exemples de boucles***

```
from lycee import *
print "Exemple 1 : ",
for voyelle in ['a','e','i','o','u','y'] :
    print voyelle,
print
print "Exemple 2 :",
for n in range(10):
    print n,
print
print "Exemple 3 : ",
for n in range(2,7):
    print n,
print
print "Exemple 4 : ",
for pair in range(100,110,2):
    print pair,
```

```
Exemple 1 :  a e i o u y
Exemple 2 : 0 1 2 3 4 5 6 7 8 9
Exemple 3 :  2 3 4 5 6
Exemple 4 :  100 102 104 106 108
```

Remarque:

- *En cas d'incohérence, la liste n'est pas générée (liste vide) et la boucle n'est donc pas exécutée. (Par exemple : `for n in range (100,110,-2):`).*
- *La variable compteur parcourt quoiqu'il arrive les valeurs demandées même si on tente de modifier celle-ci.*

Télécharger

Code: ***Modification du compteur***

```
from __future__ import division
from lycee import *
for i in range(10):
    print i,
    if i==5 :
        i=8
```

```
0 1 2 3 4 5 6 7 8 9
```

Un exemple qui calcule $\sum_{i=1}^{n} i^2$.

Télécharger

Code: ***Somme des carrés des 100 premiers entiers***

```
from lycee import *
total=0
for n in range(101):
  total=total+n*n
  if n<100:
      print n,"²+",
  else :
      print n,"²=",
print total
```

Aperçu du résultat dans la console

$0^2 + 1^2 + 2^2 + 3^2 + 4^2 + 5^2 + 6^2 + 7^2 + 8^2 + ... + 97^2 + 98^2 + 99^2 + 100^2 = 338350$

Exemple : On lance 100 000 fois de suite deux dés cubiques simultanément, et on s'intéresse au nombre de fois où la somme des deux faces vaut 7.

Télécharger

Code: ***Obtenir 7***

```
from __future__ import division
from lycee import *
j=0
for i in range (100000):
    s=randint(1,6)+randint(1,6)
    if s==7:
        j=j+1
f=j/100000.
print "le 7 est sorti",j,"fois donc avec une fréquence égale à",f
```

Aperçu du résultat dans la console

```
le 7 est sorti 16788 fois donc avec une fréquence égale à 0.16788
```

2) Boucle conditionnelle

Dans la pratique, on ne connaît que rarement le nombre d'itérations pour arriver au résultat (d'où l'intérêt d'un programme). On peut alors utiliser des boucles de type TANT QUE .... FAIRE : ...

while *condition* **:**

Exécute une instruction ou un bloc d'instructions tant que la *condition* est vérifiée. (La boucle peut donc ne jamais être exécutée si, d'entrée la *condition* n'est pas remplie).

L'exemple classique est l'algorithme d'Euclide pour calculer le PGCD de 2 nombres :

Télécharger

Code: ***Calcul du PGCD de 2 entiers strictement positifs***

```
a,b=input("Entrez deux entiers strictement positifs.")
print "PGCD(",a,",",b,")=",
while b<>0 :
    a,b=b,reste(a,b)
print a
```

Aperçu du résultat dans la console

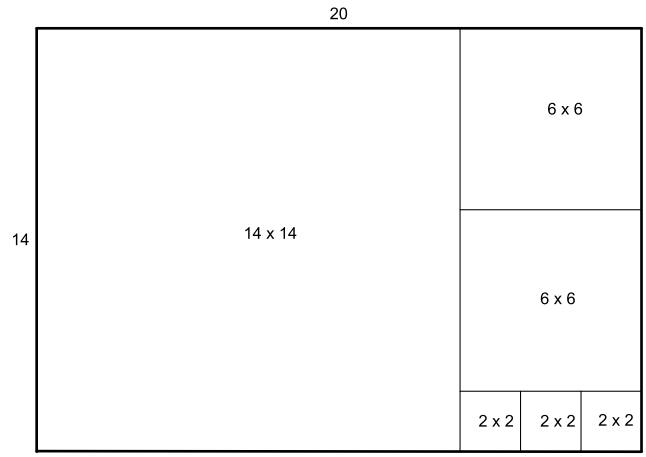
```
PGCD( 12 , 14 )= 2
```

On veut décomposer le produit de deux nombres en somme de carrés comme l'explique le schéma ci-dessous dont on peut trouver plus de détails sur le site Mathématiques magiques de Thérèse Eveilleau. Sur le schéma ci-dessous, l'algorithme donne :
$14 \times 20 = 14^2 + 6^2 + 6^2 + 2^2 + 2^2 + 2^2$.

Télécharger

Code: ***Multiplication babylonienne***

```
from lycee import *
Longueur=demande("Premier nombre")
Largeur=demande("Deuxième nombre")
print Longueur," x ",Largeur," = ",
if Longueur<Largeur :
    Longueur,Largeur=Largeur,Longueur
while Largeur>0 :
    print Largeur,"²+",
    Longueur=Longueur-Largeur
    if Longueur<Largeur :
        Longueur,Largeur=Largeur,Longueur
print 0
```



Bonus : Modifier cet exemple pour que la réponse ne se termine plus par « +0 »

Exemple de modélisation de la marche aléatoire d'un robot sur une table carrée (20cm par 20cm), située à un mètre du sol. Ce robot initialement placé au centre de la table bouge dans deux directions perpendiculaires, en avant ou en arrière, à une vitesse de 1 cm par seconde.

Télécharger

Code: ***Marche aléatoire d'un robot sur une table***

```
from lycee import *
x,y=0,0
j=0
while x>=-10 and x<=10 and y>=-10 and y<=10:
    a=randint(1,4)
    if a==1: x=x+1
    if a==2: y=y+1
    if a==3: x=x-1
    if a==4: y=y-1
    j=j+1
print "le robot est tombé de la table au bout de",j,"secondes."
```

Aperçu du résultat dans la console

```
le robot est tombé de la table au bout de 153 secondes.
```

Remarque:

- *Lorsqu'il n'y a qu'une instruction à exécuter, il n'est pas nécessaire de retourner à la ligne et de l'indenter (attention cependant à garder une bonne lisibilité du programme).*
- *Pour s'amuser, on peut remplacer les 4 lignes contenant des if par :* `x,y=x+(2-a)*reste(a,2),y+(3-a)*reste(a+1,2)`
- *Au fait, le robot tombe-t-il toujours de la table ?*

Bonus : Une visualisation du parcours du robot peut se programmer avec la tortue.

Chapitre 3

Les fonctions mathématiques

Sommaire

I) Puissance et factorielle d'un entier

puissance(a,n) *lycee*

Donne le résultat de a^n où a et n sont deux nombres.

Télécharger

Code: ***Recherche des entiers distincts tels que*** $x^y = y^x$.

```
#On se limite aux entiers inférieurs à 100
from lycee import *
for x in range(1,100):
    for y in range(x+1,100):
        if puissance(x,y)==puissance(y,x):
            print x,"^",y,"=",y,"^",x
```

Aperçu du résultat dans la console

```
2 ^ 4 = 4 ^ 2
```

Remarque:

Nous avons choisi de reprogrammer la fonction puissance afin d'éviter toute confusion chez les élèves mais pour obtenir 2^8 *on peut tout simplement taper* `2**8`.

factorial(n) *lycee,math*

Donne le résultat $n! = 1 \times 2 \times 3 \times \cdots \times (n-1) \times n$. où n est un nombre entier.

Soient $(u_n)_{n\geqslant 1}$ et $(v_n)_{n\geqslant 1}$ deux suites définies pour $n \in \mathbb{N}^*$ par $u_n = \sum_{k=0}^{n} \frac{1}{k!}$ et $v_n = u_n + \frac{1}{n!}$. On peut montrer que ces deux suites sont adjacentes. Estimer une valeur approchée de leur limite commune à 10^{-5} près.

Télécharger

Code: ***Estimation du nombre e par suites adjacentes.***

```
from __future__ import division
from lycee import *
u,n,ecart=0,0,1
while ecart > puissance(10,-5) :
    u=u+1/factorial(n)
    v=u+1/factorial(n)
    print "u",n,"=",u," et v",n,"=",v
    ecart=v-u
    n=n+1
print "la limite de (u_n) et (v_n) vaut ",u,"à 0.00001 près"
```

Aperçu du résultat dans la console

```
u 1 = 1.0  et v 1 = 2.0
u 2 = 1.5  et v 2 = 2.0
u 3 = 1.66666666667  et v 3 = 1.83333333333
u 4 = 1.70833333333  et v 4 = 1.75
u 5 = 1.71666666667  et v 5 = 1.725
u 6 = 1.71805555556  et v 6 = 1.71944444444
u 7 = 1.71825396825  et v 7 = 1.71845238095
u 8 = 1.71827876984  et v 8 = 1.71830357143
u 9 = 1.71828152557  et v 9 = 1.71828428131
la limite de (u_n) et (v_n) vaut 1.71828152557 à 0.00001 près
```

II) Quotient entier et reste d'une division euclidienne

reste(a,b) *lycee*

Donne le reste de la division de a par b, c'est à dire l'unique entier r tel que $a = bq + r$ où q est un entier et r un entier naturel vérifiant $0 \leqslant r < |b|$.

Les années bissextiles sont les années divisibles par 4, sauf celles divisibles par 100, sauf celles divisibles par 400. Imaginer un programme indiquant si une année est bissextile. (Cet exemple peut permettre de travailler un peu sur les opérateurs logiques)

Télécharger

Code: ***Année bissextile***

```
from __future__ import division
from lycee import *
n=demande('année ? ')

if (reste(n,4)==0 and reste(n,100)<>0) or reste(n,400)==0:
    print n,'est une année bissextile'
else :
    print n,"n'est pas une année bissextile"
```

quotient(a,b) *lycee*

Donne le quotient entier de la division de a par b, c'est à dire l'unique entier q tel que $a = bq + r$ où q est un entier et r un entier naturel vérifiant $0 \leqslant r < |b|$.

Imaginer un programme qui demande le nombre de décimales voulu et qui calcule le quotient avec la précision demandée.

Télécharger

Code: ***Une division avec un nombre arbitraire de décimales***

```
from lycee import *
n=demande("Nombre de décimales")
a=demande("Dividende")
b=demande("Diviseur")
print quotient(a,b),",",
r=reste(a,b)
for i in range(n):
  r=r*10
  q=quotient(r,b)
  print q,
  r=reste(r,b)
```

Remarque:

- *Notez que ces deux fonctions existent déjà en python, le quotient de deux entiers pouvant être obtenu par $int(a/b)$ et le reste par $a\%b$. Mais ces fonctions ne renvoient pas forcément les valeurs usuelles lorsque l'on donne un diviseur négatif. Il s'agit de définir la division dans $\mathbb{Z}$ comme renvoyant toujours un reste positif.*
- *La ligne* `print quotient(a,b),","`, *se termine par une virgule, ce qui a pour effet de ne pas retourner à la ligne dans l'affichage, ainsi le résultat apparait sur une seule ligne.*

III) Plus grand diviseur commun

L'algorithme d'Euclide a été étudié en classe de troisième, il est donc intéressant à notre avis que les élèves le programment une fois. Cependant, nous avons fait le choix d'ajouter cette fonction dont nous pouvons avoir besoin au cours de l'année.

pgcd(a,b) *lycee*

Retourne le pgcd des entiers a et b.

Un exemple de programme qui ajoute 2 fractions et retourne le résultat sous forme de fraction irréductible. Cet algorithme pourra largement être optimisé selon le niveau et la rapidité des élèves, en particulier pour tout ce qui concerne l'affichage du résultat s'il est entier par exemple.

Télécharger

Code: ***Somme de deux fractions***

```
from lycee import *
n1,d1=demande("Entrez la 1ère fraction en indiquant le numérateur et le dénominateur
                                                    séparés par une virgule")
n2,d2=demande("Entrez la 2ème fraction en indiquant le numérateur et le dénominateur
                                                    séparés par une virgule")
n,d=n1*d2+n2*d1,d1*d2
p=pgcd(n,d)
n,d=n/p,d/p
print n1,"/",d1,"+",n2,"/",d2,"=",n,"/",d
```

Cet autre programme est l'algorithme d'Euclide étendu étudié en spécialité maths de terminale scientifique. Cet exemple s'adresse donc à des élèves ayant déjà pratiqué l'activité algorithmique depuis 2 ans et est donc assez complexe à appréhender pour un débutant.

Télécharger

Code: ***Algorithme d'Euclide étendu***

```
from lycee import *
a, b, c = demande ("Résolution de ax + by = c dans N.\n Entrez les valeurs
                                    entières de a, b et c séparées par une virgule.")
d = pgcd (a, b)
if reste (c, d) > 0 :
    print "Aucune solution, car", d, " divise", a, "et", b,
    print "mais ne divise pas", c
else :
    a0, b0, c0 = a / d, b / d, c / d ;
    p, q, r, s = 1, 0, 0, 1 ;
    while b <> 0 :
        res = reste (a, b) ;
        quot = quotient (a, b) ;
        a, b = b, res ;
        p, q, r, s = r, s, p - quot * r, q - quot * s ;
    print (p * c0), "*", (a0 * d), "+ (", (q * c0), ") *", (b0 * d), "=", d * c0
```

IV) Racine carrée

sqrt(x) *lycee,math*

Retourne $\sqrt{x}$ où x est un nombre positif.

Le mathématicien grec Héron d'Alexandrie (Ier siècle après J.-C.) a trouvé une formule pour calculer l'aire d'un triangle connaissant la longueur de ces 3 côtés : $\mathscr{A} = \sqrt{p(p-a)(p-b)(p-c)}$ où a, b et c représentent les longueurs des 3 côtés de ce triangle et p son demi-périmètre. Traduisez cette formule dans un programme en Python.

Télécharger

Code: ***Calcul de l'aire d'un triangle avec la formule de Héron***

```
from __future__ import division
from lycee import *
a,b,c=demande('Entrez les longueurs des 3 côtés')
p=(a+b+c)/2
aire=sqrt(p*(p-a)*(p-b)*(p-c))
print "L'aire de ce triangle est",aire
```

Télécharger

Code: ***Calcul de la longueur de l'hypoténuse dans un triangle rectangle***

```
from lycee import *
a,b=demande("Entrer les longueurs des deux côtés les plus courts du triangle
                                        rectangle séparés par une virgule")
h=sqrt(a*a+b*b)
print "L'hypoténuse mesure",h
```

V) Fonction exponentielle et logarithme népérien

exp(x) *lycee,math*

Retourne e^x où x est un nombre réel.

ln(x) *lycee*

Retourne $\ln x$ où x est un nombre positif.

Remarque:

Dans le module `math` *la fonction logarithme népérien se définit par* `math.log(...)`*. Mais pour plus de cohérence, nous avons choisi de la renommer* `ln` *dans le module* `lycee`*.*

VI) Partie entière, Valeur absolue

floor(x) *lycee,math*

Retourne la partie entière du nombre x, c'est à dire le plus grand entier inférieur au réel x.

abs(x)

Retourne la valeur absolue x (ou distance à 0 dans le programme de collège).

VII) Trigonométrie

Par défaut dans la bibliothèque Python d'origine, comme dans tous les autres langages, les fonctions cos, sin, tan sont définies pour des réels qui représentent des mesures d'angles en radians. Cependant, le fait que la connaissance du radian ne soit plus exigible en seconde, nous a conduit à définir ces fonctions aussi pour des réels qui représentent des mesures d'angles en degrés (ainsi que les fonctions réciproques).
Par défaut, puisque c'est la seule unité que les élèves connaissent, l'unité utilisée est le degré, cependant on peut basculer d'un mode à l'autre avec la fonction suivante :

unite_angle(*choix*) *lycee*

Modifie l'unité par défaut utilisée par Python pour les fonctions trigonométriques :
`unite_angle('deg')` passe en mode degré.
`unite_angle('rad')` passe en mode radian.
`print unite_angle()` affiche l'unité en cours d'utilisation.

cos(x), sin(x) et tan(x) *lycee*

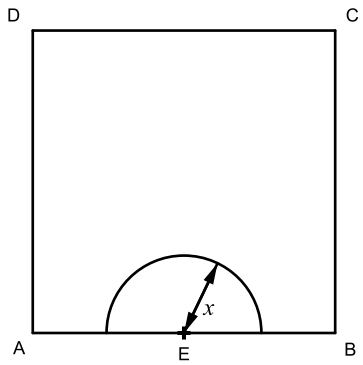
Renvoient respectivement le cosinus, sinus et la tangente de l'angle x donné dans l'unité définie par la fonction `unite_angle`.

acos(x), asin(x) et atan(x) *lycee*

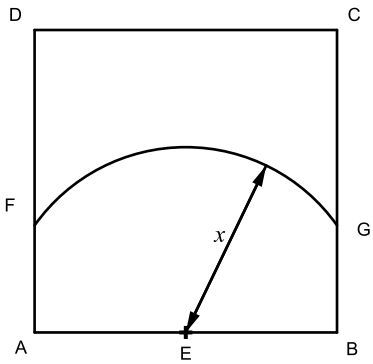
Renvoient la mesure d'un angle (dans l'unité en cours) dont le cosinus, sinus ou la tangente valent x Avec comme convention habituelle (en radians) :
acos : $[-1;1] \to \left]-\frac{\pi}{2};\frac{\pi}{2}\right]$, asin : $[-1;1] \to]-\pi;\pi]$, atan : $\mathbb{R} \to \left]-\frac{\pi}{2};\frac{\pi}{2}\right[$.

