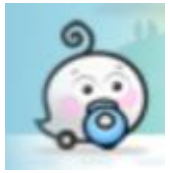
Python 3



Facile

Normal

Difficile

Professionnel

Expert

david.palermo@yantra-technologies.com

https://wiki.waze.com/wiki/Your_Rank_and_Points

Version 1.5
Copyright : Yantra Technologies 2004-2022

Python 3 : Sommaire

1 - Généralités
2 - Installation
3 - Les Notions de Base
4 - Typage dynamique
5 - Les Classes et les Objets
6 - Modèle objet
7 - Types de données
8 - Séquence
9 - Chaîne de caractères
10 - Dictionnaire
11 - Données temporelles
12 - Les fichiers
13 - Bibliothèque standard
14 - Différence python 2.7.x avec 3.x et évolution de python de 3.5 à 3.10
15 - Complément
16 - Environnement Graphique : TKINTER
17 - Environnement Graphique : PYGAME
18 - Application Web - Frameworks
19 - Le Réseau
20 - base de données
21 - Bibliographie

Version 1.5
Copyright : Yantra Technologies 2004-2022

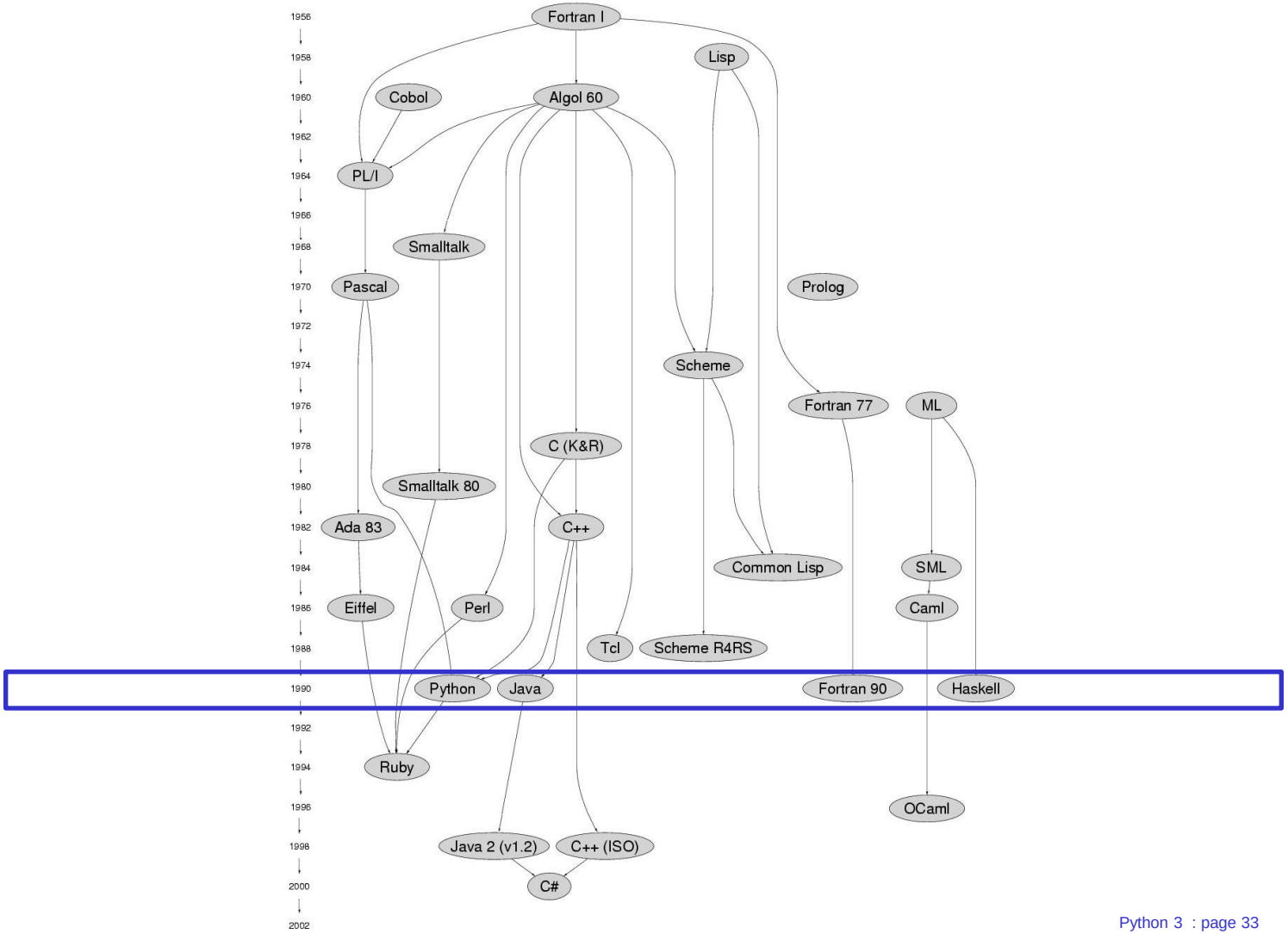
1 - Généralités

- Historique des langages
- Types de programmation
- Présentation PYTHON

http://izibook.eyrolles.com/extract/show/6213

Copyright : Yantra Technologies 2004-2022

1 – Généralités -> Historique des langages



Version 1.5
Copyright : Yantra Technologies 2004-2022

1 – Généralités -> Types de programmation

- ## Types de programmation impérative

  La programmation impérative est un paradigme de programmation qui décrit les opérations en termes de séquences d'instructions exécutées par l'ordinateur pour modifier l'état du programme. C’est le paradigme originel et le plus courant.

- ## Types de programmation orientée objet

  La programmation orientée objet consiste en la définition et l’assemblage de briques logicielles appelées objets.

- ## Types de programmation déclarative

  La programmation déclarative consiste à déclarer les données du problème, puis à demander au programme de le résoudre.

Version 1.5
Copyright : Yantra Technologies 2004-2022

1 – Généralités -> Programmation impérative

- Programmation structurée, visant à structurer les programmes impératifs pour en supprimer les instructions goto.

- Programmation procédurale
  - 1940 Langage machine
  - 1952 A-0
  - 1954 FORTRAN
  - 1963 BASIC
  - 1960 COBOL
  - 1958 Algol
  - 1971 Pascal
  - 1972 C, Smalltalk
  - 1983 Ada , C++, Objective C
  - 1987 Perl,
  - 1991 Python
  - 1995 PHP, Java

Version 1.5
Copyright : Yantra Technologies 2004-2022

1 – Généralités -> Programmation orientée objet

- Programmation orientée prototype, qui simplifie et rend plus flexible la programmation orientée objet (javascript)

- Programmation orientée classe, à comparer à la Programmation orientée prototype
  - typage dynamique : le type des objets est déterminé à l'exécution lors de la création desdits objets (Smalltalk, CLOS, **Python**, PHP…),
  - Typage statique : le type des objets est vérifié à la compilation et est soit explicitement indiqué par le développeur lors de leur déclaration (C++, Java, C#, Pascal…), soit déterminé par le compilateur à partir du contexte (Scala, OCaml, Haskell…).
  - Mécanismes de sous-typage:
    - l'héritage simple (Smalltalk, Java, C#)
    - Héritage multiple (C++, **Python**, CLOS, Eiffel, WLangage).
- Programmation orientée composant (comme en OLE)

Version 1.5
Copyright : Yantra Technologies 2004-2022

1 – Généralités -> Programmation déclarative

- Programmation descriptive, à l'expressivité réduite, qui permet de décrire des structures de données (1989 HTML, 1997 XML ou 1983 LaTeX)
- Programmation fonctionnelle, avec laquelle un programme est une fonction au sens mathématique du terme
  - Langages fonctionnels pas ou peu typés
    - 1958 Lisp
    - 1975 Scheme
    - 1984 Common Lisp
    - 1987 Caml
  - Langages fonctionnels plus récents fortement typés.
    - 1973 ML
    - 1987 Haskell,
    - 1987 Erlang (concurrent, temps réel, distribué)
    - 1996 OCaml,

- Programmation logique, consistant à exprimer les problèmes et les algorithmes sous forme de prédicats (1972 Prolog)
- Programmation par contraintes, permet de résoudre des problèmes combinatoires de grandes tailles tels que les problèmes de planification et d'ordonnancement (à comparer à la programmation logique)

Version 1.5
Copyright : Yantra Technologies 2004-2022

1 – Généralités -> Présentation PYTHON

- Histoire de Python
- Python
- Intégration avec d’autres langages

Version 1.5
Copyright : Yantra Technologies 2004-2022

1 – Généralités -> Histoire de Python

- 02/1991 : Guido van Rossum (Fan de la série télévisée des Monty Python) travaille aux Pays-Bas au CWI 1 sur le projet AMOEBA (un système d'exploitation distribué). Il conçoit Python (à partir du langage ABC) et publie la version 0.9.0 sur un forum Usenet

Guido van Rossum, créateur de Python, à la OSCON 2006.

- 1996 : sortie de Numerical Python

- 2001 : Python 2.0, naissance de la PSF (Python Software Fundation)

- 12/2008 : sorties simultanées de Python 2.6 et de Python 3.0

  pour nettoyer la bibliothèque standard de ses éléments obsolètes et redondants, Python a choisi de casser la compatibilité ascendante dans la nouvelle version majeure

- 08/2014 : versions en cours Python 2.7.8 et Python 3.4.1

Copyright : Yantra Technologies 2004-2022

1 – Généralités -> Python

Python est un langage avec une syntaxe minimale, explicite, claire et simple.

Il est utilisé :

- comme langage de script pour automatiser des tâches simples mais fastidieuses,
- dans un logiciel de conception assistée par ordinateur afin d'automatiser certains enchaînements d'actions répétitives,
- comme langage de développement de prototype lorsqu'on a besoin d'une application fonctionnelle avant de l'optimiser avec un langage de plus bas niveau,
- en programmation POO ou/et en mode procédural.

Il est particulièrement répandu dans le monde scientifique, et possède de nombreuses extensions destinées aux applications numériques.

- Open-source et gratuit.
- C’est un langage interprété sans typage statique.
- C’est un langage objet dès sa création.
- Il permet de créer des programmes.
- Il est possible d'associer des bibliothèques à Python afin d'étendre ses possibilités.
- Il est portable, c'est à dire qu'il peut fonctionner sous différents systèmes d'exploitation (Windows, Linux, Mac OS X,…).

Copyright : Yantra Technologies 2004-2022

1 – Généralités -> Lien Internet

Histoire python :

http://izibook.eyrolles.com/extract/show/6213

Python Tkinter

http://www.youtube.com/watch?v=IGSmLvfEz18

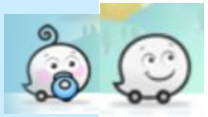
Python Pygame :

https://www.youtube.com/watch?v=IGSmLvfEz18

https://www.python.org/

Version 1.5
Copyright : Yantra Technologies 2004-2022

1 – Généralités -> Intégration avec d'autres langages



- Extensions C
- Intégration de programmes écrits en C
- Intégration de programmes Python dans du C
- Intégration de programmes écrits en Java
- Intégration de programmes Python dans Java
- Autres intégrations: Fortran, Lisp, Scheme… .

Copyright : Yantra Technologies 2004-2022

2 – Installation

- Installer Python
- Installer des bibliothèques externes
- Installer un IDE

http://www.formation-django.fr/python/comment-installer-python.html

http://python.developpez.com/cours/DiveIntoPython/php/frdiveintopython/installing_python/index.php

Version 1.5
Copyright : Yantra Technologies 2004-2022

2 – Installation -> Installer Python

- Installation Windows
- Installation Mac
- Installation sous Linux

Copyright : Yantra Technologies 2004-2022

2 – Installation -> Installation Windows

Sous Windows

- Cliquez sur le lien Download dans le menu principal de la page.
- Sélectionnez la version de Python que vous souhaitez utiliser.
- On vous propose un (ou plusieurs) lien(s) vers une version Windows : sélectionnez celle qui conviendra à votre processeur.
- Enregistrez puis exécutez le fichier d'installation et suivez les étapes.
- Une fois l'installation terminée, vous pouvez vous rendre dans le menu Démarrer > Tous les programmes. Python devrait apparaître dans cette liste.
- Python est installé sous Windows.

https://docs.python.org/3/using/windows.html
https://www.youtube.com/watch?v=O_xBNq833bM

Version 1.5
Copyright : Yantra Technologies 2004-2022

Sous Mac OS X

- Téléchargez la dernière version de Python. Ouvrez le fichier .dmg et faites un double-clic sur le paquet d'installation Python.mpkg
- Un assistant d'installation s'ouvre, laissez-vous guider
- Python est maintenant installé !

https://www.youtube.com/watch?v=QhutbLBKBok

http://python.developpez.com/cours/DiveIntoPython/php/frdiveintopython/installing_python/macosx.php

Version 1.5
Copyright : Yantra Technologies 2004-2022

2 – Installation -> Installation sous Linux

Sous Linux

- Python est pré-installé sur la plupart des distributions Linux. Cependant, il est possible que vous n'ayez pas la dernière version en date. Pour le vérifier, tapez dans un terminal la commande python -v. Cette commande vous renvoie la version de Python actuellement installée sur votre système. Il est très probable que ce soit une version **2.x**, comme 2.6 ou 2.7
- Cliquez sur download et téléchargez la dernière version de Python (actuellement « Python 3.4 gzipped source tarball »). Ouvrez un terminal, puis rendez-vous dans le dossier où se trouve l'archive
- Décompressez l'archive en tapant : tar -xzf Python-3.4.0.tar.bz2
- Exécutez le script configure en tapant ./configure dans la console
- Une fois que la configuration s'est déroulée, il n'y a plus qu'à compiler en tapant make puis make install en tant que super-utilisateur.

De manière automatique utiliser le gestionnaire du système

- sudo aptitude install python3
- sudo yum install python3
- l'interface graphique (exemple avec Synaptic)

https://fr.wikibooks.org/wiki/Python_3_par_l%27exemple/install

http://www.unixmen.com/howto-install-python-3-x-in-ubuntu-debian-fedora-centos/

Version 1.5
Copyright : Yantra Technologies 2004-2022

2 – Installation -> Installer un IDE

EDI en français ou *IDE* pour I*ntegrated Development Environement*

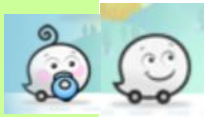
https://wiki.python.org/moin/IntegratedDevelopmentEnvironments

1. **PyCharm : https://www.jetbrains.com/pycharm/**
2. **Visual Studio Code : https://code.visualstudio.com/**
3. Jupyter : https://jupyter.org/
4. Atom : https://atom.io/
5. SublimeText : https://www.sublimetext.com/
6. PyDev : https://www.pydev.org/
7. **IDLE : https://docs.python.org/3/library/idle.html**
8. Spyder : https://github.com/spyder-ide/spyder
9. Thonny : https://thonny.org/
10. Vim : https://www.vim.org/download.php
11. Emacs : https://www.gnu.org/software/emacs/download.html

https://www.commentcoder.com/ide-python/

Version 1.5
Copyright : Yantra Technologies 2004-2022

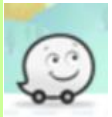
3 - Les Notions de Base



- Délimiteurs
- Instructions
- Constructions fonctionnelles
- Gestion des exceptions
- Divers

Version 1.5
Copyright : Yantra Technologies 2004-2022

3 - Les Notions de Base-> Délimiteurs



- Instructions
- Instructions sur plusieurs ligne
- Les mots clés
- Indentation
- Symboles
- Opérateurs arithmétiques
- Opérateurs logiques
- Opérateurs binaires

Copyright : Yantra Technologies 2004-2022

3 - Les Notions de Base-> Instructions

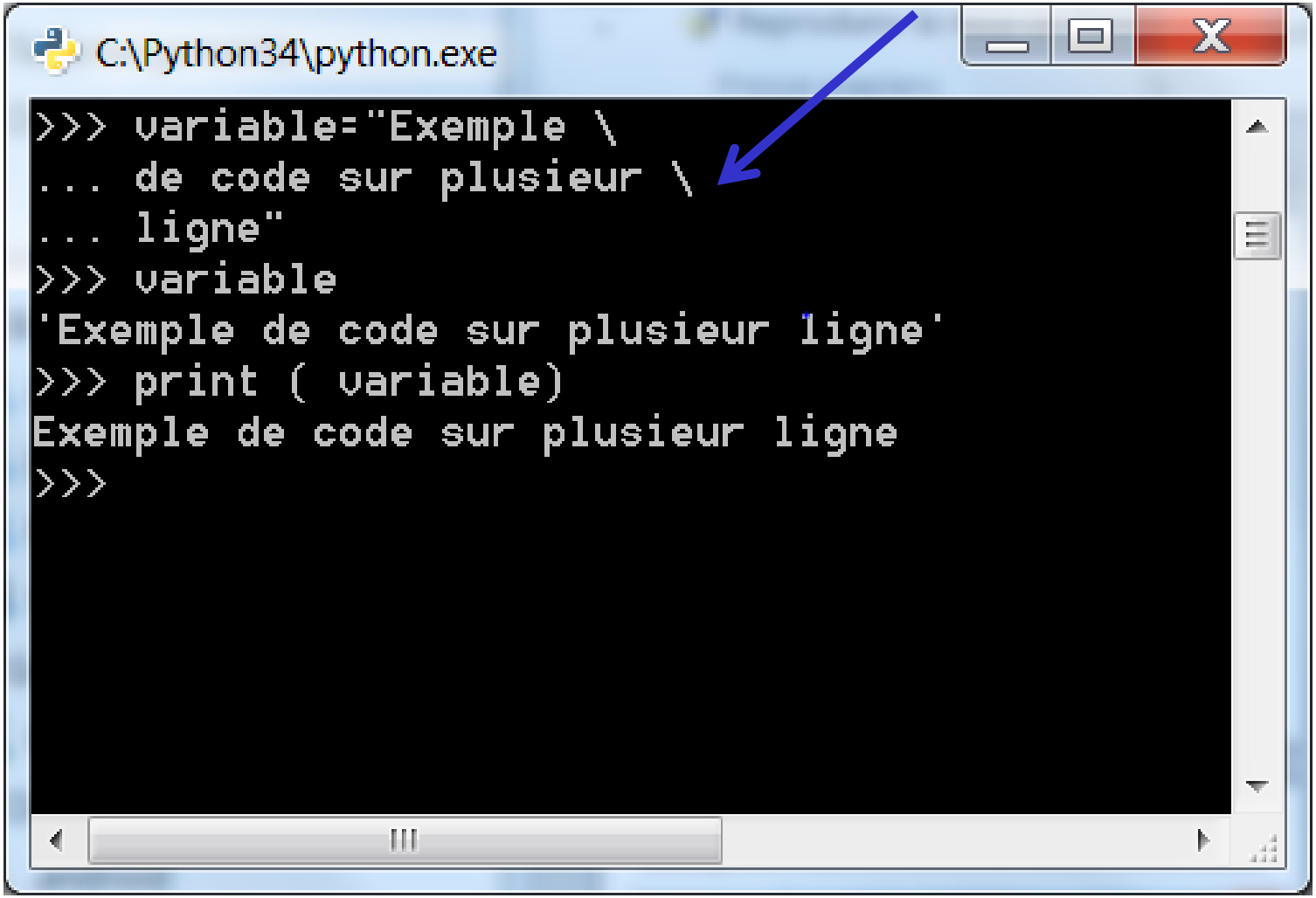
```
C:\Python34\python.exe
>>> # commentaire : ceci est mon premier commentaire
...
>>> print ("Bonjour !!!!!!!!!!!!!!!!")
Bonjour !!!!!!!!!!!!!!!!
>>>
```

Une ligne de code = une instruction

Pour mettre une ligne en commentaire il faut la commencer par #

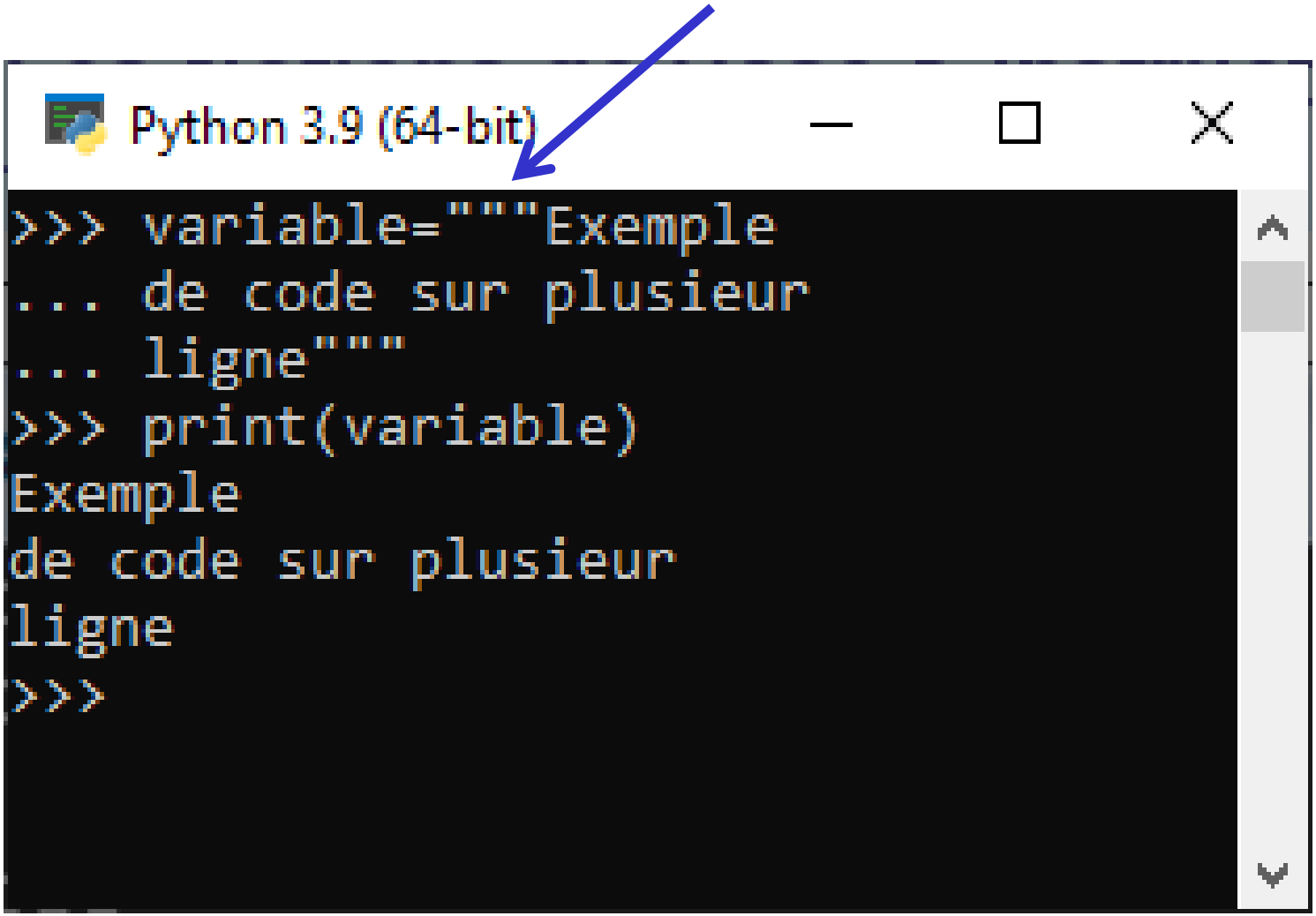
Version 1.5
Copyright : Yantra Technologies 2004-2022

3 - Les Notions de Base
-> Instructions sur plusieurs lignes



Version 1.5
Copyright : Yantra Technologies 2004-2022

3 - Les Notions de Base -> Instructions sur plusieurs lignes



Copyright : Yantra Technologies 2004-2022

3 - Les Notions de Base -> Les mots clés

Mot	Définition	Mot	Définition	Mot	Définition	Mot	Définition
and	Opérateur ET booléen logique	elif	Condition contraire	if	Condition	pass	
as		else	Contraire	import	Importation de module	print	***Afficher*** ***En python 2 mots cle*** ***En python 3*** **des fonctions du module builtins deviennent un mot réservé**
assert		except	Sauf (à utiliser après "try")	in	Contient	raise	
break	Sortie de boucle	exec	***En python 2 mots clés*** ***En python 3*** **des fonctions du module builtins deviennent un mot réservé**	is	Est	return	Stopper la fonction courante (renvoyer sa valeur)
class	Définition de classe d'objet	finally		is not	N'est pas	sort	Classer par ordre alphabétique
continue		for	Boucle	lambda	Définition d'une fonction Lambda	try	Essayer (généralement suivi de "except" : sauf)
def	Définition de fonction	from	De	not	Négation logique	while	Boucle
del	Suppression de	global	Définition dans une fonction d'une variable globale	or	Opérateur de choix OU booléen logique	yield	S'emploie uniquement dans une fonction, et renvoie son résultat régénéré
with							
True	**Nouveau mots clé Python 3**	False	**Nouveau mots clé Python**	None	**Nouveau mots clé Python**	nonlocal	**Nouveau mots clé Python**

Version 1.5
Copyright : Yantra Technologies 2004-2022

3 - Les Notions de Base -> Les mots clés : keyword

help

C:\Python34\python.exe

```
>>> import keyword
>>> keyword.kwlist
['False', 'None', 'True', 'and', 'as', 'assert', 'break', 'class', 'co
ntinue', 'def', 'del', 'elif', 'else', 'except', 'finally', 'for', 'fr
om', 'global', 'if', 'import', 'in', 'is', 'lambda', 'nonlocal', 'not'
, 'or', 'pass', 'raise', 'return', 'try', 'while', 'with', 'yield']
>>>
>>> help ("and")
Boolean operations
******************

   or_test  ::= and_test | or_test "or" and_test
   and_test ::= not_test | and_test "and" not_test
   not_test ::= comparison | "not" not_test

In the context of Boolean operations, and also when expressions are
used by control flow statements, the following values are interpreted
as false: "False", "None", numeric zero of all types, and empty
strings and containers (including strings, tuples, lists,
dictionaries, sets and frozensets).  All other values are interpreted
as true.  User-defined objects can customize their truth value by
providing a "__bool__()" method.

The operator "not" yields "True" if its argument is false, "False"
otherwise.

The expression "x and y" first evaluates *x*; if *x* is false, its
value is returned; otherwise, *y* is evaluated and the resulting value
is returned.

The expression "x or y" first evaluates *x*; if *x* is true, its value
is returned; otherwise, *y* is evaluated and the resulting value is
returned.

(Note that neither "and" nor "or" restrict the value and type they
```


Version 1.5
Copyright : Yantra Technologies 2004-2022

3 - Les Notions de Base -> Indentation

Indentation

4 espaces pour l'indentation.

