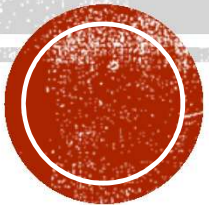
PYTHON3

Réalisé par M. Souissi Abdelmoghit



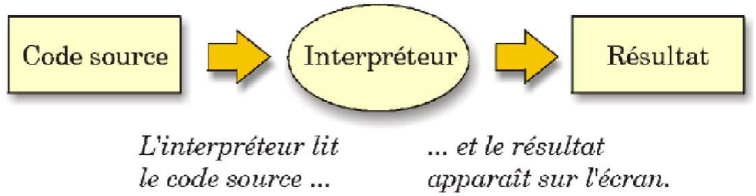
LE LANGAGE PYTHON

1. développé en 1989 par Guido van Rossum
2. Open-source
3. Portable
4. Orienté objet
5. Dynamique
6. Extensible
7. Support pour l'intégration d'autres langages

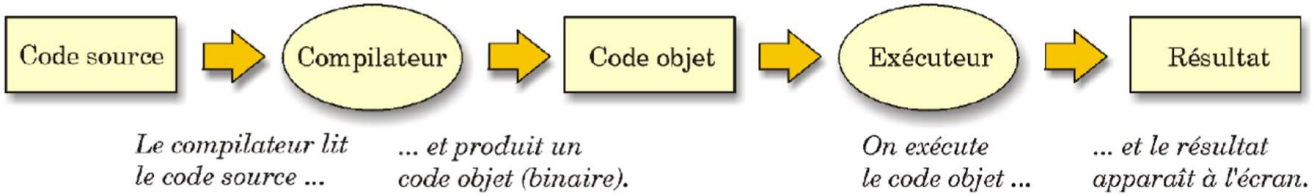
COMMENT FAIRE FONCTIONNER MON CODE SOURCE?

Il existe 2 techniques principales pour effectuer la traduction en langage machine de d'un code source :

- ❑ Interprétation



- ❑ Compilation



ET PYTHON

- **CPython**

Implémentation de base basé sur le langage C ANSI

- **Jython**

Implémentation permettant de mixer Python et java dans la même JVM

- **IronPython**

Implémentation permettant d'utiliser Python pour Microsoft .NET

- **PyPy**

Implémentation de Python en Python

QUE PEUT-ON FAIRE AVEC PYTHON?

- **web**
  Django, TurboGears, Zope, Plone, ...
- **bases de données**
  MySQL, PostgrSQL, Oracle, ...
- **réseaux**
  TwistedMatrix, PyRO, ...
- **Gui**
  Gtk, Qt, Tcl/Tk, WxWidgets
- **représentation graphique**
  gnuplot, matplotlib, VTK, ...
- **calcul scientifique**
  numpy, scipy, sage, ...

POURQUOI UTILISER PYTHON POUR LE CALCUL SCIENTIFIQUE?

- ✓ peut être appris en quelques jours
- ✓ permet de faire des tests rapides
- ✓ alternative à Matlab, Octave, Scilab, …
- ✓ parallélisassions
- ✓ tourne sur la plupart des plateformes
- ✓ très bon support pour le calcul scientifique

INSTALLATION

SYNTAXE PYTHON

- Commentaires & Ecrire sur l'écran

```
#This is a comment.
print("Hello, World!")
```

print("%d X %d = %d"%(num,i,num*i)); """

- Exécution d'un fichier

```
python myfile.py
```

- Instruction conditionnelle

```
if 5 > 2:
  print("Five is greater than two!")
```

- Déclaration des variables

```
x = 5
y = "Hello, World!"
```

- Commentaires multi ligne

```
"""This is a comment
written in
more than just one line
"""
print("Hello, World!")
```

VARIABLES

```
#Déclaration très simple
x = 5
y = "python"
print(x)
print(y)
```

```
#Legal variable names:
myvar = "John"
my_var = "John"
_my_var = "John"
myVar = "John"
MYVAR = "John"
myvar2 = "John"
```

```
# is the same as
x = “python"
x = ‘python'
```

```
#Illegal variable names:
2myvar = "John"
my-var = "John"
my var = "John"
```

DATA TYPE

En programmation, le type de données est un concept important. Les variables peuvent stocker des données de différents types et différents types peuvent faire différentes choses.