Un exemple issu d'un problème du mois de l'académie d'Amiens, « Le pré et la chèvre » : *Une personne possède un pré de forme carrée de 10m de côté. Il attache une chèvre par une corde reliée à un piquet planté au milieu d'un des côtés. Il souhaite que la chèvre broute une surface d'aire égale à la moitié de l'aire du pré. Quelle longueur de corde doit-il laisser ?* Ici on peut imaginer un programme qui permet, en fonction de la longueur x de la corde, d'afficher la surface que la chèvre peut brouter.

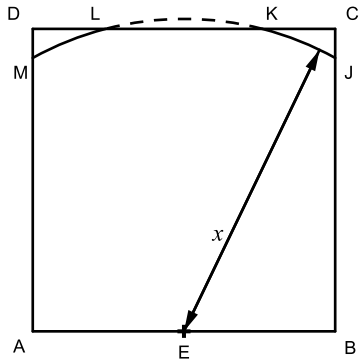
1er cas



2eme cas



3eme cas



Télécharger

Code: ***Le pré et la chèvre.***

```
from __future__ import division
from lycee import *
x=demande("Longueur de la corde ?")
if x<5:
    Aire=pi*x*x/2
elif x<10:
    AngleAEF=acos(5/x)
    AngleFEG=180-2*AngleAEF
    Aire=5*sqrt(x*x-5*5)+pi*x*x/360*AngleFEG
elif x<sqrt(125):
    AngleMEA=acos(5/x)
    AngleMEJ=180-2*AngleMEA
    AngleLEK=2*acos(10/x)
    AngleMEL=1/2*(AngleMEJ-AngleLEK)
    LK=2*x*sin(acos(10/x))
    Aire=5*sqrt(x*x-5*5)+pi*x*x/180*AngleMEL+5*LK
else:
    Aire=100
print "L'aire mesure",Aire
```

VIII) Les constantes

pi *lycee,math*

Constante qui vaut une valeur approchée du nombre π.

Télécharger

Code: ***Calculer le périmètre d'un cercle***

```
from lycee import *
r=demande("Entrer le rayon du cercle")
print "La circonférence du cercle mesure",2*pi*r
```

Remarque:

Attention : pi est une variable comme les autres mais qui a été initialisée avec une certaine valeur. Ainsi si dans un programme vous tapez pi=2, 2 sera alors la nouvelle valeur de π pour le programme. De même, vous pouvez au besoin définir le nombre e=exp(1) au début d'un programme niveau Terminale.

Le savant grec Archimède en 250 avant Jésus-Christ utilisa des polygones réguliers de 96 côtés, et détermina que le rapport de la circonférence d'un cercle à son diamètre a une valeur proche de $\frac{22}{7}$. Parmi toutes les fractions comprenant un dénominateur à un ou deux chiffres, quelles sont celles qui représentent une meilleure approximation de π ?

Télécharger

Code: ***Approximation historique du nombre π***

```
from __future__ import division
from lycee import *
err=22/7-pi
print "erreur historique",err
for d in range(1,100):
    for n in range (3*d,4*d):
        f=n/d
        if f<pi:
            e=pi-f
        else:
            e=f-pi
        if e<err :
            err=e
            print "J'ai trouvé mieux :",n,"/",d," L'erreur est :",err
```

Aperçu du résultat dans la console

```
erreur historique 0.00126448926735
J'ai trouvé mieux : 179 / 57 L'erreur est : 0.00124177639681
J'ai trouvé mieux : 201 / 64 L'erreur est : 0.000967653589793
J'ai trouvé mieux : 223 / 71 L'erreur est : 0.000747583167258
J'ai trouvé mieux : 245 / 78 L'erreur est : 0.000567012564152
J'ai trouvé mieux : 267 / 85 L'erreur est : 0.000416183001558
J'ai trouvé mieux : 289 / 92 L'erreur est : 0.000288305763706
J'ai trouvé mieux : 311 / 99 L'erreur est : 0.000178512175652
```

IX) Le hasard

Il existe différentes manières de tirer un nombre au hasard selon ce que l'on a besoin de faire dans un programme. Voici 3 fonctions qui devraient déjà répondre à la majorité de vos besoins :

random()

lycee, random

Cette fonction renvoie un nombre décimal de l'intervalle $[0\,;\,1[$, choisi selon une densité uniforme sur cet intervalle.
Cette fonction ne nécessite pas d'argument, pensez cependant à ouvrir et fermer les parenthèses. (Comme pour la fonction ALEA() sur le tableur)

Exemple : Dans un repère orthonormé, on considère les 3 zones suivantes :

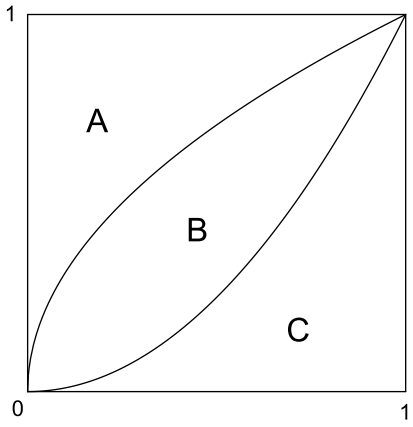
$$A = \{(x,y) \in \mathbb{R}^2 / 0 \leqslant x \leqslant 1 \text{ et } \sqrt{x} \leqslant y \leqslant 1\}$$
$$B = \{(x,y) \in \mathbb{R}^2 / 0 \leqslant x \leqslant 1 \text{ et } x^2 \leqslant y \leqslant \sqrt{x}\}$$
$$C = \{(x,y) \in \mathbb{R}^2 / 0 \leqslant x \leqslant 1 \text{ et } 0 \leqslant y \leqslant x^2\}$$

Ecrire un algorithme qui permet de simuler le choix de 10 000 points du carré ci-contre et ainsi d'estimer la probabilité d'être dans chacune des zones A, B et C.

Télécharger

Code: ***Un carré découpé en 3 zones***

```
from lycee import *
a,b,c=0,0,0
for i in range(10000):
    x,y=random(),random()
    if y>sqrt(x) : a=a+1
    elif y>x*x : b=b+1
    else : c=c+1
print "On est dans la zone A",a,"fois sur 10 000"
print "On est dans la zone B",b,"fois sur 10 000"
print "On est dans la zone C",c,"fois sur 10 000"
```



Si on a besoin de tirer un nombre décimal dans un autre intervalle que [0 ;1[, on utilisera la fonction suivante :

uniform(*min*,*max*) *lycee, random*

Cette fonction renvoie un nombre décimal de l'intervalle [*min* ; *max*[, choisi selon une densité uniforme sur cet intervalle.

Enfin, pour des tirages sur des nombres entiers (typiquement des lancers de dés), une fonction est déjà programmée :

randint(*min*,*max*) *lycee, random*

Renvoie un nombre entier de l'intervalle [*min* ; *max*], avec un tirage équiprobable.

Exemple (Extrait d'un exercice de Math O' Lycée) : Dans une urne contenant 100 boules numérotées de 1 à 100, on tire une boule au hasard, on note le résultat, on la remet, on en tire à nouveau une et on note le deuxième résultat. Le but du jeu étant d'obtenir un score de 36, laquelle de ces deux règles offre la plus grande probabilité de gagner ?

- Règle 1 : Le score est obtenu en calculant l'écart entre les deux résultats.
- Règle 2 : Le score est obtenu en conservant le plus petit des deux résultats.

Déterminer un programme simulant 10000 jeux et estimer les probabilités de gagner pour chacune des règles ? Que peut-on conjecturer ?

Télécharger

Code: ***100 boules***

```
from lycee import *
regle1,regle2=0,0
for i in range(10000):
    boule1=randint(1,100)
    boule2=randint(1,100)
    if boule1>boule2 :
        score1=boule1-boule2
        score2=boule2
    else :
        score1=boule2-boule1
        score2=boule1
    if score1==36 : regle1=regle1+1
    if score2==36 : regle2=regle2+1
print "Avec la règle 1, on a gagné",regle1,"fois sur 10 000."
print "Avec la règle 2, on a gagné",regle2,"fois sur 10 000."
```

Remarque:

D'autres fonctions existent dans le module `math` *de Python, vous pouvez retrouver l'intégralité des fonctions disponibles sur la page officielle. Le module* `math` *étant chargé en même temps que le module* `lycee`*, vous pouvez utiliser les fonctions de cette manière :* `math.fonction(...)`.

Chapitre 4

La tortue

Sommaire

Souvenirs peut-être du temps où on étudiait le langage LOGO à l'école... Le langage Python possède lui aussi une bibliothèque graphique qui permet de faire déplacer une tortue à l'écran, en lui donnant des instructions simples (avancer, tourner à gauche, ...). Programmer avec la tortue, c'est souvent l'occasion de travailler un peu avec de la géométrie classique ou repérée... Il peut aussi trouver à notre avis tout son intérêt au collège pour travailler sur les grands théorèmes (Calculs d'angles et de distances).

I) Avancer, reculer, tourner

tortue.forward(n) et tortue.back(n) *xturtle,lycee*

Fait respectivement avancer ou reculer la tortue dans la direction où elle regarde de n pas (n pouvant être entier ou non)

tortue.left(a) et tortue.right(a) *xturtle,lycee*

Fait respectivement tourner la tortue vers la gauche ou la droite de a degrés. (Aucun tracé n'est effectué, juste une rotation de la tête)

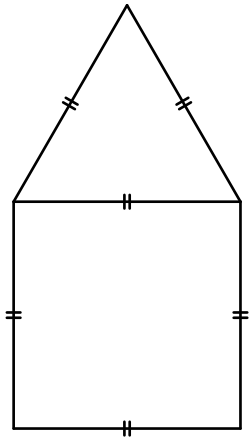
On peut commencer par demander une figure simple pour permettre aux élèves de comprendre le principe.

Télécharger

Code: ***Une maison.***

```
from lycee import *
tortue.forward(100)
tortue.right(90)
tortue.forward(100)
tortue.right(90)
tortue.forward(100)
tortue.right(90)
tortue.forward(100)
tortue.right(30)
tortue.forward(100)
tortue.right(120)
tortue.forward(100)
```

Représenter la figure ci-contre à l'aide de la tortue.



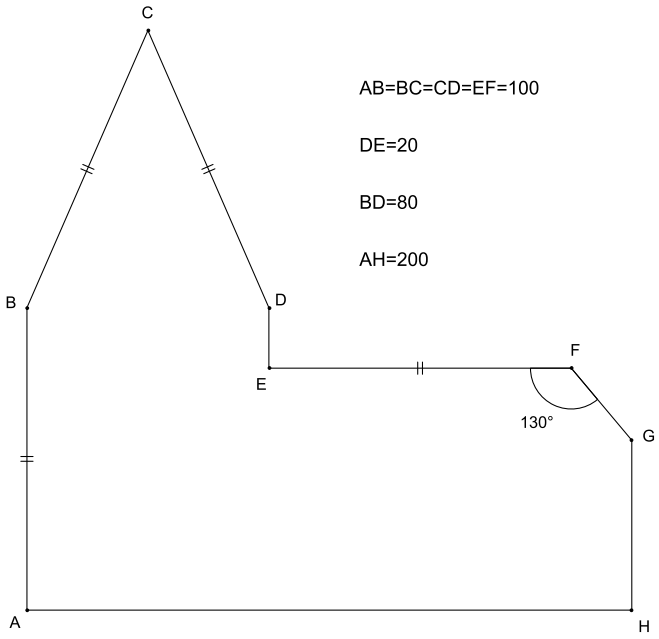
Puis enchaînez sur des figures où il est nécessaire de faire des calculs pour avancer :

Télécharger

Code: ***Une église.***

```
from lycee import *
tortue.reset()
tortue.left(90)
tortue.forward(100)
B=acos(40.0/100)
tortue.right(90-B)
tortue.forward(100)
C=180-2*B
tortue.right(180-C)
tortue.forward(100)
tortue.right(90-B)
tortue.forward(20)
tortue.left(90)
tortue.forward(100)
tortue.right(50)
FG=20/cos(40)
tortue.forward(FG)
tortue.right(40)
HG=80-sqrt(FG*FG-20*20)
tortue.forward(HG)
tortue.right(90)
tortue.forward(200)
```

Représenter la figure ci-contre à l'aide de la tortue.



Télécharger

Code: ***Tracer un polygone régulier à n côtés.***

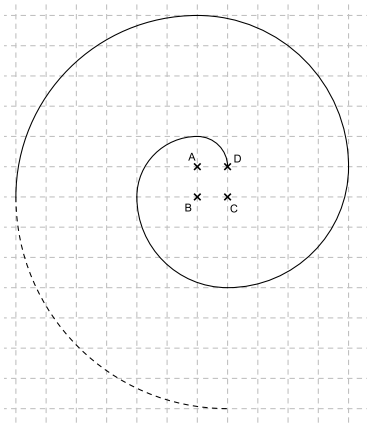
```
from lycee import *
n=demande("Nombre de côtés (au moins 3)")
for i in range(n):
    tortue.forward(50)
    tortue.left(360/n)
```

II) Tracer des cercles

tortue.circle(*rayon*) ou tortue.circle(*rayon*,*angle*) *xturtle,lycee*

- Si *rayon*>0 : Trace un cercle de rayon *rayon* à partir de la position de la tortue et en tournant dans le sens trigonométrique.
- Si *rayon* < 0 : Trace un cercle de rayon |*rayon*| dans le sens anti-horaire.
- Si *angle* est précisé, trace un arc de cercle de rayon |*rayon*| avec une ouverture de *angle* (en degré). Si *angle* n'est pas précisé, le cercle est tracé dans son intégralité.

La spirale ci-contre est obtenue en traçant bout à bout des quarts de cercle de centres successifs A, B, C et D.



Télécharger

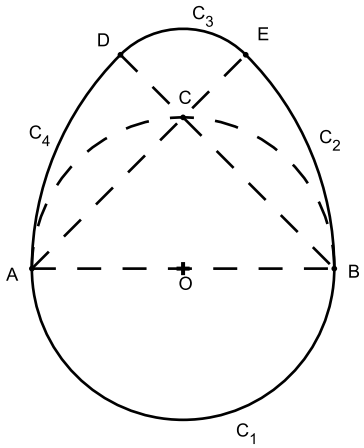
Code: ***Tracer une spirale.***

```
from lycee import *
n=10
for i in range(n):
    tortue.circle(5*i,90)
```

Remarque:

Attention, on ne connaît a priori pas le centre de ce cercle... c'est d'ailleurs ce qui peut faire l'intérêt de l'algorithme !

On veut tracer l'œuf de Pâques ci-contre.
Données : $OA = OB = OC = r$, C_1 est un demi-cercle de diamètre $[AB]$, C_2 est un arc de cercle de centre A et passant par B et E, C_3 est un arc de cercle de centre C et passant par E et D, enfin C_4 est un arc de cercle de centre B et passant par D et A.



Télécharger

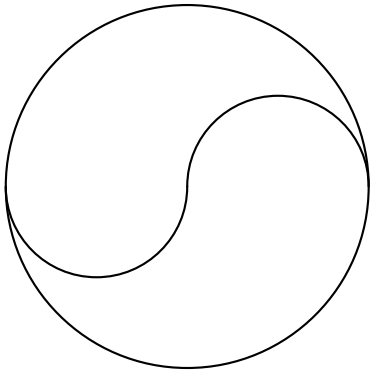
Code: ***Tracer un œuf de Pâques.***

```
from lycee import *
r=100
tortue.right(90)
tortue.circle(r,180)
tortue.circle(2*r,45)
tortue.circle(r*(2-sqrt(2)),90)
tortue.circle(2*r,45)
```

Télécharger

Code: ***Le yin et le yang.***

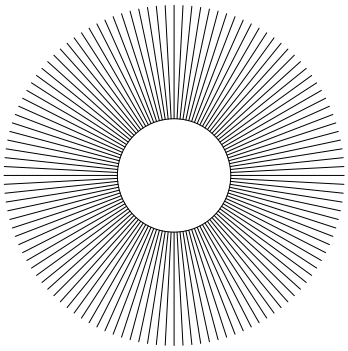
```
from lycee import *
r=100
tortue.up()
tortue.forward(r)
tortue.down()
tortue.left(90)
tortue.circle(2*r)
tortue.circle(r,180)
tortue.circle(-r,180)
tortue.hideturtle()
```



Télécharger

Code: ***Un soleil avec 120 rayons.***

```
from lycee import *
i=0
while i<120:
    tortue.right(90)
    tortue.forward(100)
    tortue.right(180)
    tortue.forward(100)
    tortue.right(90)
    tortue.circle(50,3)
    i=i+1
```



III) La tortue : Afficher, Cacher, Vitesse

tortue.showturtle() tortue.hideturtle() *xturtle,lycee*

A pour effet de respectivement cacher ou montrer la tortue à l'écran. Pour des questions d'esthétisme, on peut par exemple vouloir cacher la tortue en fin de tracé.

tortue.speed(v) *xturtle,lycee*

Permet de régler la vitesse de la tortue. v est un nombre entier entre 1 et 10, 1 étant la vitesse la plus lente et 10 la plus rapide.

IV) Le crayon : lever, baisser, taille, couleur

On peut pour certains dessins avoir besoin de déplacer la tortue sans laisser de trace.

tortue.up() tortue.down() *xturtle,lycee*

A pour effet de respectivement lever et baisser le crayon

tortue.pencolor(*texte*) ou tortue.pencolor(*rouge*,*vert*,*bleu*) *xturtle,lycee*

Définit la couleur du crayon.

- On peut entrer un texte entre guillemets parmi (entre autres) : 'aqua', 'beige', 'black', 'blue', 'brown', 'chocolate', 'fuchsia', 'gold', 'gray', 'green', 'indigo', 'khaki', 'maroon', 'orange', 'red', 'white', ...
- On peut aussi définir sa propre couleur en paramétrant les composantes *rouge*, *vert* et *bleu* de la couleur (chaque composante étant un nombre entre 0 et 255).