C:\Python34\python.exe

```
>>> help ( "if")
The "if" statement
******************

The "if" statement is used for conditional execution:

   if_stmt ::= "if" expression ":" suite
               ( "elif" expression ":" suite )*
               ["else" ":" suite]

It selects exactly one of the suites by evaluating the expressions one
by one until one is found to be true (see section *Boolean operations*
for the definition of true and false); then that suite is executed
(and no other part of the "if" statement is executed or evaluated).
If all expressions are false, the suite of the "else" clause, if
present, is executed.

Related help topics: TRUTHVALUE

>>> i=0
>>> if i == 0 :
...     print ("OK")
... else:
...     print ("KO")
...
OK
>>>
```

```
#include <iostream>

using namespace std;

int main()
{
    int i = 0;
    if ( i == 0) {
        std::cout << "OK" << std::endl;
    } else {
        std::cout << "KO" << std::endl;
    }
    return 0;
}
```

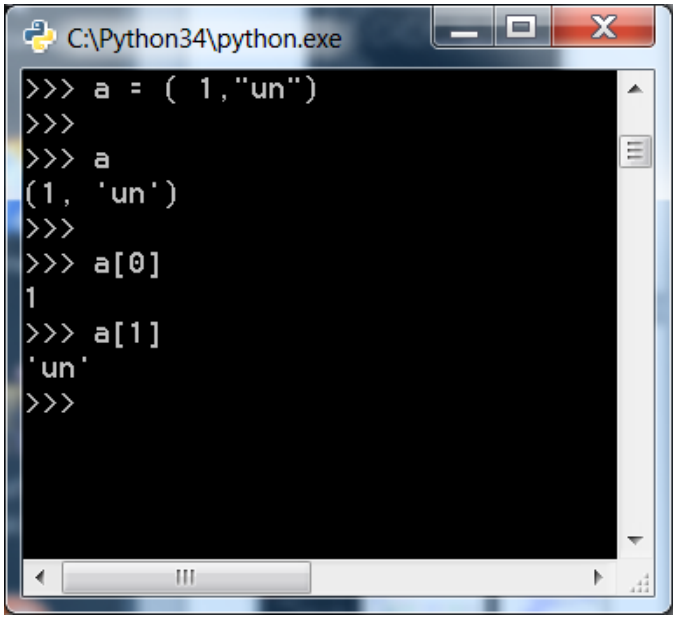
```
PROGRAM
INTEGER I
I=0
IF (I.EQ.0) THEN
  WRITE(*,*) "OK"
ELSE
 WRITE(*,*) "KO"
END IF
END PROGRAM
```


Version 1.5
Copyright : Yantra Technologies 2004-2022

3 - Les Notions de Base -> Symboles

,	()	[]	{}	.	:	Espace
	Tuple	liste	Ensemble dictionnaire			

Les Tuple : non modifiables



https://fr.wikibooks.org/wiki/Programmation_Python/Tuples

Version 1.5
Copyright : Yantra Technologies 2004-2022

3 - Les Notions de Base -> Symboles

,	()	[]	{}	.	:	Espace
	Tuple	liste	Ensemble dictionnaire			

liste

```
>>> l = [1,"un",2,3.5,4.8]
>>>
>>> print(l)
[1, 'un', 2, 3.5, 4.8]
>>>
>>> print("l[0] = ",l[0],"l[1] = ",l[1])
l[0] =  1 l[1] =  un
>>>
>>> print(l[2:])
[2, 3.5, 4.8]
>>>
>>> l[1]=1235456
>>>
>>> print("l[1] = ",l[1])
l[1] =  1235456
>>>
```

https://fr.wikibooks.org/wiki/Programmation_Python/Listes

Version 1.5
Copyright : Yantra Technologies 2004-2022

3 - Les Notions de Base -> Symboles

,	()	[]	{}	.	:	Espace
	Tuple	liste	Ensemble dictionnaire			

Ensemble

```
>>> ensemble= { "a",0, "b", 2, 100.1, 3, 4 }
>>>
>>> print( ensemble)
{0, 'b', 3, 2, 100.1, 4, 'a'}
>>>
>>> for i in ensemble:
...     print(i)
...
0
b
3
2
100.1
4
a
>>>
```

https://fr.wikibooks.org/wiki/Programmation_Python/Ensembles

Version 1.5
Copyright : Yantra Technologies 2004-2022

3 - Les Notions de Base -> Symboles

,	()	[]	{}	.	:	Espace
	Tuple	liste	Ensemble dictionnaire			

Dictionnaire

```
>>> dictionnaire = { "Cle1":"valeur" , 0:"valeur pour 0", "numero secu" : "nom" }
>>>
>>> print(dictionnaire)
{'Cle1': 'valeur', 0: 'valeur pour 0', 'numero secu': 'nom'}
>>>
>>> print(dictionnaire[0])
valeur pour 0
>>>
>>> dictionnaire["40640057"]="Kuriboh"
>>>
>>> print(dictionnaire["40640057"])
Kuriboh
>>>
```

https://fr.wikibooks.org/wiki/Programmation_Python/Dictionnaires

Version 1.5
Copyright : Yantra Technologies 2004-2022

3 - Les Notions de Base -> Symboles

,	()	[]	{}	.	:	Espace
	Tuple	liste	Ensemble dictionnaire			

C:\Python34\pyth...

```
>>>
>>> type(int)
<class 'type'>
>>> i=2
>>> type(i)
<class 'int'>
>>> f=1.0
>>> type(f)
<class 'float'>
>>> 1..real
1.0
>>> (1).real
1
```

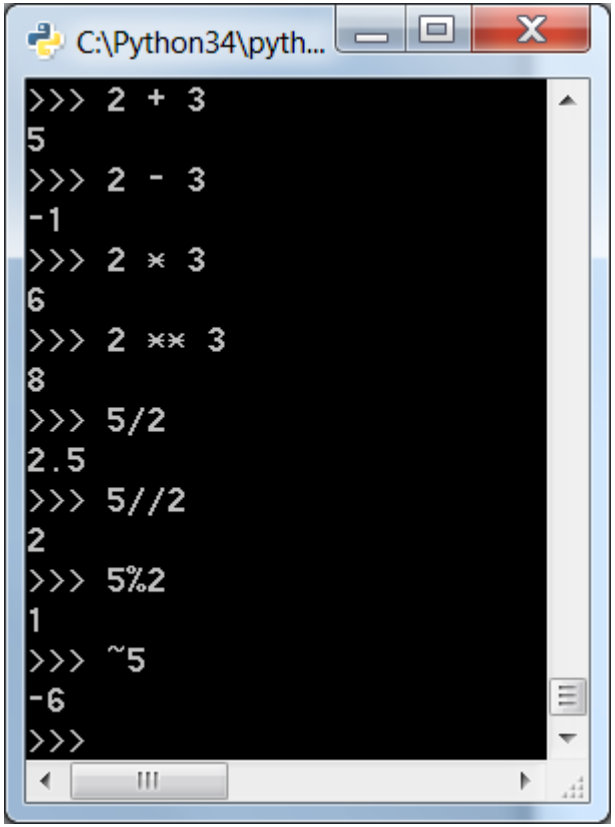
Pour connaître le type d'une donnée ou le type de la valeur d'une variable, il suffit d'utiliser la fonction ***type().***

https://courspython.com/types.html

Version 1.5
Copyright : Yantra Technologies 2004-2022

3 - Les Notions de Base -> Opérateurs arithmétiques

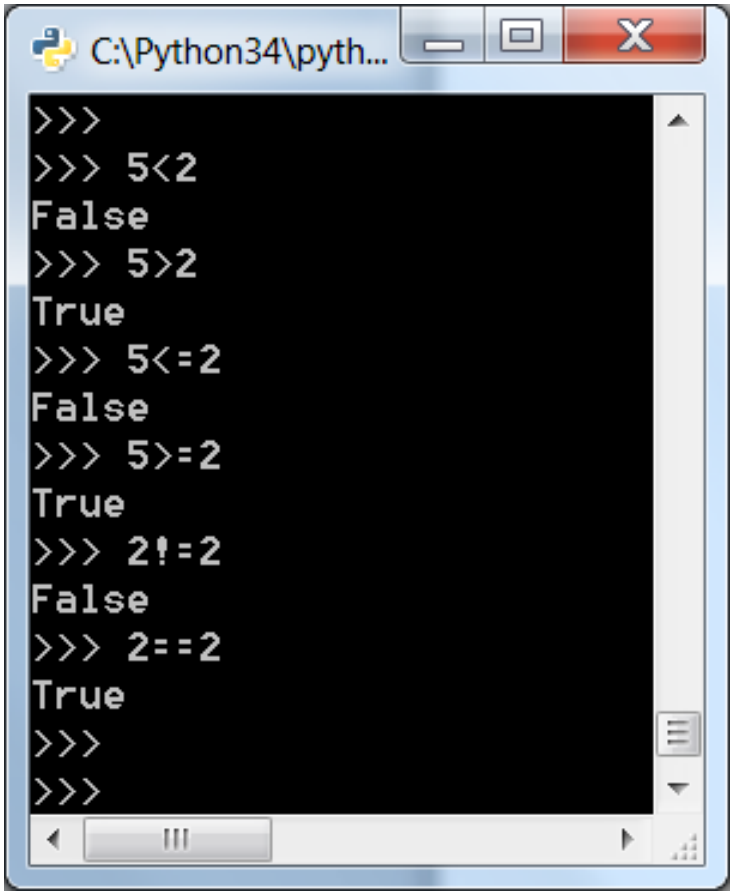
+	-	*	**	/	//	%	~
			puissance	Division réel	Division entier	modulo	~x = (-x)-1



Version 1.5
Copyright : Yantra Technologies 2004-2022

3 - Les Notions de Base -> Opérateurs logiques

<	>	<=	>=	==	!=	not	or	and



Version 1.5
Copyright : Yantra Technologies 2004-2022

3 - Les Notions de Base -> Opérateurs binaires

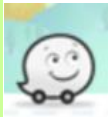
&	|	^	<<	>>

C:\Python34\python.exe

```
>>>
>>>
>>> 5&2  #and 101 010 000
0
>>> 5&6  #or 101 110 100
4
>>> 7|5  #ou 7|5 111 101 111
7
>>> 7^5  #xor 111 101 010
2
>>> 2<<2 #decalage a gauche 0010 1000
8
>>> 8>>2 #decalge a droite 1000 0010
2
>>>
```


Version 1.5
Copyright : Yantra Technologies 2004-2022

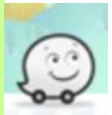
3 - Les Notions de Base -> Instructions



- Définition des fonctions
- Fonctions
- Fonctions lambda
- Classe
- Instruction vide / Suppression
- Instruction conditionnelle : if (): else:
- Interruption
- Itérations : for & while
- Rupture de séquence => break, continue, return
- Générateur

Version 1.5
Copyright : Yantra Technologies 2004-2022

3 - Les Notions de Base -> Instructions : Définition des fonctions



def <Nom Fonction > ([parametres]) :
<tabulation> instructions

```
>>> def bonjour_5():
...     i = 0
...     while i < 5:
...         print("Bonjour ",i)
...         i = i + 1
...
>>> bonjour_5()
Bonjour  0
Bonjour  1
Bonjour  2
Bonjour  3
Bonjour  4
>>> bonjour_5()
Bonjour  0
Bonjour  1
Bonjour  2
Bonjour  3
Bonjour  4
>>>
```

```
>>> def bonjour(val):
...     i = 0
...     while i < val:
...         print("Bonjour ",i)
...         i = i + 1
...
>>> bonjour(4)
Bonjour  0
Bonjour  1
Bonjour  2
Bonjour  3
>>> bonjour(5)
Bonjour  0
Bonjour  1
Bonjour  2
Bonjour  3
Bonjour  4
```

https://courspython.com/fonctions.html

Version 1.5
Copyright : Yantra Technologies 2004-2022

3 - Les Notions de Base -> Instructions : Définition des fonctions

```
>>> def message(mess,val):
...     i = 0
...     while i < val:
...         print(mess ,i)
...         i = i + 1
...
>>> message("Bonjour",4)
Bonjour 0
Bonjour 1
Bonjour 2
Bonjour 3
>>>
>>> message("Hello",5)
Hello 0
Hello 1
Hello 2
Hello 3
Hello 4
```

```
>>> def message(mess : str,val :int ):
...     i = 0
...     while i < val:
...         print(mess ,i)
...         i = i + 1
...
>>> message("Bonjour",4)
Bonjour 0
Bonjour 1
Bonjour 2
Bonjour 3
>>>
>>> message("Hello",5)
Hello 0
Hello 1
Hello 2
Hello 3
Hello 4
```

Nouveau dans la version 3.5
-> le typage

Version 1.5
Copyright : Yantra Technologies 2004-2022

3 - Les Notions de Base -> Instructions : Définition des fonctions

```
>>>
>>> def message(mess : str,val :int ):
...     i = 0
...     while i < val:
...         print(mess ,i)
...         i = i + 1
...
>>>
>>> information=message
>>>
>>> information("news ",1)
news  0
>>>
>>> print( type ( information ) )
<class 'function'>
>>> print ( type ( message ) )
<class 'function'>
>>>
```


Version 1.5
Copyright : Yantra Technologies 2004-2022

3 - Les Notions de Base -> Instructions : Définition des fonctions

```
>>> def plus(a, b) :
...     res = a + b
...     return res
...
>>> print (plus(2.,3.))
5.0
>>>
```

```
>>>
>>> def plus_type(a:float, b:float) -> float:
...     res = a + b
...     return res
...
>>> print (plus_type(2.,3.))
5.0
>>>
```


Version 1.5
Copyright : Yantra Technologies 2004-2022

3 - Les Notions de Base -> Instructions : Fonction

C:\Python34\python.exe

```
>>> def afficher_plus(a,b) :
...     def plus(a,b) :
...             return a + b
...     print("%.1f"%a," + %.1f"%b,"= %.1f "%plus(a,b)
...
>>>
... afficher_plus(1,2)
1.0  + 2.0 = 3.0
>>>
>>>
>>>
>>>
>>>
```


Version 1.5
Copyright : Yantra Technologies 2004-2022

3 - Les Notions de Base -> Instructions : Fonctions lambda

lambda <paramètres> : <instructions>

C:\Python34\python.exe

```
>>> f = lambda a,b,x : a*x+b
>>>
>>> f(1,1,3)
4
>>>
>>> def afficher_puissance(a,b) :
...     print("%.1f"%a," ^ %.1f"%b,"= %.1f "% ( lambda x,p : x**p )(a,
b) )
...
>>> afficher_puissance(2,4)
2.0  ^ 4.0 = 16.0
>>>
>>>
>>>
```


Version 1.5
Copyright : Yantra Technologies 2004-2022

3 - Les Notions de Base -> Instructions : Instruction vide / Suppression

C:\Python34\python.exe

```
>>> def fonction_vide() :
...     pass
...
>>> class Classe_vide() :
...     pass
...
>>> print (fonction_vide())
None
>>>
```

C:\Python34\python.exe

```
>>> a=5
>>> print(a)
5
>>> del a
>>> print(a)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'a' is not defined
>>>
```


Version 1.5
Copyright : Yantra Technologies 2004-2022

3 - Les Notions de Base -> Instructions : Instruction conditionnelle : if ():else :

```
C:\Python34\python.exe
>>> def negatif(val) :
...     if ( val < 0 ) :
...             return True
...     return False
...
>>> negatif(5)
False
>>> negatif(-5)
True
>>>
```

If <test> :
<tabulation> instructions
else :
<tabulation> instructions

```
C:\Python34\python.exe
>>>
>>> def print_negatif(val):
...     if ( negatif(val) ):
...             print("Le chiffre %.1f est negatif"%val)
...     else :
...             print("Le chiffre %.1f n'est pas negatif"%val)
...     pass
...
>>> print_negatif(5)
Le chiffre 5.0 n'est pas negatif
>>>
>>> print_negatif(-5)
Le chiffre -5.0 est negatif
>>>
```


Version 1.5
Copyright : Yantra Technologies 2004-2022

3 - Les Notions de Base -> Instructions : Instruction conditionnelle : if ():else:

If <test> :
<tabulation> instructions
elif <test>:
<tabulation> instructions
else :
<tabulation> instructions

C:\Python34\python.exe

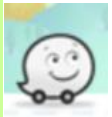
```
>>>
>>> def negatif(val) :
...     if ( val < 0 ) :
...             return True
...     return False
...
>>> def positif(val) :
...     if ( val > 0 ) :
...             return True
...     return False
...
>>> def print_signe(val):
...     if ( negatif(val) ):
...             print("Le chiffre %.1f est negatif"%val)
...     elif ( positif(val) ) :
...             print("Le chiffre %.1f est positif"%val)
...     else :
...             print("Le chiffre est egal a 0")
...     pass
...
>>> print_signe(5)
Le chiffre 5.0 est positif
>>>
>>> print_signe(-5)
Le chiffre -5.0 est negatif
>>>
>>> print_signe(0.0)
Le chiffre est egal a 0
>>>
```


Version 1.5
Copyright : Yantra Technologies 2004-2022

Match case (depuis la version 3.10)

Version 1.5
Copyright : Yantra Technologies 2004-2022

3 - Les Notions de Base -> Instructions : Interruption



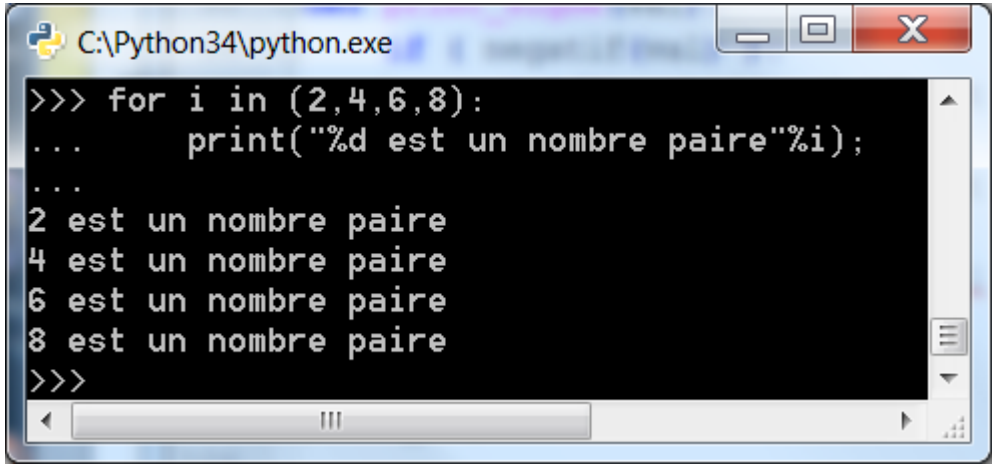
- **Bloc conditionnel *if* :**
  - ***Return[valeur]*** : permet de quitter immédiatement la fonction en cours.

Python 3 : page 99

- **Bloc non conditionnel *while, for* :**
  - ***break*** : permet de passer à l'instruction suivant l'instruction *while*, *for* la plus imbriquée.
  - ***continue*** : saute directement à la dernière ligne de l'instruction *while* ou *for* la plus imbriquée.

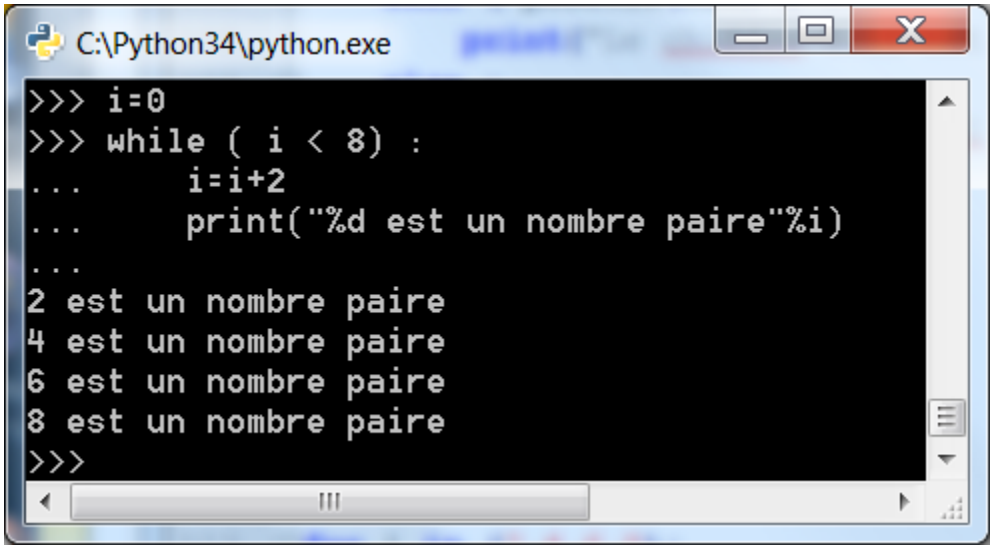
Version 1.5
Copyright : Yantra Technologies 2004-2022

3 - Les Notions de Base -> Instructions : Itérations : for & while



for <variables> in <liste>:
<tabulation> instructions

while test :
<tabulation> instructions



Version 1.5
Copyright : Yantra Technologies 2004-2022

3 - Les Notions de Base -> Instructions : Rupture de séquence => break, continue, return

C:\Python34\python.exe

```
>>> for letter in 'Le Python':
...     if letter == 'y':
...             continue
...     print ('Current Letter : %c'%letter)
...
Current Letter : L
Current Letter : e
Current Letter :
Current Letter : P
Current Letter : t
Current Letter : h
Current Letter : o
Current Letter : n
>>> var = 11
>>> while var > 0:
...     var = var -1
...     if var%2 == 1:
...             continue
...     print ('Valeur courant :%d'%var)
...
Valeur courant :10
Valeur courant :8
Valeur courant :6
Valeur courant :4
Valeur courant :2
Valeur courant :0
>>>
```

C:\Python34\python.exe

```
>>> for letter in 'Le Python':
...     if letter == 'y':
...             break
...     print ('Current Letter : %c'%letter)
...
Current Letter : L
Current Letter : e
Current Letter :
Current Letter : P
>>> var = 11
>>> while var > 0:
...     var = var -1
...     if var%2 == 1:
...             break
...     print ('Valeur courant :%d'%var)
...
Valeur courant :10
>>>
```


Version 1.5
Copyright : Yantra Technologies 2004-2022

3 - Les Notions de Base -> Instructions : Rupture de séquence => break, continue, return

C:\Python34\python.exe

```
>>> def positif(val) :
...     if ( val > 0 ) :
...             return True
...     return False
...
>>> positif(2)
True
>>>
```

C:\Python34\python.exe

```
>>> for num in range(1,20):
...     for i in range(2,num):
...         if num%i == 0:
...             j=num/i
...             print ('%d equals %d * %d' % (num,i,j))
...             break
...     else:
...         print ("%d est un nombre premier"%num)
...
1 est un nombre premier
2 est un nombre premier
3 est un nombre premier
4 equals 2 * 2
5 est un nombre premier
6 equals 2 * 3
7 est un nombre premier
8 equals 2 * 4
9 equals 3 * 3
10 equals 2 * 5
11 est un nombre premier
12 equals 2 * 6
13 est un nombre premier
14 equals 2 * 7
15 equals 3 * 5
16 equals 2 * 8
17 est un nombre premier
18 equals 2 * 9
19 est un nombre premier
>>>
```


Version 1.5
Copyright : Yantra Technologies 2004-2022

Les itérateurs et les générateurs python

Un itérateur est une sorte de curseur qui a pour mission de se déplacer dans une séquence d'objets.

L'itérateur permet de parcourir chaque object d'une séquence sans se préoccuper de la structure interne.

```
>>> liste = [1,2,3,4,5,6,7,8,9,10]
>>> for x in liste:
...     print(x)
...
1
2
3
4
5
6
7
8
9
10
```

https://python.doctor/page-iterateurs-iterator-generateur-generator-python

Version 1.5
Copyright : Yantra Technologies 2004-2022

Les générateurs permettent de créer plus facilement des itérateurs.

```
def generateur():
    yield "a"
    yield "b"
    yield "c"
```

```
>>> i=generateur()
>>> for v in i:
...     print(v)
...
a
b
c
```

```
def generateur(n):
    for i in range(n):
        if i == 5:
            print "Quoi déjà 5eme tour?"
        yield i+1
```

```
>>> i=generateur(10)
>>> for v in i:
...     print(v)
...
1
2
3
4
5
Quoi déjà 5eme tour?
6
7
8
9
10
```

https://python.doctor/page-iterateurs-iterator-generateur-generator-python

Version 1.5
Copyright : Yantra Technologies 2004-2022

Les itérateurs et les générateurs python

C:\Python34\python.exe

```
>>> def creerGenerateur(val) :
...     mylist = range(val)
...     for i in mylist:
...         yield (i,i*i)
...
>>> generateur = creerGenerateur(5)
>>>
>>> print(generateur)
<generator object creerGenerateur at 0x0000000002657EE8>
>>>
>>> for i in generateur:
...     print(i)
...
(0, 0)
(1, 1)
(2, 4)
(3, 9)
(4, 16)
```

https://realpython.com/introduction-to-python-generators/

Version 1.5
Copyright : Yantra Technologies 2004-2022

Les itérateurs et les générateurs python

```
>>> def creerGenerateur(val):
...     mylist=range(val)
...     for i in mylist:
...         yield(i,i*i)
...
>>> generateur = creerGenerateur(5)
>>>
>>> while(True):
...     print(next(generateur))
...
(0, 0)
(1, 1)
(2, 4)
(3, 9)
(4, 16)
Traceback (most recent call last):
  File "<stdin>", line 2, in <module>
StopIteration
>>>
```

```
>>> def creerGenerateur(val):
...     mylist=range(val)
...     for i in mylist:
...         yield(i,i*i)
...
>>> generateur = creerGenerateur(5)
>>>
>>> while(True):
...     try:
...         print(next(generateur))
...     except StopIteration:
...         print("Iteration terminer")
...         break
...
(0, 0)
(1, 1)
(2, 4)
(3, 9)
(4, 16)
Iteration terminer
>>>
```


Version 1.5
Copyright : Yantra Technologies 2004-2022

Les itérateurs et les générateurs python

générateur Tuple

```
C:\Python34\python.exe
>>> g = ( i**2 for i in range(10) )
>>>
>>> for j in g :
...     print (j)
...
0
1
4
9
16
25
36
49
64
81
>>>
```

générateur Liste

```
C:\Python34\python.exe
>>> k = [ i**2 for i in range(10) ]
>>> k
[0, 1, 4, 9, 16, 25, 36, 49, 64, 81]
>>>
```


Version 1.5
Copyright : Yantra Technologies 2004-2022

Les itérateurs et les générateurs python

générateur Ensemble

```
>>> multipleDe2 = { i*2 for i in range(10) }
>>>
>>> for i in multipleDe2:
...     print(i)
...
0
2
4
6
8
10
12
14
16
18
```

```
>>> multipleDe2 = { str(i*2) for i in range(10) }
>>>
>>> for i in multipleDe2:
...     print(i)
...
18
4
14
12
8
10
6
2
16
0
>>>
```

générateur Dictionnaire

```
>>> dictionnaire = { chr(i):i for i in range (65,91) }
>>>
>>> print(dictionnaire)
{'A': 65, 'B': 66, 'C': 67, 'D': 68, 'E': 69, 'F': 70, 'G': 71, 'H': 72, '
I': 73, 'J': 74, 'K': 75, 'L': 76, 'M': 77, 'N': 78, 'O': 79, 'P': 80, '
Q': 81, 'R': 82, 'S': 83, 'T': 84, 'U': 85, 'V': 86, 'W': 87, 'X': 88, 'Y
': 89, 'Z': 90}
>>>
>>> for i in dictionnaire:
...     print(i,dictionnaire[i])
...
A 65
B 66
C 67
D 68
E 69
F 70
G 71
H 72
I 73
J 74
K 75
L 76
M 77
N 78
O 79
P 80
Q 81
R 82
S 83
T 84
U 85
V 86
W 87
X 88
Y 89
Z 90
>>>
```


Version 1.5
Copyright : Yantra Technologies 2004-2022

3 - Les Notions de Base -> Gestion des exceptions

try :
<tabulation> instructions
except:
<tabulation> instruction

```
>>>
>>> while True:
...         x = int(input("Entrer un entier : "))
...         break
...
Entrer un entier : David
Traceback (most recent call last):
  File "<stdin>", line 2, in <module>
ValueError: invalid literal for int() with base 10: 'David'
>>>
```

```
>>> while True:
...     try:
...         x = int(input("Entrer un entier : "))
...         break
...     except ValueError:
...         print("Erreur la valeur n'est pas un entier, veuiller recomencer")
...
...
Entrer un entier : David
Erreur la valeur n'est pas un entier, veuiller recomencer
Entrer un entier : 5
>>>
```

https://docs.python.org/fr/3/tutorial/errors.html

Version 1.5
Copyright : Yantra Technologies 2004-2022

3 - Les Notions de Base -> Gestion des exceptions

C:\Python34\python.exe

```
>>> cp=0
>>> try :
...     while (True):
...             next(generateur)
...             cp+=1
... except:
...     print ("le generateur contenait %d element"%cp)
...
(0, 0)
(1, 1)
(2, 4)
(3, 9)
(4, 16)
le generateur contenait 5 element
>>>
```


Version 1.5
Copyright : Yantra Technologies 2004-2022

3 - Les Notions de Base -> Gestion des exceptions

C:\Python34\python.exe

```
>>> while (True):
...     a=input("Rentrer un chiffre : ")
...     if ( a== "stop" or a == "halt"):
...             break;
...     try :
...             res=float(a)
...             print ("le chiffre rentrer est un chiffre"%res)
...     except:
...             print ("le chiffre rentrer '%s' n'est pas un chiffre "
%a)
...     else:
...             print ( "%f au carre = %f"%(res,res**2))
...     finally:
...             print ("pour sortir taper stop ou halt")
...
Rentrer un chiffre : 5
le chiffre rentrer '5' n'est pas un chiffre
pour sortir taper stop ou halt
Rentrer un chiffre : p
le chiffre rentrer 'p' n'est pas un chiffre
pour sortir taper stop ou halt
Rentrer un chiffre : stop
>>>
```


Version 1.5
Copyright : Yantra Technologies 2004-2022

3 - Les Notions de Base -> Gestion des exceptions : raise

C:\Python34\python.exe

```
>>> def racine(val):
...     if val < 0:
...             raise Exception('Fonction racine : Valeur Negative', v
al)
...     return math.sqrt(val)
...
>>> try:
...     racine(-5)
... except Exception as e:
...     print(e)
...
('Fonction racine : Valeur Negative', -5)
>>> try:
...     racine(5)
... except Exception as e:
...     print(e)
...
2.23606797749979
>>>
```

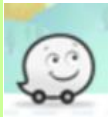

Version 1.5
Copyright : Yantra Technologies 2004-2022

3 - Les Notions de Base -> Divers

- Import
- Global
- Help
- Eval, exec

Copyright : Yantra Technologies 2004-2022

3 - Les Notions de Base -> Divers : Import



Un module est un fichier ayant pour extension .py contenant des définitions de constantes et fonctions.

