

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 → Chaîne 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

```
x="12" # x est de type string
y=3    # y est de type integer
z=x+y  # erreur
z=int(x)+y # donne 15
z=x+str(y) # donne "123"
```

```
a=(1+1>2) # a booléen qui vaut False
b=(2*3==6) # b booléen qui vaut True
```

```
a,b=12,15 # équivaut à a=12 et b=15
x+=2      # équivaut à x=x+2
x-=1      # équivaut à x=x-1
```

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 : $a^{0.5} = \sqrt{a}$)
abs() → valeur absolue
round(x,d) → arrondi le nombre x à d décimales

```
a=input("Texte?") # a type string
b=float(input("Nombre?")) # b type float
```

```
a=45*2
print("Coût=",a,"€") # affiche "Coût= 90 €"
```

```
d=11/4 # d vaut 2.75
q=11//4 # q vaut 2
r=11%4 # r vaut 3 car 11= 2*4+3
p=4**3 # p vaut 64
r=16**0.5 # r vaut racine(16) soit 4
x=abs(7-10) # x vaut 3
y=round(4/3,4) # y vaut 1,3333
```

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 chaîne
- **longueur d'une chaîne :**
len() → renvoie le nombre de caractères d'une chaîne, espaces compris.
- **Indexation** Chaque caractère de la chaîne est indexé (numéroté) **en commençant par 0**
Chaîne[i] → renvoie le caractère de rang i
🔗 astuces :
 MaChaîne[-1] → dernier caractère
 MaChaîne[-2] → avant dernier caractère, etc...
 MaChaîne[i:j] → caractères indexés 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

```
T1="Ceci est" # variable type string
T2=" un test." # variable type string
T=T1+T2      # T vaut "Ceci est un test"
```

```
T="ceci est \nun test" # Imprime: ceci est
print(T)               # un test
T="Il m'a dit:\nAttention\\""
print(T) # imprime : Il m'a dit:"Attention"
```

```
T="Ceci est un test"
L=len(T) # entier valant 16
print(T[0]) # écrit 'C'
x=T[2] # x vaut 'c'
y=T[L] # erreur de dépassement
z=T[L-1] # z vaut 't' (dernière lettre)
z=T[-1] # z vaut 't' (dernier index)
z=T[-2] # z vaut 't' (avant dernier index)
z=T[1:6] # z vaut 'eci e' (indexe 1 à 5)
```

```
T="Python"
T[1]="y" # erreur: affectation non autorisée
```

```
x=chr(65) # x vaut 'A'
y=ord('B') # y vaut 66
```

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 élément
`NomListe.pop(i)` → supprime élément d'indexe i

```
L=[5,8,"Julie"] # Liste de 3 éléments  
a=L[0] # lecture: a vaut 5  
L[1]=10 # Ecriture: L vaut [5,10,"Julie"]  
b=L[-1] # b vaut 'Julie'  
x=len(L) # x vaut 3  
c=L[3] # erreur dépassement  
L.append(3) # L vaut [5,10,'Julie',3]  
L.insert(2,"P") # L vaut [5,10,'P','Julie',3]  
L.pop() # L vaut [5,10,'P',Julie]  
L.pop(2) # L vaut [5,10,'Julie']
```

- **Tuples** → un tuple est une **liste non modifiable**

```
NomTuple = (élément1, élément2, élément3, ...)  
NomTuple[i] → élément d'indexe i en lecture seule
```

```
T=(4,8,10)  
x=T[0] # x vaut 4  
T[0]=5 # erreur, T non modifiable
```

- **Liste de listes** → une liste peut contenir des listes !!