Télécharger

Code: ***Le drapeau européen.***

```
from lycee import *
tortue.pensize(2)
tortue.pencolor(230,230,40)
tortue.bgcolor('blue')
for etoile in range(12):
  tortue.down()
  for branche in range(5):
      tortue.forward(30)
      tortue.left(144)
  tortue.up()
  tortue.forward(50)
  tortue.left(30)
```

V) L'écran : effacer, colorer le fond, afficher un texte

tortue.reset() *xturtle,lycee*

Efface l'écran et repositionne la tortue dans sa position initiale

tortue.clear() *xturtle,lycee*

Efface l'écran mais la position du crayon reste inchangée

Remarque:

Notez que si la fenêtre de dessin n'est pas fermée et qu'un nouveau programme est exécuté, les tracés continuent sur la feuille actuelle à partir de la dernière position de la tortue. Il peut donc être préférable de commencer un nouveau programme par `tortue.reset()`.

tortue.write(*texte*) *xturtle,lycee*

Affiche le texte *texte* à l'emplacement de la tortue. Celle-ci ne se déplace pas lors de l'affichage.

Chapitre 5

Les graphiques

Sommaire

Si la tortue permet de revisiter la géométrie de collège, vous remarquerez rapidement que les possibilités graphiques sont très limitées pour un usage en classe. Nous vous présentons ici un nouvel objet inclus dans AmiensPython, l'objet **repere** qui va nous permettre de tracer des graphes de fonctions et autres diagrammes. Comme pour la tortue, les actions (appelées méthodes en programmation objet) seront exécutées par des commandes du type « repere.action »...

I) Placer des points, Afficher le repère, Les couleurs

repere.plot(*x*,*y*,*options*) *matplotlib,lycee*

Place dans la fenêtre graphique un point de coordonnées (*x*,*y*). Si aucune échelle n'est spécifiée, une échelle adaptée est proposée.
Le point est représenté avec la couleur et le style défini par la chaine *options*. Cette chaine comprend usuellement deux caractères : le premier étant une couleur, le deuxième le style, comme l'indique le tableau ci-dessous :

Couleur		Style	
b	bleu	-	ligne continue
g	vert	- -	tirets
r	rouge	:	pointillés
c	cyan	.	des points
m	magenta	o	des billes
y	jaune	x	des croix
k	noir	v	des triangles
w	blanc	-.	points-tirets
...		...	

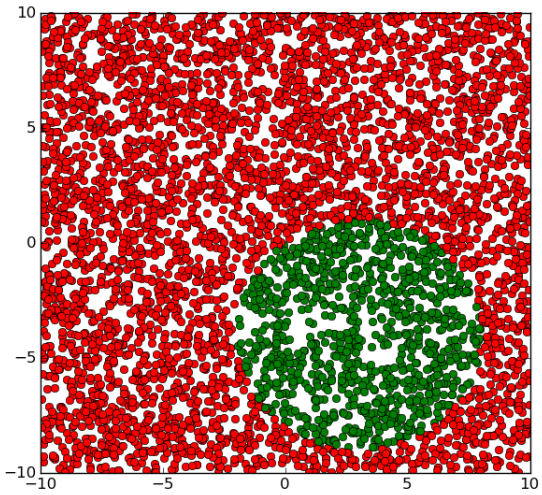
Exemple:
la commande ***repere.plot(2,3,'gx')*** *place un point sous forme de croix verte de coordonnées* (2,3).

Remarque:

Les options '-' ou '–' n'ont pas d'effet sur les points, mais seront utiles pour tracer des courbes, comme nous le verrons pas la suite.

Exemple:

On cherche, dans un repère orthonormé, tous les points de coordonnées (x,y) tels que $x*(6-x)<y*(8+y)$.



Télécharger

Code: ***Ligne de niveau.***

```
from lycee import *
for i in range(10000):
    x=uniform(-10,10)
    y=uniform(-10,10)
    if x*(6-x)<y*(8+y):
        repere.plot(x,y,'ro')
    else :
        repere.plot(x,y,'go')
repere.show()
```

Remarque:

Le programme se termine par l'instruction ***repere.show()*** *qui a pour effet, vous l'aurez déjà deviné, d'afficher la fenêtre.*

II) Nuage de points ou diagrammeXY

repere.plot(X,Y) *matplotlib,lycee*

Dessine le nuage de points de coordonnées (x_i,y_i) où les nombres x_i et y_i sont respectivement les éléments de la liste X et la liste Y.

De part la simplicité de l'utilisation des listes, il devient alors très facile de tracer la représentation graphique d'une fonction affine :

Télécharger

Code: ***Tracer d'une droite.***

```
from lycee import *
a,b=demande("Entrez le coefficient directeur et l'ordonnée à l'origine")
x = np.arange(-10, 10, 0.1)
repere.plot(x, a*x+b)
```

Remarque:

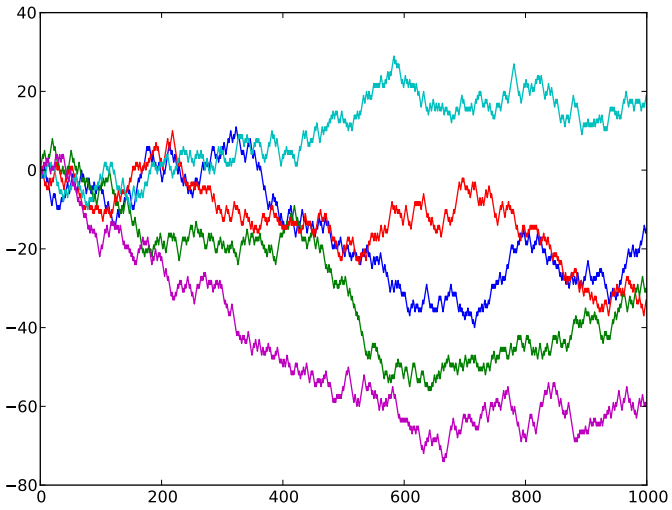
Précision technique : ici, on a recours à l'instruction `np.arange(debut,fin,pas)`*, qui crée une liste de valeurs entre* `debut` *et* `fin` *avec un pas pouvant être décimal (contrairement au* `range`*). Le gros avantage de cet objet, c'est que l'on peut alors effectuer des opérations comme* $x*x$ *ou* $3*x+4$*... (Consultez l'aide sur les objets de type NDArray de la bibliothèque numpy pour plus d'informations.)*

Autre exemple : une puce située à l'origine d'un axe gradué effectue 1 000 sauts successifs. A chaque saut, elle avance ou recule aléatoirement d'une unité sans préférence pour un sens ou l'autre. Représentez le chemin parcouru par la puce.

Télécharger

Code: ***Marche aléatoire.***

```
from lycee import *
p,x,y=0,[],[]
for i in range(1000):
    x.append(i)
    y.append(p)
    if randint(0,1)==0:
        p=p+1
    else :
        p=p-1
repere.plot(x,y)
repere.show()
```



Remarque:

Ici le programme donné ne trace qu'une seule courbe, mais si vous décidez de ne pas fermer la fenêtre graphique et de relancer le programme, les graphiques se superposent et une échelle adaptée est proposée ce qui représente un apport pédagogique. Le paragraphe suivant vous explique néanmoins comment forcer le nettoyage de la fenêtre en début de programme.

III) Les axes et la grille

repere.clf() *matplotlib,lycee*

Efface et ré-initialise le contenu de la fenêtre graphique.

Par défaut, si aucune échelle n'est imposée, celle-ci est calculée automatiquement. Quelquefois, il peut s'avérer nécessaire de la modifier manuellement.

repere.axis($[x_{min},x_{max},y_{min},y_{max}]$**)** *matplotlib,lycee*

Impose une nouvelle échelle au repère.

Vous pouvez aussi choisir d'afficher une grille ou non en plus :

repere.grid(*mode*) *matplotlib,lycee*

si le booléen *mode* vaut *True*, la grille est affichée, si *mode* vaut *False*, elle est masquée, comme c'est le cas par défaut.

IV) Titres et légendes

repere.title(*texte*) *matplotlib,lycee*

Ajoute le titre *texte* au graphique.

repere.xlabel(*texte*) et repere.ylabel(*texte*) *matplotlib,lycee*

Affiche le texte *texte* sur l'axe des abscisses ou l'axe des ordonnées.

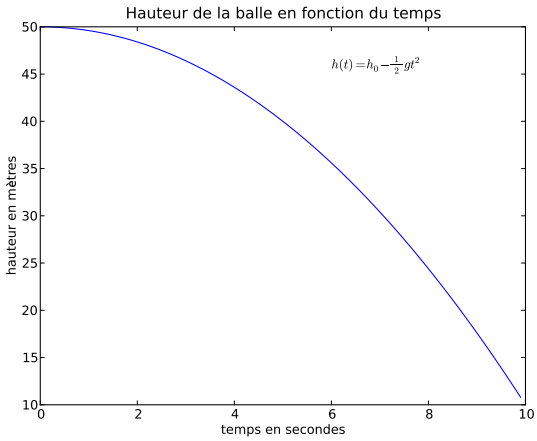
repere.text(*x*,*y*,*texte*) *matplotlib,lycee*

Affiche le texte *texte* sur le graphique à la position (x,y).

Télécharger

Code: ***Chute d'une balle.***

```
from lycee import *
t = np.arange(0, 10, 0.1)
repere.plot(t,50-0.5*0.8*t**2)
repere.title('Hauteur de la balle
                en fonction du temps')
repere.ylabel(u"hauteur en mètres")
repere.xlabel(u"temps en secondes")
repere.text(6,45,r'$h=h_0-
                    \frac{1}{2}gt^2$')
repere.show()
```



Remarque:

Vous aurez remarqué qu'en ajoutant un r à la chaine de texte, on peut faire afficher des formules mathématiques pour peu que l'on connaisse un minimum la syntaxe LaTeX... Enfin, on est davantage dans la décoration qu'autre chose !

V) Repères multiples

Il existe une instruction permettant de fractionner la fenêtre graphique en plusieurs sous graphiques. C'est peut-être un peu complexe pour des élèves, nous vous laissons donc juger de l'utilité de leur présenter cette fonction.

repere.subplot(n)

matplotlib,lycee

Fragmente la fenêtre graphique en plusieurs repères. n est un nombre entier de 3 chiffres construit ainsi :
- Le chiffre des centaines indique le nombre de lignes à créer.
- Le chiffre des dizaines indique le nombre de colonnes à créer.
- Le chiffre des unités indique quel est le repère actif.

Exemple:

L'instruction ***repere.subplot(234)*** *découpe la fenêtre graphique en 6 parties numérotées ainsi :*

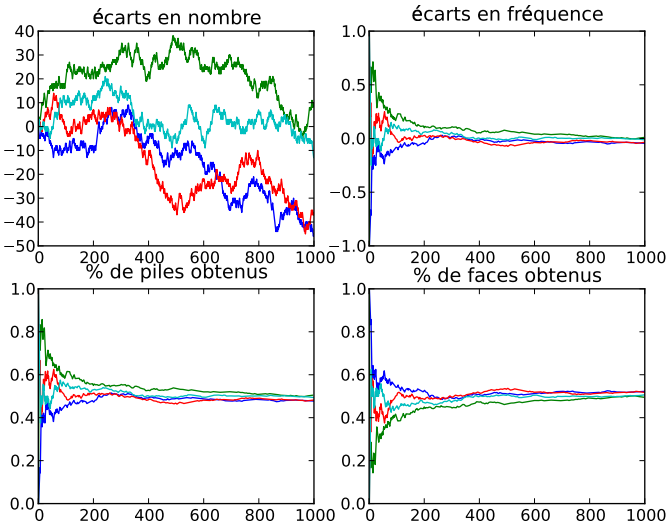
1	*2*	*3*
4	*5*	*6*

Pour l'enseignant cela peut être pertinent avec le vidéoprojecteur pour donner du sens à la notion de probabilité comme dans l'exemple ci-dessous : faire apparaître ce qui est convergent et ce qui est chaotique...

Télécharger

Code: ***Pile ou face.***

```
from __future__ import division
from lycee import *
p,f,e=0,0,0
n,ecart,ecartm,freqP,freqF=[],[],[],[],[]
for i in range(1,1000):
    if randint(0,1)==0:
        p=p+1
    else :
        f=f+1
    e=p-f;
    n.append(i)
    ecart.append(e)
    ecartm.append(e/i)
    freqP.append(p/i)
    freqF.append(f/i)
repere.subplot(221)
repere.plot(n,ecart)
repere.title("écarts en nombre")
repere.subplot(222)
repere.plot(n,ecartm)
repere.title("écarts en
                    fréquence")
repere.subplot(223)
repere.plot(n,freqP)
repere.title("% de piles obtenus")
repere.subplot(224)
repere.plot(n,freqF)
repere.title("% de faces obtenus")
repere.show()
```



Chapitre 6

Les listes

Sommaire

I) Définition

Une liste est une suite d'éléments numérotés dont le premier indice est 0. Une liste n'a donc (presque) pas de limite de taille. Les listes existent dans le langage Python, sans besoin de la bibliothèque `lycee`.

Python affiche la liste comme un « vecteur ligne »dont les composantes sont les éléments de cette liste séparés par une virgule. La liste est délimitée par des crochets.

Pour atteindre l'élément d'indice i de la liste L, il suffit d'écrire `L[i]`.

L'exemple suivant permet de démontrer que tous les ans, il y a au moins un vendredi 13 dans l'année.

Télécharger

Code: ***Vendredi 13***

```
from __future__ import division
from lycee import *
jours=["Lundi","Mardi","Mercredi","Jeudi","Vendredi","Samedi","Dimanche"]
mois=["Janvier","Février","Mars","Avril","Mai","Juin","Juillet","Août","Septembre","Oct
nbjours=[31,28,31,30,31,30,31,31,30,31,30,31]
for jan in range(7):
    print "Si le 13 Janvier est un",jours[jan]," : ",
    m=0;j=jan;
    while reste(j,7)<>4 and m<12:
        j=j+nbjours[m]
        m=m+1;
    if reste(j,7)==4:
        print "Le 13",mois[m],"sera un Vendredi 13"
    else :
        print "il n'y a pas de Vendredi 13"
```

Aperçu du résultat dans la console

```
Si le 13 Janvier est un Lundi  :  Le 13 Juin sera un Vendredi 13
Si le 13 Janvier est un Mardi  :  Le 13 Février sera un Vendredi 13
Si le 13 Janvier est un Mercredi  :  Le 13 Août sera un Vendredi 13
Si le 13 Janvier est un Jeudi  :  Le 13 Mai sera un Vendredi 13
Si le 13 Janvier est un Vendredi  :  Le 13 Janvier sera un Vendredi 13
Si le 13 Janvier est un Samedi  :  Le 13 Avril sera un Vendredi 13
Si le 13 Janvier est un Dimanche  :  Le 13 Septembre sera un Vendredi 13
```

Les éléments d'une liste peuvent être des nombres, du texte, voire des listes eux-mêmes. On peut aussi mélanger les types d'objets dans une même liste comme ci-dessous, lorsque cela semble utile.

Ici on propose de stocker dans une liste $[A,x_A,y_A,B,x_B,y_B,C,x_C,y_C,D,x_D,y_D]$ et l'on teste si $ABCD$ est un parallélogramme.

Télécharger

Code: ***Reconnaître un parallélogramme***

```
from __future__ import division
from lycee import *
quad=['C',1,2,'O',4,3,'Q',5,0,'P',2,-3]
if quad[4]-quad[1]==quad[7]-quad[10] and quad[5]-quad[2]==quad[8]-quad[11]:
    print "Le quadrilatère "+quad[0]+quad[3]+quad[6]+quad[9]+" est un parallélogramme."
else:
    print "Le quadrilatère "+quad[0]+quad[3]+quad[6]+quad[9]+" n'est pas un parallélogr
```

Aperçu du résultat dans la console

```
>>> Le quadrilatère COQP n'est pas un parallélogramme.
```

Remarque:

ATTENTION*! Respectez bien la syntaxe entre crochets pour écrire votre liste. Il existe en effet un autre objet en Python : le tuple (N-uplet) qui se note entre parenthèses, il ressemble comme deux gouttes d'eau à la liste à ceci près par exemple que l'on ne peut pas modifier les éléments. Nous ne parlerons pas de cet objet dans cet ouvrage.*

On peut connaître la taille de la liste avec la fonction `len(L)`.

len(*list*)

Renvoie le nombre d'éléments de la liste *list*

liste_demande(*prompt*) *lycee*

Ouvre une fenêtre contenant le texte *prompt*, attend une liste dont les éléments sont séparés par des virgules et renvoie cette liste.