Tout programmeur python peut réaliser un module.

Importer un module permet d'utiliser ses constantes et fonctions.

https://docs.python.org/fr/3.5/tutorial/modules.html
http://sametmax.com/les-imports-en-python/
https://courspython.com/modules.html

```
IndentationError: unexpected indent
>>>
>>> from math import sqrt
>>> sqrt(2)
1.4142135623730951
>>> from math import pi, sin
>>> pi
3.141592653589793
>>> from math import *
>>> cos (pi)
-1.0
>>>
>>> import math
>>> math.sqrt(3)
1.7320508075688772
>>>
>>> #alias
...
>>> import math as mymath
>>> mymath.cos(mymath.pi)
-1.0
>>>
```


Version 1.5
Copyright : Yantra Technologies 2004-2022

3 - Les Notions de Base -> Divers : global

C:\Python34\python.exe

```
>>> def f():
...     var = 2
...
>>> var
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'var' is not defined
>>> f()
>>> var
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'var' is not defined
>>>
>>> #-----------------------------------
...
>>> def f():
...     global var
...     var = 3
...
>>> var
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'var' is not defined
>>> f()
>>> var
3
>>>
```


Version 1.5
Copyright : Yantra Technologies 2004-2022

3 - Les Notions de Base -> Divers : help

C:\Python34\python.exe

```
>>>
>>>
>>> help("help")

Welcome to Python 3.4's help utility!

If this is your first time using Python, you should definitely check o
ut
the tutorial on the Internet at http://docs.python.org/3.4/tutorial/.

Enter the name of any module, keyword, or topic to get help on writing

Python programs and using Python modules.  To quit this help utility a
nd
return to the interpreter, just type "quit".

To get a list of available modules, keywords, symbols, or topics, type

"modules", "keywords", "symbols", or "topics".  Each module also comes

with a one-line summary of what it does; to list the modules whose nam
e
or summary contain a given string such as "spam", type "modules spam".


>>>
```


Version 1.5
Copyright : Yantra Technologies 2004-2022

3 - Les Notions de Base -> Divers : Eval, exec

C:\Python34\python.exe

```
>>> exec ('print ("Hello World")')
Hello World
>>>
>>> res=eval ( '2×3' )
>>> res
6
>>>
>>> eval  ('print ("Hello World")')
Hello World
>>>
>>> exec ( 'res=2×3' )
>>> res
6
```


Version 1.5
Copyright : Yantra Technologies 2004-2022

4 - Typage dynamique

- Primitive type et nature du type
- Typage python : espace global
- Typage python : espace local
- Notion de bloc
- Fonctions
- Fonctions paramètres extensibles
- Classe
- Module

Version 1.5
Copyright : Yantra Technologies 2004-2022

4 - Typage dynamique-> Primitive type et nature du type

C:\Pyth...

```
>>>
>>> a=[]
>>> type(a)
<class 'list'>
>>> t=type(a)
>>> t
<class 'list'>
>>> type(t)
<class 'type'>
>>>
```

type est une classe qui permet de connaitre le type d'une variable

C:\Python34\python.exe

```
>>> dir(type)
['__abstractmethods__', '__base__', '__bases__', '__basicsize__
all__', '__class__', '__delattr__', '__dict__', '__dictoffset__
ir__', '__doc__', '__eq__', '__flags__', '__format__', '__ge__'
tattribute__', '__gt__', '__hash__', '__init__', '__instanceche
'__itemsize__', '__le__', '__lt__', '__module__', '__mro__', '_
', '__ne__', '__new__', '__prepare__', '__qualname__', '__reduc
__reduce_ex__', '__repr__', '__setattr__', '__sizeof__', '__str
_subclasscheck__', '__subclasses__', '__subclasshook__', '__tex
ture__', '__weakrefoffset__', 'mro']
>>>
>>> t.__name__
'list'
>>> t.__bases__
(<class 'object'>,)
>>> t.__base__
<class 'object'>
>>>
```


Version 1.5
Copyright : Yantra Technologies 2004-2022

4 - Typage dynamique-> Primitive type et nature du type

C:\Python34\python.exe

```
>>> set(dir(type))-set(dir(object))
{'__call__', '__prepare__', '__abstractmethods__', '__subclasscheck__'
, '__dictoffset__', '__mro__', '__subclasses__', '__bases__', '__dict_
_', '__module__', '__qualname__', '__instancecheck__', '__base__', '__
basicsize__', '__text_signature__', '__weakrefoffset__', 'mro', '__ite
msize__', '__name__', '__flags__'}
>>>
```


Version 1.5
Copyright : Yantra Technologies 2004-2022

4 - Typage dynamique-> Typage python : espace global

globals() contient l'ensemble des variables déclarées et des imports

```
C:\Python34\pytho...
>>> c="nom"
>>> c=1
>>> c=1.4586655
>>> c=[]
>>>
```

```
C:\Python34\python.exe
>>>
>>> globals()
{'__builtins__': <module 'builtins' (built-in)>, '__spec__': None, '__
doc__': None, '__package__': None, '__loader__': <class '_frozen_impor
tlib.BuiltinImporter'>, '__name__': '__main__', 'c': [], 't': <class '
list'>, 'a': []}
>>>
>>>
```


Version 1.5
Copyright : Yantra Technologies 2004-2022

4 - Typage dynamique->
Typage python : espace global / local

C:\Python34\python.exe

```
>>> if True:
...     cor=5
...
>>> cor
5
>>>
>>> #probleme
...
>>> if False:
...     err=45
...
>>> err
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'err' is not defined
>>>
>>> if 'err' in globals().keys():
...     print(err)
...     del err
...
>>>
```


Version 1.5
Copyright : Yantra Technologies 2004-2022

4 - Typage dynamique-> Notion de bloc

C:\Python34\python.exe

```
>>> def f():
...     global var_global
...     var_global = 3
...     var_local  = 5
...     print("liste des variables local : ",locals())
...     print("liste des variables global : ",globals())
...
>>>
>>> f()
liste des variables local :  {'var_local': 5}
liste des variables global :  {'f': <function f at 0x0000000002A86048>
, '__builtins__': <module 'builtins' (built-in)>, '__doc__': None, 'va
r_global': 3, '__loader__': <class '_frozen_importlib.BuiltinImporter'
>, 'var': 5, 't': <class 'list'>, 'cor': 5, '__spec__': None, '__packa
ge__': None, 'c': [], '__name__': '__main__', 'a': []}
>>>
>>>
```


Version 1.5
Copyright : Yantra Technologies 2004-2022

4 - Typage dynamique-> Fonctions

C:\Python34\python.exe

```
>>>
>>> def f1():
...     pass
...
>>> set(dir(f1))-set(dir(object))
{'__kwdefaults__', '__call__', '__annotations__', '__code__', '__closu
re__', '__dict__', '__module__', '__qualname__', '__get__', '__globals
__', '__name__', '__defaults__'}
>>>
>>> f1.__module__
'__main__'
>>>
>>> f1.__name__
'f1'
>>> g1 = f1
>>>
>>> g1.__name__
'f1'
>>>
```


Version 1.5
Copyright : Yantra Technologies 2004-2022

4 - Typage dynamique-> Fonctions paramètres extensibles

C:\Python34\python.exe

```
>>> def f1(*args):
...     print("args = ",args)
...     for i in range(len(args)):
...             print ( "args[",i,"]",args[i])
...
>>> f1(1,2,3,4)
args =  (1, 2, 3, 4)
args[ 0 ] 1
args[ 1 ] 2
args[ 2 ] 3
args[ 3 ] 4
>>>
```


Version 1.5
Copyright : Yantra Technologies 2004-2022

4 - Typage dynamique-> Fonctions paramètres extensibles

C:\Python34\python.exe

```
>>> def f2(**kargs):
...     print("kargs = ",kargs)
...     for key, value in kargs.items():
...             print("kargs[",key,"]",value)
...
>>> f2(a=5,b=6)
kargs =  {'b': 6, 'a': 5}
kargs[ b ] 6
kargs[ a ] 5
>>>
>>>
>>>
```


Copyright : Yantra Technologies 2004-2022

4 - Typage dynamique-> Fonctions paramètres extensibles

C:\Python34\python.exe

```
>>> def f3(x,y=2,*args,**kwargs):
...     print(kwargs)
...     print(args)
...     locals().update(kwargs)
...
>>> f3(1,2,3,4,a=5,b=6)
{'b': 6, 'a': 5}
(3, 4)
>>> f3(*[1,2,3,4],**{'a':5,'b':6})
{'b': 6, 'a': 5}
(3, 4)
>>>
```


Version 1.5
Copyright : Yantra Technologies 2004-2022

3 - Les Notions de Base -> Instructions : Classe

C:\Python34\python.exe

```
>>> class Classe1 :
...     attribut1="Class1"
...     attribut2=24
...     def methode1(self):
...             return ( self.attribut1, self.attribut2 )
...     def methode2(self,att1,att2):
...             self.attribut1 = att1
...             self.attribut2 = att2
...             pass
...
>>> c = Classe1()
>>>
>>> c.methode1()
('Class1', 24)
>>>
>>> c.methode2("Instance a",25)
>>>
>>>
>>> c.methode1()
('Instance a', 25)
>>>
```


Version 1.5
Copyright : Yantra Technologies 2004-2022

4 - Typage dynamique -> Classe

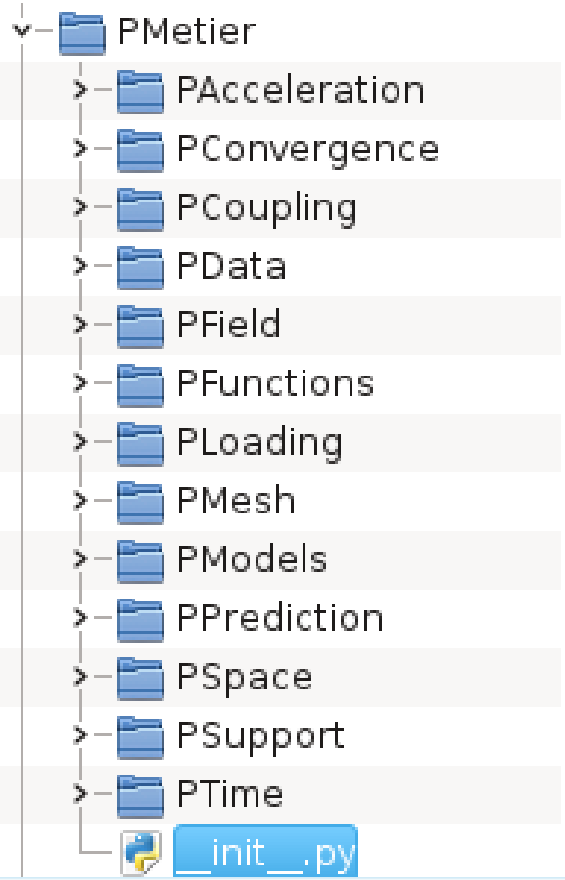
C:\Python34\python.exe

```
>>> class ModelA(object):
... #Debut block local
...     print("Debut bloc ",locals())
...
... #attribut
...     temperature=273.15
...     unite='K'
... #methode
...     def getTemperature(self):
...             return self.temperature
...
...     def getUnite(self):
...             return self.unite
...
...     print("Fin bloc ",locals())
... #Fin block local
...
Debut bloc  {'__module__': '__main__', '__qualname__': 'ModelA'}
Fin bloc  {'temperature': 273.15, 'getTemperature': <function ModelA.g
etTemperature at 0x0000000002A866A8>, 'unite': 'K', 'getUnite': <funct
ion ModelA.getUnite at 0x0000000002A86730>, '__module__': '__main__',
'__qualname__': 'ModelA'}
>>>
>>>
>>> a = ModelA()
>>> dir(a)
['__class__', '__delattr__', '__dict__', '__dir__', '__doc__', '__eq__
', '__format__', '__ge__', '__getattribute__', '__gt__', '__hash__', '
__init__', '__le__', '__lt__', '__module__', '__ne__', '__new__', '__r
educe__', '__reduce_ex__', '__repr__', '__setattr__', '__sizeof__', '_
_str__', '__subclasshook__', '__weakref__', 'getTemperature', 'getUnit
e', 'temperature', 'unite']
>>> a.unite
'K'
>>> a.getUnite()
'K'
>>>
```


Version 1.5
Copyright : Yantra Technologies 2004-2022

4 - Typage dynamique -> Module

https://docs.python.org/fr/3.5/tutorial/modules.html
http://sametmax.com/les-imports-en-python/
https://courspython.com/modules.html



__init__.py

```
"""Package contenant l'ensemble des classes relatives a PMetier.

"""

__all__ = ["PAcceleration","PConvergence","PCoupling","PData","PField","PFunctions","PLoading","PMesh","PModels","PPrediction","PSpace","PTime","PSupport"]
```


Version 1.5
Copyright : Yantra Technologies 2004-2022

4 - Typage dynamique -> Module

__init__.py

```
"""Package contenant l'ensemble des classes relatives a PSsupport
"""

__all__ = ["psupport"]
#import Pleiades.PMetier.PSupport.psupport
```


Copyright : Yantra Technologies 2004-2022

4 - Typage dynamique -> Module : reload

```
import imp
import test


imp.reload(test)
```

https://docs.python.org/3/library/imp.html

Copyright : Yantra Technologies 2004-2022

Yantra Technologies

www.yantra-technologies.com

Programmation Orientée Objet
Les Concepts

carole.grondein@yantra-technologies.com
david.palermo@yantra-technologies.com

1

Version 1.5
Copyright : Yantra Technologies 2004-2022

6 - Modèle objet

- Déclaration Classes
- Classes & instanciation
- Classe déclaration par prototype
- Classes & affectation méthode
- Méthodes de Classe
- Méthodes et attributs spéciaux
- Les méthodes de conteneur
- Héritage
- Surcharge des méthodes
- Propriétés
- Métaclasse

Version 1.5
Copyright : Yantra Technologies 2004-2022

6 - Modèle objet -> Déclaration Classes

```
class ModelA(object):
    """Description de ma classe : ModelA"""
#attribut
    temperature=273.15
    unite='K'
#methode
    def getTemperature(self):
        return self.temperature

    def getUnite(self):
        return self.unite
```

```
class ModelB(object):
    """Description de ma classe : ModelB"""
#attribut
    temperature=0.0
#methode
    def getTemperature(self):
        return self.temperature

    def getUnite(self):
        return "C"
```


Version 1.5
Copyright : Yantra Technologies 2004-2022

6 - Modèle objet -> Classes & instanciation

C:\Python34\python.exe

```
>>> import pmodel
>>> #acces documentation
...
... pmodel.ModelA.__doc__
'Description de ma classe : ModelA'
>>>
>>> #acces attribut
...
>>> pmodel.ModelA.temperature
273.15
>>>
>>> #acces methode
...
>>> pmodel.ModelA.getTemperature
<function ModelA.getTemperature at 0x0000000002A670D0>
>>>
>>> #instanciation
...
>>> m = pmodel.ModelA()
>>>
>>> #acces documentation
...
... m.__doc__
'Description de ma classe : ModelA'
>>>
>>> #acces attribut
...
>>> m.temperature
273.15
>>>
>>> #acces methode
...
>>> m.getTemperature()
273.15
>>> m.getUnite()
```


Version 1.5
Copyright : Yantra Technologies 2004-2022

6 - Modèle objet -> Classe déclaration par prototype

```
class ModelC(object):
    pass

ModelC.__doc__="""Description de ma classe : ModelC"""

ModelC.temperature=273.15
ModelC.unite='K'

def getTemperature(self):
    return self.temperature

ModelC.getTemperature=getTemperature

def getUnite(self):
    return self.unite

ModelC.getUnite=getUnite
```


Version 1.5
Copyright : Yantra Technologies 2004-2022

6 - Modèle objet -> Classes & affectation méthode

C:\Python34\python.exe

```
>>> import pmodel
>>>
>>> pmodel.ModelA.temperature
273.15
>>> pmodel.ModelA.getUnite
<function ModelA.getUnite at 0x00000000021A7158>
>>> pmodel.ModelB.temperature
0.0
>>> pmodel.ModelB.getUnite
<function ModelB.getUnite at 0x00000000021A7268>
>>>
>>> pmodel.ModelA.temperature=pmodel.ModelB.temperature
>>> pmodel.ModelA.getUnite=pmodel.ModelB.getUnite
>>>
>>> pmodel.ModelA.temperature
0.0
>>> pmodel.ModelA.getUnite
<function ModelB.getUnite at 0x00000000021A7268>
>>>
```


Version 1.5
Copyright : Yantra Technologies 2004-2022

6 - Modèle objet -> Méthodes de Classe

Une méthode de classe permet d'appeler une méthode de la classe directement

Une méthode statique a une fonction agrégée à la classe

```
class ModelA(object):
    """Description de ma classe : ModelA"""
#attribut
    temperature=273.15
    unite='K'
#methode
    def getTemperature(self):
        return self.temperature

    def getUnite(self):
        return self.unite
#methode de classe

    @classmethod
    def getNomClasse(cls,*args,**kwargs):
        return cls.__name__

#methode static

    @staticmethod
    def Creation(*args,**kwargs):
        print(kwargs)
        print(args)
        return ModelA()
```


Version 1.5
Copyright : Yantra Technologies 2004-2022

6 - Modèle objet -> Méthodes de Classe

C:\Python34\python.exe

```
>>> import pmodel
>>>
>>> pmodel.ModelA.getNomClasse()
'ModelA'
>>>
>>> pmodel.ModelA.Creation()
{}
()
<pmodel.ModelA object at 0x0000000002A94208>
>>>
```


Version 1.5
Copyright : Yantra Technologies 2004-2022

6 - Modèle objet -> Méthodes Protected de Classe

Example: Protected Attributes

Copy

```
class Student:
    _schoolName = 'XYZ School' # protected class attribute

    def __init__(self, name, age):
        self._name=name  # protected instance attribute
        self._age=age # protected instance attribute
```

Example: Access Protected Members

Copy

```
>>> std = Student("Swati")
>>> std.name
'Swati'
>>> std.name = 'Dipa'
>>> std.name
'Dipa'
>>> std._name # still accessible
```

https://www.tutorialsteacher.com/python/public-private-protected-modifiers

Version 1.5
Copyright : Yantra Technologies 2004-2022

6 - Modèle objet -> Méthodes Private de Classe

Example: Private Attributes

Copy

```
class Student:
    __schoolName = 'XYZ School' # private class attribute

    def __init__(self, name, age):
        self.__name=name  # private instance attribute
        self.__salary=age # private instance attribute
    def __display(self):  # private method
        print('This is private method.')
```

Example:

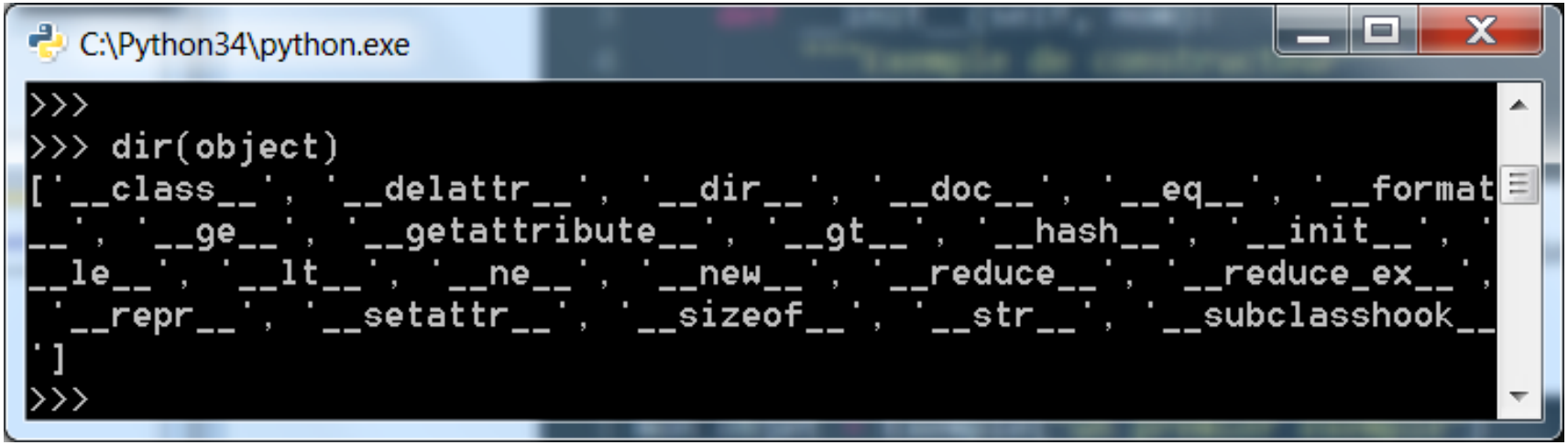
Copy

```
>>> std = Student("Bill", 25)
>>> std.__schoolName
AttributeError: 'Student' object has no attribute '__schoolName'
>>> std.__name
AttributeError: 'Student' object has no attribute '__name'
>>> std.__display()
AttributeError: 'Student' object has no attribute '__display'
```

https://www.tutorialsteacher.com/python/public-private-protected-modifiers

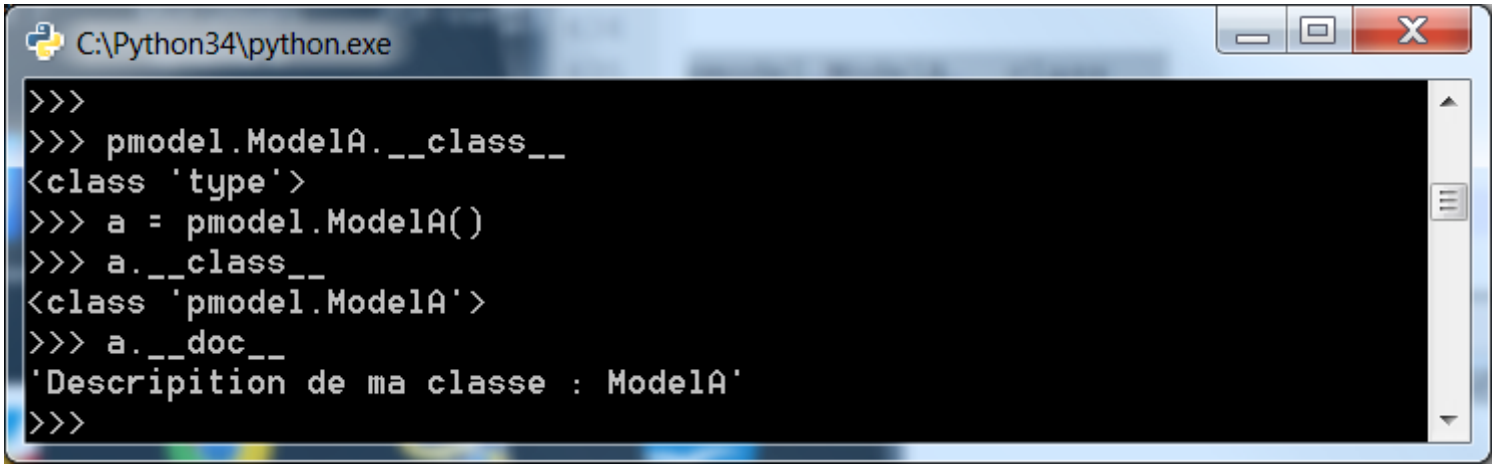
Version 1.5
Copyright : Yantra Technologies 2004-2022

6 - Modèle objet -> Méthodes et attribut s spéciaux



Copyright : Yantra Technologies 2004-2022

6 - Modèle objet -> Attributs spéciaux : __Class__ & __doc__



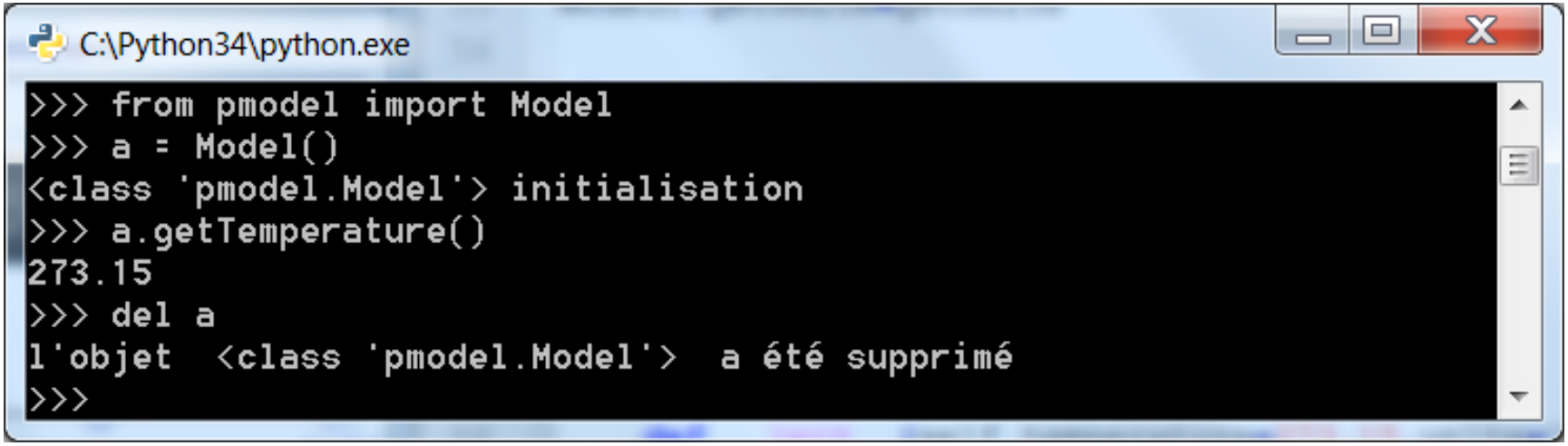
Version 1.5
Copyright : Yantra Technologies 2004-2022

6 - Modèle objet -> Méthodes spéciales : __init__ & __del__

```
class Model(object):
    """Description de ma classe : Model"""
#attribut
    temperature=None
    unite=None

#initialisation
    def __init__(self,temperature=273.15,unite="K"):
        print(self.__class__,"initialisation")
        self.temperature=temperature
        self.unite=unite
#supression
    def __del__(self):
        """Méthode appelée quand l'objet est supprimé"""
        print("l'objet ", self.__class__," a été supprimé")

#methode

    def getTemperature(self):
        return self.temperature

    def getUnite(self):
        return self.unite
```


Version 1.5
Copyright : Yantra Technologies 2004-2022

6 - Modèle objet -> Méthodes spéciales : __init__ & __del__

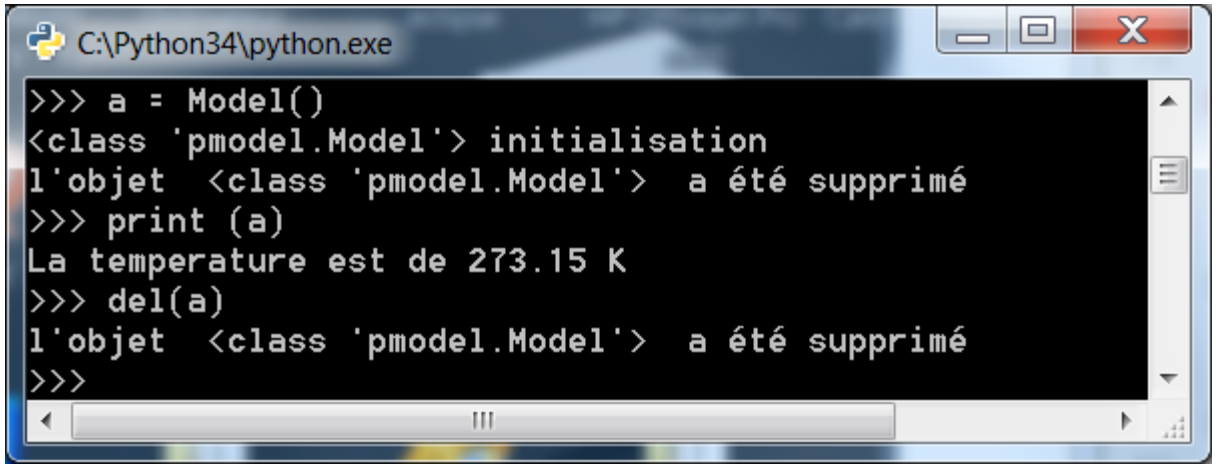


Version 1.5
Copyright : Yantra Technologies 2004-2022

6 - Modèle objet -> Méthodes spéciales: __str__

Méthode d'affichage

```
def __str__(self):
    """Méthode permettant d'afficher notre objet"""
    return "La temperature est de {} {}".format(self.temperature,self.unite)
```



