PYTHON 3 pour une utilisation en classe

I. Introduction

Traduire les algorithmes en un langage de programmation et les faire fonctionner sur un ordinateur est certainement une nécessité pour rendre l'apprentissage de l'algorithmique plus vivant et plus attractif. La possibilité pour l'élève de tester son algorithme, de le corriger, favorise son autonomie et participe à l'apprentissage d'une démarche scientifique. Pour autant, les connaissances techniques à apporter (liées au langage de programmation) ne doivent pas être excessives. Ce document, destiné aux professeurs, propose une initiation au langage Python (version 3). Les techniques et les connaissances non nécessaires ou trop éloignées de celles utilisées en classe ont été proscrites.

Pour compléter votre formation sur Python, voici une liste de sites francophones sur lesquels vous trouverez la documentation nécessaire :

- Le site Developpez.com : http://python.developpez.com/

  Sur ce site vous trouverez l'excellent ouvrage de Gérard Swinnen « Apprendre à programmer avec Python »

  https://python.developpez.com/cours/apprendre-python3/

  Et aussi un cours pour débuter au lycée :

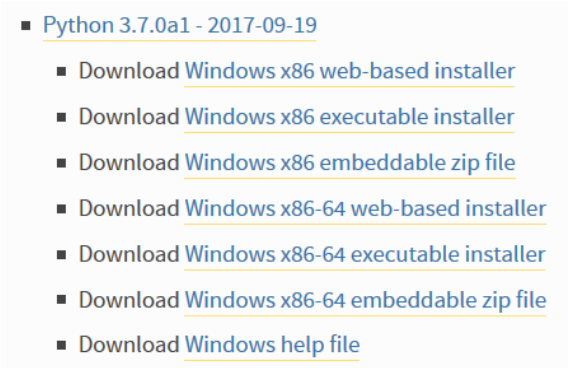
  https://python.developpez.com/tutoriels/debuter-avec-python-au-lycee/

- Un complément à l'ouvrage de Gérard Swinnen : http://www.apprendre-en-ligne.net/python3/

II. Installation (plateforme windows)

- Sur le site www.python.org, cliquez sur Download puis choisissez une version de Python 3 en fonction de votre système d'exploitation.

  Notez que le site propose directement le téléchargement pour windows de la version 3.6.2 (32 bit) ou de façon plus détaillée différentes versions pour différents systèmes d'exploitation, dont la dernière version pour windows :



Si vous disposez d'un système d'exploitation windows 32 bits, cliquez sur le lien « Download Windows x86 executable installer ». Si vous disposez d'un système d'exploitation windows 64 bits, cliquez sur le lien « Download Windows x86-64 executable installer».

Remarques concernant l'installation :

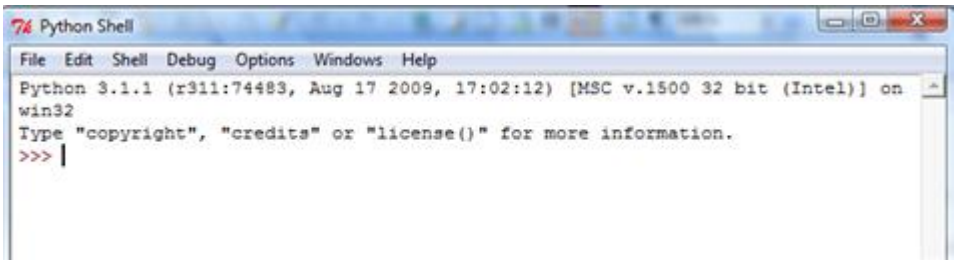
- Le premier lien fonctionnera avec un windows 32 bits ou 64 bits.
- Comment connaître son type de processeur sous windows:

  Cliquez sur  , faites un clic droit sur « ordinateur », puis sélectionnez « Propriétés » dans le menu contextuel. Les caractéristiques de votre ordinateur s'affichent alors.
  Ou cliquez sur « Démarrer » puis sur « Panneau de configuration » puis sur système. Les caractéristiques de votre processeur s'affichent alors.
- Cette installation sous Windows n'inclut pas les bibliothèques d'extension NumPy, SciPy et Matplotlib (permettant de travailler sur des tableaux, matrices et tout type de représentation graphique). On peut les installer par la suite mais si on envisage dès le départ de travailler avec ces outils scientifiques plus élaborés, on préférera l'installation d'une distribution de type pyzo qui facilite l'ajout de ces bibliothèques complémentaires.

- Exécutez le fichier téléchargé, l'installation se fait ensuite sans aucune difficulté (cliquez sur next à chaque ouverture de fenêtre)

III. Prise en main en mode interactif

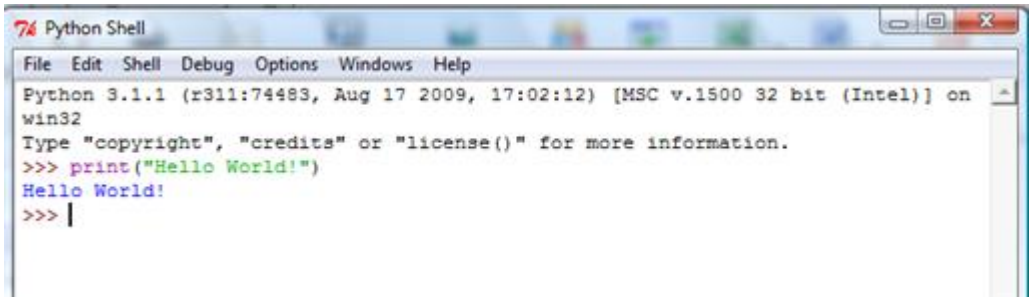
Cliquez sur bouton Démarrer puis sur « Tous les programmes » puis double-cliquez sur le dossier Python 3.x puis cliquez sur « IDLE (Python GUI) » (nom de l'environnement de travail).
Une fenêtre « Python Shell » s'ouvre alors.



L'invite de commande >>> signifie que Python est prêt à exécuter une commande.

Premiers exemples

- Tapez print("Hello World !"), puis appuyez sur la touche Entrée ;
  Python exécute cette commande. Le résultat de cette exécution est l'affichage de la chaîne de caractères Hello World !
  Une nouvelle invite de commande apparaît alors.



- Calculer avec Python :

```
>>> 3+2*5.2
13.4
>>> 5/2
2.5
>>> 3*(5+1/2)
16.5
>>> 2**5
32
>>> 12%5
2
```

Notez que la virgule des nombres décimaux doit être remplacée par le point.
2 exposant 5 s'écrit 2**5
12%5 renvoie le reste de la division euclidienne de 12 par 5.

Variable, nom, affectation, affichage, typage.

- Nom, affectation

```
>>> pi=3.1415
>>> R=3
>>> Adisque=pi*R**2
```

L'exécution de la première ligne crée une variable nommée pi contenant la valeur réelle 3.1415.
L'exécution de la deuxième ligne crée une variable nommée R contenant la valeur entière 3.
L'exécution de la troisième ligne crée une variable nommée Adisque contenant le résultat du calcul pi*R².

Le nom d'une variable est une séquence de lettres ordinaires (accents, cédilles ne sont pas autorisées) et de chiffres qui doit commencer par une lettre. Le caractère souligné _ est autorisé. Le nom d'une instruction (print, if, then, …) est interdit comme nom de variable.
Attention à la casse ! Le nom de variable Aire est différent du nom aire.

Le symbole = permet d'affecter une valeur à une variable.

- Affichage du contenu d'une variable

Pour afficher la valeur d'une variable, il suffit de taper son nom puis d'appuyer sur la touche Entrée ou bien taper print(Nom_de_la_variable).

```
>>> pi=3.1415

>>> R=3

>>> Adisque=pi*R**2
```

```
>>> Adisque
28.273500000000002
>>> print(Adisque)
28.2735
>>> print("L'aire d'un disque de rayon",R,"cm est égale à",Adisque,"cm²")
L'aire d'un disque de rayon 3 cm est égale à 28.2735 cm²
>>>
```

Notez qu'il est possible d'afficher plusieurs éléments à la suite en utilisant une seule fois l'instruction print. Il suffit pour cela de séparer les différents éléments à afficher par une virgule, comme le montre l'exemple ci-dessus.