Télécharger

Code: ***produit de 2 polynômes***

```
from __future__ import division
from lycee import *
A=liste_demande('entrez les coefficients de A(x) par ordre des puissances croissantes')
B=liste_demande('entrez les coefficients de B(x) par ordre des puissances croissantes')
DegA,DegB=len(A)-1,len(B)-1
C=[]
for degre in range(DegA+DegB+1) :
    k,coef=0,0
    while k<=degre :
        if k<=DegA and degre-k<=DegB :
            coef=coef+A[k]*B[degre-k]
        k=k+1
    C.append(coef)
print'('+affiche_poly(A)+')('+affiche_poly(B)+')='+affiche_poly(C)
```

Aperçu du résultat dans la console

```
(1+2X+3X^2)(3-X+2X^2)=3+5X+9X^2+X^3+6X^4
```

Notez la présence de la fonction `affiche_poly` qui transforme une liste en un polynôme pour plus de lisibilité. Cette fonction n'existe pas dans le langage Python d'origine :

affiche_poly(L) *lycee*

Renvoie une chaîne de caractères représentant la liste L sous forme d'un polynôme,$L[n]$ étant le coefficient de degré n.
Par exemple `affiche_poly`([1,0,4]) donne $1+4X^2$

II) Créer une liste, ajouter des éléments

Il existe différentes manières de créer des listes :

- En définissant la liste en extension : On liste les éléments, par exemple les jours de la semaine dans l'exemple du vendredi 13.
- Avec l'instruction range que nous avons déjà vue en utilisant les boucles `for`.
- Avec une formule, lorsque l'on connaît une expression explicite des éléments de la liste, exemple pour obtenir la liste des 10 premiers carrés parfaits : `L` = [`i * i for i in range(10)`]

- Enfin, on peut partir d'une liste existante ou vide (notée : [ ]) et ajouter les éléments au fur et à mesure, grâce aux instructions `append` et `insert` détaillées ci-dessous :

list.**append**(*a*)

Ajoute un élément *a* à la liste *list* en fin de liste

Télécharger

Code: ***Conjecture d'Euler***

La conjecture d'Euler (cas $n = 5$) : Elle fut énoncée par le mathématicien suisse Leonhard Euler en 1769 et peut dans le cas $n = 5$ s'énoncer ainsi : « on ne peut pas trouver 4 nombres entiers tels que la somme de chacun de ses nombres à la puissance 5 soit à son tour la puissance de 5 d'un nombre entier ». Cette conjecture fut infirmée par L. J. Lander et T. R. Parkin en 1966 grâce au contre-exemple suivant : $27^5 + 84^5 + 110^5 + 133^5 = 144^5$. L'exemple suivant cherche et trouve ce cas (Attention, si vous exécutez le programme c'est un peu long).

```
from __future__ import division
from lycee import *
L=[puissance(i,5) for i in range(1,5)]
max=5
trouve=0
while trouve==0:
  L.append(puissance(max,5))
  ind1=0;
  while ind1<max and trouve==0:
    ind2=ind1
    while ind2<max and trouve==0:
      ind3=ind2
      while ind3<max and trouve==0 :
        ind4=ind3
        while ind4 <max and trouve==0:
          if L[ind1]+L[ind2]+L[ind3]+L[ind4]==L[max-1]:
            print ind1+1,"^5+",ind2+1,"^5+",ind3+1,"^5+",ind4+1,"^5=",max,"^5"
            print L[ind1],"+",L[ind2],"+",L[ind3],"+",L[ind4],"=",L[max-1]
            trouve=1
          ind4=ind4+1
        ind3=ind3+1
      ind2=ind2+1
    ind1=ind1+1
  max=max+1
  print "max=",max
```

```
27 ^5+ 84 ^5+ 110 ^5+ 133 ^5= 144 ^5
14348907 + 4182119424 + 16105100000 + 41615795893 = 61917364224
```

Si vous avez besoin d'insérer un élément à un autre endroit qu'en fin de liste, on utilisera la fonction suivante (en particulier pour insérer en début de liste) :

list.**insert**(*j*,*a*)

Insère l'élément *a* au rang *j* de la liste *list*

Télécharger

Code: ***Nombre de Kaprekar***

```
from __future__ import division
from lycee import *
n=demande("Entrez un nombre entier strictement positif")
N=n*n
L=[]
trouveG,trouveD=-1,-1
while N<>0 :
    L.insert(0,reste(N,10))
    N=quotient(N,10)
for i in range(len(L)) :
    gauche=0
    droite=0
    for j in range(i) :
        gauche=gauche*10+L[j]
    for j in range(i,len(L)) :
        droite=droite*10+L[j]
    if gauche+droite==n :
        trouveG,trouveD=gauche,droite
if trouveG>-1:
    print n,"est nombre de Kaprekar",
    print n,"²=",n*n," et ",n,"=",trouveG,"+",trouveD
else :
    print n,"n'est pas nombre de Kaprekar"
```

Un nombre de Kaprekar est un nombre qui, lorsqu'il est élevé au carré, peut être séparé en une partie gauche et une partie droite (non nulle) telles que la somme donne le nombre initial. Exemple : 4879 est un nombre de Kaprekar car,

$$4879^2 = 23804641$$
et
$$238 + 04641 = 4879$$

```
13 n'est pas nombre de Kaprekar
703 est nombre de Kaprekar 703 ²= 494209  et  703 = 494 + 209
```

En réalité, l'utilisation de listes trouve son intérêt quand on a besoin de conserver les valeurs pour la suite du programme (extraire une moyenne, réutiliser des valeurs, ...). A vous de trouver l'équilibre entre lisibilité du programme et performance de celui-ci.

Par exemple : Que dire de la suite (u_n) définie par $u_0 = 2$, $u_1 = 5$ et pour tout entier positif n, $u_{n+1} = \sqrt{u_{n-1} \times u_n + 9(u_n + 1)}$?

On peut choisir soit d'utiliser une liste pour conserver toutes les valeurs, ou bien se limiter à deux variables `a` et `b` qui contiendront à chaque étape les valeurs de u_{n-1} et u_n :

Télécharger

Code: ***Suite avec une liste***

```
from __future__ import division
from lycee import *
U=[2,5]
n=1
while n<20 :
    U.append(sqrt(U[n-1]*U[n]+9*(U[n]+1)))
    n=n+1
print U
```

```
[2, 5, 8.0, 11.0, 14.0, 17.0, 20.0, 23.0 ...
```

Télécharger

Code: ***Suite sans liste***

```
from __future__ import division
from lycee import *
a,b,n=2,5,1
while n<=20 :
    print a,
    a,b=b,sqrt(a*b+9*(b+1))
    n=n+1
```

```
2 5 8.0 11.0 14.0 17.0 20.0 23.0 ...
```

Dans l'exemple suivant, l'utilisation d'une liste semble difficilement évitable dans la mesure où le calcul de u_{n+1} fait intervenir tous les termes précédents :

On considère la suite (u_n) définie par $u_0 = 2$ et pour tout entier naturel n,

$$\begin{aligned} u_{n+1} &= u_n - 2u_{n-1} + 3u_{n-2} - \cdots (-1)^{n+1}u_0 \\ &= \sum_{k=0}^{n} (-1)^{k+1} u_{n-k} \end{aligned}$$

a) Calculer les 20 premiers termes de cette suite à l'aide d'un algorithme.

b) Que peut-on conjecturer ? Le prouver.

c) Peut-on généraliser ce résultat pour u_0 quelconque ?

Télécharger

Code: ***Suite***

```
from __future__ import division
from lycee import *
U=[2]
n=1
while n<20 :
  k=1
  total=0
  while k<=n :
    total=total+puissance(-1,k+1)*k*U[n-k]
    k=k+1
  U.append(total)
  n=n+1
print U
```

```
[2, 2, -2, 0, 2, -2, 0, 2, -2, 0, ...
```

III) Retirer des éléments

list.**remove**(*e*)

Supprime la première occurrence de l'élément *e* dans la liste *list*

Indication:

Attention, si vous essayez de retirer un élément d'une liste alors qu'il n'en fait pas partie, un message d'erreur sera renvoyé par la console python :

```
Traceback (innermost last):
File "<interactive input>", line 1, in ?
ValueError: list.index(x): x not in list
```

Il peut être pertinent de tester si l'élément appartient à une liste à l'aide de la fonction suivante :

e **in** *list*

Teste si l'élément *e* est dans la liste *list* et renvoie VRAI ou FAUX

Le Crible d'Ératosthène permet de lister la liste des premiers nombres premiers.

Télécharger

Code: ***Crible d'Ératosthène***

```
from __future__ import division
from lycee import *
N=demande("Obtenir les nombres premiers inférieurs à ?")
indice=0
liste=range(2,N)
while indice<len(liste):
    nombre=liste[indice]
    n=2
    while n*nombre<N:
        multiple=n*nombre
        if multiple in liste:
            liste.remove(multiple)
        n=n+1
    indice=indice+1
print liste
```

1	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50
51	52	53	54	55	56	57	58	59	60
61	62	63	64	65	66	67	68	69	70
71	72	73	74	75	76	77	78	79	80
81	82	83	84	85	86	87	88	89	90
91	92	93	94	95	96	97	98	99	100

Vous pouvez aussi savoir qu'un élément a **n'est pas** dans la liste L avec le test `a not in L`
Enfin, si vous souhaitez supprimer le $i^{ème}$ élément d'une liste, il faut utiliser la commande `pop` :

***list*.pop(*i*)**

Supprime l'élément d'indice i de la liste *list*.

Télécharger

Code: ***Polynome dérivé***

```
from __future__ import division
from lycee import *
P=liste_demande("Entrez les coef du polynome, séparés par des virgules")
print "Le polynôme dérivé de",affiche_poly(P),"est",
d=1
while d<len(P):
    P[d]=d*P[d]
    d=d+1
P.pop(0)
print affiche_poly(P)
```

Aperçu du résultat dans la console

```
Le polynôme dérivé de 1+4X-3X^2+7X^4 est 4-6X+28X^3
```

IV) Rechercher, Compter, Ordonner

***list*.index(*e*)**

Renvoie l'indice de la première occurrence de l'élément e dans la liste *liste*. Si l'élément e n'est pas dans la liste, une erreur se produit, pensez donc à tester avec la commande `in` la présence de l'élément dans la liste.

list.**count**(*e*)

Renvoie le nombre d'occurrences de la valeur *e* dans la liste *list*

Télécharger

Code: ***Compter le nombre de 6***

```
from __future__ import division
from lycee import *
De=[]
for i in range(1000) :
    De.append(randint(1,6))
n=De.count(6)
print De
print 'Sur 1 000 lancers, vous avez obtenu',n,'fois le chiffre 6.'
if n>0 :
    print 'Le premier 6 a été obtenu au tirage numéro',De.index(6)+1
```

Aperçu du résultat dans la console

```
[1, 5, 3, 3, 6, 3, 5, 3, 1, 1, 6, 1, 1, 3, 6, 6....
Sur 1 000 lancers, vous avez obtenu 177 fois le chiffre 6.
Le premier 6 a été obtenu au tirage numéro 5
```

list.**sort**()

Ordonne la liste *list* (par ordre croissant des valeurs)

sorted(*list*)

Retourne une liste ordonnée qui contient les éléments d'une liste de départ *list*

Remarque:

Notez bien la différence entre `L.sort()` *qui modidifie la liste* `L` *et* `sorted(L)` *qui renvoie la liste triée, mais* `L` *n'est pas modifiée.*

Un exemple avec l'algorithme de Kaprekar : On considère un nombre N de 4 chiffres (éventuellement des 0) que l'on transforme en un nombre $N = n_2 - n_1$ où n_1 est le nombre formé des 4 chiffres de n triés dans l'ordre croissant et n_2 celui formé des 4 chiffres de n triés dans l'ordre décroissant.

On crée une liste des entiers de 0 à 9999. à chaque étape, on applique l'algorithme de Kaprekar à tous les éléments de la liste. Combien la liste comporte-elle de nombres distincts

a) Après le 1er passage ?

b) Après le 2ème passage ?

c) Après le 5ème passage ?

d) Après le 10ème passage ?

e) Après le 100ème passage ?

Télécharger

Code: ***Algorithme de Kaprekar***

```
from __future__ import division
from lycee import *
L=range(10000)
for passage in range(1,20):
    L1=[]
    for n in L :
        d=reste(n,10)
        c=reste(quotient(n,10),10)
        b=reste(quotient(n,100),10)
        a=quotient(n,1000)
        a,b,c,d=sorted([a,b,c,d])
        n1=a*1000+b*100+c*10+d
        n2=d*1000+c*100+b*10+a
        nindex{liste3=n2-n1
        if not n3 in L1 :
            L1.append(n3)
    print "Passage",passage,":",len(L1),"possibilités",L1
    L=L1
```

Indication:

Remarque importante *: La dernière ligne du programme* `L = L1` *indique que l'adresse de la liste* `L` *est celle de la liste* `L1`*, mais ne recopie pas celle-ci, ce qui n'a pas ici d'importance, car une nouvelle liste* `L1` *est créée à l'itération suivante.*
En effet, contrairement aux autres variables étudiées jusqu'à présent, les listes (dont on ne connaît pas la taille a priori) sont des pointeurs (on utilise plutôt le terme d'alias en Python) et pointent donc vers des adresses mémoire. Ainsi, si vous indiquez dans un programme `L1 = L2`*, celui-ci comprendra que les listes* `L1` *et* `L2` *sont les mêmes objets, dans ce cas modifier* `L2` *modifiera* `L1` *et inversement. Pour pallier ce problème, vous devrez saisir* `L2 = L1[:]` *pour obtenir une copie indépendante de la liste* `L1`*.*

Télécharger

Code: ***copie de l'adresse d'une liste***

```
from __future__ import division
from lycee import *
L1=["Bleu","Blanc","Rouge"]
L2=L1
L2[1]="Vert"
print "L1=",L1
print "L2=",L2
```

```
L1= ['Bleu', 'Vert', 'Rouge']
L2= ['Bleu', 'Vert', 'Rouge']
```

Télécharger

Code: ***copie du contenu d'une liste***

```
from __future__ import division
from lycee import *
L1=["Bleu","Blanc","Rouge"]
L2=L1[:]
L2[1]="Vert"
print "L1=",L1
print "L2=",L2
```

```
L1= ['Bleu', 'Blanc', 'Rouge']
L2= ['Bleu', 'Vert', 'Rouge']
```

Un dernier exemple qui permet de calculer les coefficients binomiaux grâce au triangle de Pascal. Ici on a recours à deux listes : la `listeA` contient la liste au rang $n-1$ et la `listeB` celle au rang n.
Comme vous le verrez dans le chapitre suivant, on peut heureusement calculer les coefficients $\begin{pmatrix} n \\ p \end{pmatrix}$ à l'aide de la fonction `binomial`.

Télécharger

Code: ***Triangle de Pascal***

```
from __future__ import division
from lycee import *
n=demande('quelle puissance ?')
listeA=[1,0]
print [1]
listeB=listeA
for l in range (2,n+2):
    listeA=listeB[:]
    for i in range (1,l):
        listeB[i]=listeA[i-1]+listeA[i]
    print listeB
    listeB.append(0)
```

V) Opérations sur les listes

L1 + L2

Renvoie une liste contenant les éléments de la liste $L1$ suivis des éléments de la liste $L2$.

On peut par exemple l'utiliser pour concaténer différents échantillons (voir partie statistiques et probabilités).

L ∗ n

Renvoie une liste contenant les éléments de la liste L répétés n fois.

On utilise souvent cette instruction pour initialiser une liste. Par exemple, pour simuler une urne contenant 5 boules vertes et 3 rouges, on pourra taper : `Urne` = ["`Verte`"] ∗ `5` + ["`Rouge`"] ∗ `3`. (Cf le chapitre statistiques et probabilités.)

Ce dernier exemple implémente la méthode de Hörner, autrefois étudiée en première S. Cette méthode propose un algorithme pour factoriser un polynôme lorsque l'on en connaît une racine. Le schéma ci-dessous illustre le cas du polynôme $X^3-7X^2+17X-14$ dont une racine évidente est 2. (Pour plus d'informations, nous vous invitons à lire quelques explications de Serge SAUTON) :

$a=2$

1	−7	17	−14
	×a → 2	+ −10	14
1	−5	7	0

$$X^3-7X^2+17X-14$$
$$=(X-2)(X^2-5X+7)$$

Télécharger

Code: ***Méthode de Hörner***

```
from __future__ import division
from lycee import *
P=liste_demande('entrez les coefficients de P(x) par ordre des puissances croissantes')
r=demande('Entrez une racine évidente')
Q=[0]*(len(P)-1)
v=0
for d in range(len(P)-2,-1,-1):
    v=P[d+1]+r*v
    Q[d]=v
print affiche_poly(P)+'=('+affiche_poly([-r,1])+')('+affiche_poly(Q)+')';
```

Aperçu du résultat dans la console

```
1+X+X^2+X^3=(1+X)(1+X^2)
```

Chapitre 7

Probabilités et statistiques

Sommaire

L'une des principales utilisations que nous pouvons faire d'une liste de valeurs est le traitement statistique de celle-ci. Nous avons vu comment travailler sur les listes. A présent, découvrons les outils statistiques que nous avons à notre disposition.

Ces fonctions ont été ajoutées dans le module *lycee* d'AmiensPython. Ce ne sont donc pas des fonctions de base de Python, et vous devrez donc les recréer ou les recopier si vous souhaitez les utiliser avec un autre environnement.

I) Statistiques descriptives

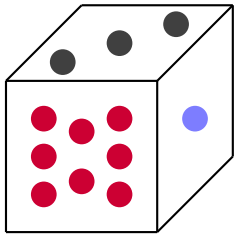
Pour commencer, nous allons reprendre les opérations statistiques étudiées au collège et complétées au lycée.

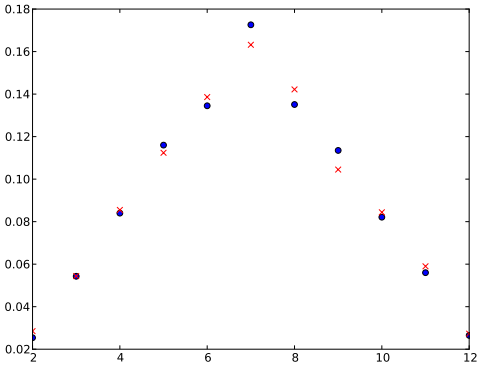
compte(*L*,*option*) *lycee*

Renvoie la liste *liste* ordonnée par ordre croissant sans doublons et (éventuellement) :

- Si *option* vaut `'effectif'`, la liste des effectifs correspondant.
- Si *option* vaut `'frequence'`, la liste des fréquences correspondante.