`NomListe[i][j]` désigne l'élément d'index j de la liste d'index i

```
Tab=[[4,2,9],[0,7,8]]  
x=Tab[0] # x vaut [4,2,1] : index 0 de Tab  
y=Tab[0][2] # y vaut 9 : index 2 de Tab[0]
```

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

```
a=float(input("Donne un nombre"))  
if a==0:  
    texte="nul"  
elif a>=0:  
    texte="positif"  
else:  
    texte="négatif"  
print(texte)
```

Boucle « Tant que »

- **While** test : # Tant que ...

| Bloc répété tant

que test vérifié

suite du programme

```
# comparateurs  
== # égal  
!= # différent  
> # supérieur  
>= # sup ou égal
```

```
a=5  
while a<10:  
    print(a)  
    a=a+2 # affiche 5,7 et 9
```

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

```
texte=input("12*5=?")  
while texte!="60":  
    print("C'est faux!")  
    texte= input("12*5=?")  
print("Correct")
```

```
for k in [4,5,8]:  
    x=k*k  
    print(x) # affiche 16, 25 et 64  
  
for i in range(3):  
    print(i) # affiche 0,1 et 2  
  
for i in range(2,5):  
    print(i) # affiche 2,3 et 4  
  
for i in range(2,10,3):  
    print(i) # affiche 2,5 et 8
```

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

```
a=(1==2) # a booléen valant False  
x=3  
b=(x>0) # b booléen valant True  
c=(a or b) # c vaut True  
d=(a and b) # d vaut False  
e= not a # e vaut True  
L=[2,4,6]  
f=(3 in L) # f vaut False
```

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

```
def bonjour(nom): # nom est argument
    # nom, text variables internes à la procédure
    text="Bonjour M."+nom+" ,comment va?"
    print(text) # action de la procédure
# programme principal
x=input("votre nom?")
bonjour(x) # appel de la fonction
```

```
def pythagore(x,y): # x,y arguments de la fct
    # x,y et z variables internes à la fct
    z=(x**2+y**2)**0.5
    return z # valeur retournée par la fct
# programme principal
x=pythagore(3,4) # appel+ affectation valeur
# x variable du programme principal
print(x) # affiche 5
```

Importation de bibliothèques – Bibliothèques utiles

Importer une bibliothèque : plusieurs méthodes

- import** MaLibrairie # Importation d'un ensemble de fcts
MaLibrairie.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

```
import turtle # importe la librairie turtle
turtle.forward(50) # forward() fct de turtle
turtle.exitonclick() # fct de turtle
```

```
import math as m # m désigne la librairie math
x=m.sqrt(2) # racine carrée
y=m.sin(1.25) # sinus (angle en radian)
```

```
from math import sqrt, sin # seulement 2 fcts
x=sqrt(2) # pas de math.sqrt
y=sin(1.25)
```

```
from math import * # toutes les fcts de math
x=sqrt(2) # pas de math.sqrt
y=sin(1.25)
```

Mathématiques

- Librairie **math** → **fonctions mathématiques**
sqrt() → racine carrée **sin()** → sinus(radian), etc...

Nombres aléatoires

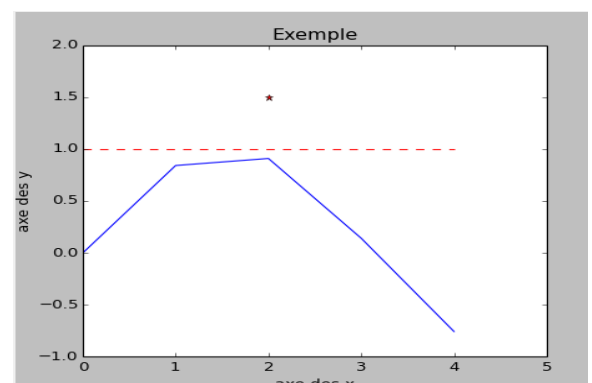
- Librairie **random** → **génération de nombres aléatoires**
randint(a,b) → entier dans [a, b]
random() → décimal (float) dans [0, 1[
uniform(a,b) → décimal (float) dans [a, b]
choice(maList) → élément de la liste maList

```
from random import *
x=random() # décimal dans [0,1[
y=uniform(1,5) # décimal dans [1,5[
z=randint(1,4) # entier 1,2,3 ou 4
L=[2,4,'e']
t=choice(L) # 2,4 ou 'e'
```

Graphiques mathématiques

- Librairie **pylab** combine deux bibliothèques : 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

```
import pylab as pl
from math import *
pl.axis([0,5,-1,2]) # fenêtre
pl.plot(2,1.5,'r*') # point rouge en *
pl.plot([0,4],[1,1],'r--')
# segment (0,1) à (4,1) en ---
X=[0,1,2,3,4] # liste des x
Y=[sin(0),sin(1),sin(2),sin(3),sin(4)]
pl.plot(X,Y,"b-") # création polygone
pl.xlabel("axe des x") # label axe des x
pl.ylabel("axe des y")
pl.title("Exemple") # titre du graphique
pl.show() # affichage du graphique
```



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

Variable
qui identifie
le fichier.