- Text Type: str
- Numeric Types: int, float, complex
- Sequence Types: list, tuple, range
- Mapping Type: dict
- Set Types: set, frozenset
- Boolean Type: bool
- Binary Types: bytes, bytearray, memoryview

DATA TYPE

```
x = 5
print(type(x))
```

Exemple	Data Type
`x = "Hello World"`	str
`x = 20`	int
`x = 20.5`	float
`x = 1j`	complex
`x = ["apple", "banana", "cherry"]`	list
`x = ("apple", "banana", "cherry")`	tuple
`x = range(6)`	range
`x = {"name" : "John", "age" : 36}`	dict
`x = {"apple", "banana", "cherry"}`	set
`x = frozenset({"apple", "banana", "cherry"})`	frozenset
`x = True`	bool
`x = b"Hello"`	bytes
`x = bytearray(5)`	bytearray
`x = memoryview(bytes(5))`	memoryview

PYTHON CASTING

```
x = int(1)   # x will be 1
y = int(2.8) # y will be 2
z = int("3") # z will be 3
```

```
x = float(1)     # x will be 1.0
y = float(2.8)   # y will be 2.8
z = float("3")   # z will be 3.0
w = float("4.2") # w will be 4.2
```

```
x = str("s1") # x will be 's1'
y = str(2)    # y will be '2'
z = str(3.0)  # z will be '3.0'
```

FONCTION TEXT

Fonction	Signification
a = "Hello" print(a)	Return la valeur de a
a = "Hello, World!" print(len(a))	Return la taille de a
print(a[1]) print(b[2:5]) print(b[-5:-2])	Return le 1 er caractère Return entre le 2ème et 5ème caractère Return entre le 5ème et 2ème caractère en inverse
print(a.lower()) print(a.upper())	Conversion en minuscule Conversion en majuscule
a = "Hello, World!" print(a.replace("H", "J"))	Remplacer H par J
a = "Hello, World!" print(a.split(",")) #	Return ['Hello', ' World!']

FONCTIONS TEXTE

Fonction	Signification

txt = "The rain in Spain stays mainly in the plain"
x = "ain" in txt # or x = "ain" not in txt
print(x)
	Return True or False

a = "Hello" b = "World" c = a + " " + b print(c)	Concaténer 2 textes
age = 36 txt = "My name is John, and I am {}" print(txt.format(age)) quantity = 3 itemno = 567 price = 49.95 myorder = "I want to pay {2} dollars for {0} pieces of item {1}." print(myorder.format(quantity, itemno, price))	Concaténer texte et chiffre

CARACTÈRE D'ÉCHAPPEMENT

```
txt = "We are the so-called \"Vikings\" from the north."
```

Code	Resultat
\'	Single Quote
\\	Backslash
\n	New Line
\r	Carriage Return
\t	Tab
\b	Backspace
\f	Form Feed
\ooo	Octal value
\xhh	Hex value

OPÉRATIONS ARITHMITIQUES

Operations	Example	Same As	Return
=	x = 5	x = 5	affectation
+=	x += 3	x = x + 3	Addition
-=	x -= 3	x = x - 3	soustraction
*=	x *= 3	x = x * 3	Multiplication
/=	x /= 3	x = x / 3	Division
%=	x %= 3	x = x % 3	Modulo: reste de la division
//=	x //= 3	x = x // 3	Division sans virgule
**=	x **= 3	x = x ** 3	Multiplication 2 x

OPÉRATIONS DE COMPARAISON

Operator	Name	Example
==	Equal	x == y
!=	Not equal	x != y
>	Greater than	x > y
<	Less than	x < y
>=	Greater than or equal to	x >= y
<=	Less than or equal to	x <= y

OPÉRATIONS LOGIQUE

Operator	Description	Example
and	Retourne True si les 2 conditions sont vrais	x < 5 and x < 10
or	Retourne True si l'un des conditions est vrai	x < 5 or x < 4
not	Retourne True si les 2 conditions sont faux	not(x < 5 and x < 10)
is	Retourne True si les 2 conditions ont les memes objet	x is y
is not	Retourne True si les 2 conditions n'ont pas les memes objet	x is not y
in	Retourne True si le premier objet se trouve dans le 2ème objet	x in y
not in	Retourne True si le premier objet ne se trouve pas dans le 2ème objet	x not in y

LISTS

Collections Python (tableaux)
Il existe quatre types de données de collecte dans le langage de programmation Python:

- La liste est une collection qui est commandée et modifiable. Autorise les membres en double.
- Tuple est une collection commandée et immuable. Autorise les membres en double.
- Set est une collection non ordonnée et non indexée. Aucun membre en double.
- Dictionary est une collection non ordonnée, modifiable et indexée. Aucun membre en double.