Les dés de Sicherman sont deux dés à 6 faces : les faces du premier sont consituées des numéros 1, 2, 2, 3, 3 et 4 et celles du second 1, 3, 4, 5, 6 et 8. Lorsque l'on jette ces 2 dés et que l'on ajoute les résultats des faces, on obtient non seulement les mêmes possibilités qu'avec un dé classique (de 2 à 12), mais avec les mêmes fréquences d'apparition ! Le programme ci-dessous lance mille fois deux dés de Sicherman ainsi que deux dés classiques et compare graphiquement les fréquences des sommes obtenues. *(Sur une idée de Jean-Philippe BLAISE).*





Télécharger

Code: ***Dés de Sicherman***

```
from __future__ import division
from lycee import *
De1=[1,2,2,3,3,4]
De2=[1,3,4,5,6,8]
L1=[];L2=[]
for i in range(10000):
    L1.append(randint(1,6)+randint(1,6))
    L2.append(De1[randint(0,5)]+De2[randint(0,5)])
X,F=compte(L1,'frequence')
repere.plot(X,F,'bo')
X,F=compte(L2,'frequence')
repere.plot(X,F,'rx')
```

Ce précédent exemple pourrait être ensuite repris et adapté pour répondre aux 2 défis suivants :
Défi 1 : Imaginer un programme pour déterminer la loi de probabilité. (Une preuve papier/crayon est ici largement pertinente.)
Défi 2 : Existe-t-il d'autres dés à 6 faces composées d'entiers naturels et ayant les mêmes propriétés ? Imaginer un programme pour répondre à cette question.

moyenne(*xi*,*ni*) *lycee*

Renvoie la moyenne arithmétique, il existe 3 manières d'utiliser cette fonction :

- Soit on indique une liste *xi* de données brutes
- Soit on indique une liste *xi* de valeurs et une liste *ni* de même taille correspondant aux effectifs.
- Soit on indique une liste *xi* **ordonnée** de taille N et une liste *ni* de taille $N-1$, dans ce cas, la liste *xi* est considérée comme les extrémités de classes et d'effectifs la liste *ni*. La moyenne est alors calculée en prenant les centres de ces classes.

mediane(*xi*,*ni*,*option*) *lycee*

Renvoie la valeur médiane de la série *xi* éventuellement pondérée des effectifs *ni* correspondants si celle-ci est donnée.
Vous pouvez aussi préciser un dernier paramètre *option* précisant le mode de calcul :

- Si *option* vaut 1 ou n'est pas précisé, la médiane est la valeur centrale de la série (valeur centrale de la série ou moyenne arithmétique des deux valeurs centrales)
- Si le paramètre *option* est 2, la médiane est alors la valeur pour laquelle on dépasse 50% des valeurs.
- la fonction `mediane` n'est pas prévue pour fonctionner avec des classes, mais seulement des valeurs discrètes.

ECC(*xi*,*ni*) et FCC(*xi*,*ni*) *lycee*

Renvoient la liste *xi* ordonnée et respectivement la liste formée des effectifs ou fréquences cumulés croissants de la liste *xi*. Si une liste *ni* (de même taille) est précisée les effectifs cumulés croissants sont calculés en tenant compte des effectifs *ni*.
Remarque : Si aucune liste *ni* n'est précisée, les données *xi* n'ont pas besoin d'être ordonnées.

quartile(xi,ni,val) *lycee*

xi est une série éventuellement pondérée des effectifs ni correspondants. Selon les valeurs du paramètre val :

- Si $val = 1$ renvoie le premier quartile de la série.
- Si $val = 3$ renvoie le troisième quartile de la série.
- Si val n'est pas précisé, renvoie le premier et troisième quartile.

decile(xi,ni,val) *lycee*

xi est une série éventuellement pondérée des effectifs ni correspondants. Selon les valeurs du paramètre val :

- Si $val = 1$ renvoie le premier décile de la série.
- Si $val = 9$ renvoie le neuvième décile de la série.
- Si val n'est pas précisé, renvoie le premier et neuvième décile.

En Novembre 1976 dans un comté du sud du Texas, Rodrigo Partida était condamné à huit ans de prison. Il attaqua ce jugement au motif que la désignation des jurés de ce comté était discriminante à l'égard des Américains d'origine mexicaine. Alors que 79,1% de la population de ce comté était d'origine mexicaine, sur les 870 personnes convoquées pour être jurés lors d'une certaine période de référence, il n'y eut que 339 personnes d'origine mexicaine. *(D'après un énoncé du document ressources pour faire la classe en mathématiques au lycée professionnel)* .

Enoncé : Effectuer 1000 simulations sur un tirage de 870 personnes dans une population composée à 79,1% de mexicains et donner l'intervalle interdéciles. Que peut-on en conclure ?

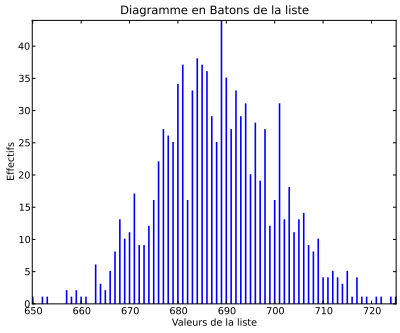
Télécharger

Code: ***Appel de jugement***

```
from __future__ import division
from lycee import *
Nb=[]
for i in range(1000):
    Mexicain=0
    for n in range(870):
        if random()<0.791 :
            Mexicain=Mexicain+1
    Nb.append(Mexicain)
dec1,dec9=decile(Nb)
print "80% des valeurs se situent entre",
print dec1,'et',dec9
```

Remarque:

Comme on le verra quelques pages plus loin, en ajoutant simplement `baton(Nb)` *en fin de programme, on peut visualiser la distribution.*



variance(xi,ni) & ecartype(xi,ni) *lycee*

Renvoie respectivement la variance et l'écart type des valeurs de la série xi éventuellement pondérée des effectifs ni correspondants si la liste ni est donnée. On peut aussi travailler avec des classes en donnant la liste des valeurs xi contenant alors une valeur de plus que la liste des effectifs ni. Dans ce cas, le calcul est effectué en utilisant les centres des classes.

II) Simulation et échantillonnage

binomial(n,p) *lycee*

Retourne le coefficient binomial $\binom{n}{p}$, c'est-à-dire le nombre de chemins de l'arbre réalisant p succès pour n répétitions.

(D'après un document de l'inspection générale) : Monsieur Z, chef du gouvernement d'un pays lointain, affirme que 52% des électeurs lui font confiance. On interroge 100 électeurs au hasard (la population est suffisamment grande pour considérer qu'il s'agit de tirages avec remise) et on souhaite savoir à partir de quelles fréquences, au seuil de 5%, on peut mettre en doute le pourcentage annoncé par Monsieur Z, dans un sens, ou dans l'autre.

On fait l'hypothèse que Monsieur Z dit vrai et que la proportion des électeurs qui lui font confiance dans la population est $p = 0{,}52$. La variable aléatoire X, correspondant au nombre d'électeurs lui faisant confiance dans un échantillon de 100 électeurs, suit la loi binomiale de paramètres $n = 100$ et $p = 0{,}52$. Déterminons donc a et b tels que :

- a est le plus petit entier tel que $\mathrm{P}(X \leqslant a) > 0{,}025$
- b est le plus petit entier tel que $\mathrm{P}(X \leqslant b) > 0{,}975$.

Télécharger

Code: ***Elections***

```
from __future__ import division
from lycee import *
p,a,b=0,-1,-1
for k in range(101):
    p=p+binomial(100,k)*puissance(0.52,k)*puissance(0.48,100-k)
    if a==-1 and p>0.025 :
        a=k
    if b==-1 and p>=0.975 :
        b=k
print "On peut considérer l'affirmation de monsieur Z exacte au seuil de 5%, si le
 nombre de personnes ayant répondu positivement est dans l'intervalle [",a,";",b,"]."
```

Aperçu du résultat dans la console

```
On peut considérer l'affirmation de monsieur Z exacte au seuil de 5%, si le nombre
de personnes ayant répondu positivement est dans l'intervalle [ 42 ; 62 ].
```

choice(L) *lycee*

Renvoie un élément de la liste L avec équiprobabilité.

Télécharger

Code: ***Un tirage de LOTO***

```
from __future__ import division
from lycee import *
boules=range(1,50)
for i in range(5):
    b=choice(boules)
    print "boule",i+1,":",b
    boules.remove(b)
print "Numéro chance :",randint(1,10)
```

L'exemple ci-contre simule un tirage de LOTO, la difficulté est de ne pas tirer une boule déjà sortie. Il suffit donc de créer une liste contenant les 49 boules de départ. A chaque tirage, on supprime de la liste la boule tirée grâce à l'instruction `remove`.
Notez la simplicité du code Python comparée au casse-tête qu'est la programmation d'un tirage sans remise au tableur !

Bien évidemment, la commande `choice` ne se limite pas aux listes numériques. Dans cet exemple de jeu : on mise 1€ pour faire une partie. 30 boules blanches et 20 noires sont placées dans une urne. On tire 3 boules successivement et sans remise. Si on obtient 3 boules de la même couleur, on gagne 5€, sinon rien. Estimer le gain moyen d'une partie.

Remarque:

Ici l'utilisation de la fonction moyenne n'est pas réellement utile. Un compteur totalisant le gain à chaque jeu serait suffisant. Par contre pour calculer l'écart type, il semble pratique de stocker tous les résultats dans une liste.

Télécharger

Code: ***Une urne de 50 boules***

```
from __future__ import division
from lycee import *
Gain=[]
for i in range(10000) :
    Urne=["Blanche"]*30+["Noire"]*20
    B1=choice(Urne)
    Urne.remove(B1)
    B2=choice(Urne)
    Urne.remove(B2)
    B3=choice(Urne)
    if B1==B2 and B2==B3 :
        Gain.append(4)
    else :
        Gain.append(-1)
print "Gain moyen : ",moyenne(Gain)
print "Ecart type : ",ecartype(Gain)
```

listeRand(n) *lycee*

Retourne une liste de n nombres décimaux aléatoires dans l'intervalle [0,1[.

listeRandint(min,max,n) *lycee*

Retourne une liste de n nombres entiers aléatoires dans l'intervalle $[min, max]$.

(D'après un énoncé du document ressources pour faire la classe en mathématiques au lycée professionnel) : On admet que toute personne réservant une place d'avion a une chance sur 10 de ne pas se présenter à l'embarquement. Une compagnie aérienne dispose d'un avion de 100 places et vend 107 réservations. L'objectif est d'évaluer la probabilité de surréservation de cette compagnie, c'est-à-dire de répondre à la question : *Quel est le risque que plus de 100 passagers se présentent à l'embarquement ?*

BONUS : On estime que lorsqu'un avion est complet, les personnes en surnombre embarquent toutes dans le suivant. Estimer le nombre de personnes qui ne peuvent pas embarquer au cinquantième départ.

Télécharger

Code: ***surréservation***

```
from __future__ import division
from lycee import *
surbook=0
for simul in range(10000):
    P=listeRandint(0,9,107)
    venus=107-P.count(0)
    if venus>100 :
        surbook=surbook+1
print "Dans",surbook/100,"% des vols
 certaines personnes n'ont pu embarquer"
```

III) Diagrammes

baton(*xi*,*ni*) *lycee*

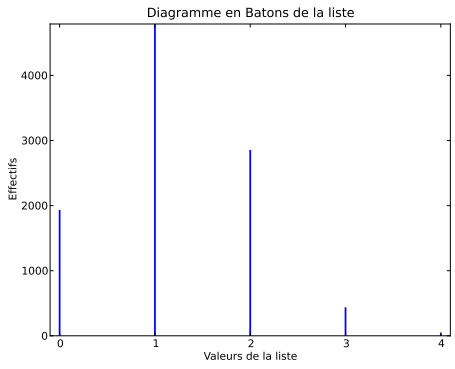
Génère et affiche le diagramme en bâtons relatif à la liste de valeurs *xi*, en tenant compte de la liste optionnelle des pondérations *ni*.

On suppose que pour une carte bleue donnée, tous les codes de 4 chiffres sont équiprobables. Représenter à l'aide d'un diagramme en bâtons, la répartition du nombre de touches en commun que peuvent avoir deux utilisateurs d'un même distributeur.

Télécharger

Code: ***Codes de carte bleue***

```
from __future__ import division
from lycee import *
ListeMeme=[]
for i in range(10000):
    Carte1=listeRandint(0,9,4)
    Carte2=listeRandint(0,9,4)
    meme=0
    for touche in range(10):
        if touche in Carte1 and touche in Carte2 :
            meme=meme+1
    ListeMeme.append(meme)
print "En moyenne, on a",moyenne(ListeMeme),"touches en commun"
baton(ListeMeme)
```



barre(*liste*,*a*,*pas*) *lycee*

Génère et affiche le diagramme en barre relatif à la liste de valeurs *liste*.

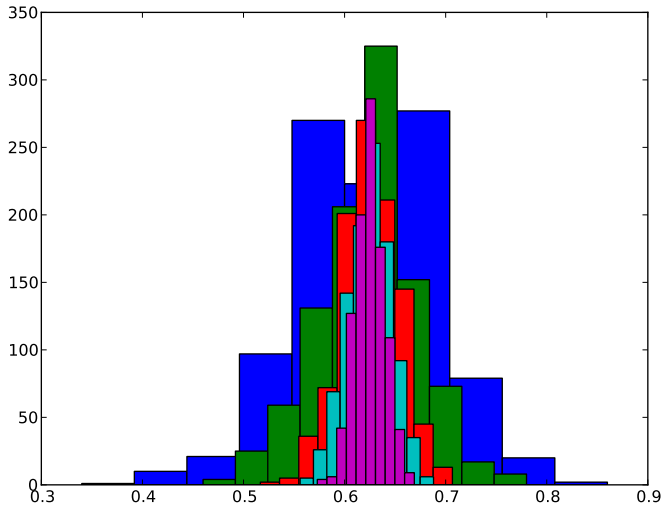
- Si seulement *liste* est renseigné, les valeurs seront réparties en 10 classes de même amplitude.
- Si *liste* et *a* sont renseignés, les valeurs seront réparties en *a* classes de même amplitude.
- Si les trois paramètres sont renseignés :
  - *a* est le centre de la première classe,
  - *pas* est un nombre indiquant l'amplitude des classes.

On réalise un programme permettant d'étudier un échantillon de N familles de 5 enfants. Parmi celles-ci, on regarde la fréquence du nombre de familles comportant 2 garçons et 3 filles ou inversement. On reconduit 1000 fois cette simulation. Que dire de la distribution des fréquences selon les valeurs de n ?

Télécharger

Code: ***Avoir une full d'enfants***

```
from __future__ import division
from lycee import *
freq=[]
N=demande("Nombre de familles ?")
for echan in range(1000) :
    nbFull=0
    for famille in range(n) :
        De=listeRandint(0,1,5)
        xi,ni=compte(De,'effectif')
        if ni==[3,2] or ni==[2,3] :
            nbFull=nbFull+1
    freq.append(nbFull/N)
barre(freq)
```



Après avoir montré que la probabilité qu'une famille de 5 enfants soit dans cette situation est $p = \frac{5}{8}$ soit 62,5%, et visualisé ce diagramme obtenu avec des échantillons de taille 50, 100, 300, 500 et 1 000, on observe la fluctuation des fréquences autour de cette valeur p. Au fur et à mesure de l'augmentation de N, l'étendue diminue. D'ailleurs, on pourrait améliorer ce programme pour compter le nombre de simulations où la fréquence observée est dans l'intervalle de fluctuation au seuil de 95% $F_{95} = \left[p - \frac{1}{\sqrt{N}}; p - \frac{1}{\sqrt{N}}\right]$ comme dans l'exemple en vidéo sur le site d' AmiensPython.

histop(*liste*,*classes*) *lycee*

Génère et affiche l'histogramme relatif à la liste *liste* dont l'aire totale des rectangles vaut 1.

- Si seule *liste* est renseignée, les valeurs seront réparties en 10 classes de même amplitude.
- Si la variable *classes* est un entier, les valeurs seront réparties en ce nombre de classes.
- Sinon, vous pouvez choisir vos classes d'amplitudes variées en indiquant dans la liste *classes* la liste ordonnée des bornes.

Remarque:

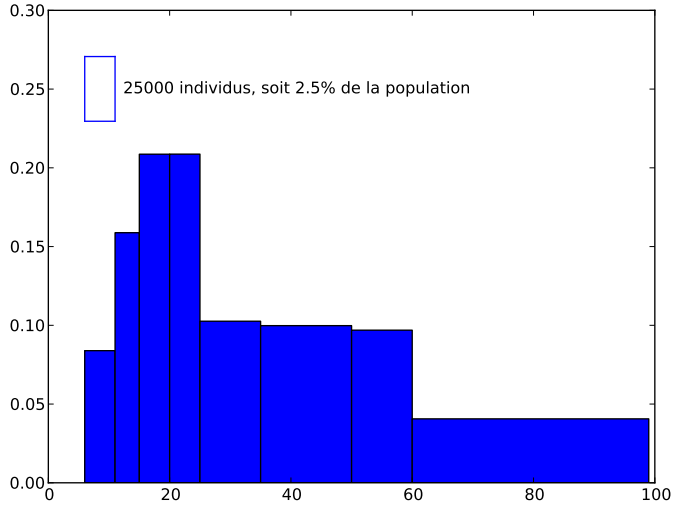
- *Nous avons choisi le nom* `histop` *pour donner une connotation probabiliste à cette fonction. En effet, comme l'aire totale vaut 1, la probabilité de choisir un individu dans une classe donnée est l'aire du rectangle correspondant.*
- *Pour les fonctions* `barre` *et* `histop`*, nous avons utilisé les conventions usuelles, c'est-à-dire des intervalles semi-ouverts sauf pour le dernier : Si* $x_1, x_2, \ldots, x_N$ *sont les bornes, les classes seront :* $[x_1\ ;\ x_2[, [x_2\ ;\ x_3[, \ldots, [x_{N-1}\ ;\ x_N]$.