Version 1.5
Copyright : Yantra Technologies 2004-2022

6 - Modèle objet -> Méthodes spéciales: __setattr__

```
def __setattr__(self, nom_attr, val_attr):
    """Méthode appelée quand on fait objet.nom_attr = val_attr."""
    if not hasattr(self, nom_attr):
        raise Exception ("L'attribut {} n'existe pas").format(nom_attr)
    print("Affecter l'attribut {} = {}".format(nom_attr,val_attr))
    object.__setattr__(self, nom_attr, val_attr)
```

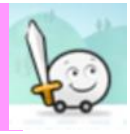
C:\Python34\python.exe

```
>>>
>>>
>>> a = Model()
<class '__main__.Model'> initialisation
Affecter l'attribut temperature = 273.15
Affecter l'attribut unite = K
>>> a.temperature=1000.
Affecter l'attribut temperature = 1000.0
>>> str(a)
'La temperature est de 1000.0 K'
>>> del a
l'objet  <class '__main__.Model'>  a été supprimé
>>>
```

https://infoforall.fr/python/python-act230.html
https://www.it-swarm-fr.com/fr/python/utiliser-setattr-dans-python/942229718/

Version 1.5
Copyright : Yantra Technologies 2004-2022

6 - Modèle objet -> Méthodes spéciales: __getattr__



```
def __getattr__(self, nom):
    """Si Python ne trouve pas l'attribut nommé nom, il appelle cette méthode. """
    print("L'attribut {} n'existe pas!".format(nom))
```

C:\Python34\python.exe

```
>>> a = Model()
<class '__main__.Model'> initialisation
Affecter l'attribut temperature = 273.15
Affecter l'attribut unite = K
>>> a.temperature=1000.
Affecter l'attribut temperature = 1000.0
>>> str(a)
'La temperature est de 1000.0 K'
>>> del a
l'objet  <class '__main__.Model'>  a été supprimé
```


Version 1.5
Copyright : Yantra Technologies 2004-2022

6 - Modèle objet -> Méthodes spéciales: __delattr__

```
def __delattr__(self, nom_attr):
    """On ne peut supprimer d'attribut, on lève l'exception AttributeError"""
    raise AttributeError("Vous ne pouvez supprimer aucun attribut de cette classe")
```

C:\Python34\python.exe

```
>>>
>>> a = Model()
<class '__main__.Model'> initialisation
Affecter l'attribut temperature = 273.15
Affecter l'attribut unite = K
>>> del a.unite
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "<stdin>", line 35, in __delattr__
AttributeError: Vous ne pouvez supprimer aucun attribut de cette class
e
>>> del a
>>>
```


Version 1.5
Copyright : Yantra Technologies 2004-2022

6 - Modèle objet -> Méthodes spéciales : set_attr__,get_attr__,__delattr__

```
# On crée une instance de la classe Model
objet = Model()

# Semblable à objet.temperature
getattr(objet, "temperature")

# objet.temperature = 1000. ou objet.__setattr__("temperature", 1000.)
setattr(objet, "temperature", 1000.)

# del objet.nom ou objet.__delattr__("temperature")
delattr(objet, "temperature")

# Renvoie True si l'attribut "temperature" existe, False sinon
hasattr(objet, "temperature")
```


Version 1.5
Copyright : Yantra Technologies 2004-2022

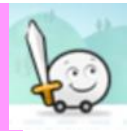
6 - Modèle objet -> Méthodes spéciales

Opérateur	Méthode spéciale	Résumé
==	def __eq__(self, bjet_a_comparer):	Opérateur d'égalité (*equal*). Renvoie True si self et objet_a_comparer sont égaux, False sinon.
!=	def __ne__(self, objet_a_comparer):	Différent de (*non equal*). Renvoie True si self et objet_a_comparer sont différents, False sinon.
>	def __gt__(self, objet_a_comparer):	Teste si self est strictement supérieur (*greather than*) à objet_a_comparer.
>=	def __ge__(self, objet_a_comparer):	Teste si self est supérieur ou égal (*greater or equal*) à objet_a_comparer.
<	def __lt__(self, objet_a_comparer):	Teste si self est strictement inférieur (*lower than*) à objet_a_comparer.
<=	def __le__(self, objet_a_comparer):	Teste si self est inférieur ou égal (*lower or equal*) à objet_a_comparer.

http://fr.openclassrooms.com/informatique/cours/apprenez-a-programmer-en-python/les-methodes-speciales-1

Version 1.5
Copyright : Yantra Technologies 2004-2022

6 - Modèle objet -> Méthodes spéciales



```
def __ge__(self,objet):
    if (self.unite != objet.unite) : raise Exception("Unite incompatible")
    if (self.temperature >= objet.temperature ):
        return True;
    return False;

def __gt__(self,objet):
    if (self.unite != objet.unite) : raise Exception("Unite incompatible")
    if (self.temperature > objet.temperature ):
        return True;
    return False;

def __le__(self,objet):
    if (self.unite != objet.unite) : raise Exception("Unite incompatible")
    if (self.temperature <= objet.temperature ):
        return True;
    return False;

def __lt__(self,objet):
    if (self.unite != objet.unite) : raise Exception("Unite incompatible")
    if (self.temperature < objet.temperature ):
        return True;
    return False;
```


Version 1.5
Copyright : Yantra Technologies 2004-2022

6 - Modèle objet -> Méthodes spéciales

C:\Python34\python.exe

```
>>>
>>> a = Model(125.,"K")
<class '__main__.Model'> initialisation
Affecter l'attribut temperature = 125.0
Affecter l'attribut unite = K
l'objet  <class '__main__.Model'>  a été supprimé
>>> b = Model(250.,"K")
<class '__main__.Model'> initialisation
Affecter l'attribut temperature = 250.0
Affecter l'attribut unite = K
l'objet  <class '__main__.Model'>  a été supprimé
>>> c = Model(125.,"C")
<class '__main__.Model'> initialisation
Affecter l'attribut temperature = 125.0
Affecter l'attribut unite = C
l'objet  <class '__main__.Model'>  a été supprimé
>>>
>>> a < b
True
>>> b <= b
True
>>> c > a
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "<stdin>", line 44, in __gt__
Exception: Unite incompatible
>>> c >= c
True
>>>
```


Version 1.5
Copyright : Yantra Technologies 2004-2022

6 - Modèle objet -> Les méthodes de conteneur

objet[item]	def __getitem__(self, index):
objet[item] = valeur	def __setitem__(self, index,valeur):
del objet[item]	def __delitem__(self, index):

Copyright : Yantra Technologies 2004-2022

6 - Modèle objet -> Héritage

```
class ModelHerite(Model):
    pass
```

C:\Python34\python.exe

```
>>> b=ModelHerite()
<class '__main__.ModelHerite'> initialisation
Affecter l'attribut temperature = 273.15
Affecter l'attribut unite = K
>>> dir(ModelHerite)
['__class__', '__del__', '__delattr__', '__dict__', '__dir__', '__doc__', '__eq__', '__format__', '__ge__', '__getattr__', '__getattribute__', '__gt__', '__hash__', '__init__', '__le__', '__lt__', '__module__', '__ne__', '__new__', '__reduce__', '__reduce_ex__', '__repr__', '__setattr__', '__sizeof__', '__str__', '__subclasshook__', '__weakref__', 'getTemperature', 'getUnite', 'temperature', 'unite']
>>>
```


Version 1.5
Copyright : Yantra Technologies 2004-2022

6 - Modèle objet -> Surcharge des méthodes

C:\Python34\python.exe

```
>>> class Point1D      :
...     x=0.0
...     def __init__(self,x):
...             self.x=x
...     def distance(self,point):
...             return math.sqrt( (point.x - self.x)**2 )
...
>>> p1= Point1D(1.0)
>>> p2= Point1D(5.0)
>>> p1.distance(p2)
4.0
>>>
```

C:\Python34\python.exe

```
>>> class Point2D(Point1D)      :
...     y=0.0
...     def __init__(self,x,y):
...             self.y=y
...             Point1D.__init__(self,x)
...     def distance(self,point):
...             return math.sqrt( Point1D.distance(self,point)**2 + (p
oint.y - self.y)**2 )
...
>>> p21= Point2D(0.,0.0)
>>> p22= Point2D(1.0,1.0)
>>> p21.distance(p22)
1.4142135623730951
>>>
```


Version 1.5
Copyright : Yantra Technologies 2004-2022

6 - Modèle objet -> Propriétés

Un propriété s'utilise comme un attribut de classe

C:\Python34\python.exe

```
>>>
>>>
>>>
>>> class Propriete:
...     notes=None
...     def __init__(self,*notes):
...             self.notes=list(notes)
...     @property
...     def moyenne(self):
...             if ( len(self.notes) != 0):
...                     return sum(self.notes)/len(self.notes)
...             return 0
...
>>>
>>> p = Propriete()
>>> p.moyenne
0
>>> p = Propriete(1,2,3,4,5)
>>> p.moyenne
3.0
>>>
```


Version 1.5
Copyright : Yantra Technologies 2004-2022

6 - Modèle objet -> Métaclasse

Hérite de type

C:\Python34\python.exe

```
>>> class MaMetaClass(type):
...     def __new__(meta, name, bases, dct):
...             print ('< metaclasse >')
...             print("MaMetaClass.__new__ : Allocationde la memoire p
our la classe ", name)
...             print(meta)
...             print(bases)
...             print(dct)
...             print('</ metaclasse >')
...             return type.__new__(meta, name, bases, dct)
...     def __init__(cls, name, bases, dct):
...             print ('< metaclasse >')
...             print ("MaMetaClass.__init__ : Initisalisation de la c
lasse class", name)
...             print (cls)
...             print (bases)
...             print (dct)
...             print('</ metaclasse >')
...             type.__init__(cls,name, bases, dct)
...     def __call__(cls, *args, **kwds):
...             print ('< metaclasse >')
...             print ('MaMetaClass.__call__ de ', str(cls))
...             print ('MaMetaClass.__call__ *args=', str(args))
...             print ('MaMetaClass.__call__ **kwds=', str(kwds))
...             print('</ metaclasse >')
...             return (type.__call__(cls, *args, **kwds))
```


Version 1.5
Copyright : Yantra Technologies 2004-2022

6 - Modèle objet -> Métaclasse

C:\Python34\python.exe

```
>>> class MaClass(metaclass=MaMetaClass):
...     def __new__(mcs, *args,**kwargs):
...             print ('< classe >')
...             print ("MaClass.__new__ avec : args=",str(args),", kwa
rgs=",str(kwargs))
...             print('</ classe >')
...             return object.__new__(mcs)
...     def __init__(self,*args,**kwargs):
...             print ('< classe >')
...             print ("MaClass.__init__  avec : args=",str(args),", k
wargs=",str(kwargs))
...             print('</ classe >')
...             return object.__init__(self)
...
```


Version 1.5
Copyright : Yantra Technologies 2004-2022

6 - Modèle objet -> Métaclasse

```
< metaclasse >
MaMetaClass.__new__ : Allocationde la memoire pour la classe  MaClass
<class '__main__.MaMetaClass'>
()
{'__new__': <function MaClass.__new__ at 0x00000000022A7378>, '__init_
_': <function MaClass.__init__ at 0x00000000022A7400>, '__module__': '
__main__', '__qualname__': 'MaClass'}
</ metaclasse >
< metaclasse >
MaMetaClass.__init__ : Initisalisation de la classe class MaClass
<class '__main__.MaClass'>
()
{'__new__': <function MaClass.__new__ at 0x00000000022A7378>, '__init_
_': <function MaClass.__init__ at 0x00000000022A7400>, '__module__': '
__main__', '__qualname__': 'MaClass'}
</ metaclasse >
>>>
```


Version 1.5
Copyright : Yantra Technologies 2004-2022

6 - Modèle objet -> Métaclasse

C:\Python34\python.exe

```
>>> a = MaClass("un","deux")
< metaclasse >
MaMetaClass.__call__ de  <class '__main__.MaClass'>
MaMetaClass.__call__ *args= ('un', 'deux')
MaMetaClass.__call__ **kwds= {}
</ metaclasse >
< classe >
MaClass.__new__ avec : args= ('un', 'deux') , kwargs= {}
</ classe >
< classe >
MaClass.__init__  avec : args= ('un', 'deux') , kwargs= {}
</ classe >
>>>
```


Copyright : Yantra Technologies 2004-2022

7 - Types de données

- Types : int
- Types : float
- Types : float & int - méthodes communes
- Types : float & int - Méthodes dédiées pour int
- Types : float & int - Méthodes dédiées pour float
- Type : complexe
- Operateurs mathématiques généraux
- Module de math
- Module de cmath
- Représentation d’un nombre
- Conversions

Version 1.5
Copyright : Yantra Technologies 2004-2022

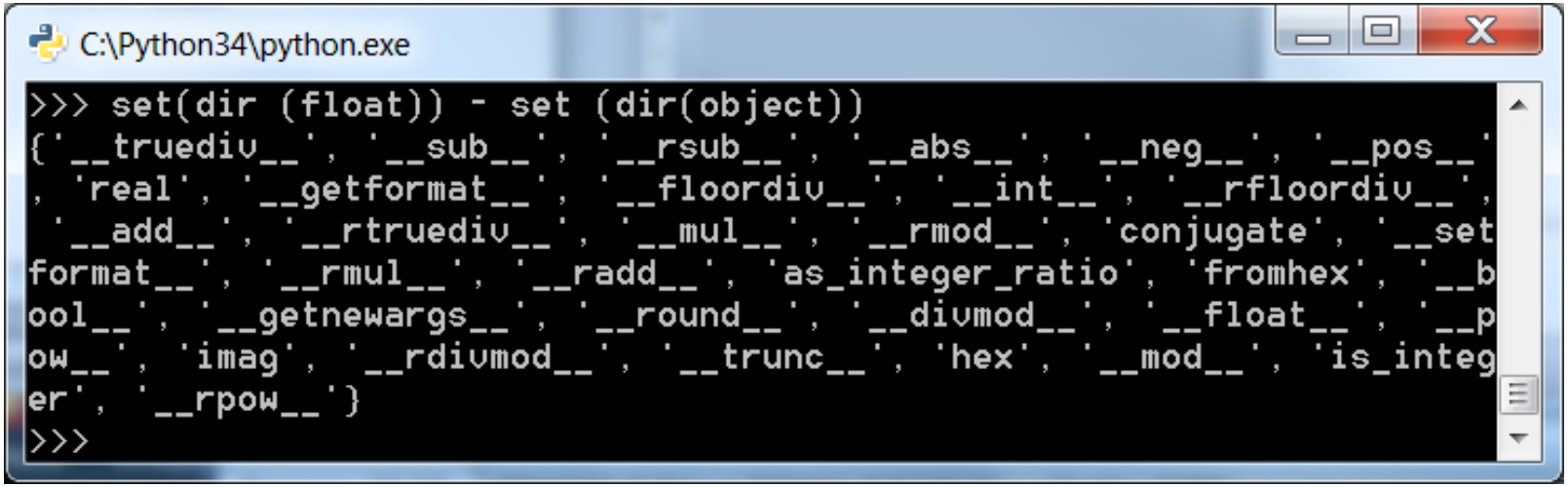
7 - Types de données -> Types : int

C:\Python34\python.exe

```
>>> set(dir (int)) - set (dir(object))
{'__rxor__', '__lshift__', 'numerator', '__truediv__', '__ror__', '__r
sub__', '__sub__', '__abs__', '__neg__', '__pos__', 'real', '__floordi
v__', 'bit_length', '__int__', '__or__', '__rfloordiv__', '__add__', '
__rtruediv__', '__invert__', 'denominator', '__mul__', '__rmod__', 'co
njugate', '__rlshift__', '__rmul__', '__radd__', '__floor__', '__xor__
', '__bool__', 'from_bytes', '__getnewargs__', '__round__', '__and__',
 '__divmod__', '__ceil__', '__float__', '__pow__', '__rand__', 'imag',
 '__rdivmod__', '__trunc__', '__rshift__', '__rrshift__', '__index__',
 '__mod__', 'to_bytes', '__rpow__'}
>>>
```

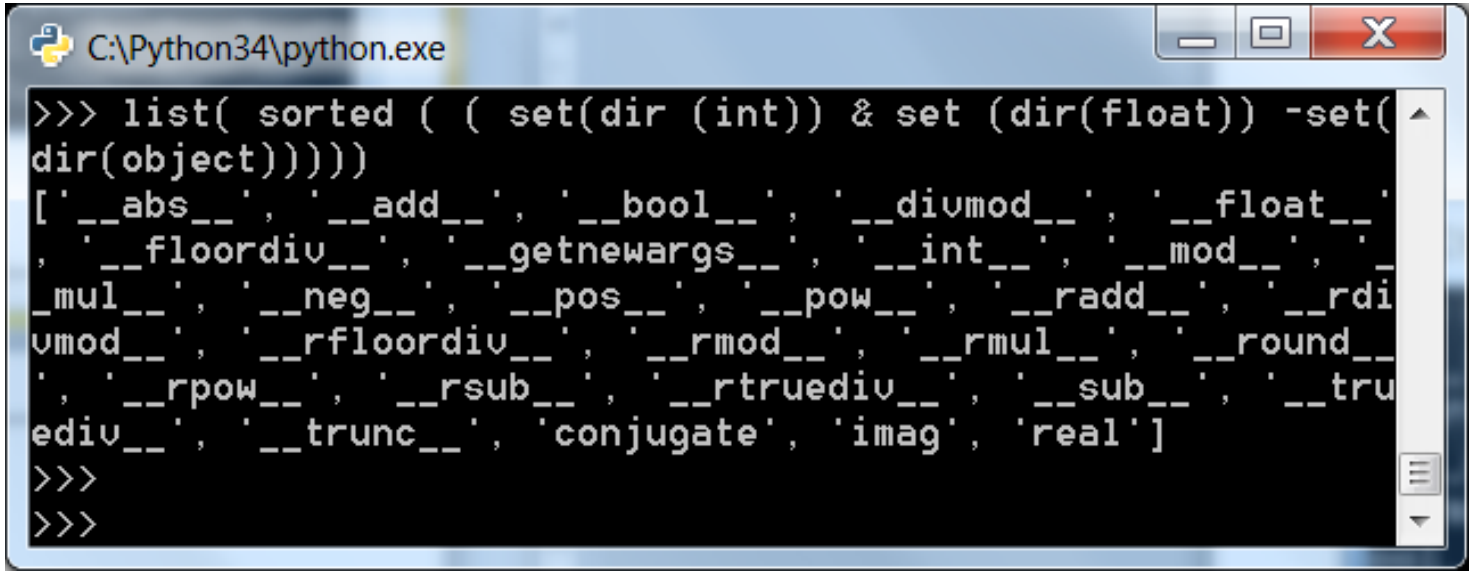

Version 1.5
Copyright : Yantra Technologies 2004-2022

7 - Types de données -> Types : float



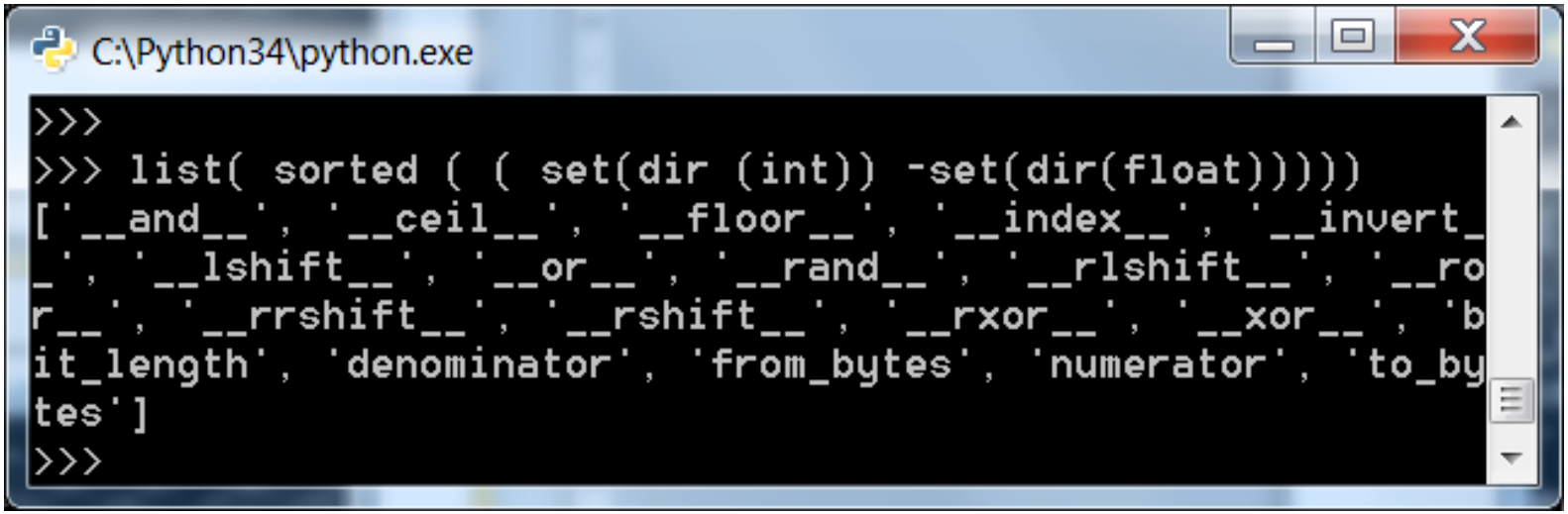
Version 1.5
Copyright : Yantra Technologies 2004-2022

7 - Types de données -> Types : float & int – méthodes communes



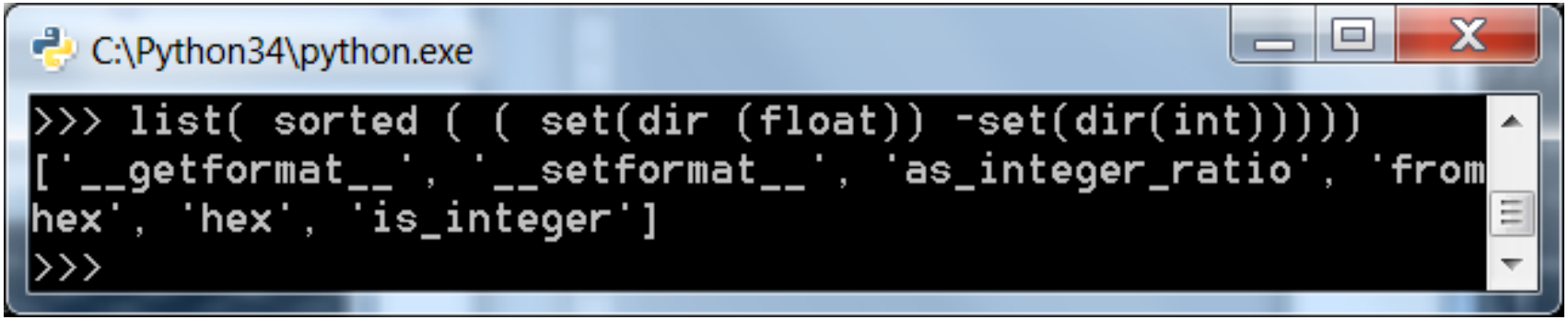
Copyright : Yantra Technologies 2004-2022

7 - Types de données -> Types : float & int – méthodes dédiées pour int



Version 1.5
Copyright : Yantra Technologies 2004-2022

7 - Types de données -> Types : float & int – méthodes dédiées pour float



Copyright : Yantra Technologies 2004-2022

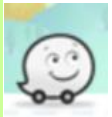
7 - Types de données -> Types : complexe

C:\Python34\python.exe

```
>>> type(1j)
<class 'complex'>
>>> c1=2+1j
>>> c2=1+2j
>>> c1
(2+1j)
>>> c2
(1+2j)
>>> c1+c2
(3+3j)
>>> c1*c2
5j
>>> c1.imag
1.0
>>> c1.real
2.0
>>> set(dir(complex))-set(dir(object))
{'__truediv__', '__sub__', '__rsub__', '__abs__', '__neg__', '__pos__'
, 'real', '__floordiv__', '__int__', '__rfloordiv__', '__add__', '__rt
ruediv__', '__mul__', '__rmod__', 'conjugate', '__rmul__', '__radd__',
 '__bool__', '__getnewargs__', '__divmod__', '__float__', '__pow__', '
imag', '__rdivmod__', '__mod__', '__rpow__'}
>>>
```


Version 1.5
Copyright : Yantra Technologies 2004-2022

7 - Types de données -> Opérateurs mathématiques généraux



Opérateur	Méthode	Exemple
+	__add__ ou __radd__	11+42 (53)
-	__sub__ ou __rsub__	10-42 (-32)
*	__mul__ ou __rmul__	10*42 (420)
/	__truediv__ ou __rtruediv__	105/42 (2,5)
//	__floordiv__ ou __rfloordiv__	105//42 (2)
%	__mod__ ou __rmod__	105%42 (21)
**	__pow__ ou __rpow__	42**2 (1764)
Operateur unaire +	__pos__	+42
Operateur unaire -	__neg__	-42
abs	__abs__	abs(-52) (52)
round	__round__	round(3.56,1) (3.6)
min		min([1,2,3,4]) ou min(1,2,3,4) (1)
max		max([1,2,3,4]) ou max‘(1,2,3,4) (4)
sum		sum(1,2,3,4) ou sum([1,2,3,4]) (10)

Version 1.5
Copyright : Yantra Technologies 2004-2022

7 - Types de données -> Module de math

```
C:\Python34\python.exe
>>> import math
>>> dir(math)
['__doc__', '__loader__', '__name__', '__package__', '__spec__', 'acos
', 'acosh', 'asin', 'asinh', 'atan', 'atan2', 'atanh', 'ceil', 'copysi
gn', 'cos', 'cosh', 'degrees', 'e', 'erf', 'erfc', 'exp', 'expm1', 'fa
bs', 'factorial', 'floor', 'fmod', 'frexp', 'fsum', 'gamma', 'hypot',
'isfinite', 'isinf', 'isnan', 'ldexp', 'lgamma', 'log', 'log10', 'log1
p', 'log2', 'modf', 'pi', 'pow', 'radians', 'sin', 'sinh', 'sqrt', 'ta
n', 'tanh', 'trunc']
>>>
```

```
C:\Python34\python.exe
>>> math.pi
3.141592653589793
>>> math.e
2.718281828459045
>>>
```


Version 1.5
Copyright : Yantra Technologies 2004-2022

7 - Types de données -> Module de math

Syntaxe	Description
acos(*x*)	Cette fonction trigonométrique retourne l'«*ArcCosinus*».
asin(*x*)	Cette fonction trigonométrique retourne l'«*ArcSinus*».
atan(*x*)	Cette fonction trigonométrique retourne l'«*ArcTangente*».
atan2(*y*,*x*)	Cette fonction trigonométrique retourne l'«*ArcTangente*» de *Y*/*X*.
ceil(*x*)	Cette fonction retourne la valeur maximale d'un nombre, soit l'entier le plus proche supérieur ou égal au nombre.
cos(*x*)	Cette fonction trigonométrique retourne le «*Cosinus*».
cosh(*x*)	Cette fonction trigonométrique retourne le «*Cosinus*» hyperbolique.
degrees(*x*)	Cette fonction permet de convertir un angle radian en degré.
exp(*x*)	Cette fonction permet de calculer la valeur exponentielle spécifiée.
fabs(*x*)	Cette fonction permet de calculer la valeur absolue d'un nombre réel.
floor(*x*)	Cette fonction permet de retourner la valeur minimale d'un nombre, soit l'entier le plus proche inférieur ou égal au nombre.
fmod(*x*, *y*)	Cette fonction permet de retourner le reste d'une division de *x*/*y*.
frexp(*x*)	Cette fonction permet d'effectuer la séparation de la mantisse et de l'exposant.

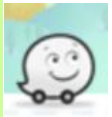
Version 1.5
Copyright : Yantra Technologies 2004-2022

7 - Types de données -> Module de math

Syntaxe	Description
hypot(*x*, *y*)	Cette fonction permet d'effectuer le calcul de la longueur de l'hypoténuse.
ldexp(*x*, *i*)	Cette fonction retourne la valeur de produit par la puissance 2.
log(*x*[, *base*])	Cette fonction retourne le logarithme naturel ou népérien.
log10(*x*)	Cette fonction retourne le logarithme décimal.
modf(*x*)	Cette fonction transforme un nombre réel en partie entière et en décimal (fraction).
pow(*x*, *y*)	Cette fonction retourne le calcul de *x* à la puissance *y*.
radians(*x*)	Cette fonction permet de convertir un degré en radian.
sin(*x*)	Cette fonction trigonométrique retourne le «*Sinus*».
sinh(*x*)	Cette fonction trigonométrique retourne le «*Sinus*» hyperbolique.
sqrt(*x*)	Cette fonction retourne la racine carrée du nombre spécifié.
tan(*x*)	Cette fonction trigonométrique retourne la «*tangente*».
tanh(*x*)	Cette fonction trigonométrique retourne la «*tangente*» hyperbolique.