- Typage :

Il existe différents types de variable : le type entier (int), le type nombre à virgule (float), le type chaîne de caractères (str),le type liste (list), le type Booléen (bool)…

Saisissez les lignes suivantes :

```
>>> a="Hello World !"
>>> b=3
>>> c=2.5
>>> d=[7,3,145]
>>> e=False
>>>
```

- La variable a contient une chaîne de caractères, elle sera de type str.

  A noter : dès que la valeur d'affection d'une variable est entre guillemets, la variable sera du type « chaîne de caractères » (str).
  Par exemple, si vous saisissez a="3" (ou a='3'), la variable a est du type chaîne de caractères et la valeur de a n'est pas considérée comme un nombre mais comme du texte ! (effectuer l'opération a+2 n'aurait aucun sens)

- La variable b contient un entier, elle sera de type int.
- La variable c contient un nombre à virgule, elle sera de type float.
- La variable d contient une liste, elle sera du type list.
- La variable e contient un booléen, elle sera du type bool (une variable de type bool peut prendre 2 valeurs True ou False).

  A noter : Avec Python, il n'est pas nécessaire de définir préalablement le type de la variable. Le typage se fait automatiquement lors de l'affectation d'une valeur à la variable.

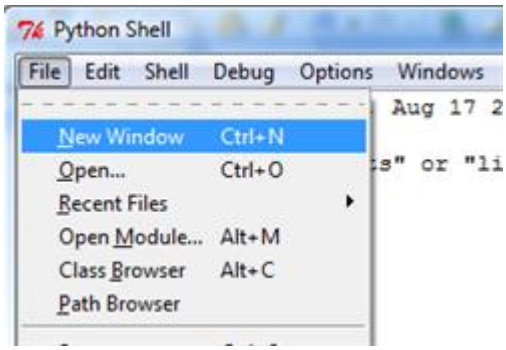
Pour connaître le type d'une variable il suffit de taper type(nom_de_la_variable) :

```
>>> a="Hello World!"
>>> b=3
>>> c=2.5
>>> d=[7,3,145]
>>> e=False
>>> type(a)
<class 'str'>
>>> type(b)
<class 'int'>
>>> type(c)
<class 'float'>
>>> type(d)
<class 'list'>
>>> type(e)
<class 'bool'>
>>>
```

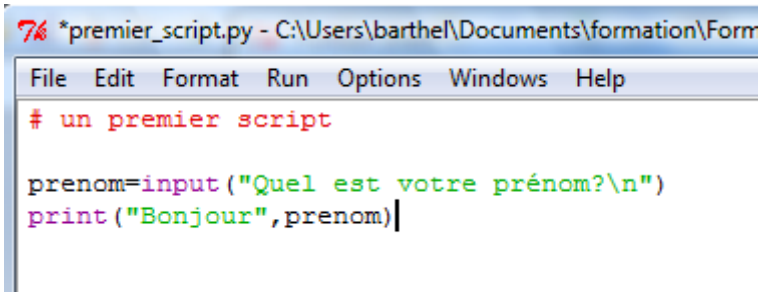
IV. Ecrire, conserver, ouvrir, exécuter un programme

En mode interactif, les lignes d'instruction ne sont plus accessibles une fois exécutées. Dans cette partie, vous apprendrez à conserver un programme pour pouvoir l'exécuter à loisir ou pour le modifier ultérieurement.

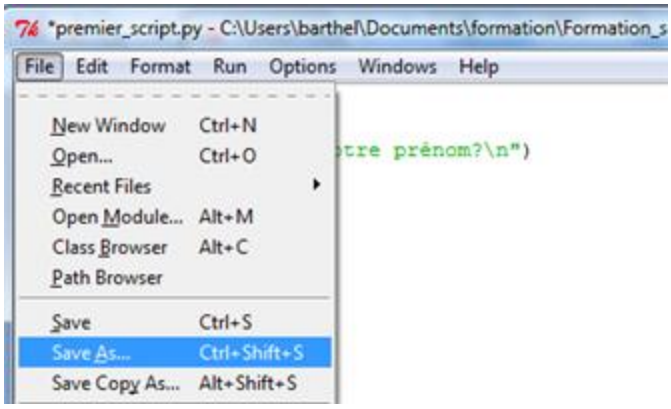
- Dans la fenêtre Python Shell (celle du mode interactif), sélectionnez New Window dans le menu File :



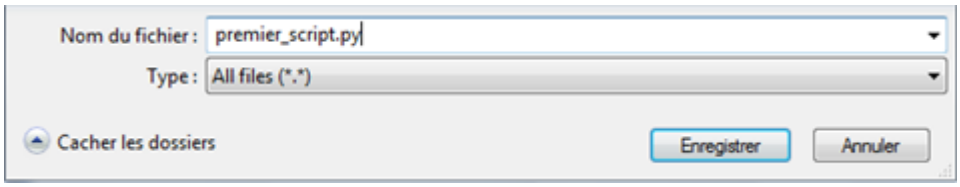
- Une nouvelle fenêtre s'ouvre alors. C'est dans cette fenêtre que vous écrirez votre premier programme. Tapez le script suivant :



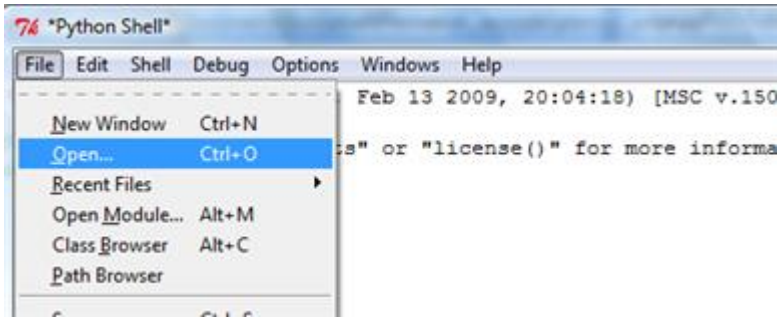
- Enregistrement du programme :
  Sélectionnez « Save as » dans le menu File



une fenêtre d'enregistrement s'ouvre alors. Choisissez dans l'arborescence le dossier dans lequel vous voulez enregistrer le programme, puis dans le champ d'enregistrement du fichier saisissez le nom du programme suivi de l'extension .py, puis cliquez sur enregistrer :

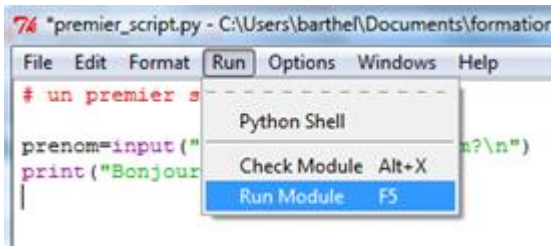


- Pour ouvrir un programme python, il suffira de sélectionner Open dans le menu File de l'environnement IDLE puis de chercher dans vos dossiers le fichier python à ouvrir.



- Exécution du programme :

  Pour exécuter le programme, il suffit de sélectionner « Run Module » dans le menu Run (si une modification du script a été effectuée, on vous proposera d'enregistrer le script modifié avant de l'exécuter).



  Le programme s'exécute dans la fenêtre Python shell.

V. Des exemples pour apprendre Python

Programme 1 :

sujets abordés : lignes de commentaires, fonction input(), fonction print()

Reprenons le script précédent :

```
# un premier script


prenom=input("Quel est votre prénom?\n")

print("Bonjour",prenom)
```

Première ligne : # un premier script

La première ligne est une ligne de commentaires. Les lignes précédées du symbole # sont ignorées par Python ; elles servent à une meilleure compréhension du programme par le lecteur du script. Prenez l'habitude d'insérer des lignes de commentaires afin de rendre votre programme plus intelligible.

Deuxième ligne : prenom=input("Quel est votre prénom ?\n")

La fonction input suspend l'exécution du programme et attend que l'utilisateur saisisse une chaîne de caractères au clavier, puis valide sa saisie en appuyant sur la touche Entrée.
Dans notre exemple, cette chaîne de caractères saisie par l'utilisateur est la valeur de la variable prenom (qui sera du type str).
Le texte entre guillemets "Quel est votre prénom ?" est affiché avant la demande de saisie.
\n à la fin du texte est interprété par Python comme un retour chariot.