On a généré un fichier représentant les âges d'un million de spectateurs de cinéma de plus de 6 ans à partir des données du site cnc.fr On souhaite afficher cette répartition selon les tranches suivantes : 6-10 ; 11-14 ; 15-19 ; 20-24 ; 25-34 ; 35-49 ; 50-60 et plus de 60 ans.

Télécharger

Code: ***Entrées au cinéma***

```
from __future__ import division
from lycee import *
R=CSV2liste('A')
Ages=L=[6,11,15,20,25,35,50,60,99]
histop(R,Ages)
```



Notez la puissance de calcul de Python, on a ici un fichier d'un million de valeurs de plus de 4 méga-octets, on obtient avec ce programme simple l'histogramme en 10 secondes environ avec un ordinateur portable.

polygoneECC(*xi*,*ni*,*coul*) & polygoneFCC(*xi*,*ni*,*coul*) *lycee*

Identique à la fonction `baton`, mais trace respectivement le polygone des effectifs cumulés croissants et des fréquences cumulées croissantes.

- Ce type de graphique ne présente d'intérêt que pour une répartition par classe, la liste *xi* doit donc comporter une valeur de plus que la liste *ni*.
- le paramètre optionnel *coul* est un caractère utilisant le code couleur présenté au chapitre 5.

Remarque:

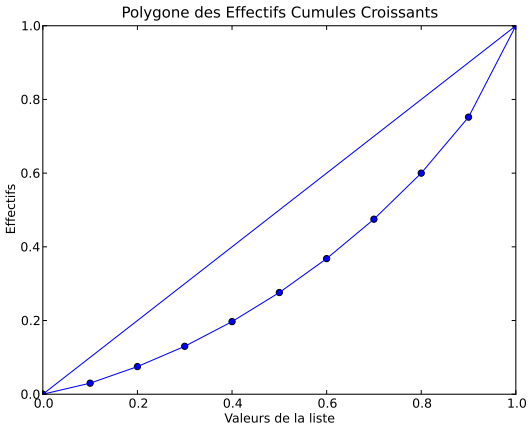
Ne perdez pas de vue que ces diagrammes ne sont que des graphiques étudiés au chapitre 5. Vous pouvez donc changer le titre et les noms des axes grâce aux fonctions `repere.title`, `repere.xlabel` *et* `repere.ylabel`

Un exemple de calcul du coefficient de Gini, dont vous pouvez trouver une présentation sur le site Statistix.

Télécharger

Code: ***Coefficient de Gini***

```
from __future__ import division
from lycee import *
listedeciles=[0,0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,(
proportions=[0.03,0.045,0.055,0.067,0.079,0.092,(
listedeciles,cumul=ECC(listedeciles,proportions)
polygoneECC(listedeciles,proportions)
repere.plot([0,1],[0,1])
somme=0
for i in range (9):
    somme=somme+0.1*cumul[i]
somme=somme+0.05*cumul[9]
gini=1-2*somme
print "Le coefficient de Gini vaut",gini, "."
```



IV) Conversion

Le travail sur les statistiques n'a d'intérêt en classe que s'il s'opère sur de grandes séries de valeurs dont on veut extraire une information. Aussi il nous est apparu utile d'offrir la possibilité de charger un fichier type tableur dans une liste. Cependant les fichiers excel ou OpenOffice sont trop complexes à analyser pour être chargés tels quels, vous devrez donc avant toutes choses convertir le fichier tableur au format CSV, pour cela il vous suffit de faire enregistrer-sous, puis de choisir CSV comme type de fichier.

CSV2liste(*num*,*fichier*) *lycee*

Retourne une liste composée d'une colonne ou d'une ligne d'un fichier CSV.

- *num* peut être un nombre (pour indiquer une ligne) ou un caractère (pour indiquer une colonne).
- Si *fichier* n'est pas renseigné, vous serez invité à en choisir un par l'intermédiaire d'un explorateur de fichiers
- Les cellules ne comptant pas de nombre sont ignorées

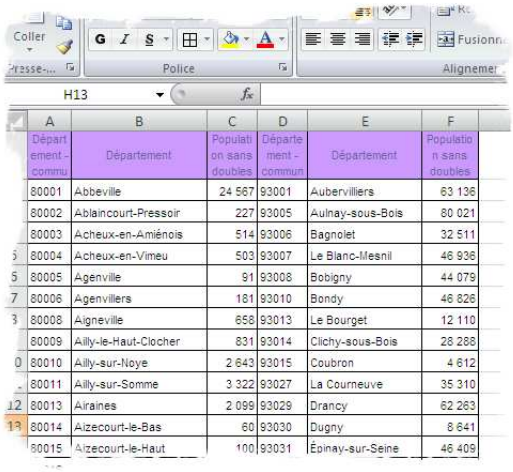
liste2CSV(*L*,*fichier*) *lycee*

Enregistre le contenu de la liste *L* dans le fichier *fichier*. Ce fichier peut alors être ouvert par Excel ou OpenOffice

- Si *fichier* n'est pas renseigné, vous serez invité à en choisir un par l'intermédiaire d'un explorateur de fichiers

L'exemple suivant compare les polygones des effectifs cumulés croissants du nombre d'habitants par commune de la Somme et de la Seine Saint-Denis. Les données proviennent du site de l'INSEE et ont été placées dans un tableur (télécharger le fichier).

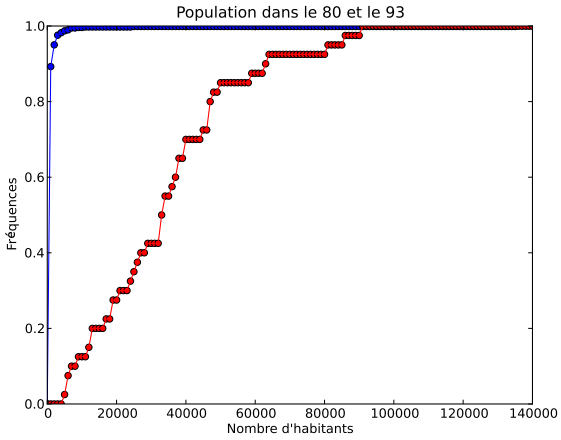
Télécharger



	A	B	C	D	E	F
	Département - commu	Département	Population sans doubles	Département - commun	Département	Population sans doubles
	80001	Abbeville	24 567	93001	Aubervilliers	63 136
	80002	Ablaincourt-Pressoir	227	93005	Aulnay-sous-Bois	80 021
	80003	Acheux-en-Amiénois	514	93006	Bagnolet	32 511
5	80004	Acheux-en-Vimeu	503	93007	Le Blanc-Mesnil	46 936
6	80005	Agenville	91	93008	Bobigny	44 079
7	80006	Agenvillers	181	93010	Bondy	46 826
8	80008	Aigneville	658	93013	Le Bourget	12 110
	80009	Ailly-le-Haut-Clocher	831	93014	Clichy-sous-Bois	28 288
0	80010	Ailly-sur-Noye	2 643	93015	Coubron	4 612
	80011	Ailly-sur-Somme	3 322	93027	La Courneuve	35 310
12	80013	Airaines	2 099	93029	Drancy	62 263
13	80014	Aizecourt-le-Bas	60	93030	Dugny	8 641
	80015	Aizecourt-le-Haut	100	93031	Épinay-sur-Seine	46 409

Code: ***Comparaison de deux départements***

```
from __future__ import division
from lycee import *
etendue=1000
nbclasse=quotient(140000,etendue)
L=CSV2liste('C')
Pop80=[0]*nbclasse
for v in L :
    classe=quotient(v,etendue)
    Pop80[classe]=Pop80[classe]+1
L=CSV2liste('F')
Pop93=[0]*nbclasse
for v in L :
    classe=quotient(v,etendue)
    Pop93[classe]=Pop93[classe]+1
Pop=[i*etendue for i in range(nbclasse+1)]
polygoneFCC(Pop,Pop80,'b')
polygoneFCC(Pop,Pop93,'r')
```



Chapitre 8

Les chaînes de caractères

Sommaire

En programmation une chaîne est l'équivalent d'une phrase, un caractère l'équivalent d'une lettre. D'un point de vue algorithmique, il est très formateur de travailler sur ces objets avec les élèves, car on peut faire réaliser des programmes simples à décrire sans connaissances mathématiques autres que l'algorithmique.

Les chaînes de caractères se définissent à l'aide de guillemets doubles ou simples : ex : `'bonjour'`, `"J'ai faim"`.

I) Longueur et caractères d'une chaîne

len(*ch*) *lycee*

renvoie le nombre de caractères de la chaîne *ch*.

Remarquez l'analogie de l'instruction **len** avec les listes, d'ailleurs comme pour celles-ci, on obtient le caractère à la position i en écrivant $ch[i]$:

$ch[i]$ *lycee*

renvoie le caractère d'indice i de la chaîne *ch*.

Attention : la première lettre est à l'indice 0. Il faut donc bien réfléchir aux indices de fin ! Voici deux programmes qui utilisent ces instructions, le premier compte le nombre de voyelles dans un mot, le second écrit une phrase à l'envers :

Télécharger

Code: ***Nombre de voyelles***

```
from __future__ import division
from lycee import *
mot="anticonstitutionnellement"
nb=0;i=0;
while i <len(mot):
    if mot[i] in ["a","e","i","o","u","y"] :
        nb=nb+1
    i=i+1
print mot,"a",nb,"voyelle(s)."
```

Télécharger

Code: ***A l'envers***

```
from __future__ import division
from lycee import *
phrase="Coucou, il fait beau !"
p=len(phrase)-1
while p>=0:
    print phrase[p],
    p=p-1
```

Si, à présent, on veut pouvoir stocker le mot à l'envers dans une variable pour tester, par exemple si le mot est un palindrome, nous allons coller bout à bout (on dit **concaténer**) les lettres du mot.

II) Opérations sur les chaînes

1) Coller, répéter

L'opérateur +

$ch1 + ch2$: Concatène les chaînes $ch1$ et $ch2$, exemple "bon"+"jour" donne "bonjour"

Voici deux exemples différents qui permettent de savoir si un mot est un palindrome :

Télécharger

Code: ***Palindrome1***

```
from __future__ import division
from lycee import *
phrase="kayak"
envers="";p=len(phrase)-1
while p>=0:
    envers=envers+phrase[p]
    p=p-1
if phrase==envers :
    print phrase,"est un palindrome."
```

Télécharger

Code: ***Palindrome2***

```
from __future__ import division
from lycee import *
phrase="kayak"
envers="";p=0
while p<len(phrase):
    envers=phrase[p]+envers
    p=p+1
if phrase==envers :
    print phrase,"est un palindrome."
```

Remarque:

A la quatrième ligne, l'instruction `envers = ""` *permet d'initialiser la variable* `envers` *comme étant une chaîne vide.*

L'opérateur *

$ch1 * n$: Répète n fois la chaîne $ch1$, exemple : "ha ! "*3 donne "ha ! ha ! ha ! "

Exemple de programme qui double toutes les voyelles d'un mot :

Télécharger

Code: ***Doubler les voyelles***

```
from __future__ import division
from lycee import *
mot="glouglou dit le dindon"
double=""
i=0;
while i <len(mot):
    if mot[i] in ("a","e","i","o","u","y") :
        double=double+mot[i]*2
    else :
        double=double+mot[i]
    i=i+1
print double
```

2) Transformer

L'exemple précédent ne fonctionne que si le texte est entré en minuscules, évidemment on pourrait adapter la ligne de test en `in` (”a”,”e”,”i”,”o”,”u”,”y”,”A”,”E”,”I”,”O”,”U”,”Y”) mais il existe déjà, en Python, deux fonctions qui transforment une chaîne en majuscules ou minuscules :

ch.upper(), ch.lower() *lycee*

renvoient respectivement la chaîne de caractères *ch* en majuscules et en minuscules.

Remarque:

- *Les fonctions* `upper()` *et* `lower()` *s'appliquent aussi aux caractères, dans l'exemple précédent, on peut changer* `if mot[i] in` (”a”,”e”,”i”,”o”,”u”,”y”,”A”,”E”,”I”,”O”,”U”,”Y”) *en* `if mot[i].lower() in` (”a”,”e”,”i”,”o”,”u”,”y”)
- *Les fonctions* `upper()` *et* `lower()` *n'ont aucun effet sur les caractères spéciaux(marqués d'un accent).*

Dans l'exemple suivant, on veut transformer un nombre inférieur ou égal à 255 en écriture hexadécimale (base 16), il nous faut alors concaténer des chiffres (ou nombres) à des caractères (ou chaînes). Il est alors nécessaire que tous aient la même nature : des chaînes de caractères.

str(n)

renvoie le nombre n transformé en chaîne de caractères.

Par exemple `str(25)` renvoie la chaîne de caractères ”25”.

Télécharger

Code: ***Hexadécimal 1***

```
from __future__ import division
from lycee import *
n=demande("Entrez n < 256")
hexa=""
c=quotient(n,16)
for i in range(2):
    if c<10 : hexa=hexa+str(c)
    elif c==10 : hexa=hexa+"A"
    elif c==11 : hexa=hexa+"B"
    elif c==12 : hexa=hexa+"C"
    elif c==13 : hexa=hexa+"D"
    elif c==14 : hexa=hexa+"E"
    elif c==15 : hexa=hexa+"F"
    c=reste(n,16)
print n,"s'écrit",hexa,"en base 16"
```

Et une version bien plus courte, utilisant un alphabet composé des symboles utilisés en base 16 :

Télécharger

Code: ***Hexadécimal 2***

```
from __future__ import division
from lycee import *
n=demande("Entrez n < 256")
alphabet="0123456789ABCDEF"
q=quotient(n,16)
r=reste(n,16)
hexa=alphabet[q]+alphabet[r]
print n,"s'écrit",hexa,"en base 16"
```

DEFI : Imaginer un algorithme pour convertir n'importe quel nombre décimal en base 16.

Inversement, pour transformer une chaîne de caractères en nombre :

eval(*ch*)

renvoie la valeur numérique de la chaîne de caractères *ch*.

Remarque:

- *Notez que* `eval` *est bien plus puissante que transformer la chaîne* "`12`" *en nombre* `12`*, car vous pouvez utiliser aussi les opérations et les fonctions de Python. Par exemple,* `eval("quotient(12,5)*sqrt(3)")` *renvoie* `3.46410161514`.
- *En ce qui concerne les changements de base, en pratique, on utilise plutôt une liste qu'une chaîne pour stocker un nombre en base a. Le nombre est d'ailleurs écrit « à l'envers » : le premier nombre de la liste est le chiffre des unités. Ainsi l'élément de la liste d'indice i est à multiplier par a^i pour obtenir la valeur du nombre. L'addition de deux nombres est ainsi assez simple à programmer(penser aux retenues). Par exemple, le nombre A76 en base 16 sera défini par la liste [6,7,A] et vaut $6\times16^0+7\times16^1+10\times16^2$.*

III) Codage d'un caractère

Les algorithmes de cryptographie fonctionnent souvent sur le modèle : Prendre une lettre → Lui associer un nombre → Transformer ce nombre → Afficher la lettre correspondant à cette image. En Python comme dans tous les langages, il existe déjà les fonctions qui associent à chaque caractère un nombre et inversement. Ce nombre est appelé le code ASCII *(**A**merican **S**tandard **C**ode for **I**nformation **I**nterchange)*, dont voici quelques correspondances :

Lettre	espace	A	B	...	Z	0	1	...	9	a	b	...	z
Code ASCII	32	65	66	...	90	48	49	...	57	97	98	...	122

ord(*car*) et str(*n*)

- **ord**(*car*) : Renvoie le code ASCII du caractère *car*.
- **str**(*n*) : Renvoie le caractère dont le code ASCII est *n*.

Dans la pratique, vous verrez que ce code pose des soucis. En effet, les premiers caractères sont des caractères spéciaux. Pour éviter tout problème d'affichage, la console Python que vous utilisez les remplace par le caractère générique □, ce qui a pour conséquence immédiate de ne plus distinguer visuellement ces 30 premiers caractères... Difficile alors de décrypter le message □□□□□ ! Pour pallier ce problème, nous vous proposons un autre codage qui correspond à un alphabet que nous avons baptisé l'***A****lphabet* ***A****miens****P****ython* (AAP). Il se compose de 102 caractères.

Code AAP	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Caractère		A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
Code AAP	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33
Caractère	Q	R	S	T	U	V	W	X	Y	Z	a	b	c	d	e	f	g
Code AAP	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50
Caractère	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x
Code AAP	51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67
Caractère	y	z	0	1	2	3	4	5	6	7	8	9	.	,	:	;	!
Code AAP	68	69	70	71	72	73	74	75	76	77	78	79	80	81	82	83	84
Caractère	?	&	à	â	é	è	ê	ë	î	ï	ù	#	'	(	-	_	)
Code AAP	85	86	87	88	89	90	91	92	93	94	95	96	97	98	99	100	101
Caractère	{	}	[	]	|	\	@	=	+	°	§	$	<	>	%	*	/

AlphabetAP

lycee

AlphabetAP est la chaîne de caractères contenant tous les caractères ordonnés de l'alphabet AmiensPython.

