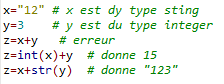
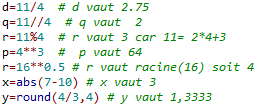
## Memento PYTHON

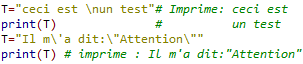
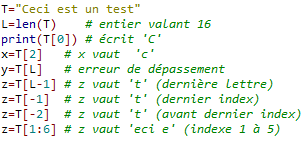
Types de variables - fonctions de conversions

* Type integer  Nombre entier   
  int( )  convertit si possible un décimal ou texte en entier
* Type float  Nombre décimal   
  float( )  convertit si possible un entier ou texte en décimal
* Type string  Chaine de caractères ( texte)  
  suite de signes définie en la délimitant par des guillemets   
  str( )  convertit un nombre en chaîne
* Type boolean  Logique   
  ne prend que deux valeurs : True et False
* Affectation =  
   x =…  lire « x prend la valeur….» ☞ astuces

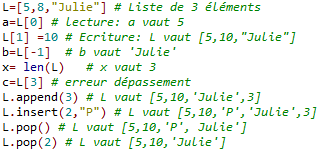
Entrées, sorties console, opérations numériques

* Entrée  input ( " message" ) : lit un texte saisi au clavier .   
  ☞ Renvoie donc toujours une chaîne de caractères.   
  ☞ conversion possible en nombre par int( ) ou float( )
*  Sortie en console  print( , , …) : affiche en console les valeurs de tout type en les séparant par une tabulation.
* Opérations sur les nombres  
  /  division décimale  
  //  quotient de la division entière   
  %  reste de la division entière  
  \*\*  puissance (remarque : )  
  abs( )  valeur absolue  
  round( x,d )  arrondi le nombre x à d décimales

Chaînes de caractères

* Concaténation +  attache les textes pour n’en former qu’un
* Caractères d'échappement   
  le signe \  permet de transformer le caractère qui suit   
  \n   saut de ligne (new). \t   tabulation  
  \" ou \’   guillemet qui ne ferme pas la chaine
* longueur d’une chaine :   
  len( )  renvoie le nombre de caractères d’une chaine, espaces compris.
* Indexation Chaque caractère de la chaîne est indexé (numéroté) en commençant par 0   
  Chaine[ i ]  renvoie le caractère de rang i  
  ☞ astuces :  
   MaChaine[ -1 ]  dernier caractère  
   MaChaine[ -2 ]  avant dernier caractère,etc…  
   MaChaine[i : j]  caractères indéxés de i à j-1   
  ☞ Attention : on en peut pas modifier un caractère d’une chaîne par son index, seulement le lire !
*  Code ASCII  
  chr( x)  renvoie le caractère de code ASCII x  
  ord( char )  renvoie le code ASCII du caractère char   
  ☞ chr(10) ou chr(13)  saut de ligne. chr(9) tabulation

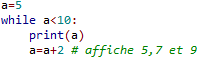
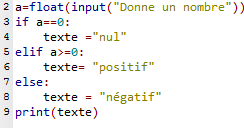
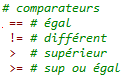
Listes et tuples

*  Liste  suite indexée et modifiable d’éléments de tout type Attention : l’indexation commence à 0  
  NomListe = [ élément1 , élément2 , élément3,…]   
  NomListe[ i ]  élément d’index i ( lecture ou écriture)  
  NomListe[ 0 ]  premier élément  
  NomListe[ -1]  dernier élément
* Principales fonctions :  
  Longueur : len( )  renvoie le nombre d’éléments   
  Ajout : Nomliste.append(x)  ajoute x en fin de liste  
  Insertion : Nomliste.insert(i,x)  insert x à l’index i  
  Suppression: Nomliste.pop( ) supprime le dernier élémt  
   Nomliste.pop(i)  supprime élément d’indexe i
* Tuples  un tuple est une liste non modifiable  
   NomTuple = ( élément1 , élément2 , élément3,..)   
  NomTuple[ i]  élément d’indexe i en lecture seule

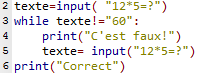
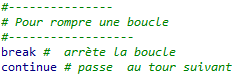


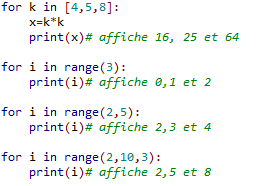
* Liste de listes  une liste peut contenir des listes !!  
   NomListe[ i][ j ] désigne l’élément d’index j de la liste d’index i

Tests

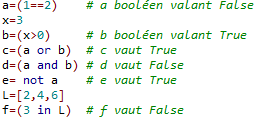
* if test 1 : # un test est une valeur booléenne (logique)  
   | bloc si test1 vérifié  
  elif  test 2 : # (facultatif). Sinon si :  
   | bloc si test 1 non vérifié mais test2 vérifié   
   …..  
  else : # (facultatif). Sinon  
   | bloc si aucun des tests précédent n’est vérifié  
   suite du programme

Boucle « Tant que »

* While test : # Tant que …  
   | Bloc répété tant que test vérifié  
  suite du programme

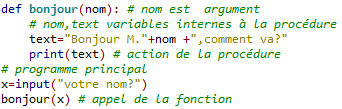
Boucle « Pour… »

* For variable in liste : # Pour chaque … dans… :   
   | Bloc répété pour chaque valeur de la variable parcourant la liste  
  suite du programme
* Génération de listes d’entiers  
  range(a)  tous les entiers de [0 ; a[   
  range(a,b)  tous les entiers de [a ; b[  
  range(a,b,p)  tous les entiers de [a ; b[ de p en p