Troisième ligne : print("Bonjour",prenom)

Cette ligne affiche le texte Bonjour suivi de la valeur de la variable prenom.

Programme 2 :

sujet abordé : structure itérative (instruction while, instruction for)

Un épargnant place 10000 € sur un compte rémunérant à un taux annuel de 3% à intérêts composés. Tous les ans, le jour de la perception des intérêts, il dépose 1000 euros sur ce compte. Cet épargnant ne retire jamais d'argent de ce compte.

L'objectif du programme suivant est de calculer la valeur du capital acquis au bout de 10 ans de placement après y avoir déposé une dernière fois les 1000 euros.

L'algorithme permettant le calcul du capital acquis au bout de 10 ans

Affecter à S la valeur 10000.

Affecter à n la valeur 1.

Tant que n≤10 faire :

- Affecter à S la valeur 1,03S+1000.
- Affecter à n la valeur n+1.

Afficher S.

Traduction de l'algorithme en langage Python

```
#Calcul du capital acquis au bout de 10 ans


S=10000

n=1

while n<=10:

   S=1.03*S+1000

   n=n+1

print("Le capital acquis au bout de 10 ans s'élève à",S,"euros")
```

A noter :

- Dans l'utilisation de l'instruction while (tant que) la fin de la condition est marquée par le double-point et un retour à la ligne.
- Le bloc d'instructions à exécuter sous condition doit nécessairement être indenté. (sous l'environnement IDLE, l'indentation est automatique)

<= est un opérateur de comparaison. Les opérateurs de comparaison sont :
x==y (x est égal à y)
x != y (x est différent de y)
x<y (x est strictement inférieur à y)
x<=y (x est inférieur ou égal à y)
x>y (x est strictement supérieur à y)
x>=y (x est supérieur ou égal à y)

Remarque importante :
Lors de la création d'une structure itérative avec while, il faut bien vérifier que l'exécution du bloc d'instructions s'effectue un nombre fini de fois. Si vous oubliez la ligne n=n+1 dans le programme précédent, la condition n<=10 sera toujours vérifiée et le programme tournera en boucle à l'infini ! (où du moins jusqu'à surcharge de la mémoire)

Instruction for utilisée avec range :

Exemple:

```
for i in range(10) :
    print(i)
```

range(10) est la liste des 10 premiers entiers naturels.
La traduction littérale de ce script est : Pour l'entier i variant de 0 à 9 avec un pas de 1, afficher i.
Le résultat de l'exécution de ce script est l'affichage des entiers de 0 à 9.

Voir les créations de liste avec la fonction range au paragraphe VI.9).

Autre exemple :

```
for i in range(4,20,2) :
    print(i)
```

Pour i variant de 4 à 18 avec un pas de 2, afficher i.
Le résultat de l'exécution de ce script est l'affichage des nombres : 4, 6,8,…,18.

Le programme précédent du calcul du capital au bout de 10 ans peut s'écrire avec la structure itérative for de la manière suivante :

```
#Calcul du capital acquis au bout de 10 ans


S=10000
for n in range(1,11):
   S=1.03*S+1000


print("Le capital acquis au bout de 10 ans s'élève à",S,"euros")
```

Programme 3 :

sujet abordé : convertir un type de variable.

On reprend les hypothèses du placement bancaire précédent.
L'objectif du programme suivant est de déterminer le capital acquis au bout d'un nombre d'années donné par l'utilisateur (nombre d'années compris entre 1 et 50).

L'algorithme permettant le calcul du capital acquis au bout d'un nombre d'années (compris entre 0 et 50 ans) choisi par l'utilisateur de l'algorithme.

Affecter à S la valeur 10000.

Affecter à nbre_an une valeur entière, comprise entre 0 et 50, choisie par l'utilisateur.

Affecter à n la valeur 1.

Tant que n≤nbre_an faire :

- Affecter à S la valeur 1,03S+1000.
- Affecter à n la valeur n+1.

Afficher S.

Traduction de l'algorithme en langage Python :

```
# Calcul du capital acquis au bout d'un nbre d'années fixé
# par l'utilisateur

S=10000
nbre_an=int(input("Saisissez le nombre d'années de placement(entre 0 et 50)\n"))
n=1
while n<=nbre_an:
    S=1.03*S+1000
    n=n+1
print("Le capital acquis au bout de",nbre_an,"ans s'élève à",S,"euros")
```

Nous avons vu lors de l'étude du programme 1 que l'affectation d'une variable à l'aide de l'instruction input impliquait nécessairement que cette variable soit de type str (chaîne de caractères).
Si vous tapez : nbre_an=input("Saisir le nombre d'années de placement"), la valeur de la variable nbre_an ne sera pas considérée comme un nombre mais comme du texte !

Or ici nous souhaitons que la variable nbre_an soit de type entier. Pour cela il faut convertir la chaîne de caractères saisie en un entier à l'aide de la fonction int().
Remarque : si l'utilisateur avait eu à saisir un nombre décimal, on aurait converti la chaîne de caractères en nombre décimal en utilisant la fonction float() :
float("3.2") convertit la chaîne de caractères saisie entre guillemets en le nombre décimal 3.2

Exercice 1: Réaliser un programme Python qui demande à l'utilisateur la somme du capital à atteindre et qui renvoie le nombre minimum d'années de placement pour que le capital acquis soit supérieur à cette somme.

Programme 4 :

Sujets abordés : Instruction if-else, imbrication des instructions, opérateur de comparaison « == »

Le programme suivant a pour objectif de déterminer des valeurs approchées des racines éventuelles d'un trinôme du second degré $ax^2 + bx + c$ où a,b,c sont des constantes décimales dont les valeurs sont demandées à l'utilisateur.

```
#Racine d'un trinôme du second degré


from math import *


print("Racines éventuelles de ax²+bx+c")
a=float(input("Saisir la valeur de a\n"))
b=float(input("Saisir la valeur de b\n"))
c=float(input("Saisir la valeur de c\n"))
Delta=b**2-4*a*c
print("Delta=",Delta)
if Delta<0:
   print("Ce trinôme n'a pas de racines")
else:
   if Delta==0:
      x1=-b/(2*a)
      print("Delta=0, ce trinôme a une seule racine",x1)
   else:
      x1=(-b-sqrt(Delta))/(2*a)
      x2=(-b+sqrt(Delta))/(2*a)
      print("Delta>0, Ce trinôme a deux racines\n",x1,"\n", x2)
```

L'utilisation de l'instruction if (si) et else (sinon) se fait de la même manière que l'instruction while (double-point après la condition, indentation du bloc d'instructions à exécuter sous la condition).

Notez que l'opérateur de comparaison « est égal » est ==. Il se distingue de l'opérateur = d'affectation d'une variable.

Observez l'imbrication des instructions !

On aurait également pu utiliser l'instruction « elif » qui est une contraction de else if (sinon si) et qui permet d'obtenir un script plus lisible :

```
#Racine d'un trinôme du second degré


from math import *


print("Racines éventuelles de ax²+bx+c")
a=float(input("Saisir la valeur de a\n"))
b=float(input("Saisir la valeur de b\n"))
c=float(input("Saisir la valeur de c\n"))
Delta=b**2-4*a*c
print("Delta=",Delta)
if Delta<0:
    print("Ce trinôme n'a pas de racines")
elif Delta==0:
    x1=-b/(2*a)
    print("Delta=0, ce trinôme a une seule racine",x1)
else:
    x1=(-b-sqrt(Delta))/(2*a)
    x2=(-b+sqrt(Delta))/(2*a)
    print("Delta>0, Ce trinôme a deux racines\n",x1,"\n", x2)
```

Programme 5 : Suite de Syracuse

Sujet abordé : imbrication des instructions, opérateur de comparaison « != »

La suite de Syracuse est définie par récurrence de la manière suivante :

Le premier terme u_0 est un entier naturel supérieur ou égal à 1 et pour tout entier n :

si u_n est pair $u_{n+1} = \frac{u_n}{2}$

si u_n est impair $u_{n+1} = 3u_n + 1$.

Quelle que soit la valeur u_0 choisie au départ, cette suite prendra la valeur 1 (propriété conjecturée qui n'a pas encore été démontrée à ce jour).
L'objectif du programme suivant est de déterminer le plus petit indice n pour lequel u_n =1 en fonction de la valeur de u_0 donnée par l'utilisateur.