LISTS

```
thislist = ["apple", "banana", "cherry", "orange", "kiwi", "melon", "mango"]
print(thislist)

print(thislist[1])
print(thislist[-1])
print(thislist[2:5])
print(thislist[:4])
print(thislist[2:])
print(thislist[-4:-1])
thislist[1] = "blackcurrant"
for x in thislist:
  print(x)
```

CONDITIONS PYTHON ET INSTRUCTIONS IF

Python prend en charge les conditions logiques habituelles des mathématiques:

Egalité: a == b
Inégalité: a != b
Inférieur à: a < b
Inférieur ou égale à : a <= b
Supérieur à: a > b
Supérieur ou égale à : a >= b

```
a = 33
b = 200
if b > a:
  print("b is greater than a")
```

```
a = 200
b = 33
if b > a:
  print("b is greater than a")
elif a == b:
  print("a and b are equal")
else:
  print("a is greater than b")
```

```
a = 200
b = 33
c = 500
if a > b or a > c:
  print("At least one of the conditions
is True")
```

PYTHON WHILE LOOPS

Instruction répétitive sans break

```
i = 1
while i < 6:
  print(i)
  i += 1
```

Instruction répétitive avec break

```
i = 1
while i < 6:
  print(i)
  if i == 3: #condition de sortie
    break  # sortie de la boucle
  i += 1
```

PYTHON FOR LOOPS

```
fruits = ["apple", "banana", "cherry"]
for x in fruits:
  print(x)
```

```
for x in range(6):
  print(x)
else:
 print("Finally finished!")
```

```
for x in range(2, 6):
  print(x)
```

```
for x in range(2, 30, 3):
  print(x)
```

FONCTIONS

- **Déclaration de la fonction**

```
def my_function(x,y,z):
```

Paramètres de la fct

```
return instruction opérations
```

- **Appel de la fonction**

```
my_function(a,b,c)
```

Valeurs d'entrée

- **Exemple**

```
def my_function(x):
  return 5 * x

print(my_function(3))
print(my_function(5))
print(my_function(9))
```

LAMBDA

Une fonction lambda est une petite fonction anonyme. Elle peut prendre n'importe quel nombre d'arguments, mais ne peut avoir qu'une seule expression.

```
x = lambda a, b : a * b
print(x(5, 6))
```

```
x = lambda a, b, c : a + b + c
print(x(5, 6, 2))
```

```
def myfunc(n):
  return lambda a : a * n

mydoubler = myfunc(2)

print(mydoubler(11))
```

POO AVEC PYTHON

Python est un langage de programmation orienté objet. Presque tout en Python est un objet, avec ses propriétés et ses méthodes. Une classe est comme un constructeur d'objet ou un «plan directeur» pour créer des objets.

Contenu de la class

```
class MyClass:
  x = 5
```

Instanciation

```
p1 = MyClass()
print(p1.x)
```

```
class Person:
  def __init__(self, name, age):
    self.name = name
    self.age = age

p1 = Person("John", 36)

print(p1.name)
print(p1.age)
```

Initialisation

POO/MÉTHOD

```
class Person:
  def __init__(self, name, age):
    self.name = name
    self.age = age

  def myfunc(self):
    print("Hello my name is " + self.name)

p1 = Person("John", 36)
p1.myfunc()
```

Modification d'un objet

```
p1.age = 40
```

Suppression d'un objet

```
del p1.age
```

POO/HÉRITAGE

```
class Person:
  def __init__(self, fname, lname):
    self.firstname = fname
    self.lastname = lname

  def printname(self):
    print(self.firstname, self.lastname)

class Student(Person):
  def __init__(self, fname, lname, year):
    super().__init__(fname, lname)
    self.graduationyear = year

  def welcome(self):
    print("Welcome", self.firstname, self.lastname, "to the class of", self.graduationyear)

x = Student("Mike", "Olsen", 2019)
x.welcome()
```

- Class mère
- Class fils
- Méthode de la classe fils
- Instanciation de la classe fils

ITERATORS

Un littérateur est un objet qui contient un nombre dénombrable de valeurs. Un littérateur est un objet qui peut être itéré, ce qui signifie que vous pouvez parcourir toutes les valeurs. Techniquement, en Python, un littérateur est un objet qui implémente le protocole d'itération, qui se compose des méthodes **__iter __ ()** et **__next __ ().**

```
mytuple = ("apple",
"banana", "cherry")
myit = iter(mytuple)

print(next(myit))
print(next(myit))
print(next(myit))
```

```
class MyNumbers:
  def __iter__(self):
    self.a = 1
    return self
  def __next__(self):
    x = self.a
    self.a += 1
    return x
myclass = MyNumbers()
myiter = iter(myclass)
print(next(myiter))
print(next(myiter))
print(next(myiter))
```

MODULE

Un module est identique à une bibliothèque de code.
Un fichier contenant un ensemble de fonctions que vous souhaitez inclure dans votre application.