Copyright : Yantra Technologies 2004-2022

7 - Types de données -> Module de math



Syntaxe	Description
fsum(x)	somme (meilleure précision que sum)
isinf(x)	renvoie True si x est infini
isnan(x)	renvoie True si x n'est pas un nombre
copysign(x, y)	renvoie x avec le signe de y
erf	fonction d'erreur dédiée aux statistiques
erfc	complément de la fonction d'erreur
gamma(x)	fonction gamma
lgamma(x)	logarithme de la valeur absolue de la fonction gamma

Version 1.5
Copyright : Yantra Technologies 2004-2022

7 - Types de données -> Module de cmath

C:\Python34\python.exe

```
>>>
>>> dir (cmath)
['__doc__', '__loader__', '__name__', '__package__', '__spec__', 'acos
', 'acosh', 'asin', 'asinh', 'atan', 'atanh', 'cos', 'cosh', 'e', 'exp
', 'isfinite', 'isinf', 'isnan', 'log', 'log10', 'phase', 'pi', 'polar
', 'rect', 'sin', 'sinh', 'sqrt', 'tan', 'tanh']
>>>
>>> set(dir(cmath))-set((dir(math)))
{'rect', 'phase', 'polar'}
>>>
>>> set(dir(cmath))&set((dir(math)))
{'cosh', 'e', 'acos', '__spec__', 'tan', 'atan', '__doc__', 'asinh', '
sin', 'isinf', 'atanh', 'log10', 'exp', 'asin', '__loader__', 'acosh',
 'isfinite', 'log', 'cos', 'tanh', 'isnan', 'pi', '__name__', 'sinh',
'__package__', 'sqrt'}
>>>
```


Version 1.5
Copyright : Yantra Technologies 2004-2022

7 - Types de données -> Module de cmath

Syntaxe	Description
acos(x)	Cette fonction trigonométrique retourne l'«*ArcCosinus*».
acosh(x)	Cette fonction trigonométrique retourne la valeur de l'«ArcCosinus» hyperbolique.
asin(x)	Cette fonction trigonométrique retourne l'«*ArcSinus*».
asinh(x)	Cette fonction trigonométrique retourne la valeur de l'«ArcSinus» hyperbolique.
atan(x)	Cette fonction trigonométrique retourne l'«*ArcTangente*».
atanh(x)	Cette fonction trigonométrique retourne la valeur de l'«ArcTangente» hyperbolique.
cos(x)	Cette fonction trigonométrique retourne le «*Cosinus*».
cosh(x)	Cette fonction trigonométrique retourne le «*Cosinus*» hyperbolique.
exp(x)	Cette fonction calcule l'exponentiel de la valeur «*x*».
log(x[, base])	Cette fonction retourne le logarithme naturel ou népérien.
log10(x)	Cette fonction retourne le logarithme décimal.
sin(x)	Cette fonction trigonométrique retourne le «*Sinus*».
sinh(x)	Cette fonction trigonométrique retourne le «*Sinus*» hyperbolique.
sqrt(x)	Cette fonction retourne la racine carrée du nombre «*a*».
tan(x)	Cette fonction trigonométrique retourne la «*tangente*».
tanh(x)	Cette fonction trigonométrique retourne la «*tangente*» hyperbolique.

Version 1.5
Copyright : Yantra Technologies 2004-2022

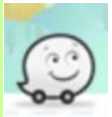
7 - Types de données -> Module de cmath

Syntaxe	Description
polar(ci)	donne une représentation polaire d'un nombre complexe
phase(ci)	la fonction phase fait partie du module cmath et permet de calculer l'argument d'un nombre complexe
rect(x,y)	la fonction permet de passer d'une représentation polaire à une représentation rectangulaire

C:\Python34\python.exe

```
>>> import cmath
>>> cmath.polar(1+1j)
(1.4142135623730951, 0.7853981633974483)
>>> cmath.phase(1+1j)
0.7853981633974483
>>> cmath.rect(2**0.5,cmath.pi/4.)
(1.0000000000000002+1j)
>>>
```


Version 1.5
Copyright : Yantra Technologies 2004-2022



Représentation

- Decimal
- Hexadécimal : hex(x)
- Octale : oct(x)

Copyright : Yantra Technologies 2004-2022

7 - Types de données -> Conversions

C:\Python34\python.exe

```
124.0
>>> int(425.258974256)
425
>>> complex(1.0)
(1+0j)
>>> complex(2)
(2+0j)
>>> int(1+0.j)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: can't convert complex to int
>>> bool(45)
True
>>> bool(0)
False
>>> bool(-58)
True
>>> bool(2+1j)
True
>>> bool(0j)
False
>>>
```

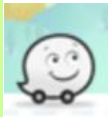

Version 1.5
Copyright : Yantra Technologies 2004-2022

8 - Types de séquence

- Listes
- N-uplet
- Array
- Ensemble

Copyright : Yantra Technologies 2004-2022

8 - Type de séquence -> listes



C:\Python34\python.exe

```
>>> l0=[]
>>> l0
[]
>>> len(l0)
0
>>> l5=[1,2,3,4,5]
>>> l5
[1, 2, 3, 4, 5]
>>> len(l5)
5
>>> l5[0]=10
>>> l5
```


Version 1.5
Copyright : Yantra Technologies 2004-2022

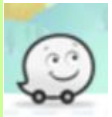
8 - Type de séquence -> listes

C:\Python34\python.exe

```
>>>
>>> dir(list)
['__add__', '__class__', '__contains__', '__delattr__', '__delitem__',
 '__dir__', '__doc__', '__eq__', '__format__', '__ge__', '__getattribu
te__', '__getitem__', '__gt__', '__hash__', '__iadd__', '__imul__', '_
_init__', '__iter__', '__le__', '__len__', '__lt__', '__mul__', '__ne_
_', '__new__', '__reduce__', '__reduce_ex__', '__repr__', '__reversed_
_', '__rmul__', '__setattr__', '__setitem__', '__sizeof__', '__str__',
 '__subclasshook__', 'append', 'clear', 'copy', 'count', 'extend', 'in
dex', 'insert', 'pop', 'remove', 'reverse', 'sort']
>>> type.mro(list)
[<class 'list'>, <class 'object'>]
>>>
>>> set(dir(list))-set(dir(object))
{'__setitem__', 'insert', 'copy', 'index', 'sort', 'count', '__iadd__'
, 'clear', '__add__', '__mul__', 'append', '__len__', '__rmul__', '__r
eversed__', 'extend', '__imul__', '__getitem__', '__delitem__', '__con
tains__', 'remove', 'reverse', 'pop', '__iter__'}
>>>
```


Version 1.5
Copyright : Yantra Technologies 2004-2022

8 - Type de séquence -> listes



Python Expression	Résultats	Description
len ([1, 2, 3])	3	Longueur
[1, 2, 3] + [4, 5, 6]	[1, 2, 3, 4, 5, 6]	Enchaînement
['Salut!'] * 4	['Salut!', 'Salut!', 'Salut!', 'Salut!']	Répétition
3 in [1, 2, 3]	True	Adhésion
for x in [1, 2, 3]: print(x)	1 2 3	Itération
L=['mac','PC','Linux']		
L [2]	Linux	Décalages commençant à zéro
L [-2]	PC	Négatif: comptage de la droite
L [1:]	['mac','PC']	Tranchage : récupère sections

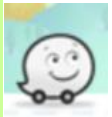
Version 1.5
Copyright : Yantra Technologies 2004-2022

8 - Type de séquence -> listes

Fonction	Description
cmp (list1, list2)	Compare les éléments des deux listes.
len (liste)	Donne la longueur totale de la liste.
max (liste)	Retourne l'élément de la liste de valeur max.
min (liste)	Retourne l'élément de la liste de valeur min.
liste (éléments)	Convertit un tuple dans la liste.

Version 1.5
Copyright : Yantra Technologies 2004-2022

8 - Type de séquence -> listes



Les méthodes	Description
list.append (obj)	Ajoute l'objet obj à la liste
list.count (obj)	Retourne combien de fois obj est dans la liste
list.extend (suivant)	Ajoute le contenu de suivant à la liste
list.index (obj)	Renvoie l'index le plus bas dans la liste où apparaît obj
list.insert (indice, obj)	Insère obj à la liste à l'index de décalage
list.pop (obj = liste [-1])	Supprime et renvoie le dernier objet ou obj de la liste
list.remove (obj)	Supprime l'objet obj de la liste
list.reverse ()	Inverse les objets de la liste en place
list.sort ([FUNC])	Trie les objets de la liste

Version 1.5
Copyright : Yantra Technologies 2004-2022

8 - Type de séquence -> N-uplet

C:\Python34\python.exe

```
>>>
>>> l0=()
>>> l0
()
>>> len(l0)
0
>>> l5=(1,2,3,4,5)
>>> l5
(1, 2, 3, 4, 5)
>>> l5
(1, 2, 3, 4, 5)
>>> len(l5)
5
>>>
>>> #tuple est une liste dont les elements ne peuvent pas etre modifié
s
... l5[0]=10
Traceback (most recent call last):
  File "<stdin>", line 2, in <module>
TypeError: 'tuple' object does not support item assignment
>>> l5
(1, 2, 3, 4, 5)
>>>
```


Version 1.5
Copyright : Yantra Technologies 2004-2022

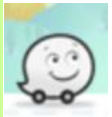
8 - Type de séquence -> N-uplet & liste

C:\Python34\python.exe

```
>>>
>>> dir(tuple)
['__add__', '__class__', '__contains__', '__delattr__', '__dir__', '__
doc__', '__eq__', '__format__', '__ge__', '__getattribute__', '__getit
em__', '__getnewargs__', '__gt__', '__hash__', '__init__', '__iter__',
 '__le__', '__len__', '__lt__', '__mul__', '__ne__', '__new__', '__red
uce__', '__reduce_ex__', '__repr__', '__rmul__', '__setattr__', '__siz
eof__', '__str__', '__subclasshook__', 'count', 'index']
>>> type.mro(tuple)
[<class 'tuple'>, <class 'object'>]
>>>
>>> set(dir(tuple))-set(dir(object))
{'__contains__', '__add__', 'index', '__getnewargs__', '__mul__', '__g
etitem__', '__iter__', '__len__', 'count', '__rmul__'}
>>>
>>> set(dir(tuple))-set(dir(list))
{'__getnewargs__'}
>>> set(dir(list))-set(dir(tuple))
{'__setitem__', '__reversed__', 'clear', 'remove', 'reverse', 'insert'
, 'extend', 'copy', '__imul__', 'sort', 'pop', 'append', '__delitem__'
, '__iadd__'}
>>>
```


Version 1.5
Copyright : Yantra Technologies 2004-2022

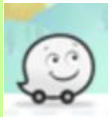
8 - Type de séquence -> N-uplet



Python Expression	Résultats	Description
len ((1, 2, 3))	3	Longueur
(1, 2, 3) + (4, 5, 6)	(1, 2, 3, 4, 5, 6)	Enchaînement
('Salut!') * 4	('Salut!', 'Salut!', 'Salut!', 'Salut!')	Répétition
3 in [1, 2, 3]	True	Adhésion
for x in (1, 2, 3): print(x)	1 2 3	Itération
L=('mac','PC','Linux')		
L [2]	Linux	Décalages commencent à zéro
L [-2]	PC	Négatif : comptage de la droite
L [1:]	['mac','PC']	Tranchage récupère sections

Copyright : Yantra Technologies 2004-2022

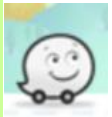
8 - Type de séquence -> N-uplet



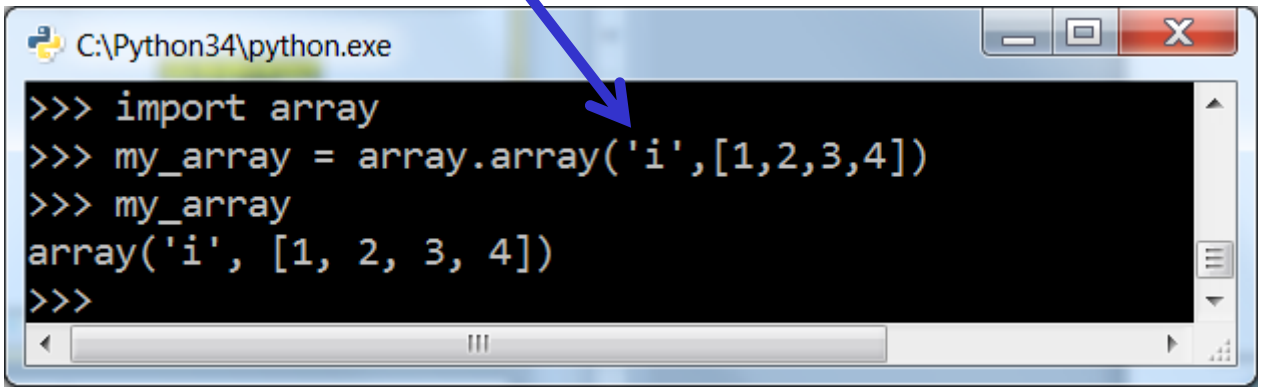
Fonction	Description
cmp (list1, list2)	Compare les éléments des deux tuples
len (liste)	Donne la longueur totale du tuple
max (liste)	Retourne l'élément du tuple de valeur max
min (liste)	Retourne l'élément du tuple de valeur min
tuple (éléments)	Convertit une liste dans un tuple

Copyright : Yantra Technologies 2004-2022

8 - Type de séquence -> array



- qui est similaire à une liste (list) avec la condition supplémentaire que tous les éléments sont du même type
- Typescodes permettant le typage



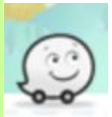
Version 1.5
Copyright : Yantra Technologies 2004-2022

8 - Type de séquence -> array

Type code	C Type	Python Type	Minimum size in bytes
'c'	char	character	1
'b'	signed char	int	1
'B'	unsigned char	int	1
'u'	Py_UNICODE	Unicode character	2 (see note)
'h'	signed short	int	2
'H'	unsigned short	int	2
'i'	signed int	int	2
'I'	unsigned int	long	2
'l'	signed long	int	4
'L'	unsigned long	long	4
'f'	float	float	4
'd'	double	float	8

Version 1.5
Copyright : Yantra Technologies 2004-2022

8 - Type de séquence -> ensemble



On distingue deux types d'ensembles :

- les **set** modifiables,
- les **frozenset** non modifiables.

C:\Python34\python.exe

```
>>>
>>> ens = set([1, 2, 3])
>>> ens
{1, 2, 3}
>>>
>>> ensnonmodifiable = frozenset([1, 2, 3])
>>> ensnonmodifiable
frozenset({1, 2, 3})
>>>
```


Version 1.5
Copyright : Yantra Technologies 2004-2022

8 - Type de séquence -> ensemble

C:\Python34\python.exe

```
>>> S1 = set([1, 2, 3, 4])
>>> S2 = set([0, 1, 2])
>>>
>>> #La réunion
... S1.union(S2)
{0, 1, 2, 3, 4}
>>> S1 | S2
{0, 1, 2, 3, 4}
>>>
>>> # L'intersection
... S1.intersection(S2)
{1, 2}
>>> S1 & S2
{1, 2}
>>>
```

```
>>> # La différence
... S1.difference(S2)
{3, 4}
>>> S1-S2
{3, 4}
>>>
>>> # La différence symétrique
... S1.symmetric_difference(S2)
{0, 3, 4}
>>> S1 ^ S2
{0, 3, 4}
>>>
>>> # inclusion
... A = set([1, 2])
>>> B = set([0, 1, 2, 3])
>>>
>>> A.issubset(B)
True
>>> B.issuperset(A)
```

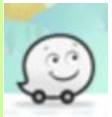

Version 1.5
Copyright : Yantra Technologies 2004-2022

9 - Chaîne de caractères

- Chaîne de caractères
- Caractères spéciaux
- Opérateurs
- Formats
- Symboles
- Méthodes

Copyright : Yantra Technologies 2004-2022

9 - Chaîne de caractères

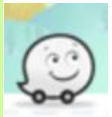


C:\Python34\python.exe

```
>>> s = "Le mot magique est 'abracadabra'"
>>>
>>> s = 'Le mot magique est "abracadabra"'
>>>
>>> s = "avec des 'guillemets simples', mais aussi" 'des "guillemets d
oubles"...'
>>>
>>> s = """Ceci est une longue
... chaine de caractéres.
... En plus, elle contient des
... guillemets de toutes les
... formes : " ' ..."""
```


Version 1.5
Copyright : Yantra Technologies 2004-2022

9 - Chaîne de caractères -> Caractères spéciaux



Backslash \ notation	Description
\ a	Cloche ou d'alerte
\ b	Retour arrière
\ cx	Control-x
\ Cx	Control-x
\ e	Évasion
\ f	Saut de page
\ M \ Cx	Meta-Control-x
\ n	Newline
\ nnn	Notation octale, où n est dans la gamme de 0,7
\ r	Retour chariot
\ s	Espace
\ t	Tabulation
\ v	Tabulation verticale
\ x	Caractère x
\ xnn	La notation hexadécimale, où n est dans la gamme de 0,9, af ou AF

Copyright : Yantra Technologies 2004-2022

9 - Chaîne de caractères -> Caractères spéciaux

C:\Python34\python.exe

```
>>> s = "Ceci est une chaine de caracteres\nsur deux lignes."
>>> print(s)
Ceci est une chaine de caracteres
sur deux lignes.
>>>
>>> s = r"Ceci est une chaine de caracteres\nsur une ligne."
>>> print(s)
Ceci est une chaine de caracteres\nsur une ligne.
>>>
>>>
>>> s = "Et a fait BIP\a BIP\a..."
>>> print(s)
Et a fait BIP BIP...
>>>
>>>
>>> s = "Zut\b\b\bTabernacle !"
>>> print(s)
Tabernacle !
>>>
>>>
>>> s = "| a\t| bb\t| ccc\t| dddd\t| eeeee\t| ffffff\t|"
>>> print(s)
| a     | bb    | ccc   | dddd  | eeeee | ffffff        |
>>>
```


Version 1.5
Copyright : Yantra Technologies 2004-2022

9 - Chaîne de caractères -> Opérateurs

Opérateur	Description
+	Concaténation - Ajoute les valeurs de chaque côté de l'opérateur
*	Répétition - Crée de nouvelles chaînes, concaténer plusieurs copies de la même chaîne
[]	Slice - donne le caractère de l'index donné
[:]	Donne les éléments de la plage donnée
in	Renvoie True si un caractère existe dans la chaîne donnée
not in	Renvoie True si un personnage n'existe pas dans la chaîne donnée
r / R	Raw String - Supprime le sens réel de caractères d'échappement. La syntaxe pour les chaînes premières est exactement le même que pour les chaînes normales, à l'exception de l'opérateur de première chaîne, la lettre «r», qui précède les guillemets. Le "r" peut être minuscules (r) ou majuscule (R) et doit être placée immédiatement avant le premier guillemet.
%	Format - Effectue le formatage

Version 1.5
Copyright : Yantra Technologies 2004-2022

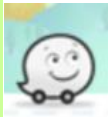
9 - Chaîne de caractères -> Opérateurs

C:\Python34\python.exe

```
>>> s1="La précipitation vient du Diable "
>>> s2="Dieu travaille lentement."
>>> s3="Proverbe Persan"
>>> s=s1+";"+s2+"\n"+s3+"\n"
>>> print(s)
La précipitation vient du Diable ;Dieu travaille lentement.
Proverbe Persan

>>> len(s)
76
>>> s[0:2]
'La'
>>> s[35:39]
'ieu '
>>> "Diable" in s
True
>>>
>>>
```


Version 1.5
Copyright : Yantra Technologies 2004-2022

9 - Chaîne de caractères -> Formats



Format de symbole	Conversion
%c	caractère
%s	conversion de chaîne via str () avant le formatage
%i	entier décimal signé
%d	entier décimal signé
%u	entier décimal non signé
%o	octal
%x	hexadécimal (lettres minuscules)
% X	hexadécimal (les lettres en majuscules)
%e	notation exponentielle (avec minuscule «e»)
% E	notation exponentielle (avec majuscule 'E')
%f	virgule flottante nombre réel
%g	la plus courte de% f et% e
% G	la plus courte de% f et% E

Copyright : Yantra Technologies 2004-2022

9 - Chaîne de caractères -> Formats

C:\Python34\python.exe

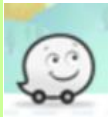
```
>>> import math
>>>
>>> "{0:.4f}".format(10.1234567890)
'10.1235'
>>>
>>> "{0:8.4f}".format(10.987654321)
' 10.9877'
>>>
>>> "sin({:.4f}) = {:.4e}".format(0.1234567890, math.sin(0.123456789))

'sin(0.1235) = 1.2314e-01'
>>>
>>> coord = (3, 5)
>>> 'X: {0[0]};  Y: {0[1]}'.format(coord)
'X: 3;  Y: 5'
>>>
>>> "repr() shows quotes: {!r}; str() doesn't: {!s}".format('test1', '
test2')
"repr() shows quotes: 'test1'; str() doesn't: test2"
```

```
>>> '{:<30}'.format('left aligned')
'left aligned                  '
>>>
>>> '{:>30}'.format('right aligned')
'                 right aligned'
>>>
>>> '{:^30}'.format('centered')
'           centered           '
>>>
>>> '{:*^30}'.format('centered')
'***********centered***********'
>>>
>>> import datetime
>>> d = datetime.datetime(2010, 7, 4, 12, 15, 58)
>>>
>>> '{:%Y-%m-%d %H:%M:%S}'.format(d)
'2010-07-04 12:15:58'
>>>
```

https://docs.python.org/3.1/library/string.html

Version 1.5
Copyright : Yantra Technologies 2004-2022

9 - Chaîne de caractères -> Symboles



Symbole	Fonctionnalité
*	argument spécifie la largeur ou la précision
-	justification à gauche
+	afficher le signe
<Sp>	laisser un espace avant un nombre positif
#	ajouter l'octal zéro («0») ou hexadécimal leader 0x »ou« 0X », selon que « x » ou « X » ont été utilisés.
0	Remplissage à gauche avec des zéros (au lieu des espaces)
%	%% Vous laisse avec un seul littéral '%'
(Var)	variable de cartographie (arguments dictionnaire)
mn	m est la largeur totale minimum et n est le nombre de chiffres à afficher après la virgule
""" chaine """	Triples guillemets de Python vient à la rescousse en permettant que chaîne s'étende sur plusieurs lignes, y compris les sauts de ligne, les tabulations et autres caractères spéciaux.

Version 1.5
Copyright : Yantra Technologies 2004-2022

9 - Chaîne de caractères -> Méthodes

Les méthodes	Description
tirer ()	Capitalise la première lettre de la chaîne
centre (largeur, FillChar)	Retourne une chaîne complétée par des espaces avec la chaîne originale centrée sur un total de colonnes en largeur
compter (str, mendier = 0, fin = len (string))	Compte combien de fois str se produit dans la chaîne ou dans une sous-chaîne de chaîne qui commence à l'indice mendier et se termine à l'indice
decode (encoding = "UTF-8 ', erreurs =' strict ')	Décode la chaîne en utilisant le codec enregistré pour l'encodage. Encodage par défaut le codage de chaîne par défaut.
encode (encoding = "UTF-8 ', erreurs =' strict ')	Retourne la version codée chaîne de chaîne; en cas d'erreur, le défaut est de soulever une ValueError
endswith (suffixe, mendier = 0, fin = len (string))	Détermine si la chaîne ou une sous-chaîne (commençant par l'indice mendier et se terminant par l'indice fin) se termine par le suffixe; renvoie true, si c'est le cas et faux sinon
expandtabs (tabsize = 8)	Développe les onglets de chaîne à plusieurs espaces; par défaut à 8 places par onglet si tabsize non prévus
trouver (str, mendier = 0 end = len (string))	Déterminer si str se trouve dans la chaîne ou dans une sous-chaîne de chaîne sous-chaîne (commençant par l'indice mendier et se terminant par l'indice fin) indice de rendement s'il est trouvé et -1 sinon
indice (str, mendier = 0, fin = len (string))	Comme find (), mais déclenche une exception si str n'est pas trouvé

Version 1.5
Copyright : Yantra Technologies 2004-2022

9 - Chaîne de caractères -> Méthodes

Les méthodes	Description
isalnum ()	Renvoie true si la chaîne a au moins 1 caractère et tous les caractères sont alphanumériques et faux sinon
isalpha ()	Renvoie true si la chaîne a au moins 1 caractère et tous les caractères sont alphabétiques et faux sinon
isdigit ()	Renvoie true si la chaîne ne contient que des chiffres et faux sinon
islower ()	Renvoie true si toutes les lettres sont en minuscule, et faux sinon (ou chaîne vide)
IsNumeric ()	Renvoie true si une chaîne Unicode contient uniquement des caractères numériques et faux sinon
isspace ()	Renvoie true si la chaîne contient des caractères blancs et seulement, faux sinon
istitle ()	Renvoie true si la chaîne est correctement "titlecased" et faux sinon
isupper ()	Renvoie true si toutes les lettres sont en majuscules et faux sinon (ou chaîne vide)
join (seq)	Fusionne (concatène) les éléments de seq dans une chaîne, avec chaîne de séparation
len (string)	Renvoie la longueur de la chaîne
ljust (n [, FillChar])	Renvoie une chaîne de longueur max(len(s),n) débutant par une copie de s, suivie de zéros ou du caractère de remplissage
inférieure ()	Convertit toutes les lettres majuscules de la chaîne en minuscules
lstrip ()	Supprime tous les principaux espaces dans la chaîne
maketrans ()	Renvoie une table de conversion à utiliser en fonction de traduction.
max (str)	Retourne le caractère alphabétique max de la chaîne str
min (str)	Retourne le caractère alphabétique min de la chaîne str

Version 1.5
Copyright : Yantra Technologies 2004-2022

9 - Chaîne de caractères -> Méthodes

Les méthodes	Description
replace(old,new [,max])	Remplace toutes les occurrences de la chaîne old avec new ou à la plupart des occurrences max si max donné
rfind (str, beg= 0, end = len (string))	Identique à trouver (), mais recherche en arrière dans la chaîne
rindex (str, beg= 0, end= len (string))	Même que indice (), mais recherche en arrière dans la chaîne
rjust (largeur, [, FillChar])	Retourne une chaîne complétée par des espaces avec la chaîne d'origine justifié à droite à un total de colonnes de largeur.
rstrip ()	Supprime tous les espaces de fin de chaîne
split (str = "", num = string.count (str))	Divise la chaîne en fonction du séparateur str (espace si non fourni) et retourne la chaîne divisée en sous-chaînes
splitlines(num = string.count ('\ n'))	Divise la chaîne à tous (ou num) les sauts de lignes et retourne une liste de chaque ligne avec des sauts de ligne retirés
startswith (str, beg= 0, end= len (string))	Détermine si la chaîne ou une sous-chaîne (commençant par l'indice beg et se terminant par l'indice end)) commence par chaîne str; renvoie true si c'est le cas et faux sinon
strip([chars])	Effectue la fois lstrip () et rstrip () sur la chaîne
swapcase ()	Inverse minuscule et majuscule pour toutes les lettres de la chaîne
title ()	Renvoie une version de la chaîne où tous les mots commencent par une majuscule, le reste en minuscules
translate(table, deletechars = "")	Traduit la chaîne en fonction de la table de traduction str (256 caractères), en supprimant ceux de la chaîne
upper()	Convertit les minuscules en majuscules
zfill (largeur)	Retourne la chaîne originale lvec des zéros à gauche à un total de caractères de largeur
isdecimal ()	Renvoie true si une chaîne Unicode contient uniquement des caractères décimaux et faux sinon

Version 1.5
Copyright : Yantra Technologies 2004-2022

10 - Dictionnaire

- Dictionnaire
- Fonctions
- Méthodes

Copyright : Yantra Technologies 2004-2022

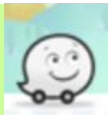
10 - Dictionnaire

C:\Python34\python.exe

```
>>> d = {"phrase":"Le bonheur est une petite chose que l'on grignote,
assis par terre, au soleil", "auteur":"Jean Giraudoux"}
>>>
>>> d
{'auteur': 'Jean Giraudoux', 'phrase': "Le bonheur est une petite chos
e que l'on grignote, assis par terre, au soleil"}
>>>
>>> d["phrase"]
"Le bonheur est une petite chose que l'on grignote, assis par terre, a
u soleil"
>>>
>>> d["auteur"]
'Jean Giraudoux'
>>>
>>> d["date"]
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
KeyError: 'date'
```


Version 1.5
Copyright : Yantra Technologies 2004-2022

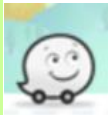
10 - Dictionnaire



Python Expression	Résultats	Description
len ({'a':1,'b',2,'c':3})	3	Longueur
a' in {'a':1,'b',2,'c':3}	True	Adhésion
for x in {'a':1,'b',2,'c':3}: print(x)	a b c	Itération
L={'a':1,'b',2,'c':3}		
L ['a']	1	