Algorithme :

Affecter à U le choix par l'utilisateur d'un entier naturel non nul.

Affecter à N la valeur 0.

Tant que U $\neq$ 1 faire :

- Si U est pair alors affecter à U la valeur U/2
  sinon affecter à U la valeur 3U+1.
- Augmenter N de 1.

Afficher le texte « Le premier indice pour lequel Un=1 est » et la valeur de N.

Traduction de l'algorithme en langage Python

```
#Suite de Syracuse

#Calcul du premier indice pour lequel Un=1 en fonction de la valeur

#U0 donnée par l'utilisateur.



U=int(input("Quelle est la valeur de U0?(entier naturel)\n"))

N=0

while U!=1:

   if U%2==0:

      U=U/2

   else:

      U=3*U+1

   N=N+1

print("Le premier indice pour lequel Un=1 est",N)
```

Exercice 2: Le plus petit indice *n* pour lequel le terme général de la suite de Syracuse est égal à 1 est appelé le temps de vol de la suite.

Ecrire un programme qui permet de déterminer la plus petite valeur de u_0 comprise entre 0 et 10000 telle que le temps de vol soit le plus grand. La valeur de u_0 et le temps de vol correspondant seront affichés.

Programme 6 : Simulation de lancers d'un dé équilibré

Sujets abordés : Module random et fonction randint de ce module, listes

Le programme ci-dessous simule 1000 lancers d'un dé équilibré et calcule la fréquence d'apparition de chacune des faces.

La liste L contient les effectifs d'apparition des différentes faces.

Algorithme

Affecter à L la liste [0,0,0,0,0,0].

Affecter à i la valeur 1.

Tant que i≤1000 faire :

- Affecter à n le choix aléatoire d'un nombre compris entre 1 et 6.
- Augmenter de 1 le nième nombre de la liste L.
- Augmenter i de 1.

Affecter à j la valeur 1.

Tant que j≤6 faire :

- Afficher la valeur du quotient du jème nombre de la liste par 1000.
- Augmenter j de 1.

Traduction en langage Python

```
from random import * #importation des fonctions du module random


i=1 #initialisation du compteur du nombre de simulations

L=[0,0,0,0,0,0] #initialisation de la liste contenant les effectifs

            # d'apparition des différentes faces


while i<=1000:

   n=randint(1,6)

   L[n-1]=L[n-1]+1 # Le nième nombre de la liste est augmenté de 1

   i=i+1


#Affichage des fréquences

j=1

while j<=6:

   print("La fréquence d'apparition de la face",j,"est égale à", L[j-1]/1000)

   j=j+1
```

A noter :

- La fonction randint(entier1,entier2) génère aléatoirement un nombre compris entre entier1 et entier2. Pour utiliser cette fonction, il faut l'importer du module random. La première ligne du programme permet cette importation :

  « from random import * » importe toutes les fonctions du module random. Ici on aurait pu se contenter d'importer uniquement la fonction randint et d'écrire « from random import randint »

- Le premier élément d'une liste L est l'élément d'index 0, L[0] ; le nième élément d'une liste est L[n-1]. Attention au décalage !

Programme 7 : Simulation de tirages sans remise

Sujets abordés : Listes, méthodes sur les listes (append, remove), fonction del, fonction choice du module random

Le programme ci-dessous simule 6 tirages successifs, sans remise, d'une boule dans une urne contenant 49 boules numérotées de 1 à 49.

Le résultat des tirages est mémorisé dans une liste.

Algorithme :

Affecter à U la liste [1,2,3,…,49].

Affecter à L la liste vide.

Affecter à j la valeur 1.

Tant que j<=6 faire :

- Affecter à Tirage le choix aléatoire d'un élément de U.
- Supprimer l'élément Tirage de la liste U.
- Ajouter Tirage à la fin de la liste L.
- Affecter à j la valeur j+1.

Afficher la liste L.

La liste U contient les entiers de 1 à 49. L'ordinateur retire un nombre de cette liste choisi au hasard et l'ajoute à la liste L et ceci six fois de suite.

Traduction de l'algorithme en langage Python

```
#Importation des fonctions du module random
from random import *


U=list(range(1,50))    #Affectation de [1,2,..,49] à la liste U
L=[]    #Affectation de la liste vide à L
j=1
while j<=6:
   Tirage=choice(U)
   U.remove(Tirage)
   L.append(Tirage)
   j=j+1
print(L)
```

A noter :

- range(a,b) où a et b désigne des entiers (a<b) crée une pseudo liste [a,a+1,…,b-1]. Cette pseudo liste est de type range, pour la convertir en type liste on utilise la fonction de conversion list() (voir fonction range() dans le paragraphe « compléments sur les listes).
- La fonction choice est une fonction du module random. choice(L) renvoie au hasard un élément de la liste L.
- U.remove(Tirage) supprime le premier élément de la liste égal à la valeur de Tirage.
- L.append(Tirage) ajoute à la fin de la liste L la valeur de Tirage. Pour appliquer cette méthode, il est nécessaire que la liste L ait été préalablement définie. C'est pourquoi on affecte à L la liste vide L=[] avant d'utiliser cette méthode pour la première fois.

Remarque : Ce programme utilise de nombreux outils de Python (range, list , choice, remove, append). Il peut être intéressant pour la formation de l'élève de construire la liste U=[1,2,…,49] sans utiliser range et de réaliser le choix au hasard d'un élément de la liste sans utiliser choice (en choisissant un index de la liste au hasard). Dans ce cas le script est plus lourd et l'écriture du programme n'est pas une simple traduction ligne par ligne de l'algorithme précédent.

Voici le programme :

```
from random import randint
#Création de la liste U=[1,2,...,49]
U=[]
i=1
while i<=49:
    U.append(i)
    i=i+1
#Simulation de 6 tirages sans remise et stockage des résultats dans L
L=[]      #Création d'une liste vide L
j=0
while j<=5:
    n=randint(0,48-j)      #choix aléatoire d'un index de la liste U
    L.append(U[n])         # Ajout à la liste L de l'élément de la liste U d'index n
    del U[n]               #Suppression de l'élément d'index n de la liste U
    j=j+1
print(L) #Affichage de la liste obtenue
```

A noter : del U[n] supprime l'élément d'index n de la liste U

Programme 8 : **Définir une fonction**

Sujets abordés : module math, fonction python, fonction eval

Le programme suivant montre la manière de définir la fonction f1 : $x \mapsto 3\sqrt{x}+2$.

L'utilisateur est invité à entrer la valeur de x dont on veut déterminer l'image par f1 puis l'ordinateur affiche la valeur de f1(x).

```
#Importation des fonctions du module math
from math import *
#Création de la fonction f1
def f1(x):
    if x<0:          #Si x n'est pas dans le domaine de définition
                     # alors la valeur False est renvoyée
        return False
    else:
        return 3*sqrt(x)+2


#Structure principale du programme
x=eval(input("Quelle est la valeur de x?\n"))
print("f1(",x,")=",f1(x))
```

A noter :

- La définition d'une fonction se fait par l'instruction def suivie du nom de la fonction, des paramètres entre parenthèses puis du double-point.
- L'élément à retourner en fonction des paramètres se fait à l'aide de l'instruction return.
- Dans ce programme, la fonction sqrt du module math est utilisée. Il est donc nécessaire d'importer cette fonction ; la première ligne du programme importe toutes les fonctions du module math.
- **Fonction eval** :
  x=float(input("Quelle est la valeur de x")) permet uniquement à l'utilisateur de saisir des nombres décimaux. Il est souhaitable que l'utilisateur puisse saisir un nombre sous forme de puissance (3**2 par exemple) ou sous forme d'une somme 2+sqrt(3). Le texte saisi doit être interprété par Python, c'est pourquoi nous utilisons ici la fonction eval qui comme son nom l'indique évalue la chaîne de caractères saisie comme une commande Python.