Logique : variables booléennes

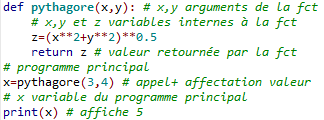
* Une variable booléenne ne prend que 2 valeurs True, False
* Opérateurs booléens  
  a or b  vaut True si et seulement si l’un au moins vaut True  
  a and b  vaut True si et seulement si les deux valent True  
  not a  contraire de a : True si a False, False si a True  
  a in Liste  vaut True si et seulement si a élément de Liste

Procédures et fonctions ☞ Essentielles pour structurer un programme

Ce sont des sous-programmes autonomes avec leurs propres variables. Ils ne sont exécutés que lorsqu’ils sont appelés par le programme principal ou par une autre fonction

* Procédure (ou sous-programme)   
  def Nom(arg1, arg2,… ) :   
   | bloc instructions  
  # programme principal  
  Nom (variable1,variable2…)  # appel de la procédure

*Les variables de ‘passage’ sont appelées* ***arguments***

*  Fonction = procédure avec retour de valeur(s)  
  def Nom(arg1, arg2,…) :   
   | bloc instructions  
   return x # x valeur ou liste de valeurs   
  # programme principal  
  a = Nom (valeur1,..,.. )  # appel + affectation de la valeur retournée

***Arguments*** *de la fonction*

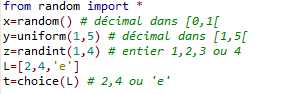
  
Importation de librairies – Librairies utiles

**Importer une librairie : plusieurs méthodes**

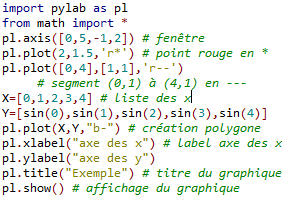
* import MaLibrairie # Importation d’un ensemble de fcts  
  MaLibraire.fonction1(var1,… ) # appel d’une fonction  
  import MaLibrairie as Lib # nom local de la librairie  
  Lib.fonction1(var1,… ) # appel d’une fonction
*  From MaLibrairie import fct1, fct2, .# liste fcts utiles,  
   From MaLibrairie import \* # toutes les fonctions  
  fonction1(var1,… ) # appel d’une fonction

**Mathématiques**

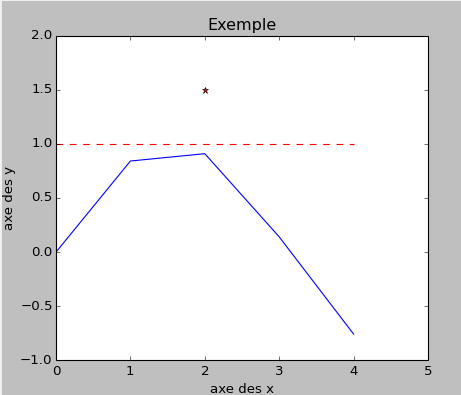
* Libraire math  fonctions mathématiques  
  sqrt( )  racine carrée sin( )  sinus(radian), etc…

**Nombres aléatoires**

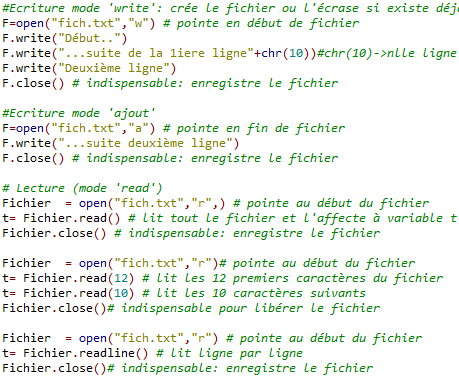
* Librairie random  génération de nombres aléatoires   
  randint(a,b)  entier dans [a, b ]  
  random( )  décimal (float) dans [ 0, 1 [  
  uniform(a,b)  décimal (float) dans [ a, b [  
  choice(maList)  élément de la liste maList



**Graphiques mathématiques**

* Librairie pylab combine deux librairies : pyplot et numpy pour les graphiques et calculs mathématiques   
  point plot(x,y,’ro’)  point de coord (x,y) rouge et rond  
  segment plot( [x1,x2] , [y1,y2], ’b -’ )   
    segment de (x1,y1) à (x2,y2) en bleu et trait plein   
  polygone plot( liste des x, liste des y, ‘g- - ‘)  
    polygone en vert et trait pointillé  
  axes axis([xmin, xmax, ymin, ymax])  
  affichage show( )  affiche le graphique  
  grille grid()  affiche la grille  
  label xlabel(’ texte’ )  Label de l’axe des x  
   ylabel( ‘texte ‘ )  Label de l’axe des y  
  titre title( ‘texte du titre’ )   
  sauvegarde savefig( ‘nomfichier’ )  sauve au format .png  
    
  + info  http://matplotlib.org/users/pyplot\_tutorial.html

Fichiers textes

* Ouverture nom = open("fichier.txt","w" )  en mode lecture ou écriture, pointe en début de fichier

(*chemin +)* ***nom fichier*** *sur disque* fichier

***Variable*** *qui identifie le fichier.*

***Mode***  *• "r" lecture   
• "w" écriture  
• " a " ajout*

* Ecriture nom.write( "texte…" )   
   Si le fichier n’existe pas, il est créé.   
   Si le fichier existe il est écrasé
* Lecture  ouvrir le fichier en mode lecture  
  x= nom.read()  lecture du fichier entier x= nom.read( n )  lecture des n caractères suivants .  
  x= nom.readline()  lecture ligne par ligne .

Interfaces graphiques   
avec la librairie tkinter