Création du module **mymodule.py**

```
def greeting(name):
  print("Hello, " + name)

person1 = {
  "name": "John",
  "age": 36,
  "country": "Norway"
}
```

Importation du module

```
import mymodule

mymodule.greeting("Jonathan")
```

```
import mymodule

a = mymodule.person1["age"]
print(a)
```

DATE/TIME

Python admet une bibliothèque très riche de formatage et de traitement des objets de type date

```
import datetime

x = datetime.datetime.now()
print(x)

print(x.year)
print(x.strftime("%A"))
```

```
x = datetime.datetime(2020, 5, 17)
print(x)
```

FORMAT DATE/TIME

Directive	Description	Example
%a	Weekday, short version	Wed
%A	Weekday, full version	Wednesday
%w	Weekday as a number 0-6, 0 is Sunday	3
%d	Day of month 01-31	31
%b	Month name, short version	Dec
%B	Month name, full version	December
%m	Month as a number 01-12	12
%y	Year, short version, without century	18
%Y	Year, full version	2018
%H	Hour 00-23	17
%I	Hour 00-12	05
%p	AM/PM	PM
%M	Minute 00-59	41
%S	Second 00-59	08
%f	Microsecond 000000-999999	548513
%z	UTC offset	+0100
%Z	Timezone	CST
%j	Day number of year 001-366	365
%U	Week number of year, Sunday as the first day of week, 00-53	52
%W	Week number of year, Monday as the first day of week, 00-53	52
%c	Local version of date and time	Mon Dec 31 17:41:00 2018
%x	Local version of date	12/31/18
%X	Local version of time	17:41:00
%%	A % character	%

JSON AND PYTHON

Python admet une bibliothèque de formatage et de traitement des objets json

```
import json
x =  '{ "name":"John", "age":30, "city":"New
York"}‘    # some JSON:
y = json.loads(x)   # parse x:
print(y["age"]) # the result is a Python dictionary:
```

```
import json
x = {
  "name": "John",
  "age": 30,
  "city": "New York"
} # a Python object (dict):
y = json.dumps(x) # convert into JSON:
print(y) # the result is a JSON string:
```

AUTRE OBJETS PYTHON

```
username = input("Enter username:")
print("Username is: " + username)
```

Entrée utilisateur

```
try:
  print(x)
except:
  print("An exception occurred")
```

Exception

Formatage du texte

```
quantity = 3
itemno = 567
price = 49
myorder = "I want {0} pieces of item number {1} for {2:.2f} dollars."
print(myorder.format(quantity, itemno, price))
```

I want 3 pieces of item number 567 for 49.00 dollars.

LIRE UN FICHIER

Pour lire un fichier existant, vous devez ajouter un paramètre à la fonction open (): "r"

```
f = open("demofile.txt", "r")
print(f.read())
```

```
print(f.readline())
```

```
f.close
```

```
for x in f:
  print(x)
```

ECRITURE ET CRÉATION D'UN FICHIER

Pour écrire dans un fichier existant, vous devez ajouter un paramètre à la fonction open ():
"a" - Ajouter - ajoutera à la fin du fichier
"w" - Écriture - écrasera tout contenu existant
"x" - Création – création d'un nouveau fichier

```
f = open("demofile2.txt", "a")
f.write("Now the file has more content!")
f.close()

#open and read the file after the appending:
f = open("demofile2.txt", "r")
print(f.read())

f = open("myfile.txt", "x")
```

MATH AND STAT

```
1.import math
2.angleInDegree = 60
3.angleInRadian = math.radians(angleInDegree)
4.print('Given angle :', angleInRadian)
5.print('cos(x) is :', math.cos(angleInRadian))
```

```
import math
number = 5e-2  # small value of of x
print('The given number (x) is :', number)
print('e^x (using exp() function) is :', math.exp(number)-1)
```

MATH AND STAT

```
import statistics
# list of positive integer numbers
datasets = [5, 2, 7, 4, 2, 6, 8]
x = statistics.mean(datasets)
# Printing the mean
print("Mean is :", x)
```

```
import statistics
datasets = [4, -5, 6, 6, 9, 4, 5, -2]
# Printing median of the
# random data-set
print("Median of data-set is : % s "
      % (statistics.median(datasets)))
```