Exemple :

```
>>> from math import *
>>> eval("sqrt(2)/2")
0.7071067811865476
>>> eval("print('Bob')")
Bob
```

VI. L'utilisation de fonctions :

Sous python on peut créer toutes sortes de fonctions introduites par le mot « def ». Une fonction peut avoir un nombre quelconque d'arguments (on dit aussi paramètres). Elle peut renvoyer zéro, un ou plusieurs éléments ; dans le cas où elle n'en retourne pas, on parle parfois de procédure. L'utilisation de procédures ou fonctions est très utile pour plusieurs raisons :

- lorsqu'elle est appelée plusieurs fois dans un même programme, une fonction améliore la lisibilité du script.
  Exemple :

```
def table(n):
    i=0
    while i<=10:
        print(n,"*",i,"=",n*i)
        i=i+1
#Structure principale du programme
#Appel de la procédure table
print("table de 7")
table(7)
print("table de 13")
table(13)
```

  Les variables définies dans une fonction sont des variables locales ! Elles n'existent qu'au sein de la fonction !

- l'utilisation d'une fonction permet de se libérer des instructions d'entrée et sortie (input et print), ce qui simplifie la saisie des variables dont on n'aura pas nécessairement à expliciter le type.

  Exemple : le programme 4 peut ainsi être réécrit à l'aide d'une fonction

```
#Racine d'un trinôme du second degré

from math import *


def degre2(a,b,c) :
        delta = b*b-4*a*c
        if delta >0 :
                x1 = (-b+sqrt(delta))/(2*a)
                x2 = (-b-sqrt(delta))/(2*a)
                return (x1,x2,s,p)
        elif delta == 0 :
                x0 = -b/(2*a)
                return x0
        else :
                return("pas de racine")
```

De nombreux exemples de fonctions sont proposés dans le document ressource « Algorithmique et programmation », accompagnant les aménagements du programme de 2nde qui introduit la notion de fonction dans les programmes de mathématiques du lycée.
http://cache.media.eduscol.education.fr/file/Mathematiques/73/3/Algorithmique_et_programmation_787733.pdf

VII. Python et les mathématiques

L'utilisation de certaines fonctions ou constantes mathématiques nécessite l'importation du module math.
Pour cela, il est nécessaire d'écrire la ligne suivante au début du programme :
from math import * (voir par exemple programme 4 du chapitre IV).
Les exemples sont donnés en mode interactif.

Constantes mathématiques

- Constante π : pi (à importer du module math)

```
>>> from math import *
>>> pi
3.1415926535897931
```

- Constante e (exp(1)) : e (à importer du module math)

```
>>> from math import *
>>> e
2.7182818284590451
```

Quelques outils et fonctions mathématiques

- Calculer une puissance : a**b

```
>>> (-3)**2
9
>>> 3**1.2
3.7371928188465517
```

(-3)**2 renvoie $(-3)^2$

3**1.2 renvoie $3^{1,2}$.

- Reste de la division euclidienne de a par b : a%b

```
>>> 12%5
2
>>> -5%2
1
```

12%5 renvoie le reste de la division euclidienne de 12 par 5
-5%2 renvoie le reste de la division euclidienne de -5 par 2

- Fonction valeur absolue : abs(x)

```
>>> abs(-3)
3
>>> abs(5)
5
```

abs(-3) renvoie 3
abs(5) renvoie 5

- Fonction partie entière : floor(x) (fonction du module math)

```
>>> from math import *
>>> floor(4.2)
4
>>> floor(-5.3)
-6
```

floor(4.2) renvoie 4
floor(-5.3) renvoie -6

- Fonction racine carrée : sqrt(x) (fonction du module math)

```
>>> from math import *
>>> sqrt(4)
2.0
```

- Fonction exp : exp(x) (fonction du module math)

```
>>> from math import *
>>> exp(3)
20.085536923187668
>>> e**3
20.085536923187664
```

Remarque : Pour calculer exp(3) on peut également saisir e**3

- Fonctions log : log(x[,base]) (fonction du module math)

  log(x[,base]) renvoie le logarithme de base « base ». Si base n'est pas spécifié c'est le logarithme népérien qui est renvoyé.

```
>>> from math import *
>>> log(e)
1.0
>>> log(10**3,10)
2.9999999999999996
```

  log(e) renvoie ln(e).
  log(10**3,10) renvoie $\log(10^3)$ (logarithme décimal)

- Fonctions sinus, cosinus, tangente : sin(x), cos(x), tan(x) (fonctions du module math)

  La mesure de l'angle x doit être en radians.

```
>>> from math import *
>>> cos(pi/4)
0.70710678118654757
>>> sin(2*pi/3)
0.86602540378443871
>>> tan(pi/5)
0.7265425280053609
```

- Fonctions de conversion : degrees(radians), radians(degrés) (fonctions du module math)

  degrees(radians) convertit en degrés un angle exprimé en radians
  radians(degrés) convertit en radians un angle exprimé en degrés

Générateurs aléatoires de nombres : fonctions du module random

Pour utiliser les fonctions du module random, il faut au préalable les importer du module random en saisissant « from random import * » au début du programme.

- **randint(entier1,entier2)**

  randint(entier1,entier2) renvoie un nombre compris entre entier1 et entier2.

```
>>> from random import *
>>> randint(1,6)
3
```

- **random()**

  random() renvoie un nombre réel compris entre 0 et 1

```
>>> from random import *
>>> random()
0.66023433559956712
```

- **uniform(a,b)**

uniform(a,b) renvoie un nombre réel compris entre a et b

```
>>> from random import *
>>> uniform(3.1,4.5)
4.417971571551254
```

- **choice(L)**

choice(L) renvoie au hasard un élément de la liste L.

```
>>> from random import *
>>> choice([1,2,3,10,12])
3
>>> choice([1,2,3,10,12])
2
```

Fonctions logiques

- **Fonction and**

condition1 and condition 2 renvoie le booléen True si les deux conditions sont vérifiées, False sinon.

Exemple :

```
>>> a=5
>>> a%2==0 and a%5==0
False
```

La condition 1 n'est pas vérifiée (a n'est pas divisible par 2), le booléen False est retourné.

and s'utilise généralement avec les instructions conditionnelles (if, while).

Exemple :

```
>>> a=20
>>> if a%2==0 and a%5==0:
        print("le nombre",a,"est divisible par 10")


le nombre 20 est divisible par 10
```

- **Fonction or**

condition 1 or condition 2 retourne le booléen False si aucune des deux conditions n'est vérifiée et True sinon.

or s'utilise généralement avec les instructions conditionnelles (if, while).

Exemple :

Le programme suivant renvoie l'image de *x* par la fonction f : $x \mapsto \sqrt{x^2-4}$ à condition que *x* appartienne au domaine de définition de f.

```
from math import *
x=float(input("saisir la valeur de x\n"))
if x<-2 or x>2:
    print("L'image de",x,"par f est",sqrt(x**2-4))
```

VIII. Complément sur les listes

Accéder à un élément d'une liste

Vous avez déjà vu que pour accéder aux éléments d'une liste, il suffisait de saisir le nom de la liste suivi de l'index de l'élément entre crochets. L'index du premier élément est 0.

Exemple :

```
>>> fruit=["pomme","poire","prune","cerise"]
>>> fruit[0]
'pomme'
>>> fruit[3]
'cerise'
```

Remplacer un élément d'une liste

L[i]=e remplace l'élément d'index i de la liste L par l'élément e

Exemple :

```
>>> fruit=["pomme","poire","prune","cerise"]
>>> fruit[2]="mirabelle"
>>> print(fruit)
['pomme', 'poire', 'mirabelle', 'cerise']
```

Ajout d'un élément à une liste

- Ajout en fin de liste avec la méthode append

  L.append(élément) ajoute l'élément donné entre parenthèses à la fin de la liste L.

  Exemple :

```
>>> Legumes=[] #La variable Legume est une liste vide
>>> Legumes.append("Tomates")
>>> print(Legumes)
['Tomates']
>>> Legumes.append("Haricots")
>>> print(Legumes)
['Tomates', 'Haricots']
```

- Ajout d'un élément à un emplacement de la liste

  L.insert(pos,e) ajoute l'élément e à la liste de telle sorte que l'index de e soit pos.