Copyright : Yantra Technologies 2004-2022

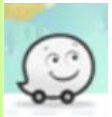
10 – Dictionnaire -> Fonctions



Fonction	Description
cmp (list1, list2)	Compare les éléments des deux tuples
len (liste)	Donne la longueur totale du tuple
str(dict)	Produit une représentation imprimable de la chaîne d'un dictionnaire
tuple (seq)	Convertit un dictionnaire dans un tuple
liste(seq)	Convertit un dictionnaire dans une liste

Version 1.5
Copyright : Yantra Technologies 2004-2022

10 – Dictionnaire -> fonctions

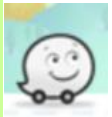


C:\Python34\python.exe

```
>>>
>>> len(d)
2
>>>
>>> "date" in d
False
>>>
>>> str(d)
'{\'auteur\': \'Jean Giraudoux\', \'phrase\': "Le bonheur est une peti
te chose que l\'on grignote, assis par terre, au soleil"}'
>>>
>>> d.keys()
dict_keys(['auteur', 'phrase'])
>>>
```


Version 1.5
Copyright : Yantra Technologies 2004-2022

10 – Dictionnaire -> Méthodes



Les méthodes	description
dict.clear ()	Supprime tous les éléments du dictionnaire *dict*
dict.copy ()	Retourne une copie du dictionnaire *dict*
dict.fromkeys	Crée un nouveau dictionnaire avec les éléments définis à la valeur.
dict.get (clé, valeur par défaut = Aucune)	Pour la *clé,* retourne la valeur par défaut ou si la clé n'est pas dans le dictionnaire
dict.has_key (clé) remplacer par in	Renvoie true si la clé est dans le dictionnaire dict, faux sinon
dict.items	Retourne une liste de *dict* (clé, valeur)
dict.keys()	Liste les clés du dictionnaire dict
dict.setdefault (clé, valeur par défaut = Aucune)	Semblable à obtenir (), mais sera mis dict [key] = défaut si *la clé* n'est pas déjà en dict
dict.update (dict2)	Ajoute le dictionnaire dict2 à dict
dict.values()	Liste des valeurs du dictionnaire *de dict*

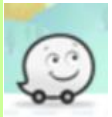
Version 1.5
Copyright : Yantra Technologies 2004-2022

11 - Données temporelles

- Données temporelles
- Module time
- Module calendar

Version 1.5
Copyright : Yantra Technologies 2004-2022

11 - Données temporelles

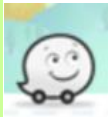


C:\Python34\python.exe

```
>>> import time;  # il faut inclure le module time .
>>>
>>> ticks = time.time()
>>> print ("nombre  de seconde depuis le  1 janvier , 1970: 12:00 am =
> \n", ticks, "s\n")
nombre  de seconde depuis le  1 janvier , 1970: 12:00 am =>
 1410079537.6355 s

>>>
>>>
```


Version 1.5
Copyright : Yantra Technologies 2004-2022

11 - Données temporelles → module time - formats



Index	Domaine	Attributs	Valeurs
0	Année à 4 chiffres	tm_year	2008
1	Mois	tm_mon	1 à 12
2	Jour	tm_mday	1 à 31
3	Heure	tm_hour	0 à 23
4	Minute	tm_min	0-59
5	Deuxième	tm_sec	0-61 (60 ou 61 sont bissextiles secondes)
6	Jour de la semaine	tm_wday	0 à 6 (0 est le lundi)
7	Jour de l'année	tm_yday	1 et 366 (jour julien)
8	Heure d'été	tm_isdst	-1, 0, 1, -1 signifie bibliothèque détermine l'heure d'été

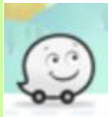
Version 1.5
Copyright : Yantra Technologies 2004-2022

11 - Données temporelles → module time - fonction

Fonction	Description
time.altzone	Le décalage du fuseau horaire DST locale, en quelques secondes à l'ouest de l'UTC, s'il est défini. C'est négatif si le fuseau horaire DST locale est à l'est de l'UTC (comme en Europe de l'Ouest, y compris le Royaume-Uni). Utiliser cette option que si la lumière du jour est différente de zéro.
time.asctime([tupletime])	Accepte un temps-tuple et retourne une chaîne de 24 caractères lisibles tels que «mar 11 décembre 18:07:14 2008».
time.clock( )	Renvoie l'heure actuelle de la CPU en tant que nombre à virgule flottante de secondes. Pour mesurer les coûts informatiques des différentes approches, la valeur de time.clock est plus utile que celle de time.time ().
time.ctime([secs])	Comme asctime (localtime (s)) et sans arguments, c'est comme asctime ()
time.gmtime([secs])	Accepte un instant exprimé en secondes depuis l'époque et retourne un temps tuple t avec le temps UTC. Remarque: t.tm_isdst est toujours 0
time.localtime([secs])	Accepte un instant exprimé en secondes depuis l'époque et retourne un temps tuple t avec l'heure locale (t.tm_isdst est 0 ou 1, selon que l'heure d'été s'applique à sec instantanées par les règles locales).
time.mktime(tupletime)	Accepte un instant exprimé en temps-uplet en heure locale et renvoie une valeur à virgule flottante à l'instant exprimé en secondes depuis l'époque.
time.sleep(secs)	Suspend le thread appelant à secs secondes.
time.strftime(fmt[,tupletime])	Accepte un instant exprime en temps-uplet en heure locale et renvoie une chaîne représentant l'instant comme spécifié par la chaîne fmt.
time.strptime(str,fmt='%a %b %d %H:%M:%S %Y')	Analyse les str selon la chaîne de format fmt et renvoie l'instant en format moment-tuple.
time.time( )	Renvoie l'instant présent, un nombre à virgule flottante de secondes depuis l'époque.
time.tzset()	Remet les règles de conversion de temps utilisés par les routines de la bibliothèque. La variable d'environnement TZ spécifie comment cela se fait.

Version 1.5
Copyright : Yantra Technologies 2004-2022

11 - Données temporelles → module time - exemples



C:\Python34\python.exe

```
>>>
>>> localtime = time.localtime(time.time())
>>> print( "temps local :", localtime)
temps local : time.struct_time(tm_year=2014, tm_mon=9, tm_mday=7, tm_h
our=10, tm_min=53, tm_sec=25, tm_wday=6, tm_yday=250, tm_isdst=1)
>>>
>>> localtime = time.asctime( time.localtime(time.time()) )
>>> print ("temps local :", localtime)
temps local : Sun Sep  7 10:53:25 2014
>>>
>>> time.strftime("%A %d %B %Y %H:%M:%S")
'Sunday 07 September 2014 10:53:26'
>>>
```


Copyright : Yantra Technologies 2004-2022

11 - Données temporelles → module calendar

C:\Python34\python.exe

```
>>> import calendar
>>>
>>> cal = calendar.month(2015, 1)
>>> print (cal);
    January 2015
Mo Tu We Th Fr Sa Su
          1  2  3  4
 5  6  7  8  9 10 11
12 13 14 15 16 17 18
19 20 21 22 23 24 25
26 27 28 29 30 31

>>>
>>> cal = calendar.HTMLCalendar(calendar.SUNDAY)
>>> print (c.formatmonth(2014, 7))
<table border="0" cellpadding="0" cellspacing="0" class="month">
<tr><th colspan="7" class="month">July 2014</th></tr>
<tr><th class="sun">Sun</th><th class="mon">Mon</th><th class="tue">Tu
e</th><th class="wed">Wed</th><th class="thu">Thu</th><th class="fri">
Fri</th><th class="sat">Sat</th></tr>
<tr><td class="noday"> </td><td class="noday"> </td><td clas
s="tue">1</td><td class="wed">2</td><td class="thu">3</td><td class="f
ri">4</td><td class="sat">5</td></tr>
<tr><td class="sun">6</td><td class="mon">7</td><td class="tue">8</td>
<td class="wed">9</td><td class="thu">10</td><td class="fri">11</td><t
d class="sat">12</td></tr>
<tr><td class="sun">13</td><td class="mon">14</td><td class="tue">15</
td><td class="wed">16</td><td class="thu">17</td><td class="fri">18</t
d><td class="sat">19</td></tr>
<tr><td class="sun">20</td><td class="mon">21</td><td class="tue">22</
td><td class="wed">23</td><td class="thu">24</td><td class="fri">25</t
d><td class="sat">26</td></tr>
<tr><td class="sun">27</td><td class="mon">28</td><td class="tue">29</
td><td class="wed">30</td><td class="thu">31</td><td class="noday">&nb
sp;</td><td class="noday"> </td></tr>
</table>

>>>
```


Version 1.5
Copyright : Yantra Technologies 2004-2022

11 - Données temporelles → module calendar - fonctions

Fonction	Description
calendar.calendar(year,w=2,l=1,c=6)	Retourne une chaîne multiligne avec un calendrier pour l'année de l'année formaté en trois colonnes séparées par des espaces c. w est la largeur en caractères de chaque jour; chaque ligne a une longueur de 21 * w + 18 + 2 * c. l est le nombre de lignes pour chaque semaine.
calendar.firstweekday( )	Retourne la valeur actuelle de la semaine qui commence chaque semaine. Par défaut, lorsque le calendrier est d'abord importé, c'est 0, ce qui signifie lundi.
calendar.isleap(year)	Renvoie True si l'année est une année bissextile; sinon Faux.
calendar.leapdays(y1,y2)	Renvoie le nombre total de jours intercalaires dans les années à portée (y1, y2).
calendar.month(year,month,w=2,l=1)	Retourne une chaîne multiligne avec un calendrier pour le mois de l'année, une ligne par semaine, plus deux lignes d'en-tête. w est la largeur en caractères de chaque jour; chaque ligne a une longueur de 7 * w + 6. l est le nombre de lignes pour chaque semaine.
calendar.monthcalendar(year,month)	Retourne une liste de listes d'entiers. Chaque sous-liste désigne une semaine. Jours dehors mois de l'année mois année sont mis à 0; jours dans le mois sont fixés à leur jour de mois, 1 an et inférieure.
calendar.monthrange(year,month)	Retourne deux entiers. Le premier est le code de la semaine pour le premier jour du mois de mois dans l'année de l'année; le second est le nombre de jours dans le mois. Codes semaine sont 0 (lundi) à 6 (dimanche); le nombre de mois sont de 1 à 12.
calendar.prcal(year,w=2,l=1,c=6)	Comme l'impression calendar.calendar (année, w, l, c).
calendar.prmonth(year,month,w=2,l=1)	Comme l'impression calendar.month (année, mois, w, l).
calendar.setfirstweekday(weekday)	Définit le premier jour de chaque semaine au code de semaine en semaine. Codes semaine sont 0 (lundi) à 6 (dimanche).
calendar.timegm(tupletime)	L'inverse de time.gmtime: accepte un instant sous forme de temps tuple et retourne au même instant comme un nombre à virgule flottante de secondes depuis l'époque.
calendar.weekday(year,month,day)	Retourne le code de la semaine pour la date donnée. Codes semaine sont 0 (lundi) à 6 (dimanche); nombre de mois sont 1 (Janvier) à 12 (Décembre).

Version 1.5
Copyright : Yantra Technologies 2004-2022

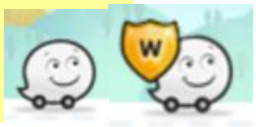
11 - Données temporelles → module calendar - exemples

C:\Python34\python.exe

```
>>> print (calendar.TextCalendar(calendar.SUNDAY).formatyear(2007, 2,
1, 1, 2))
                    2007

      January               February
Su Mo Tu We Th Fr Sa  Su Mo Tu We Th Fr Sa
    1  2  3  4  5  6                 1  2  3
 7  8  9 10 11 12 13   4  5  6  7  8  9 10
14 15 16 17 18 19 20  11 12 13 14 15 16 17
21 22 23 24 25 26 27  18 19 20 21 22 23 24
28 29 30 31           25 26 27 28

       March                 April
Su Mo Tu We Th Fr Sa  Su Mo Tu We Th Fr Sa
             1  2  3   1  2  3  4  5  6  7
 4  5  6  7  8  9 10   8  9 10 11 12 13 14
11 12 13 14 15 16 17  15 16 17 18 19 20 21
18 19 20 21 22 23 24  22 23 24 25 26 27 28
25 26 27 28 29 30 31  29 30

        May                   June
Su Mo Tu We Th Fr Sa  Su Mo Tu We Th Fr Sa
       1  2  3  4  5                    1  2
 6  7  8  9 10 11 12   3  4  5  6  7  8  9
13 14 15 16 17 18 19  10 11 12 13 14 15 16
20 21 22 23 24 25 26  17 18 19 20 21 22 23
27 28 29 30 31        24 25 26 27 28 29 30

        July                  August
Su Mo Tu We Th Fr Sa  Su Mo Tu We Th Fr Sa
 1  2  3  4  5  6  7              1  2  3  4
 8  9 10 11 12 13 14   5  6  7  8  9 10 11
15 16 17 18 19 20 21  12 13 14 15 16 17 18
22 23 24 25 26 27 28  19 20 21 22 23 24 25
```


Version 1.5
Copyright : Yantra Technologies 2004-2022

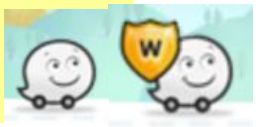
12 - Les Fichiers



- Les types de fichiers
- les différents modes d'ouverture
- Attributs
- Méthodes
- Méthodes de l'OS

Copyright : Yantra Technologies 2004-2022

12 - Les Fichiers -> Les types de Fichiers



Les fichiers textes

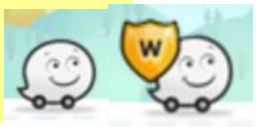
- humainement lisibles. ils contiennent du texte (lettres, ponctuations, nombres, ...). Leur contenu est souvent divisé en lignes. Ces fichiers peuvent être lus et manipulés avec les outils traditionnels Unix: vi, cat, sed, etc. En environnement Unix, les fichiers de configuration dans /etc sont des exemples de fichiers texte.

Les fichiers binaires

- ne contiennent pas (exclusivement) du texte. Ils ne peuvent être convenablement traités que par des logiciels spécialisés. Un exécutable, un fichier PDF, une image JPEG ou un mp3 sont quelques exemples de fichiers binaires.

Version 1.5
Copyright : Yantra Technologies 2004-2022

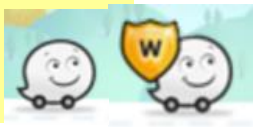
12 - Les Fichiers -> les différents modes d'ouverture



Modes	Description
r	Ouvre un fichier en lecture seule. Le pointeur de fichier est placé au début du fichier (mode par défaut)
rb	Ouvre un fichier en lecture seule dans un format binaire. Le pointeur de fichier est placé au début du fichier (mode par défaut)
r+	Ouvre un fichier pour la lecture et l'écriture. Le pointeur de fichier sera au début du fichier.
rb+	Ouvre un fichier pour la lecture et l'écriture dans un format binaire. Le pointeur de fichier sera au début du fichier.
w	Ouvre un fichier en écriture seule. Remplace le fichier si le fichier existe. Si le fichier n'existe pas, crée un nouveau fichier pour l'écriture.
wb	Ouvre un fichier en écriture seule dans un format binaire. Remplace le fichier si le fichier existe. Si le fichier n'existe pas, crée un nouveau fichier pour l'écriture.
w+	Ouvre un fichier pour l'écriture et la lecture. Remplace le fichier existant si le fichier existe. Si le fichier n'existe pas, crée un nouveau fichier pour la lecture et l'écriture.
wb+	Ouvre un fichier pour l'écriture et la lecture au format binaire. Remplace le fichier existant si le fichier existe. Si le fichier n'existe pas, crée un nouveau fichier pour la lecture et l'écriture.
a	Ouvre un fichier en ajout. Le pointeur de fichier est à la fin du fichier si le fichier existe. Autrement dit, le fichier est dans le mode d'ajout. Si le fichier n'existe pas, il crée un nouveau fichier en écriture.
ab	Ouvre un fichier en ajout au format binaire. Le pointeur de fichier est à la fin du fichier si le fichier existe. Autrement dit, le fichier est dans le mode d'ajout. Si le fichier n'existe pas, il crée un nouveau fichier en écriture.
a+	Ouvre un fichier à la fois de concaténation et de lecture. Le pointeur de fichier est à la fin du fichier si le fichier existe. Le fichier s'ouvre dans le mode d'ajout. Si le fichier n'existe pas, il crée un nouveau fichier pour la lecture et l'écriture.
ab+	Ouvre un fichier à la fois pour l'ajout et la lecture au format binaire. Le pointeur de fichier est à la fin du fichier si le fichier existe. Le fichier s'ouvre dans le mode d'ajout. Si le fichier n'existe pas, il crée un nouveau fichier pour la lecture et l'écriture.

Version 1.5
Copyright : Yantra Technologies 2004-2022

12 - Les Fichiers -> les différents modes d'ouverture : exemple

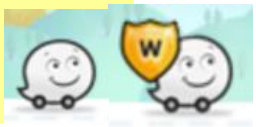


C:\Python34\python.exe

```
>>>
>>> # Ouverture du fichier source
... source = open("table.csv", "a")
>>>
>>> source.write("A1;A2;A3\n")
9
>>> source.write("B1;B2;B3\n")
9
>>>
>>> # Fermerture du fichier source
... source.close()
>>>
>>> # Ouverture du fichier source
... source = open("table.csv", "r")
>>>
>>> source.readline()
'A1;A2;A3\n'
>>> source.readline()
'B1;B2;B3\n'
>>>
```


Version 1.5
Copyright : Yantra Technologies 2004-2022

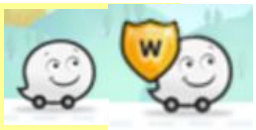
12 - Les Fichiers -> Attributs



Attribut	Description
file.closed	Retourne true si le fichier est fermé, sinon false.
file.mode	Retourne le mode d'accès avec lequel le fichier a été ouvert.
file.name	Retourne le nom du fichier.
file.softspace	Retourne false si l'espace requis pour impression, vrai sinon.

Version 1.5
Copyright : Yantra Technologies 2004-2022

12 - Les Fichiers -> Méthodes



Méthode	Description
file.close()	Fermez le fichier. Un fichier fermé ne peut pas être lu ou écrit ensuite.
file.flush()	Vide la mémoire tampon interne, comme la fflush de stdio.
file.fileno()	Renvoie le descripteur de fichier entier qui est utilisé par l'application sous-jacente de demande des opérations d'E / S du système d'exploitation.
file.isatty()	Retourne Vrai si le fichier est connecté à un périphérique tty (-like), Faux sinon.
file.next()	Retourne la ligne suivante dans le fichier à chaque fois qu'il est appelé.
file.read([size])	Lit size octets du fichier (moins si la lecture atteint EOF avant d'obtenir la taille octets).
file.readline([size])	Lit une ligne entière à partir du fichier. Un caractère de nouvelle ligne est maintenu dans la chaîne.
file.readlines([sizehint])	Lit jusqu'à EOF en utilisant readline () et retourne une liste contenant les lignes. Si l'argument sizeHint option est présente, au lieu de lire jusqu'à EOF, des lignes entières totalisant octets environ sizeHint (éventuellement après arrondissement à une taille de mémoire tampon interne) sont lues.
file.seek(offset[, whence])	Définit la position actuelle du fichier.
file.tell()	Retourne la position actuelle du fichier
file.truncate([size])	Tronque la taille du fichier. Si l'argument optionnel de taille est présent, le fichier est tronqué à (au plus) size.
file.write(str)	Écrit une chaîne dans le fichier. Il n'y a pas de valeur de retour.
file.writelines(sequence)	Ecrit une séquence de chaînes dans le fichier. La séquence peut être n'importe quel objet, généralement une liste de chaînes.

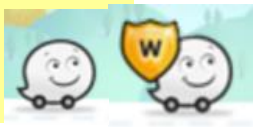
Version 1.5
Copyright : Yantra Technologies 2004-2022

12 - Les Fichiers -> Méthodes de l'OS

Méthode	Description
os.access(path, mode)	Utilise le uid / gid réel pour tester l'accès au chemin.
os.chdir(path)	Change le répertoire de travail courant au chemin
os.chflags(path, flags)	Définit les indicateurs de chemin d'accès aux drapeaux numériques.
os.chmod(path, mode)	Change le mode de chemin vers le mode numérique donné.
os.chown(path, uid, gid)	Change le propriétaire et le groupe id de chemin d'accès au uid et gid numériques.
os.chroot(path)	Change le répertoire racine du processus actuel de chemin.
os.close(fd)	Ferme le descripteur de fichier fd.
os.closerange(fd_low, fd_high)	Ferme tous les descripteurs de fichiers à partir de fd_low (inclus) au fd_high (exclusif), en ignorant les erreurs.
os.dup(fd)	Retourne une copie de descripteurs de fichier fd.
os.dup2(fd, fd2)	Duplique un descripteur de fichier à fd2, en fermant le dernier si nécessaire.

Version 1.5
Copyright : Yantra Technologies 2004-2022

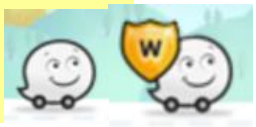
12 - Les Fichiers -> Méthodes de l'OS



Méthode	Description
os.fchdir(fd)	Change le répertoire de travail courant dans le répertoire indiqué par le descripteur de fichier fd.
os.fchmod(fd, mode)	Change le mode du fichier donné par fd au mode numérique.
os.fchown(fd, uid, gid)	Change le propriétaire et le groupe id du fichier donné par fd à l'uid et gid numériques.
os.fdatasync(fd)	Synchronisation d'écriture du fichier de descripteur fd sur le disque.
os.fdopen(fd[, mode[, bufsize]])	Retourne un objet fichier ouvert et connecté au descripteur de fichier fd.
os.fpathconf(fd, name)	Retourne le système d'informations de configuration correspondant à un fichier ouvert; name spécifie la valeur de configuration à récupérer.
os.fstat(fd)	Retour du statut descripteur de fichier, comme stat ().
os.fstatvfs(fd)	Retour d'informations sur le système de fichiers contenant le fichier associé à un descripteur de fichier, comme statvfs ().
os.fsync(fd)	Synchronisation d'écriture du fichier de descripteur fd sur le disque.
os.ftruncate(fd, length)	Tronque le fichier correspondant au descripteur de fichier fd, de sorte qu'il est au plus length octets en taille.
os.getcwd()	Retourne une chaîne représentant le répertoire de travail courant.
os.getcwdu()	Retourne un objet Unicode représentant le répertoire de travail courant.
os.isatty(fd)	Retour Vrai si le descripteur de fichier fd est ouvert et connecté à un terminal (-like), Faux sinon.

Version 1.5
Copyright : Yantra Technologies 2004-2022

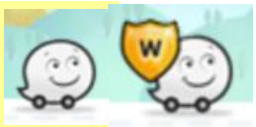
12 - Les Fichiers -> Méthodes de l'OS



Méthode	Description
os.lchflags(path, flags)	Définir les indicateurs de chemin d'accès aux drapeaux numériques, comme chflags (), mais ne pas suivre les liens symboliques.
os.lchmod(path, mode)	Changer le mode de chemin vers le mode numérique.
os.lchown(path, uid, gid)	Changer le propriétaire et le groupe id de chemin d'accès au uid et gid numériques. Cette fonction ne fera pas suivre les liens symboliques.
os.link(src, dst)	Créer un lien pointant de src à dst nommé.
os.listdir(path)	Retourne une liste contenant les noms des entrées dans le répertoire donné par le chemin.
os.lseek(fd, pos, how)	Régle la position actuelle du descripteur de fichier fd à la position pos, modifiée par la façon how.
os.lstat(path)	Comme stat (), mais ne suit pas les liens symboliques.
os.major(device)	Extraire le numéro de périphérique majeur d'un certain nombre de périphérique brut.
os.makedev(major, minor)	Composez un numéro de périphérique brut à partir des numéros périphériques majeur et mineur.
os.minor(device)	Extraire le numéro mineur de périphérique à partir d'un numéro de périphérique brut.
os.mkfifo(path[, mode])	Créer un FIFO (un tube nommé) du chemin nommé avec le mode numérique. Le mode par défaut est 0666 (octal).
os.open(file, flags[, mode])	Ouvre le fichier file et définit divers drapeaux selon flag et peut-être son mode en fonction du mode.
os.pathconf(path, name)	Retourne le système d'informations de configuration correspondant à un fichier nommé.
os.popen(command[, mode[, bufsize]])	Ouvre un pipe de commande.
os.readlink(path)	Retourne une chaîne représentant le chemin vers lequel pointe le lien symbolique.
os.removedirs(path)	Supprime les répertoires de manière récursive.

Version 1.5
Copyright : Yantra Technologies 2004-2022

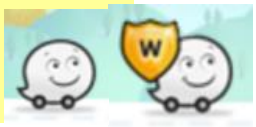
12 - Les Fichiers -> Méthodes de l'OS



Méthode	Description
os.renames(old, new)	Fonction fichier de changement de nom.
os.rmdir(path)	Retire le chemin du répertoire
os.stat(path)	Effectue un appel système stat sur le chemin donné.
os.stat_float_times([newvalue])	Détermine si stat_result représente l'horodatage comme des objets flottants.
os.statvfs(path)	Effectue un appel système statvfs sur le chemin donné.
os.symlink(src, dst)	Crée un lien symbolique src dst nommé.
os.tcgetpgrp(fd)	Retourne le groupe de processus associé à la borne donnée par fd (un descripteur de fichier ouvert, retourné par open ()).
os.tcsetpgrp(fd, pg)	Règle le groupe de processus associé à la borne donnée par fd (un descripteur de fichier ouvert, retourné par open ()) à p.
os.tempnam([dir[, prefix]])	Renvoie un nom de chemin unique qui est raisonnable pour la création d'un fichier temporaire.
os.tmpfile()	Retourne un nouvel objet fichier ouvert en mode mise à jour (w + b).
os.tmpnam()	Renvoie un nom de chemin unique qui est raisonnable pour la création d'un fichier temporaire.
os.ttyname(fd)	Retourne une chaîne qui spécifie le terminal associé à un descripteur de fichier. Si fd n'est pas associé à un dispositif terminal, une exception est levée.
os.unlink(path)	Retire le chemin du fichier.
os.utime(path, times)	Définit l'accès et les temps de modification du fichier spécifié par path.
os.walk(top[, topdown=True[, onerror=None[, followlinks=False]]])	Généré des noms de fichiers dans une arborescence de répertoires en parcourant l'arbre soit de haut en bas ou de bas en haut.
os.write(fd, str)	Écrit la chaîne str le descripteur de fichier fd. Retourne le nombre d'octets effectivement écrits.