INSTALLER PIP

Si vous n'avez pas installé PIP, vous pouvez le télécharger et l'installer à partir de cette page:

https://pypi.org/project/pip/

pip install camelcase ou uinstall pour désinstaller le module

```
import camelcase

c = camelcase.CamelCase()

txt = "hello world"

print(c.hump(txt))
```

PYTHON ET GRAPHIQUE (PLOT)

Pour installer la bibliothèque de matplot sur python tapez la commande

pip install matplotlib

conda install matplotlib

from matplotlib import pyplot as plt

Exemple1

```
x = [5, 2, 7]
y = [1, 10, 4]
plt.plot(x, y)
plt.title('Line graph')
plt.ylabel('Y axis')
plt.xlabel('X axis')
plt.show()
```

Exemple2

```
from matplotlib import pyplot as plt
plt.plot([1,2,3,4,5])
plt.ylabel("y axis")
plt.xlabel('x axis')
plt.show()
```

Exemple3

```
plt.plot([1,2,3,4,5],[1,4,9,16,25])
plt.ylabel("y axis")
plt.xlabel('x axis')
plt.show()
```

PYTHON ET GRAPHIQUE (PLOT)

Exemple1

```
from matplotlib import pyplot as plt
plt.plot([1, 2, 3, 4,5], [1, 4, 9, 16,25], 'ro')
plt.axis([0, 6, 0, 20])
plt.show()
```

Exemple2

```
from matplotlib import pyplot
names = ['Abhishek', 'Himanshu', 'Devansh']
marks= [87,50,98]
plt.figure(figsize=(9,3))
plt.subplot(131)
plt.bar(names, marks)
plt.subplot(132)
plt.scatter(names, marks)
plt.subplot(133)
plt.plot(names, marks)
plt.suptitle('Categorical Plotting')
plt.show()
```

'b'	Using for the blue marker with default shape.
'ro'	Red circle
'-g'	Green solid line
'--'	A dashed line with the default color
'^k:'	Black triangle up markers connected by a dotted line

Character	Color
'b'	**Blue**
'g'	Green
'r'	Red
'c'	Cyan
'm'	Magenta
'y'	Yellow
'k'	Black
'w'	White

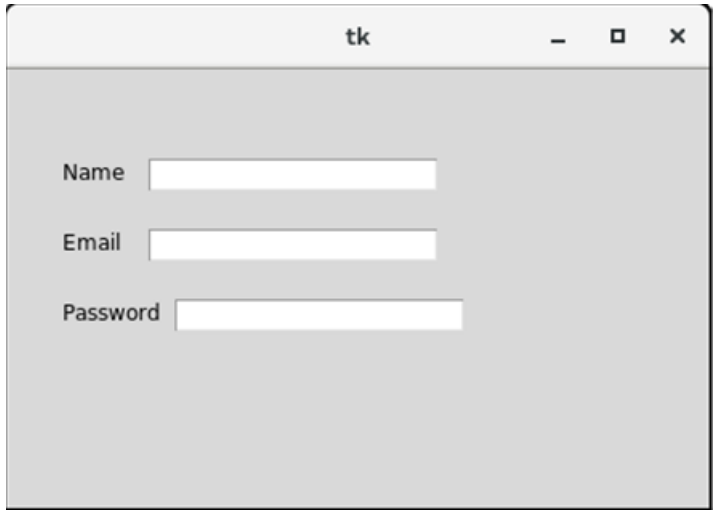
GUI TKINTER

Tkinter est conçu pour les débutants et les professionnels. Python fournit la bibliothèque standard Tkinter pour créer l'interface utilisateur graphique pour les applications de bureau. Développer des applications basées sur le bureau avec python Tkinter n'est pas une tâche complexe. Une fenêtre de niveau supérieur Tkinter vide peut être créée à l'aide des étapes suivantes.

```
1.from tkinter import *
2.#creating the application main window.
3.top = Tk()
4.#Entering the event main loop
5.top.mainloop()
```

FENETRE2

```
from tkinter import *
top = Tk()
top.geometry("400x250")
name = Label(top, text = "Name").place(x = 30,y = 50)
email = Label(top, text = "Email").place(x = 30, y = 90)
password = Label(top, text = "Password").place(x = 30, y = 130)
e1 = Entry(top).place(x = 80, y = 50)
e2 = Entry(top).place(x = 80, y = 90)
e3 = Entry(top).place(x = 95, y = 130)
top.mainloop()
```