  Exemple :

```
>>> Legumes=["Tomates","Haricots","Concombres"]
>>> Legumes.insert(2,"radis")
>>> print(Legumes)
['Tomates', 'Haricots', 'radis', 'Concombres']
>>> print(Legumes[2])
radis
```

Suppression d'un élément d'une liste connaissant son index

del L[i] supprime l'élément d'index i de la liste L

Exemple :

```
>>> print(Legumes)
['Tomates', 'Haricots', 'radis', 'Concombres']
>>> del Legumes[2]
>>> print(Legumes)
['Tomates', 'Haricots', 'Concombres']
```

Suppression d'un élément de liste connaissant sa valeur

L.remove(e) supprime le premier élément de la liste L qui a la même valeur que e. Si aucun élément n'est trouvé, une erreur est retournée.

Exemple :

```
>>> L=[1,2,3,1,5,6]
>>> L.remove(2)
>>> print(L)
[1, 3, 1, 5, 6]
>>> L.remove(1)
>>> print(L)
[3, 1, 5, 6]
```

Index d'un élément de la liste

L.index(e) renvoie l'index du premier élément dont la valeur est e. Une erreur est renvoyée si e n'est pas dans la liste.

Exemple :

```
>>> L=[5,7,8,19]
>>> L.index(8)
2
```

Nombre d'occurrences de l'élément e

L.count(e) indique le nombre d'occurrences de l'élément e.

Exemple :

```
>>> L=[3,2,3,2,2,5,2,7]
>>> L.count(2)
4
```

Longueur d'une liste

len(L) renvoie la longueur de la liste L

Exemple :

```
>>> print(Legumes)
['Tomates', 'Haricots', 'Concombres']
>>> len(Legumes)
3
```

Création d'une liste d'éléments de progression arithmétique entière

range(a,b,step) (a,b et step sont des entiers) renvoie la pseudo liste des entiers variant de l'entier a à l'entier b-1 avec un pas de step). Si le nombre step n'est pas spécifié il prendra par défaut la valeur 1. Si a et step ne sont pas spécifiés, a prendra par défaut la valeur 0 et step la valeur 1.
Sous python 3, l'objet créé avec range est du type range et il est impossible d'appliquer les méthodes des listes sur un tel objet. C'est pourquoi il est souvent utile de convertir l'objet de type range en type list avec la fonction list.

```
>>> list(range(1,12,2))
[1, 3, 5, 7, 9, 11]
>>> list(range(1,10))
[1, 2, 3, 4, 5, 6, 7, 8, 9]
>>> list(range(8))
[0, 1, 2, 3, 4, 5, 6, 7]
>>> list(range(-1,-5,-1))
[-1, -2, -3, -4]
```

Test d'appartenance d'un élément dans une liste

Exemple :

```
>>> print(Legumes)
['Tomates', 'Haricots', 'Concombres']
>>> if "Haricots" in Legumes: #Si "Haricots" est dans Legumes
        print("'Haricots' est dans cette liste")


'Haricots' est dans cette liste
```

Trier une liste numérique

- Dans l'ordre croissant

  L.sort() trie les éléments de L (liste numérique) dans l'ordre croissant.

```
>>> L=[5,4,12,1]
>>> L.sort()
>>> print(L)
[1, 4, 5, 12]
```

- Dans l'ordre décroissant
  L.reverse() permute les éléments de la liste L de telle sorte que le premier élément devient le dernier, le deuxième l'avant-dernier, etc..

  Pour trier une liste dans l'ordre décroissant il suffit donc d'appliquer successivement les méthodes sort() puis reverse() à la liste L

Exemple :

```
>>> L=[5,4,12,1]
>>> L.sort()
>>> L.reverse()
>>> print(L)
[12, 5, 4, 1]
```

Tableaux comme liste de listes

Exemple :
Considérons le tableau suivant :

5	7	12
3	2	1

On peut stocker les éléments de ce tableau dans une liste de la manière suivante :
tableau=[[5,7,12],[3,2,1]]
L'élément 3 de ce tableau est l'élément d'index 0 de l'élément d'index 1 du tableau.
tableau[1][0] renvoie l'élément d'index 0 de l'élément d'index 1 du tableau (l'élément d'index 1 du tableau est la liste [3,2,1]) c'est-à-dire 3.

```
>>> tableau=[[5,7,12],[3,2,1]]
>>> tableau[1][0]
3
```

IX. Autres exemples

Politique nataliste

Un pays a mis en place la politique nataliste suivante :
Les couples font des enfants jusqu'à ce qu'ils obtiennent un garçon.
Cette politique a-t-elle une influence sur la proportion de filles et de garçons dans ce pays ?

On va simuler les naissances d'enfants de 10000 couples après la mise en place de cette politique et déterminer les fréquences de filles et de garçons.

Exemple de programme :

```
#Politique nataliste

from random import *


k=1  # k représente le numéro du couple (k varie entre 1 et 10000)

N_f=0 # N_f contient le nombre de filles nées

N_g=0  # N_g contient le nombre de garçons nés

while k<=10000:

   sexe=-1         # la variable sexe est initialisée à -1

                    # Le chiffre 0 représente la naissance d'une fille

                    # et le chiffre 1 la naissance d'un garçon

   while sexe!=1:           # tant que l'enfant n'est pas un garçon

      sexe=randint(0,1)

      if sexe==0: # Si l'enfant est une fille

         N_f=N_f+1

      else:

         N_g=N_g+1

   k=k+1


print("la fréquence de filles est égale à",N_f/(N_f+N_g))

print("la fréquence de garçons est égale à", N_g/(N_f+N_g))
```

Valeurs approchées du minimum et du maximum d'une fonction sur un segment [a,b]

L'expression de la fonction, la borne inférieur et supérieur du segment sont demandées à l'utilisateur. L'ordinateur calcule toutes les valeurs de f(x) lorsque x balaie l'intervalle de a à b avec un pas constant égal à 0,001 et mémorise la plus grande et la plus petite valeur de f(x)

```
#Minimum et maximum d'une fonction f définie sur [a,b]

#L'utilisateur peut utiliser des fonctions ou des variables du module math

#c'est pourquoi il est nécessaire d'importer les fonctions du module math

from math import *

f=input("Saisir l'expression de f en fonction de x\n")

# f est une chaîne de caractères qui sera évaluée en fonction de la valeur de x

a=eval(input("Saisir la valeur de la borne inférieure de l'intervalle\n"))

b=eval(input("Saisir la valeur de la borne supérieure de l'intervalle\n"))

x=a

min=eval(f) #min=f(a)

max=eval(f) #max=f(a)

while x<=b:

   x=x+0.001

   f_x=eval(f)

   if f_x<min:

      min=f_x

   elif f_x>max:

      max=f_x

print("le minimum de f sur [",a,",",b,"] est",min)

print("le maximum de f sur [",a,",",b,"] est",max)
```

Un

Exemple de résultat lors de l'exécution :

```
Saisir l'expression de f en fonction de x
(x-2)*(x-3)
Saisir la valeur de la borne inférieure de l'intervalle
-5*sqrt(2)
Saisir la valeur de la borne supérieure de l'intervalle
3
le minimum de f sur [ -7.07106781187 , 3 ] est -0.249999995402
le maximum de f sur [ -7.07106781187 , 3 ] est 91.3553390593
```

On peut également créer une fonction dont la syntaxe est très proche du langage naturel et qui pourra être utilisée après avoir défini la fonction f dans la fenêtre Python Shell par exemple.

```
#Minimum et maximum d'une fonction f définie sur [a,b]

def minmax(f,a,b):
    x=a
    min=f(x)
    max=f(x)
    while x<=b:
        x=x+0.001
        if f(x)<min:
            min=f(x)
        elif f(x)>max:
            max=f(x)
    return min,max
```

```
>>> def f(x):
...     y=x**2-3
...     return y
...
...

>>> minmax(f,-1,4)
(-3.0, 13.008000999997375)

>>>
```

Le problème des chapeaux

Lors d'une soirée au théâtre, chacune des n personnes invitées dépose son chapeau au vestiaire avant le spectacle. A l'issue de celui-ci, survient une panne d'électricité. Dans ces conditions, chaque personne récupère un chapeau au hasard. Quelle est la probabilité qu'au moins une personne récupère son propre chapeau ?