Version 1.5
Copyright : Yantra Technologies 2004-2022

12 - Les Fichiers -> Méthodes de l'OS - exemple



C:\Python34\python.exe

```
>>> os.rename("table.csv","array.csv")
>>> # Ouverture du fichier source
... source = open("array.csv", "r")
>>>
>>> source.readline()
'A1;A2;A3\n'
>>> source.readline()
'B1;B2;B3\n'
>>>
>>> # Fermerture du fichier source
... source.close()
>>> os.getcwd()
'C:\\Python34'
```

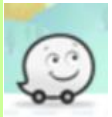

Copyright : Yantra Technologies 2004-2022

13 - Bibliothèques standards

https://docs.python.org/3/library/

Version 1.5
Copyright : Yantra Technologies 2004-2022

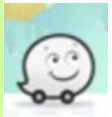
14 - Différences python 2.7.x avec 3.x



- La fonction print
- La division entière
- Unicode
- Range & xrange
- Lever une exception
- Gestion d’une exception
- Fonction next() et méthode.next()
- La boucle et l’espace de nommage global
- Comparaison non ordonnée
- L'analyse des entrées utilisateur via input
- Itération sur les listes

Copyright : Yantra Technologies 2004-2022

14 - Différences python 2.7.x avec 3.x -> La fonction print



Python 2

```
print 'Python', python_version()
print 'Hello, World!'
print('Hello, World!')
print "text", ; print 'print more text on the same line'
```

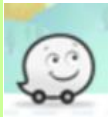
Python 3

```
print('Python', python_version())
print('Hello, World!')

print("some text,", end="")
print(' print more text on the same line')
```

http://nbviewer.ipython.org/github/rasbt/python_reference/blob/master/tutorials/key_differences_between_python_2_and_3.ipynb

Version 1.5
Copyright : Yantra Technologies 2004-2022

14 - Différences python 2.7.x avec 3.x -> La division entière



Python 2

```
:  imprimer Python ', python_version ()
print '3/2 =', 3/2
print '3 // 2 =', 3 // 2
print '3 / 2.0 =', 3 / 2.0
print '3 // 2.0 =', 3 // 2.0
```

```
Python 2.7.6
3/2 = 1
3 // 2 = 1
3 / 2,0 = 1,5
3 // 2.0 = 1.0
```

Python 3

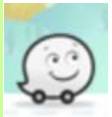
```
imprimer (Python ', python_version ())
print ("3/2 = ', 3/2)
print ('3 // 2 =', 3 // 2)
print ('3 / 2.0 =', 3 / 2.0)
print ('3 // 2.0 =', 3 // 2.0)
```

```
Python 3.4.1
3/2 = 1,5
3 // 2 = 1
3 / 2,0 = 1,5
3 // 2.0 = 1.0
```

http://nbviewer.ipython.org/github/rasbt/python_reference/blob/master/tutorials/key_differences_between_python_2_and_3.ipynb

Version 1.5
Copyright : Yantra Technologies 2004-2022

14 - Différences python 2.7.x avec 3.x -> Unicode



Python 2

```
print 'Python', python_version()
```

```
Python 2.7.6
```

```
print type(unicode('this is like a python3 str type'))
```

```
<type 'unicode'>
```

```
print type(b'byte type does not exist')
```

```
<type 'str'>
```

```
print 'they are really' + b' the same'
```

```
they are really the same
```

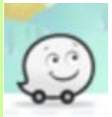
```
print type(bytearray(b'bytearray oddly does exist though'))
```

```
<type 'bytearray'>
```

http://nbviewer.ipython.org/github/rasbt/python_reference/blob/master/tutorials/key_differences_between_python_2_and_3.ipynb

Version 1.5
Copyright : Yantra Technologies 2004-2022

14 - Différences python 2.7.x avec 3.x -> Unicode



Python 3

```
print('Python', python_version())
print('strings are now utf-8 \u03BCnico\u0394é!')
```

```
Python 3.4.1
strings are now utf-8 µnicoΔé!
```

```
print('Python', python_version(), end="")
print(' has', type(b' bytes for storing data'))
```

```
Python 3.4.1 has <class 'bytes'>
```

```
print('and Python', python_version(), end="")
print(' also has', type(bytearray(b'bytearrays')))
```

```
and Python 3.4.1 also has <class 'bytearray'>
```

```
'note that we cannot add a string' + b'bytes for data'
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-13-d3e8942ccf81> in <module>()
----> 1 'note that we cannot add a string' + b'bytes for data'
```

http://nbviewer.ipython.org/github/rasbt/python_reference/blob/master/tutorials/key_differences_between_python_2_and_3.ipynb

Version 1.5
Copyright : Yantra Technologies 2004-2022

14 - Différences python 2.7.x avec 3.x -> range & xrange

Python 3

```
print('Python', python_version())

print('\ntiming range()')
%timeit test_range(n)
```

```
Python 3.4.1

timing range()
1000 loops, best of 3: 520 µs per loop
```

Python 2

```
print 'Python', python_version()

print '\ntiming range()'
%timeit test_range(n)

print '\n\ntiming xrange()'
%timeit test_xrange(n)
```

```
Python 2.7.6

timing range()
1000 loops, best of 3: 433 µs per loop


timing xrange()
1000 loops, best of 3: 350 µs per loop
```

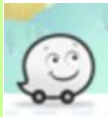
```
print(xrange(10))
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-5-5d8f9b79ea70> in <module>()
----> 1 print(xrange(10))

NameError: name 'xrange' is not defined
```

http://nbviewer.ipython.org/github/rasbt/python_reference/blob/master/tutorials/key_differences_between_python_2_and_3.ipynb

Version 1.5
Copyright : Yantra Technologies 2004-2022

14 - Différences python 2.7.x avec 3.x -> Lever une exception



Python 2

```
print 'Python', python_version()
```

```
Python 2.7.6
```

```
raise IOError, "file error"
```

```
---------------------------------------------------------------------------
IOError                                   Traceback (most recent call last)
<ipython-input-8-25f049caebb0> in <module>()
----> 1 raise IOError, "file error"

IOError: file error
```

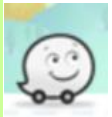
```
raise IOError("file error")
```

```
---------------------------------------------------------------------------
IOError                                   Traceback (most recent call last)
<ipython-input-9-6f1c43f525b2> in <module>()
----> 1 raise IOError("file error")

IOError: file error
```

http://nbviewer.ipython.org/github/rasbt/python_reference/blob/master/tutorials/key_differences_between_python_2_and_3.ipynb

Copyright : Yantra Technologies 2004-2022

14 - Différences python 2.7.x avec 3.x -> Lever une exception



Python 3

```
print('Python', python_version())
```

```
Python 3.4.1
```

```
raise IOError, "file error"
```

```
  File "<ipython-input-10-25f049caebb0>", line 1
    raise IOError, "file error"
                 ^
SyntaxError: invalid syntax
```

The proper way to raise an exception in Python 3:

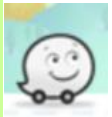
```
print('Python', python_version())
raise IOError("file error")
```

```
Python 3.4.1

---------------------------------------------------------------------------
OSError                                   Traceback (most recent call last)
<ipython-input-11-c350544d15da> in <module>()
      1 print('Python', python_version())
----> 2 raise IOError("file error")

OSError: file error
```

http://nbviewer.ipython.org/github/rasbt/python_reference/blob/master/tutorials/key_differences_between_python_2_and_3.ipynb

Version 1.5
Copyright : Yantra Technologies 2004-2022

14 - Différences python 2.7.x avec 3.x -> Gestion d'une exception



Python 2

```
print 'Python', python_version()
try:
    let_us_cause_a_NameError
except NameError, err:
    print err, '--> our error message'
```

```
Python 2.7.6
name 'let_us_cause_a_NameError' is not defined --> our error message
```

Python 3

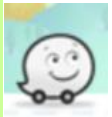
```
print('Python', python_version())
try:
    let_us_cause_a_NameError
except NameError as err:
    print(err, '--> our error message')
```

```
Python 3.4.1
name 'let_us_cause_a_NameError' is not defined --> our error message
```

http://nbviewer.ipython.org/github/rasbt/python_reference/blob/master/tutorials/key_differences_between_python_2_and_3.ipynb

Version 1.5
Copyright : Yantra Technologies 2004-2022

14 - Différences python 2.7.x avec 3.x -> Fonction next() et méthode .next()



Python 2

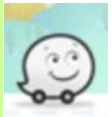
```
print 'Python', python_version()

my_generator = (letter for letter in 'abcdefg')

next(my_generator)
my_generator.next()
```

```
Python 2.7.6

'b'
```

http://nbviewer.ipython.org/github/rasbt/python_reference/blob/master/tutorials/key_differences_between_python_2_and_3.ipynb

Copyright : Yantra Technologies 2004-2022

14 - Différences python 2.7.x avec 3.x -> Fonction next() et méthode .next()



Python 3

```
print('Python', python_version())

my_generator = (letter for letter in 'abcdefg')

next(my_generator)
```

```
Python 3.4.1

'a'
```

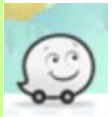
```
my_generator.next()
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-14-125f388bb61b> in <module>()
----> 1 my_generator.next()

AttributeError: 'generator' object has no attribute 'next'
```

http://nbviewer.ipython.org/github/rasbt/python_reference/blob/master/tutorials/key_differences_between_python_2_and_3.ipynb

Copyright : Yantra Technologies 2004-2022

14 - Différences python 2.7.x avec 3.x -> La boucle et l'espace de nommage global



Python 2

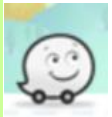
```
: print 'Python', python_version()

  i = 1
  print 'before: i =', i

  print 'comprehension: ', [i for i in range(5)]

  print 'after: i =', i
```

```
Python 2.7.6
before: i = 1
comprehension:  [0, 1, 2, 3, 4]
after: i = 4
```

http://nbviewer.ipython.org/github/rasbt/python_reference/blob/master/tutorials/key_differences_between_python_2_and_3.ipynb

Version 1.5
Copyright : Yantra Technologies 2004-2022

14 - Différences python 2.7.x avec 3.x -> La boucle et l'espace de nommage global



Python 3

```
print('Python', python_version())

i = 1
print('before: i =', i)

print('comprehension:', [i for i in range(5)])

print('after: i =', i)
```

```
Python 3.4.1
before: i = 1
comprehension: [0, 1, 2, 3, 4]
after: i = 1
```

http://nbviewer.ipython.org/github/rasbt/python_reference/blob/master/tutorials/key_differences_between_python_2_and_3.ipynb

Version 1.5
Copyright : Yantra Technologies 2004-2022

14 - Différences python 2.7.x avec 3.x -> Comparaison non ordonnée

Python 2

```
print 'Python', python_version()
print "[1, 2] > 'foo' = ", [1, 2] > 'foo'
print "(1, 2) > 'foo' = ", (1, 2) > 'foo'
print "[1, 2] > (1, 2) = ", [1, 2] > (1, 2)
```

```
Python 2.7.6
[1, 2] > 'foo' =  False
(1, 2) > 'foo' =  True
[1, 2] > (1, 2) =  False
```

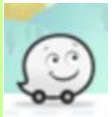
Python 3

```
print('Python', python_version())
print("[1, 2] > 'foo' = ", [1, 2] > 'foo')
print("(1, 2) > 'foo' = ", (1, 2) > 'foo')
print("[1, 2] > (1, 2) = ", [1, 2] > (1, 2))
```

```
Python 3.4.1

---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-16-a9031729f4a0> in <module>()
      1 print('Python', python_version())
----> 2 print("[1, 2] > 'foo' = ", [1, 2] > 'foo')
      3 print("(1, 2) > 'foo' = ", (1, 2) > 'foo')
      4 print("[1, 2] > (1, 2) = ", [1, 2] > (1, 2))

TypeError: unorderable types: list() > str()
```


Version 1.5
Copyright : Yantra Technologies 2004-2022

14 - Différences python 2.7.x avec 3.x -> L'analyse des entrées utilisateur via input

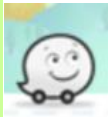


Python 2

```
Python 2.7.6
[GCC 4.0.1 (Apple Inc. build 5493)] on darwin
Type "help", "copyright", "credits" or "license" for more information.

>>> my_input = input('enter a number: ')

enter a number: 123

>>> type(my_input)
<type 'int'>

>>> my_input = raw_input('enter a number: ')

enter a number: 123

>>> type(my_input)
<type 'str'>
```


Version 1.5
Copyright : Yantra Technologies 2004-2022

14 - Différences python 2.7.x avec 3.x -> L'analyse des entrées utilisateur via input

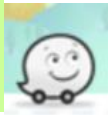


Python 3

```
Python 3.4.1
[GCC 4.2.1 (Apple Inc. build 5577)] on darwin
Type "help", "copyright", "credits" or "license" for more information.


>>> my_input = input('enter a number: ')


enter a number: 123


>>> type(my_input)
<class 'str'>
```


Version 1.5
Copyright : Yantra Technologies 2004-2022

14 - Différences python 2.7.x avec 3.x -> Itération sur les listes



Python 2

```
print 'Python', python_version()

print range(3)
print type(range(3))
```

```
Python 2.7.6
[0, 1, 2]
<type 'list'>
```

Python 3

```
print('Python', python_version())

print(range(3))
print(type(range(3)))
print(list(range(3)))
```

```
Python 3.4.1
range(0, 3)
<class 'range'>
[0, 1, 2]
```


Version 1.5
Copyright : Yantra Technologies 2004-2022

14a : évolution de Python 3.5 à Python 3.9

- https://docs.python.org/fr/3/library/typing.html#newtype
- https://zestedesavoir.com/articles/175/sortie-de-python-3-5/#2-principales-nouveautes
- https://zestedesavoir.com/articles/1540/sortie-de-python-3-6/
- https://linuxfr.org/news/sortie-de-python-3-7
- https://www.docstring.fr/blog/les-nouveautes-de-python-38/
- https://www.docstring.fr/blog/les-nouveautes-de-python-39/

https://fr.wikipedia.org/wiki/Python_(langage)

Version 1.5
Copyright : Yantra Technologies 2004-2022

14a : évolution de Python 3.5 à Python 3.9

typing — Prise en charge des annotations de type

Nouveau dans la version 3.5.

Code source : Lib/typing.py

> **Note:** Le moteur d'exécution Python n'applique pas les annotations de type pour les fonctions et les variables. Elles peuvent être utilisées par des outils tiers tels que les contrôleurs de type, les IDE, les analyseurs de code, etc.

This module provides runtime support for type hints. The most fundamental support consists of the types `Any`, `Union`, `Callable`, `TypeVar`, and `Generic`. For a full specification, please see **PEP 484**. For a simplified introduction to type hints, see **PEP 483**.

La fonction ci-dessous prend et renvoie une chaîne de caractères, et est annotée comme suit :

```
def greeting(name: str) -> str:
    return 'Hello ' + name
```

La fonction `greeting` s'attend à ce que l'argument `name` soit de type `str` et le type de retour `str`. Les sous-types sont acceptés comme arguments.

Relevant PEPs

Since the initial introduction of type hints in **PEP 484** and **PEP 483**, a number of PEPs have modified and enhanced Python's framework for type annotations. These include:

- **PEP 526**: Syntax for Variable Annotations
  Introducing syntax for annotating variables outside of function definitions, and `ClassVar`
- **PEP 544**: Protocols: Structural subtyping (static duck typing)

Copyright : Yantra Technologies 2004-2022

Alias de type

A type alias is defined by assigning the type to the alias. In this example, `Vector` and `list[float]` will be treated as interchangeable synonyms:

```
Vector = list[float]

def scale(scalar: float, vector: Vector) -> Vector:
    return [scalar * num for num in vector]

# typechecks; a list of floats qualifies as a Vector.
new_vector = scale(2.0, [1.0, -4.2, 5.4])
```

Les alias de type sont utiles pour simplifier les signatures complexes. Par exemple :

```
from collections.abc import Sequence

ConnectionOptions = dict[str, str]
Address = tuple[str, int]
Server = tuple[Address, ConnectionOptions]

def broadcast_message(message: str, servers: Sequence[Server]) -> None:
    ...

# The static type checker will treat the previous type signature as
# being exactly equivalent to this one.
def broadcast_message(
        message: str,
        servers: Sequence[tuple[tuple[str, int], dict[str, str]]]) -> None:
    ...
```

Notez que `None` comme indication de type est un cas particulier et est remplacé par `type(None)`.

Version 1.5
Copyright : Yantra Technologies 2004-2022

14a : évolution de Python 3.5 à Python 3.9

Les expressions d'affectation (l'opérateur Walrus)

:=

https://www.docstring.fr/blog/les-nouveautes-de-python-38/

Avec Python 3.7

```
f = open('mon_fichier.txt')
line = f.readline()

while line:
    print(line)
    line = f.readline()

f.close()
```

Avec Python 3.8

```
f = open('mon_fichier.txt')

while line := f.readline():
    print(line)

f.close()
```


Copyright : Yantra Technologies 2004-2022

14b : Nouveautés 3.10 : 4/10/21

- https://docs.python.org/fr/3.10/whatsnew/3.10.html
- https://www.pythoniste.fr/python/les-nouveautes-de-la-version-3-10-de-python/
- https://zestedesavoir.com/articles/4041/sortie-de-python-3-10/

Version 1.5
Copyright : Yantra Technologies 2004-2022

14b : Nouveautés 3.10 : 4/10/21

Avec le **structural pattern matching** nous pouvons remplacer les `if`, `elif` et les `else` par des `match` et des `case` :

```
http_code == "418"

match http_code:
    case "200":
        print("OK")
    case "404":
        print("Not Found")
    case "500":
        print("Internal Error")
    case _:
        print("Code not found")
```

https://www.pythoniste.fr/python/les-nouveautes-de-la-version-3-10-de-python/

Version 1.5
Copyright : Yantra Technologies 2004-2022

14b : Nouveautés 3.10 : 4/10/21

```
dict_a = {
    'id': 1,
    'meta': {
        'source': 'abc',
        'location': 'nord'
    }
}

dict_b = {
    'id': 2,
    'source': 'def',
    'location': 'nord'
}
```

```
for d in [dict_a, dict_b, "test"]:
    match d :
        case {'id': ident,
              'meta':{'source': source,
                      'location': loc}}:
            print(ident, source, loc)

        case {'id': ident,
              'source': source,
              'location': loc}:
            print(ident, source, loc)

        case _:
            print("no match")

1 abc nord
2 def nord
no match
```

https://www.docstring.fr/blog/les-nouveautes-de-python-310/
https://www.pythoniste.fr/python/les-nouveautes-de-la-version-3-10-de-python/

Version 1.5
Copyright : Yantra Technologies 2004-2022

14b : Nouveautés 3.10 : 4/10/21

```
with (open("fichier1.txt", "r") as fin,
      open("fichier2.txt", "w") as fout):

      fout.write(fin.read())
```


Copyright : Yantra Technologies 2004-2022

15 - Compléments

- Base de données
  - http://apprendre-python.com/page-database-data-base-donnees-query-sql-mysql-postgre-sqlite
  - https://wiki.python.org/moin/DatabaseProgramming/
- LDAP *(Lightweight Directory Access Protocol) protocole destiné à accéder a des données présentés dans un annuaire*
  - http://condorcet.iris.free.fr/spip.php?article43
  - http://python-ldap.org/
- XML
  - **http://apprendre-python.com/page-xml-python-xpath**
  - https://wiki.deimos.fr/Initiation_%C3%A0_quelques_modules_Python_int%C3%A9ressants
- Travailler avec des medias
  - https://wiki.python.org/moin/AudioVideo
  - http://pymedia.org/features.html

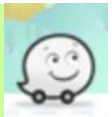
Version 1.5
Copyright : Yantra Technologies 2004-2022

15 - Compléments

- Génération contenu
  - https://pypi.python.org/pypi/pdfdocument/3.1
  - https://pypi.python.org/pypi/odfpy
- Programmation parallèle
  - http://www.parallelpython.com/
  - http://materials.jeremybejarano.com/MPIwithPython/
- Programmation système et réseau
  - http://fsincere.free.fr/isn/python/cours_python_reseau.php
  - http://apprendre-python.com/page-reseaux-sockets-python-port
- Bonnes Pratiques
  - https://larlet.fr/david/biologeek/archives/20080511-bonnes-pratiques-et-astuces-python/
  - http://www.ulb.ac.be/di/verif/tmassart/Informatique/html/chapitre16.html

Version 1.5
Copyright : Yantra Technologies 2004-2022

16 - Environnement Graphique : TKINTER



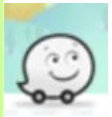
- le plus petit programme graphique
- Widget
- Widget : canevas
- Exemple : Bip Bip et Coyote

  - Cadre
  - Définition des images
  - Affichage des images
  - Insérer définition des boutons
  - Fonction bas, haut
  - Fonction gauche, droite
  - Gestion flèche clavier

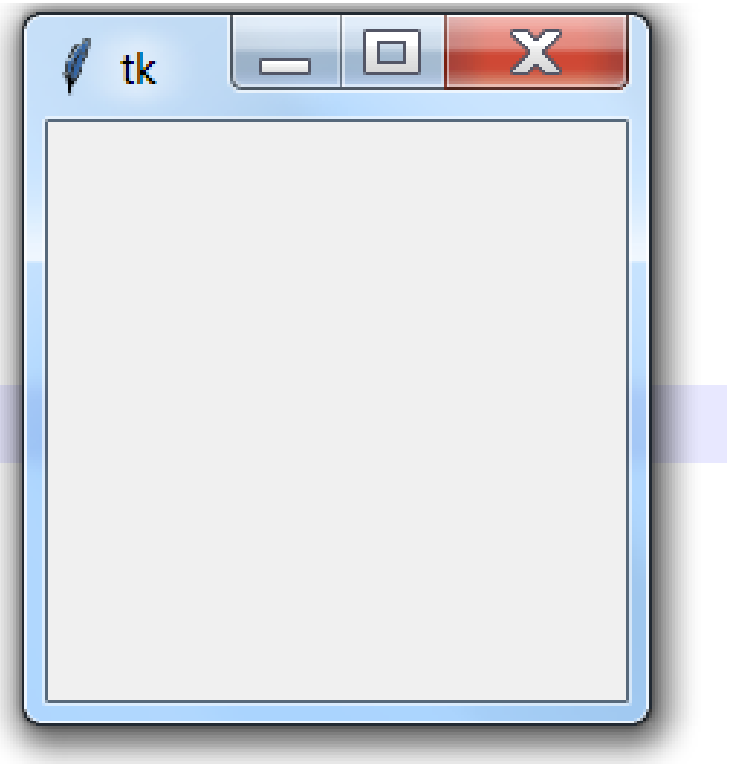
https://docs.python.org/fr/3/library/tkinter.html
https://python.doctor/page-tkinter-interface-graphique-python-tutoriel

Version 1.5
Copyright : Yantra Technologies 2004-2022

16 - Environnement Graphique : TKINTER -> le plus petit programme graphique



```
from tkinter import *

fen = Tk()
fen.mainloop()
```



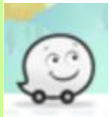
l'essentiel se passe à l'intérieur de application.mainloop()

https://fr.wikibooks.org/wiki/Programmation_Python/Tkinter
https://docs.python.org/3.4/library/tkinter.html
http://www.tutorialspoint.com/python/python_gui_programming.htm

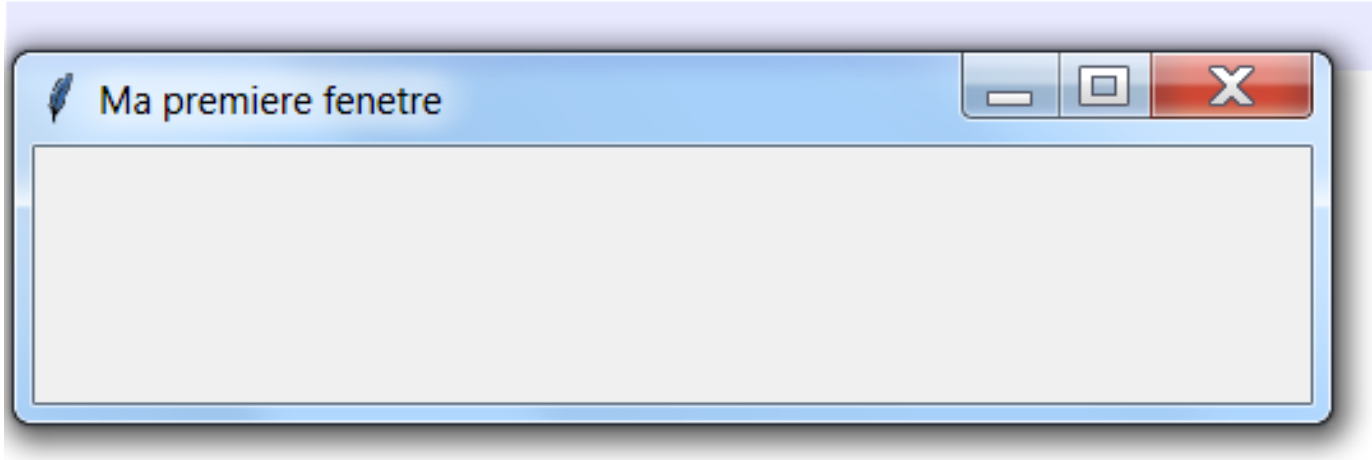
https://youtu.be/rcACl0sUJeQ

Version 1.5
Copyright : Yantra Technologies 2004-2022

16 - Environnement Graphique : TKINTER -> le plus petit programme graphique

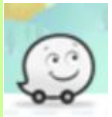


```
from tkinter import *

fen = Tk()
fen.title("Ma premiere fenetre")
fen.geometry("500x100")
fen.mainloop()
```

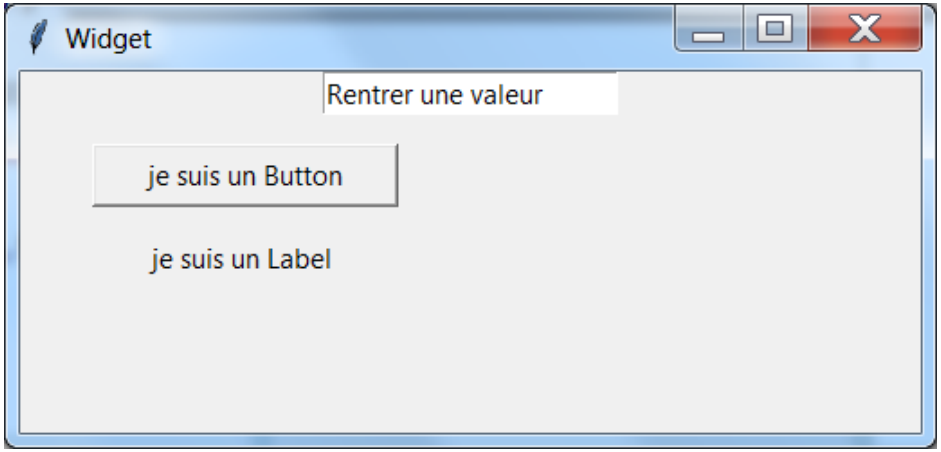


Copyright : Yantra Technologies 2004-2022

16 - Environnement Graphique : TKINTER -> Widget



```
from tkinter import *

fen = Tk()
fen.title("Widget")
fen.geometry("500x200")

bou =Button(fen,text="je suis un Button",width=20)
bou.place(x=40,y=40)

lab =Label(fen,text="je suis un Label",width=20)
lab.place(x=40,y=90)

value = StringVar()
value.set("Rentrer une valeur")

ent =Entry(fen,textvariable=value,width=20)
ent.pack()

fen.mainloop()
```



Copyright : Yantra Technologies 2004-2022

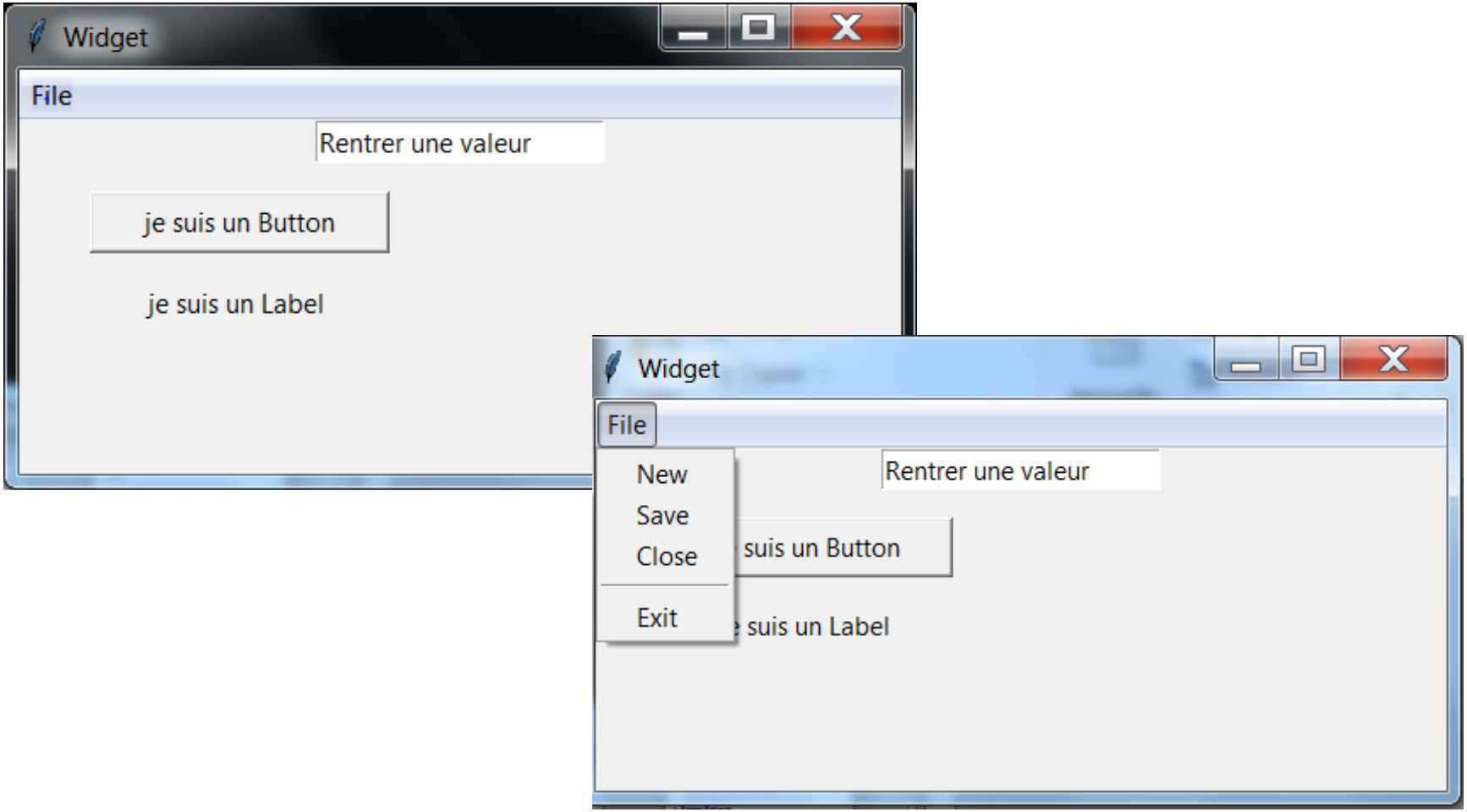
Yantra Technologies

16 - Environnement Graphique : TKINTER -> Widget

```
from tkinter import *
def neFaitRien():
    filewin = Toplevel(fen)
    button = Button(filewin, text="Ne fait rien")
    button.pack()

fen = Tk()
fen.title("Widget")
fen.geometry("500x200")

menubar = Menu(fen)
filemenu = Menu(menubar, tearoff=0)
filemenu.add_command(label="New", command=neFaitRien)
filemenu.add_command(label="Save", command=neFaitRien)
filemenu.add_command(label="Close", command=neFaitRien)
filemenu.add_separator()
filemenu.add_command(label="Exit", command=quit)
menubar.add_cascade(label="File", menu=filemenu)
fen.config(menu=menubar)

bou =Button(fen,text="je suis un Button",width=20)
bou.place(x=40,y=40)
lab =Label(fen,text="je suis un Label",width=20)
lab.place(x=40,y=90)
value = StringVar()
value.set("Rentrer une valeur")
ent =Entry(fen,textvariable=value,width=20)
ent.pack()

fen.mainloop()
```


Copyright : Yantra Technologies 2004-2022

16 - Environnement Graphique : TKINTER -> Widget



Copyright : Yantra Technologies 2004-2022

16 - Environnement Graphique : TKINTER -> Widget

Operateur	Description
Message	Le widget de message est utilisé pour afficher des champs texte multiligne pour accepter les valeurs d'un utilisateur.
Radiobutton	Le widget Radiobutton est utilisé pour afficher un certain nombre d'options que les boutons de radio. L'utilisateur peut sélectionner une seule option à la fois.
Scale	Le widget d'échelle est utilisé pour fournir un widget de curseur.
Scrollbar	Le widget de défilement est utilisée pour ajouter une capacité de défilement à divers widgets, tels que des boîtes de liste.
Text	Le widget texte est utilisé pour afficher du texte en plusieurs lignes.
Toplevel	Le widget de premier niveau est utilisé pour fournir une fenêtre récipient séparé.
Spinbox	Le widget Spinbox est une variante du widget Tkinter norme d'entrée, qui peut être utilisé pour sélectionner à partir d'un nombre fixe de valeurs.
PanedWindow	Un PanedWindow est un widget container qui peut contenir un certain nombre de volets, disposés horizontalement ou verticalement.
LabelFrame	Un labelframe est un widget simple conteneur. Son but principal est d'agir comme une entretoise ou un conteneur pour fenêtres mises en page complexes.
tkMessageBox	Ce module est utilisé pour afficher les boîtes de message dans vos applications.