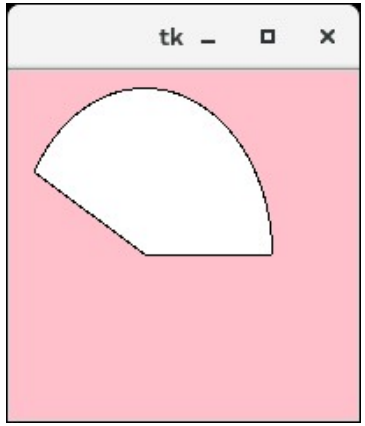


MESSAGE PACK

```
from tkinter import *

top = Tk()
top.geometry("100x100")
var = StringVar()
msg = Message( top, text = "Welcome to Javatpoint")

msg.pack()
top.mainloop()
```

CANVAS

```
from tkinter import *
top = Tk()
top.geometry("200x200")

#creating a simple canvas
c = Canvas(top,bg = "pink",height = "200",width = 200)
arc = c.create_arc((5,10,150,200),start = 0,extent = 150, fill= "white")
c.pack()
top.mainloop()
```



FENETRE1

```
from tkinter import *
parent = Tk()
def fun():
   messagebox.showinfo("Hello", "Button is clicked")
name = Label(parent,text = "Name").grid(row = 0, column = 0)
e1 = Entry(parent).grid(row = 0, column = 1)
password = Label(parent,text = "Password").grid(row = 1, column = 0)
e2 = Entry(parent).grid(row = 1, column = 1)
submit = Button(parent, command = fun, text = "Submit").grid(row = 4, column = 0)
parent.mainloop()
```

CALCULATRICE

```
import tkinter as tk
from functools import partial
 def call_result(label_result, n1, n2):
    num1 = (n1.get())
    num2 = (n2.get())
    result = int(num1)+int(num2)
    label_result.config(text="Result = %d" % result)
    return
root = tk.Tk()
root.geometry('400x200+100+200')
root.title('Calculator')
number1 = tk.StringVar()
number2 = tk.StringVar()
labelNum1 = tk.Label(root, text="A").grid(row=1, column=0)
labelNum2 = tk.Label(root, text="B").grid(row=2, column=0)
labelResult = tk.Label(root)
labelResult.grid(row=7, column=2)
entryNum1 = tk.Entry(root, textvariable=number1).grid(row=1,
column=2)
entryNum2 = tk.Entry(root, textvariable=number2).grid(row=2,
column=2)
call_result = partial(call_result, labelResult, number1,
number2)
buttonCal = tk.Button(root, text="Calculate",
command=call_result).grid(row=3, column=0)
root.mainloop()
```

LANCER ALEATOIRE

```
from tkinter import *
import random
def NouveauLance():
    nb = random.randint(1,6)
    Texte.set('Résultat -> ' + str(nb))
# Création de la fenêtre principale (main window)
Mafenetre = Tk()
Mafenetre.title('Dé à 6 faces')
Mafenetre.geometry('300x100+400+400')
# Création d'un widget Button (bouton Lancer)
BoutonLancer = Button(Mafenetre, text ='Lancer',
command = NouveauLance)
# Positionnement du widget avec la méthode pack()
BoutonLancer.pack(side = LEFT, padx = 5, pady = 5)
# Création d'un widget Button (bouton Quitter)
BoutonQuitter = Button(Mafenetre, text ='Quitter',
command = Mafenetre.destroy)
BoutonQuitter.pack(side = LEFT, padx = 5, pady = 5)
Texte = StringVar()
NouveauLance()
# Création d'un widget Label (texte 'Résultat -> x')
LabelResultat = Label(Mafenetre, textvariable =
Texte, fg ='red', bg ='white')
LabelResultat.pack(side = LEFT, padx = 5, pady = 5)
Mafenetre.mainloop()
```