Il s'agit ici de déterminer une valeur approchée de la probabilité cherchée à l'aide d'un grand nombre de simulations (la valeur de n est demandée à l'utilisateur)

Une modélisation possible :

On numérote les personnes de 1 à n et on considère la liste chapeaux=[1,2,3,…,n] où 1 représente le chapeau de la personne 1, 2 le chapeau de la personne 2 etc…

Les *n* personnes récupèrent au hasard un chapeau est équivalent à ce que les personnes récupèrent tour à tour un chapeau de la liste chapeaux. Les résultats de ces choix seront mémorisés dans une liste choix. Par exemple choix=[3,2,1,5,7,..] correspondra à : la personne 1 récupère le chapeau 3, la personne 2 récupère le chapeau 2 etc…

Il suffira ensuite de voir s'il y a un nombre de la liste choix à la « bonne place ».

```
#Problème des chapeaux
from random import *
n=int(input("combien de chapeaux?\n"))
k=1 #Compteur du nbre de simulations
nbre=0  # nbre stocke le nombre de fois au moins une personne récupère son chapeau
while k<=10000:          #10000 simulations
   chapeaux=list(range(1,n+1))  #chapeaux=[1,2,..,n]
   #Les n personnes choisissent un chapeau tour à tour
   i=1
   choix=[]
   while i<=n:
      choix_chap=choice(chapeaux)              #Choix d'un chapeau dans la liste chapeaux
      chapeaux.remove(choix_chap)              #Supression du chapeau choisi
      choix.append(choix_chap)                 #Ajout de ce chapeau à la liste choix
      i=i+1
   # Test: Si au moins une personne a choisi un bon chapeau alors bon_chap=True
   bon_chap=False
   j=1
   while j<=n and bon_chap==False: #Tant qu'aucune personne n'a choisi le bon chapeau
      if choix[j-1]==j:            # Si la jème personne a récupéré son chapeau
         bon_chap=True
         nbre=nbre+1
      j=j+1
   k=k+1
print("Une valeur approchée de la proba cherchée est",nbre/10000)
```

Remarque : On peut améliorer le programme. Le test qui permet de voir qu'au moins une personne a récupéré son chapeau peut être fait dans la boucle qui simule les tirages. Dans ce cas, dès qu'une personne a récupéré son chapeau, on arrête le tirage ; la liste choix devient inutile.

```
#Problème des chapeaux

from random import *

n=int(input("combien de chapeaux?\n"))          # Nombre total de chapeaux

k=1    # la variable k est le compteur du nombre de simulations

nbre=0  # nbre stocke le nombre de fois où au moins une personne a récupéré son #chapeau sur les #10000 expériences

while k<=10000:          # On effectue 10000 simulations de cette expérience aléatoire

   chapeaux=list(range(1,n+1))  #chapeaux=[1,2,...,n]
# Tant qu'une personne n'a pas récupéré son chapeau, les n personnes choisissent un chapeau tour à #tour

   i=1

   bon_chap=False

   while i<=n and bon_chap==False:

      choix_chap=choice(chapeaux) # choix d'un chapeau dans la liste chapeaux

      chapeaux.remove(choix_chap) # suppression du chapeau choisi de la liste chapeaux

      if choix_chap==i: #Si la ième personne a récupéré son chapeau

         bon_chap=True

         nbre=nbre+1

      i=i+1

   k=k+1

print("Une valeur approchée de la proba cherchée est",nbre/10000)
```

X. Réalisation de lignes, cercles avec le module tkinter

Le module tkinter offre la possibilité de réaliser des fenêtres graphiques performantes pouvant gérer des événements (clic de souris, enfoncement d'un bouton,…).
Il n'est pas question dans ce tutoriel d'étudier l'ensemble des fonctionnalités du module tkinter. On se limitera à l'étude de la création d'une fenêtre, d'un canevas, de lignes et de cercles.

Importation du module tkinter, fenêtre et canevas

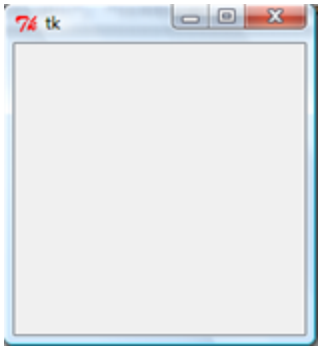
- **Importation du module tkinter**
  Pour importer les objets et les fonctions du module tkinter, il faut écrire en début de programme la ligne :

  from tkinter import *

- **Création d'une fenêtre graphique et « activation de la fenêtre »**
  fenetre=Tk() crée une fenêtre qui a pour nom « fenetre »
  fenetre.mainloop() active la fenêtre. Cette ligne doit être écrite après toutes celles qui font référence à cette fenêtre.

```
#Création et activation d'une fenêtre


from tkinter import *

fenetre=Tk()

fenetre.mainloop()
```

L'exécution du programme suivant crée une fenêtre vide comme celle-ci:



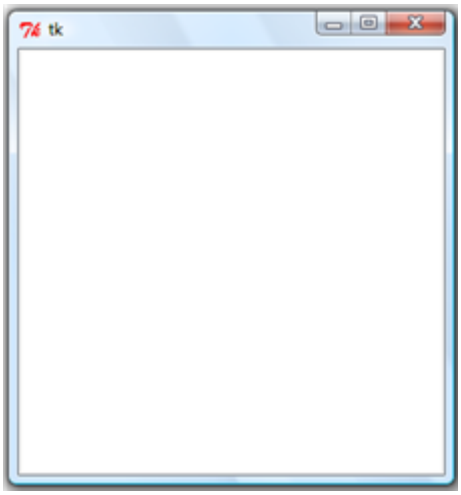
- **Création du canevas sur lequel les figures seront réalisées** :
  La fenêtre précédente est vide.
  La réalisation d'un canevas sur lequel des figures vont être réalisées se fait avec la méthode Canvas :
  can=Canvas(fenetre,bg="white",height=300, width=300)

  Le canevas a pour nom can, la couleur de fond du canevas est blanc, la hauteur du canevas est de 300 pixels et sa largeur de 300 pixels.
  Pour encapsuler le canevas dans la fenêtre, il faut utiliser la méthode pack() :

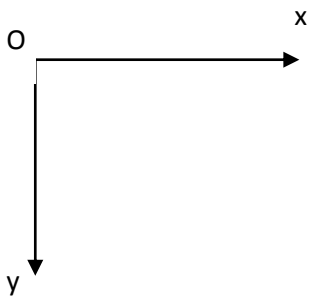
Exemple :

```
# Création d'un canevas

from tkinter import *

fenetre=Tk()

can=Canvas(fenetre,bg="white",height=300, width=300)

can.pack()

fenetre.mainloop()
```

Ce programme permet l'ouverture d'une fenêtre dans laquelle est encapsulé un canevas du nom de « can » blanc de largeur 300 pixels et de hauteur 300 pixels :



Remarque : L'origine du repère du canevas est le bord supérieur gauche, l'axe des abscisses est horizontal et orienté vers la droite ; l'axe des ordonnées est verticale et orienté vers le bas.



Créations de lignes, cercles sur le canevas

Une fenêtre tkinter a été créée ainsi qu'un canevas encapsulé dans cette fenêtre portant le nom « can », blanc, de taille 300×300.

- **Création de lignes :**
  can.create_line(0,0,300,300,width=2,fill="red")

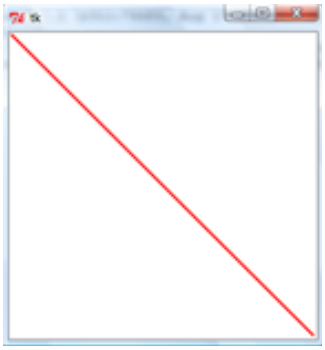
permet de tracer dans le canevas « can » un segment d'extrémités les points de coordonnées respectives (0,0) et (300,300). L'épaisseur du trait est de 2 pixels et la couleur du trait est rouge.

Remarque :

Si l'épaisseur et la couleur sont omises, le trait sera noir et d'épaisseur 1 pixel.