(chemin +)
nom
fichier sur

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.

```
#Ecriture mode 'write': crée le fichier ou l'écrit si existe déjà
F=open("fich.txt","w") # pointe en début de fichier
F.write("Début..")
F.write("...suite de la 1ère ligne"+chr(10))#chr(10)->nlle ligne
F.write("Deuxième ligne")
F.close() # indispensable: enregistre le fichier
```

```
#Ecriture mode 'ajout'
F=open("fich.txt","a") # pointe en fin de fichier
F.write("...suite deuxième ligne")
F.close() # indispensable: enregistre le fichier
```

```
# Lecture (mode 'read')
Fichier = open("fich.txt","r",) # pointe au début du fichier
t= Fichier.read() # lit tout le fichier et l'affecte à variable t
Fichier.close() # indispensable: enregistre le fichier
```

```
Fichier = open("fich.txt","r")# pointe au début du fichier
t= Fichier.read(12) # lit les 12 premiers caractères du fichier
t= Fichier.read(10) # lit les 10 caractères suivants
Fichier.close()# indispensable pour libérer le fichier
```

```
Fichier = open("fich.txt","r") # pointe au début du fichier
t= Fichier.readline() # lit ligne par ligne
Fichier.close()# indispensable: enregistre le fichier
```

Interfaces graphiques avec la librairie tkinter

```
1 # Importation de la librairie graphique tkinter
2 from tkinter import *
3 #-----
4 # Création d'une fenêtre
5 #-----
6 Mafenetre= Tk() # Création d'une fenêtre
7 Mafenetre.title("Tutoriel") # Titre de la fenêtre
8 Mafenetre.geometry("180x250") # Dimensions de la fenêtre
9 #-----
10 # Les "widgets" sont des composants que l'on positionne dans la fenêtre
11 #-----
12 # Widget "Label" (zone d'affichage de texte)
13 Affichage = Label(Mafenetre, text='Entrez un entier')# création d'un Label
14 Affichage.place(x=10,y=0)# positionnement
15 # Widget "Entry" (zone de saisie de texte)
16 Saisie = Entry(Mafenetre) # création d'une zone de saisie
17 Saisie.place(x=10, y= 20) # positionnement
18 # Widget "Button" (bouton de commande)
19 def calcul():# Procédure associée au click du bouton
20     v = Saisie.get() # récupère la valeur du widget entry 'Saisie'
21     double =2*float(v) # (-> penser à la conversion en nombre)
22     MonTexte = "Le double de " + Saisie.get() + " = " + str(double)
23     Affichage.config(text=MonTexte) # Le texte du label 'Affichage' est changé
24     # création et association du click à la procédure 'calcul'
25 bouton = Button( Mafenetre, text='Calculer', command=calcul) # création bouton
26 bouton.place(x=125,y=20) # positionnement |
27 # Image dans un Label
28 from PIL import Image, ImageTk # importation des librairies images
29 monimage = Image.open("tutorial.png") # Chargement d'une image à partir de PIL
30 photo = ImageTk.PhotoImage(monimage) # Création d'une image compatible tkinter
31 LabelImage = Label( Mafenetre,image=photo) # label contenant une image
32 LabelImage.place(x=10,y=45)# positionnement
33 #-----
34 # Un Canvas est une zone de la fenêtre pour images ou dessins
35 #-----
36 MonCanevas = Canvas(Mafenetre,bg="black",width = 150, height =100)
37 Rectangle = MonCanevas.create_rectangle(0,0,100,70,fill='red')#points opposés
38 Cercle = MonCanevas.create_oval(50,50,100,100,fill='yellow') #x-R, y-R, x+R, y+R
39 MonCanevas.place(x=10, y=140)
40 #-----
41 # IMPORTANT: La fonction mainloop i provoque le démarrage du récepteur
42 # d'événements (clics,clavier,...) associé à la fenêtre
43 #-----
44 Mafenetre.mainloop()
```