MOT DE PASSE

```
from tkinter import *
from tkinter.messagebox import * # boîte de dialogue
def Verification():
    if Motdepasse.get() == 'python27':
        # le mot de passe est bon : on affiche une boîte de
dialogue puis on ferme la fenêtre
        showinfo('Résultat','Mot de passe correct.\nAu revoir !')
        Mafenetre.destroy()
    else:
        # le mot de passe est incorrect : on affiche une boîte de
dialogue
        showwarning('Résultat','Mot de passe
incorrect.\nVeuillez recommencer !')
        Motdepasse.set('')
# Création de la fenêtre principale (main window)
Mafenetre = Tk()
Mafenetre.title('Identification requise')
# Création d'un widget Label (texte 'Mot de passe')
Label1 = Label(Mafenetre, text = 'Mot de passe ')
Label1.pack(side = LEFT, padx = 5, pady = 5)
# Création d'un widget Entry (champ de saisie)
Motdepasse= StringVar()
Champ = Entry(Mafenetre, textvariable=
Motdepasse, show='*', bg ='bisque', fg='maroon')
Champ.focus_set()
Champ.pack(side = LEFT, padx = 5, pady = 5)
# Création d'un widget Button (bouton Valider)
Bouton = Button(Mafenetre, text ='Valider',
command = Verification)
Bouton.pack(side = LEFT, padx = 5, pady = 5)
Mafenetre.mainloop()
```

RADIO BOUTON

```
from tkinter import *
def selection():
  selection = "You selected the option " + str(radio.get())
  label.config(text = selection)
top = Tk()
top.geometry("300x150")
radio = IntVar()
lbl = Label(text = "Favourite programming language:")
lbl.pack()
R1 = Radiobutton(top, text="C", variable=radio, value=1,
            command=selection)
R1.pack( anchor = W )
R2 = Radiobutton(top, text="C++", variable=radio, value=2,
            command=selection)
R2.pack( anchor = W )
R3 = Radiobutton(top, text="Java", variable=radio, value=3,
            command=selection)
R3.pack( anchor = W)
label = Label(top)
label.pack()
top.mainloop()
```

CHECK BOUTTON

```
from tkinter import *
top = Tk()
top.geometry("200x200")
checkvar1 = IntVar()
checkvar2 = IntVar()
checkvar3 = IntVar()
chkbtn1 = Checkbutton(top, text = "C", variable = checkvar1, onvalue
= 1, offvalue = 0, height = 2, width = 10)
chkbtn2 = Checkbutton(top, text = "C++", variable = checkvar2,
onvalue = 1, offvalue = 0, height = 2, width = 10)
chkbtn3 = Checkbutton(top, text = "Java", variable = checkvar3,
onvalue = 1, offvalue = 0, height = 2, width = 10)
chkbtn1.pack()
chkbtn2.pack()
chkbtn3.pack()
top.mainloop()
```

CALCUL AUTOMATIQUE

```
from tkinter import *
def carre():
   """ Calcul du carré """
   Resultat.set("Carré = "+str(float(Valeur.get())**2))
# Création de la fenêtre principale (main window)
Mafenetre = Tk()
Mafenetre.title("Spinbox widget")
Valeur = StringVar()
Valeur.set(2.0)
# Création d'un widget Spinbox
boite =
Spinbox(Mafenetre,from_=0,to=10,increment=0.5,textvariable=Valeur,width=5,com
mand=carre)
boite.pack(padx=30,pady=10)
# Création d'un widget Label
Resultat = StringVar()
carre()
Label(Mafenetre,textvariable=Resultat).pack(padx=30,pady=10)
Mafenetre.mainloop()
```

MENU

```
from tkinter import Toplevel, Button, Tk, Menu
top = Tk()
menubar = Menu(top)
file = Menu(menubar, tearoff=0)
file.add_command(label="New")
file.add_command(label="Open")
file.add_command(label="Save")
file.add_command(label="Save as...")
file.add_command(label="Close")
file.add_separator()
file.add_command(label="Exit", command=top.quit)
menubar.add_cascade(label="File", menu=file)
edit = Menu(menubar, tearoff=0)
edit.add_command(label="Undo")
edit.add_separator()
edit.add_command(label="Cut")
edit.add_command(label="Copy")
edit.add_command(label="Paste")
edit.add_command(label="Delete")
edit.add_command(label="Select All")
menubar.add_cascade(label="Edit", menu=edit)
help = Menu(menubar, tearoff=0)
help.add_command(label="About")
menubar.add_cascade(label="Help", menu=help)
top.config(menu=menubar)
top.mainloop()
```

GENERATEUR GUI

https://www.magicsplat.com/tcl-installer/index.html

Current releases:

- Download Magicsplat Tcl/Tk 8.6.10-1.10.0 for Windows (64-bit)
- Download Magicsplat Tcl/Tk 8.6.10-1.10.0 for Windows (32-bit)

https://sourceforge.net/projects/page/

ACCES BDD MYSQL

python -m pip install mysql-connector

```
1.import mysql.connector
2.
3.#Create the connection object
4.myconn = mysql.connector.connect(host = "localhost", user = "root",passwd
= "google", database = "mydb")
5.
6.#printing the connection object
7.print(myconn)
```