Exemple :

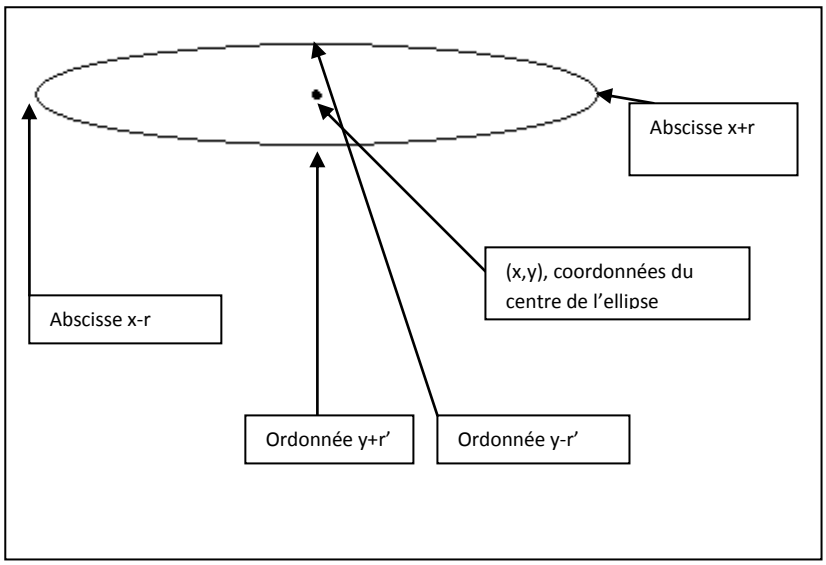
```
#Création d'une diagonale


from tkinter import *

fenetre=Tk()

can=Canvas(fenetre,bg="white",height=300, width=300)

can.pack()

can.create_line(0,0,300,300,width=2,fill="red")

fenetre.mainloop()
```



- **Création de cercles :**

can.create_oval(x-r,y-r',x+r,y+r',width=1,fill="couleur1",outline="couleur2")

crée une ellipse sur le canevas « can » caractérisée de la façon suivante :



width=1 fixe l'épaisseur de la ligne à 1 pixel, couleur1 est la couleur de remplissage, couleur2 est la couleur de la ligne.

Remarque 1 :

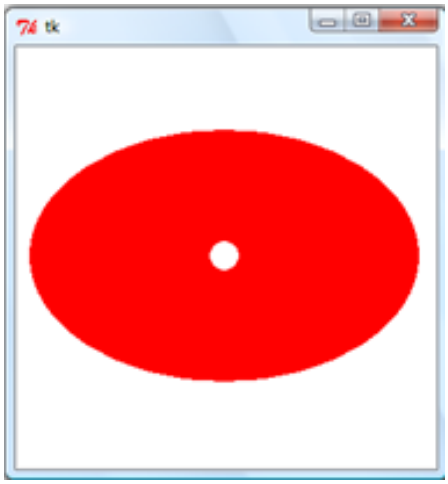
Si l'épaisseur, la couleur de la ligne, la couleur de remplissage sont omises, la ligne sera noire, d'épaisseur 1 pixel et il n'y aura pas de remplissage.

Remarque 2 :

Il suffit de prendre r=r' pour tracer un cercle.

```
# Création d'une ellipse et d'un disque


from tkinter import *

fenetre=Tk()

can=Canvas(fenetre,bg="white",height=300, width=300)

can.pack()

can.create_oval(10,60,290,240,fill="red",outline="red")

can.create_oval(150-10,150-10,150+10,150+10,fill="white",outline="white")

fenetre.mainloop()
```

Résultat de l'exécution de ce programme :



Avec les élèves

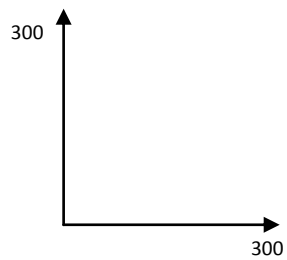
La création de figures sous python nécessite des connaissances techniques dont l'acquisition n'est pas une priorité pour les élèves.
Il est possible d'éviter pour l'élève tous les problèmes techniques inhérents à la création d'objets graphiques en créant au préalable la fenêtre, le canevas et des fonctions graphiques plus simples à utiliser.

En voici un exemple :

```
# Support élève pour la réalisation d'objets graphiques


from tkinter import *

haut_fen=300  # Hauteur de la fenêtre

larg_fen=300  # Largeur de la fenêtre

fenetre=Tk()

can=Canvas(fenetre,bg="white",height=haut_fen,width=larg_fen)

can.pack()


def segment(x1,y1,x2,y2):

    can.create_line(x1,haut_fen-y1,x2,haut_fen-y2)


def cercle(x1,y1,R):

    can.create_oval(x1-R,haut_fen-y1-R,x1+R,haut_fen-y1+R)


#Espace réservé aux lignes de programmation de l'élève

segment(0,0,300,300)

cercle(50,50,20)

# Fin de l'espace réservé aux lignes de programmation de l'élève


fenetre.mainloop()
```

Commentaires :

Dans cet exemple de support de programmation, le repère avec lequel l'élève travaille est un repère orthogonal classique.

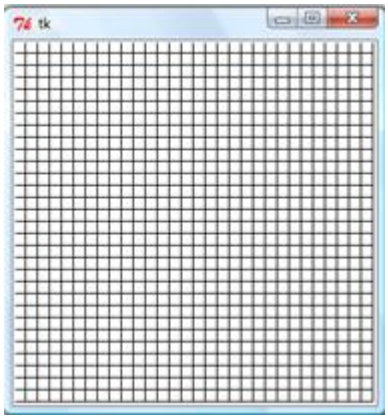


segment(x1,y1,x2,y2) crée le segment d'extrémités les points de coordonnées (x1,y1) et (x2,y2)

cercle(x1,x2,R) crée le cercle de centre le point de coordonnées (x1,x2) et de rayon R.

Résultat de l'exécution du programme ci-dessus :

Exercice 3 **:** En vous servant du support élève ci-dessus réaliser un programme dont le résultat de l'exécution est :



(distance de 10 pixels entre deux lignes consécutives)

XI. Correction des trois exercices du tutoriel

Correction de l'exercice 1 :

```
S=10000
Svoulue=float(input("Saisissez le montant du capital souhaité\n"))
n=0
while S<Svoulue:
    S=1.03*S+1000
    n=n+1
print("Le nombre minimum d'années de placement est",n)
```

Correction de l'exercice 2 :

```
Tps=0
U0=1
Umax=0
while U0<=10**4:
    N=0
    U=U0
    while U!=1:
        if U%2==0:
            U=U/2
        else:
            U=3*U+1
        N=N+1
    if N>Tps:
        Tps=N
        Umax=U0
    U0=U0+1
print("la valeur de U0 cherchée est",Umax,"et le temps de vol est égal à",Tps)
```

Explications :

```
Tps=0      #Tps contiendra le temps de vol le plus grand
U0=1       #U0 contient la valeur du premier terme Uo
Umax=0     #Umax contiendra la valeur de Uo qui donne le temps de vol le plus grand
while U0<=10**4:
    N=0
    U=U0
    while U!=1:
        if U%2==0:
            U=U/2
        else:
            U=3*U+1
        N=N+1
    if N>Tps:
        Tps=N
        Umax=U0
    U0=U0+1
print("la valeur de U0 cherchée est",Umax,"et le temps de vol est égal à",Tps)
```

Calcul de temps de vol pour toutes les valeurs de Uo compris entre 1 et 10000

A ce moment la valeur de N est égale au temps de vol pour la valeur de Uo en cours.

Si le temps de vol pour la valeur de Uo en cours est supérieur au plus grand temps de vol précédent alors la variable Tps mémorise ce nouveau "record" et la valeur de Uo correspondante.

Correction de l’exercice 3 :

```
# Support pour la réalisation d'objets graphiques


from tkinter import *
haut_fen=300  # Hauteur de la fenêtre
larg_fen=300  # Largeur de la fenêtre
fenetre=Tk()
can=Canvas(fenetre,bg="white",height=haut_fen,width=larg_fen)
can.pack()


def segment(x1,y1,x2,y2):
    can.create_line(x1,haut_fen-y1,x2,haut_fen-y2)


def cercle(x1,y1,R):
    can.create_oval(x1-R,haut_fen-y1-R,x1+R,haut_fen-y1+R)


#Espace réservé aux lignes de programmation de l'élève
#Correction de l’exercice
a=0
while a<=300:
    segment(a,0,a,300)
    segment(0,a,300,a)
    a=a+10
# Fin de l'espace réservé aux lignes de programmation de l'élève


fenetre.mainloop()
```