Remarque : Comme pour `pi`, il s'agit en réalité d'une variable que l'on a initialisée. Ainsi, vous pouvez choisir de la modifier en début de programme pour vos besoins.

codeAAP(*car*) et decodeAAP(*n*)

lycee

- **codeAAP**(*car*) : Renvoie la place dans l'Alphabet AmiensPython du caractère *car*.
- **decodeAAP**(*n*) : Renvoie le caractère de rang *n* dans l'Alphabet AmiensPython.

Un exemple de codage et décodage affine : $f(x)$ représente le reste de la division de $5x + 46$ par 102 et $g(x)$ celui de la division de $41x + 52$ par 102.

$$\text{Codage : Lettre} \overset{\texttt{CodeAAP}}{\longmapsto} \text{Nombre} \overset{f}{\longmapsto} \text{Nombre} \overset{\texttt{DecodeAAP}}{\longmapsto} \text{Lettre}$$
$$\text{Décodage : Lettre} \overset{\texttt{CodeAAP}}{\longmapsto} \text{Nombre} \overset{g}{\longmapsto} \text{Nombre} \overset{\texttt{DecodeAAP}}{\longmapsto} \text{Lettre}$$

Télécharger

Code: *codage affine*

```
from __future__ import division
from lycee import *
texte="Les carottes sont cuites !"
message=""
for i in range(len(texte)):
    lettre=texte[i]
    code=codeAAP(lettre)
    code2=reste(5*code+46,102)
    lettre2=decodeAAP(code2)
    message=message+lettre2
print message
```

Télécharger

Code: *décodage affine*

```
from __future__ import division
from lycee import *
texte="D%!t|#9uéé%!t!upét|ïQé%!të"
message=""
for i in range(len(texte)):
    lettre=texte[i]
    code=codeAAP(lettre)
    code2=reste(41*code+52,102)
    lettre2=decodeAAP(code2)
    message=message+lettre2
print message
```

IV) sous-chaîne d'une chaîne de caractères

1) Extraire une sous-chaîne

ch[*deb* : *fin*]

- *ch*[*deb* : *fin*] : Renvoie la partie de la chaîne *ch* comprise entre le caractère à la positon *deb* (inclus) et celui à la position *fin* (exclu).

Si *deb* n'est pas précisé, la sous-chaîne commence au début de la chaîne. Si *fin* n'est pas précisé la sous-chaîne s'arrête à la fin de la chaîne :

- *ch*[*deb* :] : Renvoie la partie droite de la chaîne *ch* commençant au rang *deb* (inclus).
- *ch*[: *fin*] : Renvoie la partie gauche de la chaîne *ch* jusqu'au rang *fin* (exclu).

Remarque:

- *Attention : ch*[3 : 7] *renvoie la partie de ch de la quatrième à la septième lettre mais n'a pas d'effet sur ch.*
- *Au passage on notera que ce système d'extraction est aussi valable pour une liste. (On comprend alors mieux le sens de la commande* `L2 = L1[:]` *présenté au chapitre précédent.)*

(D'après Wikipedia) : L'ISBN (International Standard Book Number) ou numéro international normalisé du livre est un numéro international qui permet d'identifier, de manière unique, chaque livre publié. Il est destiné à simplifier la gestion informatique du livre : bibliothèques, libraires, distributeurs, etc. Jusqu'en janvier 2007, le code ISBN comportait 10 chiffres, le dernier étant une clé de contrôle permettant de détecter une erreur de saisie (code détecteur) et calculé ainsi :

- On attribue une pondération à chaque position (de 10 à 2 en allant en sens décroissant) et on fait la somme des produits ainsi obtenus.
- On conserve le reste de la division euclidienne de ce nombre par 11. La clé s'obtient en retranchant ce nombre à 11.
- Si le reste de la division euclidienne est 0, la clé de contrôle n'est pas 11 (11 - 0 = 11) mais 0.
- De même si le reste de la division euclidienne est 1, la clé de contrôle n'est pas 10 mais la lettre X. Ceci permet donc d'avoir respectivement pour les restes de la division 0, 1, 2, 3, ... 10 les codes 0, X, 9, 8, ..., 1.

Par exemple :

Code ISBN	2	2	6	6	1	1	1	5	6
Pondération	10	9	8	7	6	5	4	3	2
Produit	20	18	48	42	6	5	4	15	12

Soit au total 170 dont le reste de la division euclidienne par 11 est 5. La clé de contrôle est donc $11 - 5 = 6$. L'ISBN au complet est : $2 - 266 - 11156 - 6$.

Télécharger

Code: ***Code ISBN***

```
from __future__ import division
from lycee import *
code=texte_demande("Entrer le code ISBN sans les - :")
l=len(c)
debut=code[:l-2]
#On peut écrire code[:-1]
fin=code[l-2:]
#On peut écrire code[-1]
total=0
for i in range(9):
    total=total+eval(debut[i])*(10-i)
cle=11-reste(total,11)
if cle==11 : cle=srt(0)
elif cle==10 : cle="X"
else : cle=str(cle)
if cle==fin:
    print "Ce code est valide"
else :
    print "Ce code n'est pas valide"
```

2) Rechercher, remplacer et compter

***ch*.find(*texte*)**

Indique la position dans la chaîne *ch* où se trouve *texte* et renvoie -1 si *texte* n'apparait pas dans la chaîne *ch*.

L'exemple suivant illustre comment simplifier une fraction. A la différence de l'exemple donné dans la partie mathématiques, où l'on demandait le numérateur et le dénominateur, cette fois-ci la fraction est entrée sous forme d'une chaîne (exemple "12/45"), le programme extrait alors le numérateur et le dénominateur pour simplifier la fraction.

Télécharger

Code: ***Analyser une fraction***

```
from __future__ import division
from lycee import *
frac=texte_demande("Entrez une fraction")
barre=frac.find("/")
if barre==-1 :
    print "Je n'ai pas vu de fraction"
else :
    num=eval(frac[:barre])
    den=eval(frac[barre+1:])
    d=pgcd(num,den)
    num,den=quotient(num,d),quotient(den,d)
    print frac+"="+str(num)+"/"+str(den)
```

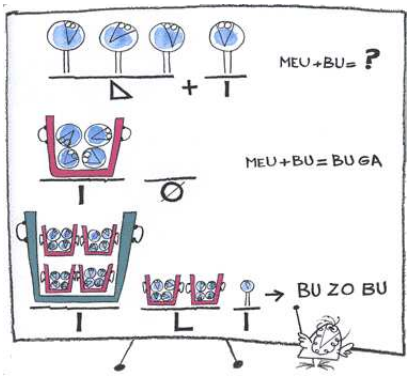
Remarque:

En informatique, l'action d'analyser une chaîne pour l'interpréter, s'appelle « parser »la chaîne (de l'anglais to parse : analyser).

ch.**replace**(*texte*1,*texte*2)

Remplace *texte*1 par *texte*2 chaque fois qu'il le trouve dans la chaîne *ch*.

Les Shadoks ne possèdent que 4 mots pour compter : GA - BU - ZO - MEU, et comptent donc en base 4...L'explication par les Shadoks eux-même est bien plus drôle. Le programme ci-contre, transforme un nombre Shadoks en nombre compréhensible par les terriens.



Télécharger

Code: ***Compter en Shadoks***

```
from __future__ import division
from lycee import *
nbS="MEU GA GA BU MEU ZO ZO"
print nbS,"=",
nbS=nbS.replace("GA","0")
nbS=nbS.replace("BU","1")
nbS=nbS.replace("ZO","2")
nbS=nbS.replace("MEU","3")
nbS=nbS.replace(" ","")
d=1;r=0
for i in range(len(nbS),0,-1):
    r=r+eval(nbS[i-1])*d
    d=d*4
print r
```

ch.**count**(*texte*)

Compte le nombre de fois où *texte* est présent dans la chaîne *ch*.

Cette fois, on essaie un codage *non affine* avec une fonction polynomiale : f définie par $f(x) = 51x^2 + 46x + 21$ en appliquant toujours le même principe :

$$\text{Lettre} \overset{\texttt{CodeAAP}}{\longmapsto} \text{Nombre} \overset{f}{\longmapsto} \text{Nombre} \overset{\texttt{DecodeAAP}}{\longmapsto} \text{Lettre}$$

Question : « A-t-on un réel codage ? », c'est à dire : « 2 lettres distinctes sont-elles codées par 2 lettres distinctes ? »ou encore pour parler mathématiquement f réalise-t-elle une bijection de $\{0,1,\cdots,101\}$ dans $\{0,1,\cdots,101\}$?

Nul besoin ici de chercher des théorèmes de Gauss ou autre puisque l'ordinateur peut se charger de lister tous les cas. Le programme ci-contre (qui n'est pas optimisé) transforme tout l'alphabet, puis compte le nombre de fois où chaque caractère apparait.

DEFI : Imaginer un programme pour répondre à la question : « *Existe-il une fonction de degré 2 qui décode ?* »

Télécharger

Code: ***Tester un codage***

```
from __future__ import division
from lycee import *
message=""
i=0
for i in range(len(AlphabetAP)):
    lettre=AlphabetAP[i]
    code=codeAAP(lettre)
    code2=reste(51*code*code+46*code+21,1
    lettre2=decodeAAP(code2)
    message=message+lettre2
doubles=0
for i in range(len(message)):
    if message.count(message[i])>1 :
        doubles=doubles+1
if doubles==0 :
    print "C'est un codage"
else :
    print "Ce n'est pas un codage"
```

V) Convertir des chaînes de caractères

1) Enregistrer et charger un fichier

Comme pour les listes, il nous a semblé important de pouvoir enregistrer une chaîne dans un fichier texte (et de pouvoir le charger !). On peut par exemple imaginer de demander aux élèves de vous envoyer un message codé par messagerie ou via un espace de dépôt...

fich2chaine(*nom*) ***(se prononce « fiche to chaine » : transformer un fichier en chaîne)*** *lycee*

Renvoie une chaîne de caractères qui correspond au contenu du fichier *nom*. Si aucun *nom* n'est précisé, une fenêtre invitant l'utilisateur à choisir un fichier s'ouvrira lors de l'exécution du programme.

Notez que si le fichier comprend plusieurs lignes, la chaîne obtenue sera composée de toutes les lignes séparées par le caractère spécial "\n".

La fonction suivante permet à l'inverse d'enregistrer une chaîne dans un fichier texte.

chaine2fich(*ch*, *nom*) *lycee*

Enregistre le contenu de la chaîne *ch* dans le fichier *nom*. Si aucun *nom* n'est précisé, une fenêtre invitant l'utilisateur à choisir un nom de fichier s'ouvrira lors de l'exécution du programme.

Remarque:

Attention *: Python sait lire un fichier de texte brut, mais ne saura pas lire un fichier Word ou OpenOffice car ces fichiers sont très complexes (gestion des images, des polices de caractères ...) Si vous souhaitez utiliser un fichier provenant d'un tel logiciel, vous devrez utilisez « enregistrer sous »et modifier le format d'enregistrement. (Choisir « texte brut »ou « .txt ».)*

2) Convertir une chaîne de caractères en liste

Il peut être parfois utile de convertir une chaîne de caractères en une liste de chaînes. Dans ce cas, il faut indiquer le séparateur qui sera utilisé.

_ch_.split(_sep_)

Renvoie une liste de chaînes de caractères obtenue en découpant la chaîne *ch* à chaque occurrence du séparateur *sep*.

Inversement,

sep.join(_list_)

Renvoie une chaîne de caractères obtenue en concaténant les éléments de la liste *list* séparés par le séparateur *sep*.

Dans l'exemple qui suit, nous allons charger dans une chaîne le poème bien connu ci-contre et transformer cette chaîne en une liste de mots en utilisant le séparateur espace, puis compter le nombre de lettres de chaque mot pour obtenir les premières décimales du nombre π.

Télécharger

Code: ***Poème version 1***

```
from __future__ import division
from lycee import *

poeme=fich2chaine()
CarSpeciaux=['.',',','!','"',"\n",';','?',':']
for c in CarSpeciaux:
    poeme=poeme.replace(c," ")
mots=poeme.split(" ")
for m in mots:
    if m<>"" : print reste(len(m),10),
```

Contenu du fichier *poeme_pi.txt* :

« Que j'aime à faire apprendre ce nombre utile aux sages !
Immortel Archimède, artiste ingénieur,
Qui de ton jugement peut priser la valeur ?
Pour moi, ton problème eut de pareils avantages.
Jadis, mystérieux, un problème bloquait
Tout l'admirable procédé, l'oeuvre grandiose
Que Pythagore découvrit aux anciens Grecs.
0 quadrature ! Vieux tourment du philosophe
Insoluble rondeur, trop longtemps vous avez
Défié Pythagore et ses imitateurs.
Comment intégrer l'espace plan circulaire ?
Former un triangle auquel il équivaudra ?
Nouvelle invention : Archimède inscrira
Dedans un hexagone ; appréciera son aire
Fonction du rayon. Pas trop ne s'y tiendra :
Dédoublera chaque élément antérieur ;
Toujours de l'orbe calculée approchera ;
Définira limite ; enfin, l'arc, le limiteur
De cet inquiétant cercle, ennemi trop rebelle
Professeur, enseignez son problème avec zèle »

Encore une fois, nous insistons sur le fait qu'il n'est pas indispensable pour les élèves de connaître ces fonctions. La démarche de chercher soi-même les espaces et de compter les lettres peut être très formatrice d'un point de vue algorithmique.

Télécharger

Code: ***Poème version 2***

```
from __future__ import division
from lycee import *
poeme=fich2chaine()
NbDecimales,pos=0,0
reponse=""
while pos<len(poeme):
    NbLettres=0
    while pos<len(poeme) and not poeme[pos]in ('.',',','!',"'","\n",';','?',':',' '):
        NbLettres=NbLettres+1
        pos=pos+1
    reponse=reponse+str(reste(NbLettres,10))
    NbDecimales=NbDecimales+1
    if NbDecimales==1 : reponse=reponse+","
    while pos<len(poeme) and poeme[pos]in ('.',',','!',"'","\n",';','?',':',' '):
        pos=pos+1
print reponse
```

Chapitre 9

Fonctions

Sommaire

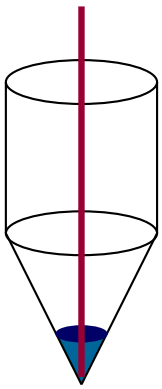
Bien que la notion de fonction n'apparaisse pas de manière claire dans le programme d'algorithmique du lycée, il nous semble important de la faire apparaitre ici pour plusieurs raisons : d'une part, la notion de fonction au sens mathématique est très présente dans tout le cycle du lycée, d'autre part, il est important de pouvoir réinvestir ce qui a été fait.

def *nom de la fonction*(*paramètres*) **:**

Définit une nouvelle fonction, comme pour le `if` et le `for`, les deux points entrainent une indentation délimitant la déclaration de la fonction. Le bloc peut servir à effectuer une série d'actions, mais le plus souvent il se termine par `return` pour renvoyer une ou plusieurs valeurs.

I) Exemple de fonctions mathématiques

Une cuve a la forme forme d'un cône surmonté d'un cylindre, chacun d'une hauteur de 20cm. Une sonde est placée pour mesurer la hauteur d'eau, comme sur le schéma ci-contre. Le diamètre du cylindre est aussi de 20cm. On souhaite réaliser un algorithme affichant le volume d'eau en fonction de la hauteur.



Pour plus de clarté, nous avons volontairement omis les tests vérifiant que la hauteur h est bien dans l'intervalle $[0; 40]$. Nous vous présentons deux versions du programme : La première sans fonction est déjà un travail conséquent en classe de seconde. (Recherche des formules de volume, théorème de Thalès, ...) La seconde définit une fonction $V(h)$, comme nous le ferions en mathématiques.

Télécharger

Code: ***La cuve sans fonction***

```
from __future__ import division
from lycee import *
h=demande("hauteur d'eau ?")
if h<20 :
    R=h/2
    V=1/3*pi*R*R*h
else :
    Vcone=1/3*pi*10*10*20
    Vcyl=pi*10*10*(h-20)
    V=Vcyl+Vcone
print "Le volume est",V
```

Télécharger

Code: ***La cuve avec fonction***

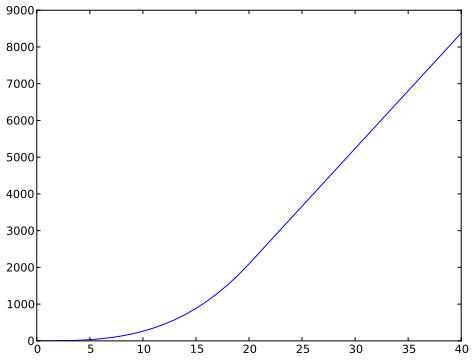
```
from __future__ import division
from lycee import *
def V(h):
    if h<20 :
        R=h/2
        return 1/3*pi*R*R*h
    else :
        Vcyl=1/3*pi*10*10*20
        Vcone=pi*10*10*(h-20)
        return Vcyl+Vcone

h=demande("hauteur d'eau ?")
print "Le volume est",V(h)
```

Remarque:

- *La fonction V peut alors être utilisée comme n'importe quelle fonction de Python. Vous pourriez par exemple demander de tracer la courbe de V, en remplaçant les 2 dernières lignes du second programme par :*

```
X=range(41)
Y=[]
for x in X : Y.append(V(x))
repere.plot(X,Y)
repere.show()
```



- *Toutes les variables à l'intérieur de la déclaration de la fonction sont des variables locales, c'est-à-dire que le fait de les modifier ou d'avoir une variable qui porte le même nom dans le corps principal du programme n'a aucun effet sur celui-ci.*
- *De la même manière, à l'intérieur d'une fonction, le logiciel ne connait aucune des variables du reste du programme, vous devez donc prévoir de placer toutes les variables dont vous aurez besoin dans les paramètres de la fonction. (Ici seul h est nécessaire.)*

II) Fonction au sens informatique

Extrait d'un concours FFJM : « Les briques de Brian sont tous les parallélépipèdes rectangles dont les dimensions sont des nombres entiers inférieurs ou égaux à 7. Brian calcule le volume de chaque brique et le divise par le carré de sa plus grande dimension. Il additionne ensuite tous les résultats. Quelle fraction irréductible obtient-il ? ».