LISTER LES BDD ET CRÉER NV BDD

```
import mysql.connector
#Create the connection object
myconn = mysql.connector.connect(host = "localhost", user =
"root",passwd = "google")
#creating the cursor object
cur = myconn.cursor()
try:
   #creating a new database
   cur.execute("create database PythonDB2")
   #getting the list of all the databases which will now include
the new database PythonDB
   dbs = cur.execute("show databases")
except:
   myconn.rollback()
for x in cur:
      print(x)
myconn.close()
```

AJOUTER UNE TABLE

```
import mysql.connector
#Create the connection object
myconn = mysql.connector.connect(host = "localhost", user = "root",passwd =
"google",database = "PythonDB")
#creating the cursor object
cur = myconn.cursor()
try:
   #adding a column branch name to the table Employee
dbs = cur.execute("create table Employee(name varchar(20) not null, id int(20) n
ot null primary key, salary float not null, Dept_id int not null")  except:
   myconn.rollback()
myconn.close()
```

AJOUTER DES DONNÉES

```
import mysql.connector
#Create the connection object
myconn = mysql.connector.connect(host = "localhost", user =
"root",passwd = "google",database = "PythonDB")
#creating the cursor object
cur = myconn.cursor()
sql = "insert into Employee(name, id, salary, dept_id,
branch_name) values (%s, %s, %s, %s, %s)"
#The row values are provided in the form of tuple
val = ("John", 110, 25000.00, 201, "Newyork")
try:
   #inserting the values into the table
   cur.execute(sql,val)
   #commit the transaction
   myconn.commit()
except:
   myconn.rollback()
print(cur.rowcount,"record inserted!")
myconn.close()
```

LISTER LES DONNÉES

```
import mysql.connector
#Create the connection object
myconn = mysql.connector.connect(host = "localhost", user = "root",passwd
= "google",database = "PythonDB")
#creating the cursor object
cur = myconn.cursor()
try:
   #Reading the Employee data
   cur.execute("select * from Employee")
   #fetching the rows from the cursor object
   result = cur.fetchall()
   #printing the result
   for x in result:
      print(x);
except:
   myconn.rollback()
myconn.close()
```

LISTER LES DONNÉES EN COLLONE

```
import mysql.connector
#Create the connection object
myconn = mysql.connector.connect(host = "localhost", user =
"root",passwd = "google",database = "PythonDB")
#creating the cursor object
cur = myconn.cursor()
try:
   #Reading the Employee data
   cur.execute("select name, id, salary from Employee")
   #fetching the rows from the cursor object
   result = cur.fetchall()
   print("Name    id    Salary");
   for row in result:
      print("%s    %d    %d"%(row[0],row[1],row[2]))
except:
   myconn.rollback()
myconn.close()
```

MISE À JOUR DES DONNÉES

```
import mysql.connector
#Create the connection object
myconn = mysql.connector.connect(host = "localhost", user =
"root",passwd = "google",database = "PythonDB")
#creating the cursor object
cur = myconn.cursor()
try:
   #updating the name of the employee whose id is 110
   cur.execute("update Employee set name = 'alex' where id = 110")
   myconn.commit()
except:
   myconn.rollback()

myconn.close()
```

SUPPRIMER DES DONNÉES

```
import mysql.connector
#Create the connection object
myconn = mysql.connector.connect(host = "localhost", user = "root",passwd =
"google",database = "PythonDB")
#creating the cursor object
cur = myconn.cursor()
try:
   #Deleting the employee details whose id is 110
   cur.execute("delete from Employee where id = 110")
   myconn.commit()
except:
   myconn.rollback()
myconn.close()
```

PYTHON ET SQLITE

Pour installer sqlite taper la commande **sudo apt-get install sqlite3 libsqlite3-dev**

```
import sqlite3
conn = sqlite3.connect('javatpoint.db')
print("Opened database successfully");


conn.execute("""CREATE TABLE Employees
      (ID INT PRIMARY KEY     NOT NULL,
      NAME          TEXT    NOT NULL,
      AGE           INT     NOT NULL,
      ADDRESS       CHAR(50),
      SALARY        REAL);""")
print ("Table created successfully");
conn.close()
```

PYTHON SQLITE

```
import sqlite3
conn = sqlite3.connect('python.db')
data = conn.execute("select * from Employees");
for row in data:
   print ("ID = ", row[0])
   print ("NAME = ", row[1])
   print ("ADDRESS = ", row[2])
   print ("SALARY = ", row[3], "\n")
conn.close();
```