En fait, nous nous apercevons qu'il suffit de calculer :

$$\sum_{i=1}^{7}\sum_{j=1}^{i}\sum_{k=1}^{j}\frac{j\times k}{i}$$

La difficulté est ici de travailler avec les fractions et de conserver des valeurs exactes. La fonction `sommefrac` attend 4 arguments n_1,d_1,$n2$ et d_2 et renvoie le numérateur et le dénominateur du résultat simplifié de $\frac{n_1}{d_1}+\frac{n_2}{d_2}$.

Télécharger

Code: ***Les briques de Brian***

```
from __future__ import division
from lycee import *
from math import *
def sommefrac(n1,d1,n2,d2):
  n,d=n1*d2+n2*d1,d1*d2
  p=pgcd(n,d)
  return quotient(n,p),quotient(d,p)

n,d=0,1
for i in range(1,8):
  for j in range(1,i+1):
    for k in range(1,j+1):
      n,d=sommefrac(n,d,j*k,i)
print n,'/',d
```

III) Une fonction peut en cacher une autre

Comme le montre l'exemple précédent, une fonction peut à son tour faire appel à une autre fonction que vous avez créée. En Python, il n'y a pas d'ordre pour déclarer les fonctions.

Télécharger

Code: ***Suite de Syracuse***

```
from __future__ import division
from lycee import *
def suivant(x):
    if reste(x,2)==0:
        return quotient(x,2)
    else :
        return 3*x+1

def vol(x):
    L=[]
    while x<>1 :
        L.append(x)
        x=suivant(x)
    return L

tmax=0
for n in range(1,100001):
    t=len(vol(n))
    if t>tmax :
        print "Temps de vol de",t,"pour n=",n
        tmax=t
```

Une suite de Syracuse N est la suite définie par

$$u_0=N \text{ et } u_{n+1}=\begin{cases}\frac{u_n}{2} \text{ si } u_n \text{ est pair}\\ 3\times u_n+1 \text{ sinon}\end{cases}.$$

On appelle **temps de vol** le premier indice p tel que $u_p=1$.

Sur les 10 000 premières suites de Syracuse, laquelle a le temps de vol maximal ?

IV) Récursivité

Une fonction peut même s'appeler elle-même. (Attention cependant aux boucles infinies !) C'est ce que l'on appelle des fonctions récursives. Prenons l'exemple le plus connu de l'algorithme d'Euclide, il peut s'énoncer ainsi : Si a et b sont 2 entiers naturels non simultanément nuls, alors :

$$PGCD(a,b) = \begin{cases} a \text{ si } b = 0 \\ PGCD(b,r) \text{ sinon} \end{cases} \quad \text{où } r \text{ est le reste de la division euclidienne de } a \text{ par } b.$$

La fonction correspondante s'énonce alors d'une manière très simple :

Télécharger

Code: ***PGCD récursif***

```
from __future__ import division
from lycee import *
def pgcd_rec(a,b):
    if b==0 :
        return a
    else :
        return pgcd(b,reste(a,b))
print pgcd_rec(12,28)
```

Un autre exemple pour calculer de manière originale a^n (sans utiliser la fonction puissance).

Ce premier exemple est basé sur le fait que :

$$a^n = \begin{cases} a \text{ si } n = 1 \\ a \times a^{n-1} \text{ sinon} \end{cases}$$

Télécharger

Code: ***Puissance récursive version 1***

```
from __future__ import division
from lycee import *
def puissV1(a,n):
    if n==1 :
        return a
    else :
        return a*puissV1(a,n-1)

print puissV1(2,10)
```

Ce second exemple est basé sur le fait que :

$$a^n = \begin{cases} a \text{ si } n = 1 \\ \left(a^{\frac{n}{2}}\right)^2 \text{ si } n \text{ est pair} \\ a \times a^{n-1} \text{ sinon} \end{cases}$$

Télécharger

Code: ***Puissance récursive version 2***

```
from __future__ import division
from lycee import *
def puissV2(a,n):
    if n==1 :
        return a
    elif reste(n,2)==0 :
        temp=puissV2(a,quotient(n,2))
        return temp*temp
    else :
        return a*puissV2(a,n-1)

print puissV2(2,10)
```

Le second algorithme est appelé algorithme des puissances indiennes et limite fortement le nombre de multiplications à effectuer, pour calculer 2^{10} avec le premier algorithme, neuf multiplications sont nécessaires alors qu'il n'en faut que quatre avec le deuxième.

Un dernier exemple avec une suite récurrente d'ordre 2 : la plus connue est sûrement la suite de Fibonacci définie par $u_0 = u_1 = 1$ et pour tout entier naturel n, $u_{n+2} = u_n + u_{n+1}$, le programme suivant renvoie le terme u_r pour r donné.

Télécharger

Code: ***Suite de Fibonacci***

```
from __future__ import division
from lycee import *
def fibo(n):
    if n==0 or n==1:
        return 1
    else:
        return fibo(n-1)+fibo(n-2)
r=demande("Indice du terme de la suite de Fibonacci?")
print fibo(r)
```

Chapitre 10

Questions fréquemment posées

Sommaire

I) A propos d'AmiensPython

1) Puis-je utiliser AmiensPython sous Mac ou Linux ?

Malheureusement non, en tout cas pas dans son intégralité. L'éditeur utilisé PyScripter, ne fonctionne que sous windows. Néanmoins, vous pouvez récupérer la bibliothèque lycee pour l'installer avec votre version personnelle de Python. Vous rencontrerez peut-être des problèmes d'accents. Tenez nous informés.

2) Quelle est la différence avec la version d'origine PortablePython ?

En effet, AmiensPython est dérivée du projet PortablePython que vous pouvez retrouver ici. Les modifications réalisées sont minimes, mais évitent à chaque utilisateur de faire ces modifications (Modification du moteur de rendu Python, pour permettre l'utilisation du mode "tortue", Ajout de la bibliothèque XTurtle, plus interactive, prise en compte des accents...

3) Pourquoi utiliser une vieille version 2.6 alors que la version 3 de python est disponible ?

A l'heure où je réponds, le Python 3.0 est assez différent du 2.6, de ce fait beaucoup de bibliothèques (Sorties graphiques entre autre) ne sont pas encore traduites.

4) Comment arrêter un programme en cours d'exécution ?

En appuyant simultanément sur la touche contrôle et la touche F2.

5) Pourquoi avoir traduit certaines fonctions ?

En réalité, n'ont été traduites que les fonctions qui ont été modifiées. Prenons l'exemple de `demande`, nous avons fait le choix de la renommer pour indiquer qu'elle avait subi une transformation par rapport à la fonction `input` de départ (dans ce cas pour gérer les accents). Mais rien ne vous empêche d'utiliser `input` avec vos élèves dans la mesure où vous n'utilisez pas d'accent dans le programme.

6) Quand je tente d'ouvrir un fichier .py, l'ordinateur me demande avec quoi l'ouvrir.

En effet, AmiensPython étant un système portable, aucune installation n'a lieu. De ce fait, les fichiers python ne sont pas associés à l'interface. Vous devez lancer AmiensPython, puis sélectionner « ouvrir » et aller chercher votre fichier.
Si vous avez des droits d'administrateur, la version 2.1 effectue l'association, ce qui n'est pas possible si vous exécutez le programme sur une clé USB.

II) A propos de Python

1) Pourquoi dites-vous que Python est un langage très puissant ?

En fait d'autres langages sont aussi très performants pour les calculs. Mais Python peut travailler sur de très grands nombres. D'ailleurs, il n'y a pas de limite de taille pour les *entiers*, il calcule avec eux sans approximation. Le dernier exemple de cette documentation est inspiré du livre *« histoire d'algorithmes, du caillou à la puce »* de Chabert *et alii*. Le test de primalité de Lucas et Lehmer sur les nombres de Mersenne permet en une vingtaine de minutes de conclure que $2^{21701}-1$ est un nombre premier(hors programme bien sûr). Ce nombre trouvé le 30 octobre 1978 est maintenant à la portée d'un simple ordinateur.

Télécharger

Code: ***Test de primalité de Lucas Lehmer.***

```
from __future__ import division
from lycee import *
p=demande('entrer p, nombre premier impair')
M=2**p-1
print "M=2^p  -1=",M
i,v=2,4
while v!=0 and i<=p:
    i=i+1
    v=v**2-2
    v=v%M
    print "v(2^", i, ") est congru à", v, "mod M"

if v==0 and i==p:
    print "Le nombre de Mersenne 2^", p, "-1=", M, "est premier."
else:
    print "Le nombre de Mersenne 2^", p, "-1=", M, "n'est pas premier."
```

2) Pourtant Python n'est pas précis dans les calculs !

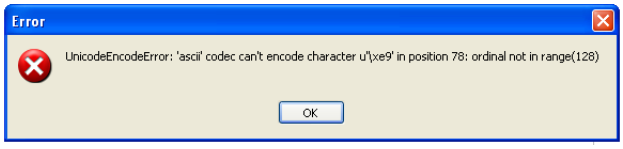
Si vous tapez `print 3 - 2.99 - 0.01`, vous obtiendrez le résultat `-2.13370987545e - 16` ce qui peut poser problème...Cela vient de la représentation des nombres dans la machine : La représentation des nombres décimaux se fait selon la norme IEEE 754 et on obtient la même erreur avec beaucoup de logiciels (Excel, OpenOffice, XCas,...). En fait pour nous le nombre 2,99 est un nombre "simple" dans son écriture décimale, mais pour un ordinateur qui pense en base 2, c'est beaucoup plus complexe ! Plus d'informations sur la documentation officielle de Python

3) Les élèves éprouvent de grandes difficultés à utiliser « `for i in range ···` »

En effet, la boucle `for` et la définition de l'ensemble des valeurs prises par la variable sont deux difficultés simultanées, qui nécessitent un temps long d'appropriation par les élèves.
Il semble plus efficace de n'utiliser que la boucle `while` dans un premier temps. Il faut alors initialiser la variable avant la boucle, et incrémenter dans la boucle par `i = i + 1`, par exemple.
Après la répétition de ce genre de boucle, on peut proposer aux élèves le `for` et l'emploi du `range`.

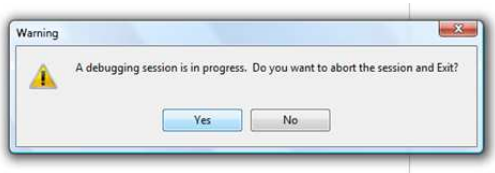
4) J'ai le message : "UnicodeEncodeError : 'ascii' codec...

Vous avez probablement utilisé un nom de fichier avec un accent pour votre programme ou dans le chemin...



5) Que signifie ce message d'erreur ?

Cela signifie qu'un programme en Python est déjà en train d'être exécuté. Il suffit de cliquer sur "YES" pour stopper l'exécution en cours.



III) AmiensPython et l'enseignement ISN

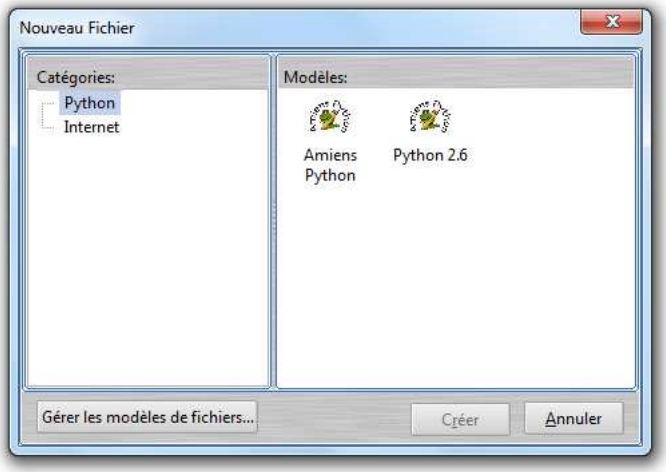
Les derniers modules python ajoutés dans la version 2.1 d'AmiensPython permettent un travail relativement complet en ISN dans différents champs comme :

- Le traitement du signal et de l'image (Numpy, Matplotlib, PIL)
- La liaison série (PySerial)
- Les bases de données (Sqlite3)
- ...

Ainsi, si les élèves utilisent AmiensPython depuis la seconde, il semble intéressant qu'ils puissent continuer à le faire puisqu'ils ont des habitudes de travail avec cet outil. Cependant les choix pédagogiques que nous avons mis en œuvre dans la bibliothèque *lycee* (nouvelles fonctions et choix d'utiliser une division décimale) ne sont pas des conventions du langage Python tel qu'un étudiant pourra les retrouver dans son cursus ultérieur.

Lors de la mise en place de l'ISN, nous trouvons donc pertinent de donner la possibilité de continuer à utiliser l'interface PyScripter d'AmiensPython pour réaliser des programmes Python en se passant de la bibliothèque *lycee*. Ce sera l'occasion d'échanger avec les élèves sur les différents types de nombres en tapant `1/3` et `1.0/3` ou en allant voir comment sont programmées les fonctions de la bibliothèque *lycee* dont on pourrait avoir besoin.

En cliquant sur « Nouveau Fichier », vous avez à présent la possibilité de choisir si vous souhaitez commencer un programme avec AmiensPython ou en Python classique.



Chapitre 11

Plus de 80 programmes

Sommaire

I) 40 programmes simples

Il nous semble important de proposer tout au long de l'année des programmes simples à réaliser, c'est pourquoi nous nous sommes efforcés d'illustrer au maximum cette documentation par des petits programmes. En voici la liste enrichie de commentaires :

Les bases

Télécharger — Que fait le programme suivant ?

Télécharger — Nombre de 6 obtenus en lançant un dé 10 fois

Télécharger — Calculer l'opposé d'un nombre

Télécharger — Calculer l'IMC

Télécharger — Nombre de zéros d'un trinôme

Télécharger — Signe d'un trinôme

Télécharger — Solutions de $ax + b = 0$

Télécharger — Image par une fonction définie par morceaux

Télécharger — Recherche d'un triplet pythagoricien d'entiers consécutifs

Télécharger — Distance sur un axe gradué

Télécharger — Volume d'eau dans une cuve

Télécharger — Le jeu du nombre mystère

Télécharger — Calcul de probabilités

Télécharger — Somme des carrés des 100 premiers entiers

Télécharger — Obtenir 7

Télécharger — Calcul du PGCD de 2 entiers strictement positifs

Télécharger — Multiplication babylonienne

Télécharger — Marche aléatoire d'un robot sur une table

Fonctions mathématiques

Télécharger — Recherche des entiers distincts tels que $x^y = y^x$

Télécharger Estimation du nombre e par suites adjacentes.
Télécharger Une division avec un nombre arbitraire de décimales
Télécharger Somme de deux fractions
Télécharger Calculer l'aire d'un triangle avec la formule de Héron
Télécharger Calcul de la longueur de l'hypoténuse dans un triangle rectangle
Télécharger Le pré et la chèvre
Télécharger Calculer le périmètre d'un disque
Télécharger Approximation historique du nombre π
Télécharger Un carré découpé en 3 zones
Télécharger 100 boules

La tortue

Télécharger Tracer d'une maison
Télécharger Tracer d'une église
Télécharger Tracer un polygone régulier à n côtés.
Télécharger Tracer une spirale.
Télécharger Tracer un œuf de pâques.
Télécharger Le yin et le yang.
Télécharger Dessiner un soleil
Télécharger Le drapeau européen.

Les graphiques

Télécharger Ligne de niveau
Télécharger Tracer d'une droite
Télécharger Marche aléatoire
Télécharger Chute d'une balle
Télécharger Pile ou face

II) 40 autres programmes plus élaborés

Télécharger Algorithme d'Euclide étendu

Les listes

Télécharger Vendredi 13
Télécharger Reconnaître un parallélogramme
Télécharger Produit de 2 polynômes
Télécharger Conjecture d'Euler
Télécharger Nombre de Kaprekar
Télécharger Suite
Télécharger Crible d'Ératosthène
Télécharger Polynome dérivé
Télécharger Compter le nombre de 6
Télécharger Algorithme de Kaprekar

Télécharger Méthode de Hörner

Probabilités et statistiques

Télécharger Dés de Sicherman
Télécharger Appel de jugement
Télécharger Elections
Télécharger Un tirage de LOTO
Télécharger Une urne de 50 boules
Télécharger Surréservation
Télécharger Codes de carte bleue
Télécharger Avoir une full d'enfants
Télécharger Entrées au cinéma
Télécharger Coefficient de Gini
Télécharger Comparaison de deux départements

Les chaînes de caractères

Télécharger Nombre de voyelles
Télécharger A l'envers
Télécharger Palindrome1
Télécharger Palindrome2
Télécharger Doubler les voyelles
Télécharger Hexadécimal 1
Télécharger Hexadécimal 2
Télécharger Codage affine
Télécharger Décodage affine
Télécharger Code ISBN
Télécharger Analyser une fraction
Télécharger Compter en Shadoks
Télécharger Tester un codage
Télécharger Poème version 1
Télécharger Poème version 2

Les fonctions

Télécharger La cuve sans fonction
Télécharger La cuve avec fonction
Télécharger Les briques de Brian
Télécharger Suite de Syracuse
Télécharger PGCD récursif
Télécharger Puissance récursive version 1
Télécharger Puissance récursive version 2
Télécharger Suite de Fibonacci

Index