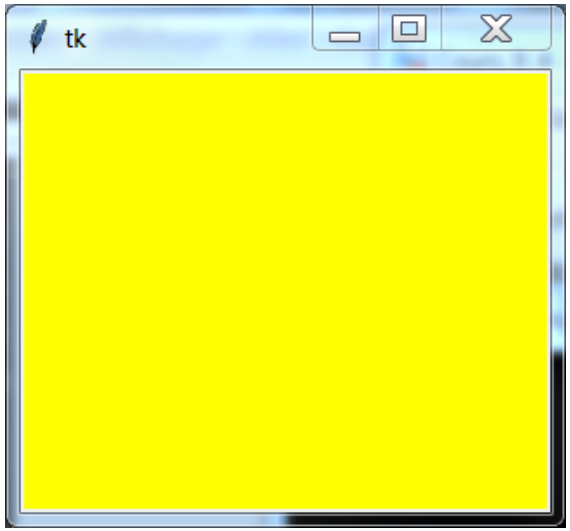
Version 1.5
Copyright : Yantra Technologies 2004-2022

16 - Environnement Graphique : TKINTER -> Widget

Operateur	Description
Button	Le widget de bouton est utilisé pour afficher des boutons dans votre application.
Canvas	Le widget Canvas est utilisé pour dessiner des formes, telles que des lignes, des ovales, des polygones et de rectangles, dans votre application.
Checkbutton	Le widget Checkbutton est utilisé pour afficher un certain nombre d'options que les cases à cocher. L'utilisateur peut sélectionner plusieurs options à la fois.
Entry	Le widget de saisie est utilisé pour afficher un champ de texte d'une seule ligne pour accepter les valeurs d'un utilisateur.
Frame	Le widget de trame est utilisé comme un widget container d'organiser d'autres widgets.
Label	Le widget label est utilisé pour fournir une légende sur une seule ligne pour d'autres widgets. Il peut aussi contenir des images.
Listbox	Le widget Listbox est utilisé pour fournir une liste d'options à un utilisateur.
Menubutton	Le widget Menubutton est utilisé pour afficher les menus dans votre application.
Menu	Le widget Menu est utilisé pour fournir diverses commandes à un utilisateur. Ces commandes sont contenues à l'intérieur Menubutton.

Version 1.5
Copyright : Yantra Technologies 2004-2022

16 - Environnement Graphique : TKINTER -> Widget : canevas

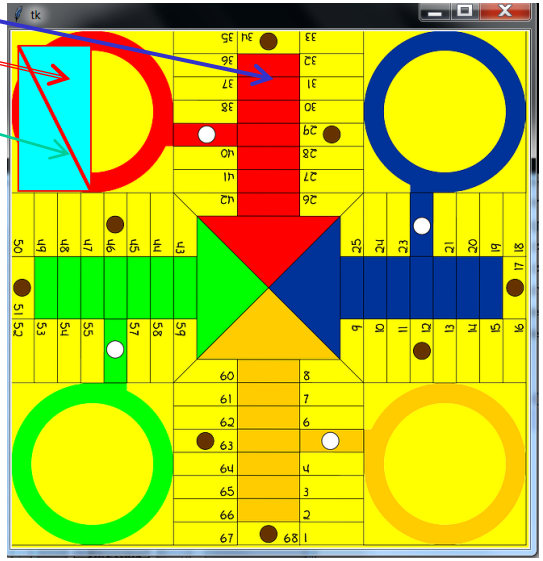



```
from tkinter import *

fenetre = Tk()
canevas = Canvas(fenetre, bg="yellow", height=250, width=300)

canevas.pack()
fenetre.mainloop()
```


Version 1.5
Copyright : Yantra Technologies 2004-2022

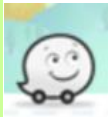
16 - Environnement Graphique : TKINTER -> Widget : canevas - create

```
from tkinter import *

fenetre = Tk()
canevas = Canvas(fenetre, bg="yellow", height=640, width=640)
x1=10
y1=20
x2=100
y2=200
rectangle = canevas.create_rectangle(x1,y1,x2,y2,width=2,fill="cyan",outline="red")
fichier=PhotoImage(file="C:\My_python_lib\ludo-148865_640.png")
ligne = canevas.create_line(x1,y1,x2,y2,width=5,fill="red")
image = canevas.create_image(0,0,image=fichier,anchor="nw")
canevas.tag_lower(image)
canevas.tag_raise(rectangle)
canevas.tag_raise(ligne)

canevas.pack()
fenetre.mainloop()
```



Version 1.5
Copyright : Yantra Technologies 2004-2022

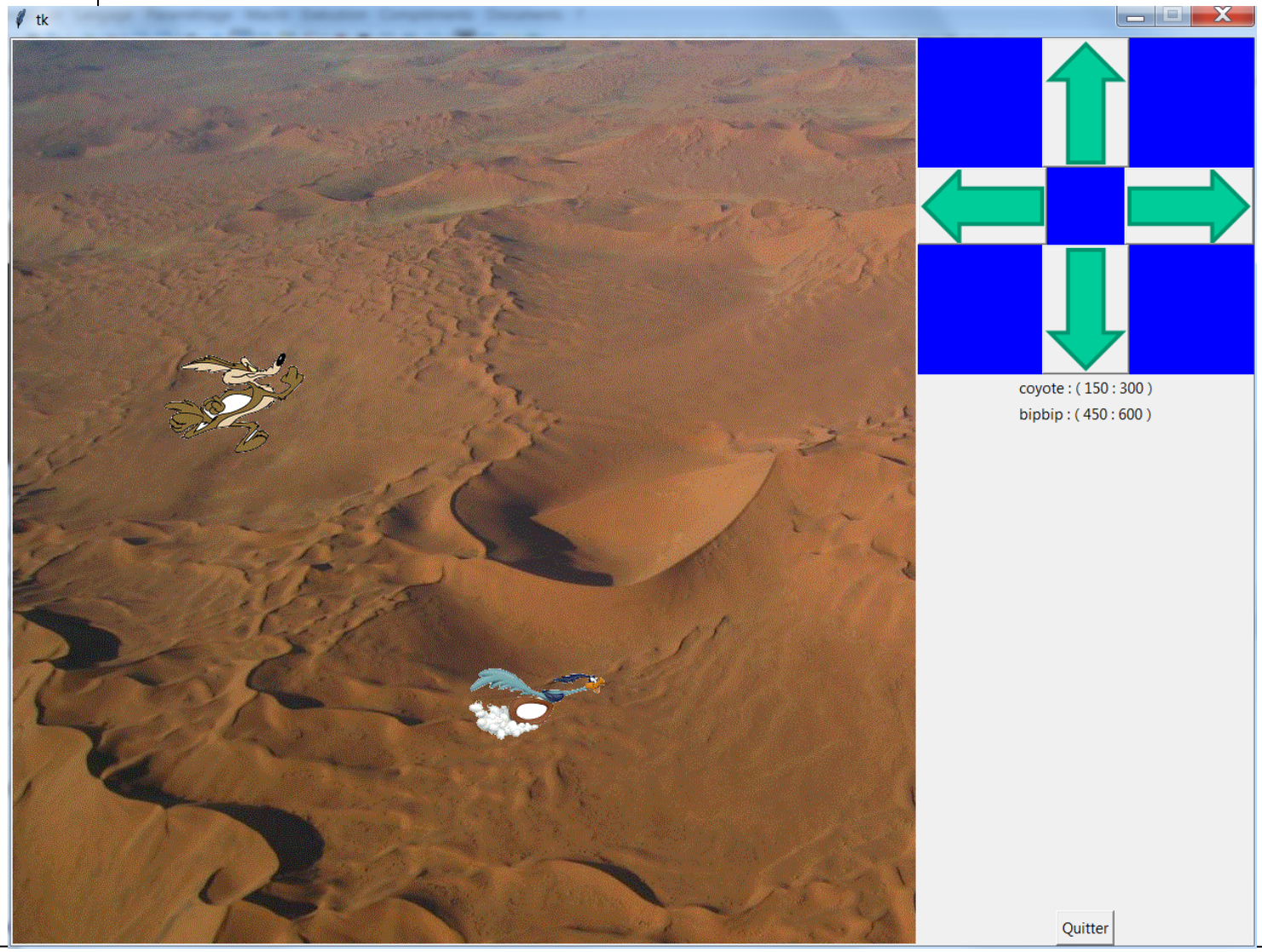
16 - Environnement Graphique : TKINTER -> Widget : canevas - create



create_arc()	arc de cercle
create_bitmap()	bitmap
create_image()	image
create_line()	ligne
create_oval()	ovale
create_polygon()	polygone
create_rectangle()	rectangle
create_text()	texte
create_window()	fenêtre

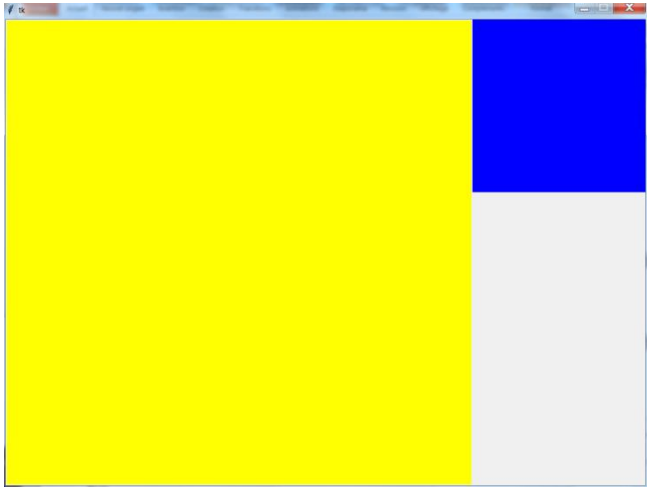
Version 1.5
Copyright : Yantra Technologies 2004-2022

16 - Environnement Graphique : TKINTER -> Exemple : Bip Bip et Coyote



Version 1.5
Copyright : Yantra Technologies 2004-2022

16 - Environnement Graphique : TKINTER -> Exemple : Cadre

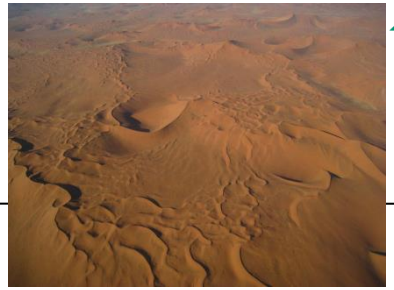


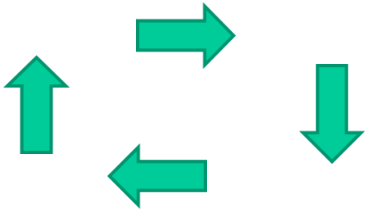
```
from tkinter import *
taille=150
rmax=6
tmax=taille*rmax

fenetre = Tk()
fenetre.resizable(width=False, height=False)
canevas = Canvas(fenetre, bg="yellow", height=tmax, width=tmax)
canevas.pack(side=LEFT)
canevas.pack_propagate(0)
canevas.pack()
#-> insérer définition des images
#-> affichage des images
bframe = Frame(fenetre,bg="blue",width=125*2+85, height=125*2+85)
bframe.pack_propagate(0)
bframe.pack()
#-> insérer définition des boutons
#-> gestion flèche clavier
fenetre.mainloop()
```


Version 1.5
Copyright : Yantra Technologies 2004-2022

16 - Environnement Graphique : TKINTER -> Définition des images

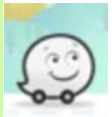
```
fdesert=PhotoImage(file="C:\My_python_lib\\desert.gif")

fcoyote_droite=PhotoImage(file="C:\My_python_lib\\coyote_min_droite.png")
fbipbip_droite=PhotoImage(file="C:\My_python_lib\\bipbip_min_droite.png")
fcoyote_gauche=PhotoImage(file="C:\My_python_lib\\coyote_min_gauche.png"
)
fbipbip_gauche=PhotoImage(file="C:\My_python_lib\\bipbip_min_gauche.png")
fcoyote_haut=PhotoImage(file="C:\My_python_lib\\coyote_min_haut.png")
fbipbip_haut=PhotoImage(file="C:\My_python_lib\\bipbip_min_haut.png")
fcoyote_bas=PhotoImage(file="C:\My_python_lib\\coyote_min_bas.png")
fbipbip_bas=PhotoImage(file="C:\My_python_lib\\bipbip_min_bas.png")

fbgauche=PhotoImage(file="C:\My_python_lib\\gauche.png")
fbdroite=PhotoImage(file="C:\My_python_lib\\droite.png")
fbhaut=PhotoImage(file="C:\My_python_lib\\haut.png")
fbbas=PhotoImage(file="C:\My_python_lib\\bas.png")
```



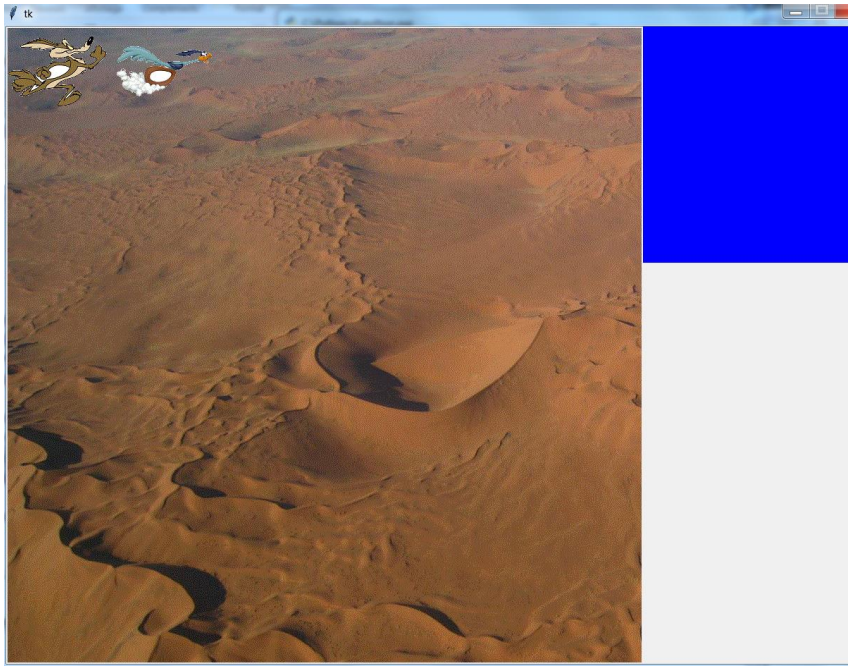


Version 1.5
Copyright : Yantra Technologies 2004-2022

16 - Environnement Graphique : TKINTER -> Affichage des images

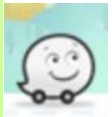


```
xb=taille
yb=xc=yc=0

desert=canevas.create_image(xc,xc,image=fdesert,anchor="nw")
imgcoyote = canevas.create_image(xc,xc,image=fcoyote_droite,anchor="nw")
imgbipbip = canevas.create_image(xb,yb,image=fbipbip_droite,anchor="nw")
```

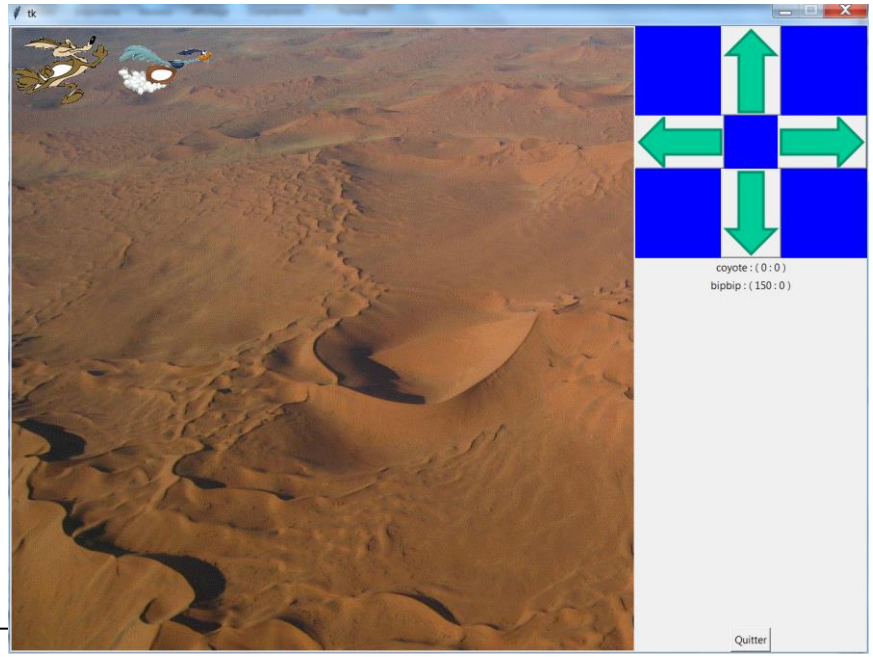


Copyright : Yantra Technologies 2004-2022

16 - Environnement Graphique : TKINTER -> Insérer définition des boutons vu



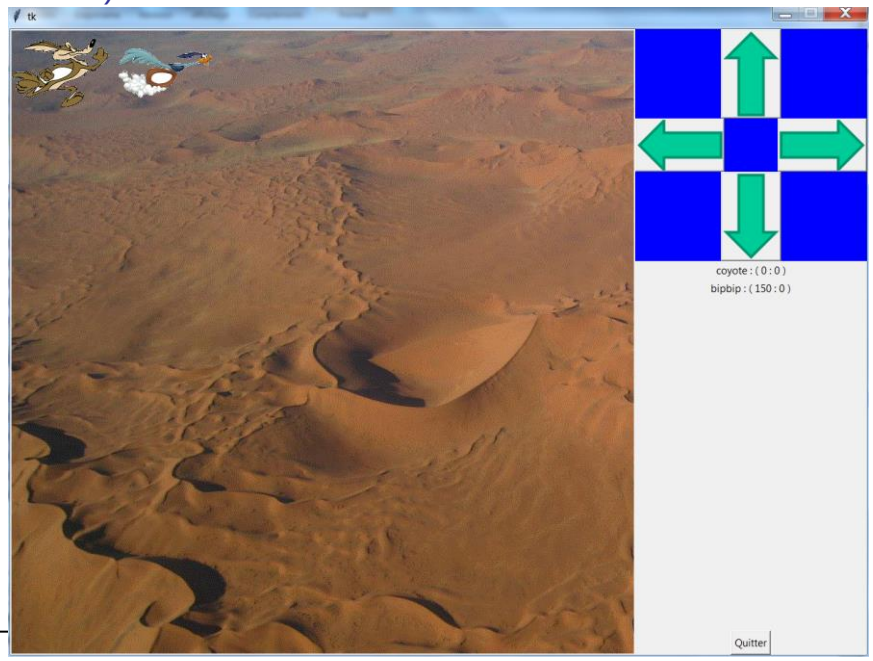
```
bhaut=Button(bframe,text="haut",image=fbhaut)
bhaut.pack()
bbas=Button(bframe,text="bas",image=fbbas)
bbas.pack(side=BOTTOM)
bgauche=Button(bframe,text="gauche",image=fbgauche)
bgauche.pack(side=LEFT)
bdroite=Button(bframe,text="droite",image=fbdroite)
bdroite.pack(side=RIGHT)

Button(fenetre,text='Quitter',command=fenetre.quit).pack(side=BOTTOM)

lab1 = Label(fenetre)
lab1.pack()
lab1.config(text="coyote : ( {0} : {1} )".format(xc,yc))
lab2 = Label(fenetre)
lab2.pack()
lab2.config(text="bipbip : ( {0} : {1} )".format(xb,yb))
```



Copyright : Yantra Technologies 2004-2022

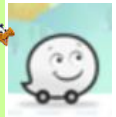
16 - Environnement Graphique : TKINTER -> Insérer définition des boutons V1

```
bbhaut=Button(bframe,text="haut",image=fbhaut, command=haut)
bhaut.pack()
bbas=Button(bframe,text="bas",image=fbbas, command=bas)
bbas.pack(side=BOTTOM)
bgauche=Button(bframe,text="gauche",image=fbgauche, command=gauche)
bgauche.pack(side=LEFT)
bdroite=Button(bframe,text="droite",image=fbdroite, command=droite)
bdroite.pack(side=RIGHT)

Button(fenetre,text='Quitter',command=fenetre.quit).pack(side=BOTTOM)

lab1 = Label(fenetre)
lab1.pack()
lab1.config(text="coyote : ( {0} : {1} )".format(xc,yc))
lab2 = Label(fenetre)
lab2.pack()
lab2.config(text="bipbip : ( {0} : {1} )".format(xb,yb))
```



Copyright : Yantra Technologies 2004-2022

16 - Environnement Graphique : TKINTER -> Fonction bas, haut



```
def bas():
        global yb,xb,xc,yc,taille
        yb=yb+taille*2
        yc=yc+taille
        if ( xb == xc and yb == yc):
                yb=yb+taille
        if ( yb >= tmax ):
                yb = 0
        if ( yc >= tmax ):
                yc = 0
                if ( xb == xc and yb == yc):
                        yb=yb+taille

        canevas.itemconfig(imgbipbip, image = fbipbip_bas)
        canevas.itemconfig(imgcoyote, image = fcoyote_bas)
        canevas.coords(imgbipbip,xb,yb)
        canevas.coords(imgcoyote,xc,yc)
        lab1.config(text="coyote : ( {0} : {1} )".format(xc,yc))
        lab2.config(text="bipbip : ( {0} : {1} )".format(xb,yb))
```

```
def haut():
        global yb,xb,xc,yc,taille
        yb=yb-taille*2
        yc=yc-taille
        if ( xb == xc and yb == yc) :
                yb=yb-taille
        if ( yb < 0 ):
                yb = tmax-150
        if ( yc < 0 ):
                yc = tmax-150
                if ( xb == xc and yb == yc):
                        yb=yb-taille
        canevas.itemconfig(imgbipbip, image = fbipbip_haut)
        canevas.itemconfig(imgcoyote, image = fcoyote_haut)
        canevas.coords(imgbipbip,xb,yb)
        canevas.coords(imgcoyote,xc,yc)
        lab1.config(text="coyote : ( {0} : {1} )".format(xc,yc))
        lab2.config(text="bipbip : ( {0} : {1} )".format(xb,yb))
```


Version 1.5
Copyright : Yantra Technologies 2004-2022

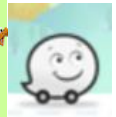
16 - Environnement Graphique : TKINTER -> Fonction gauche, droite

```
def gauche():
        global yb,xb,xc,yc,taille
        xb=xb-taille*2
        xc=xc-taille
        if ( xb == xc and yb == yc) :
                xb=xb-taille
        if ( xb < 0 ):
                xb = tmax-150
        if ( xc < 0 ):
                xc = tmax-150
                if ( xb == xc and yb == yc) :
                        xb=xb-taille
        canevas.itemconfig(imgbipbip,image = fbipbip_gauche)
        canevas.itemconfig(imgcoyote,image =fcoyote_gauche)
        canevas.coords(imgbipbip,xb,yb)
        canevas.coords(imgcoyote,xc,yc)
        lab1.config(text="coyote : ( {0} : {1} )".format(xc,yc))
        lab2.config(text="bipbip : ( {0} : {1} )".format(xb,yb))
```

```
def droite():
        global yb,xb,xc,yc,taille
        xb=xb+taille*2
        xc=xc+taille
        if ( xb == xc and yb == yc):
                xb=xb+taille
        if ( xb >=tmax ):
                xb = 0
        if ( xc >=tmax ):
                xc = 0
                if ( xb == xc and yb == yc):
                        xb=xb+taille
        canevas.itemconfig(imgbipbip, image = fbipbip_droite)
        canevas.itemconfig(imgcoyote, image = fcoyote_droite)
        canevas.coords(imgbipbip,xb,yb)
        canevas.coords(imgcoyote,xc,yc)
        lab1.config(text="coyote : ( {0} : {1} )".format(xc,yc))
        lab2.config(text="bipbip : ( {0} : {1} )".format(xb,yb))
```


Version 1.5
Copyright : Yantra Technologies 2004-2022

16 - Environnement Graphique : TKINTER -> Gestion flèche clavier

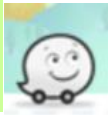


```
fenetre.bind("<Up>", chaut) # Flèche haut
fenetre.bind("<Down>", cbas) # Bas
fenetre.bind("<Left>", cgauche) # Gauche
fenetre.bind("<Right>", cdroite) # Droite
```

```
def cbas (event):
        bas()
def chaut (event):
        haut()
def cgauche (event):
        gauche()
def cdroite (event):
        droite()
```


Version 1.5
Copyright : Yantra Technologies 2004-2022

17 - Environnement Graphique : PYGAME



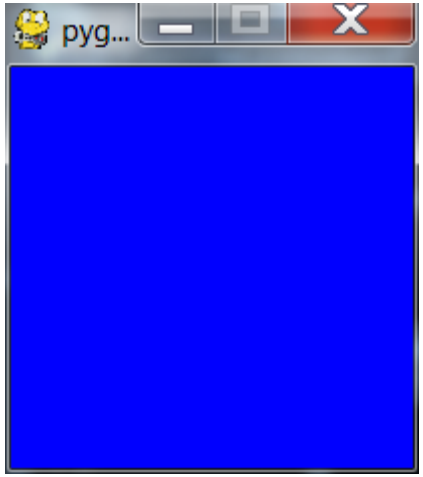
- le plus petit programme graphique
- Widget
- pygame.draw
- Exemple : Bip Bip et Coyote
  - Cadre
  - Définition des images
  - Affichage des images
  - Insérer définition des boutons
  - Gestion évènement : souris, clavier
  - Fonction event : gauche, droite
  - Fonction event : bas, haut
  - Fonction bas, haut, gauche droite
  - Son

https://he-arc.github.io/livre-python/pygame/index.html
https://pythonforge.com/pygame-creer-des-jeux-avec-python/
https://realpython.com/pygame-a-primer/

Version 1.5
Copyright : Yantra Technologies 2004-2022

17 - Environnement Graphique : PYGAME
-> le plus petit programme graphique

```
import pygame
from pygame.locals import *

blue = 0,0,255

pygame.init()
fenetre_pygame = pygame.display.set_mode((200,200))
fenetre_pygame.fill(blue)

pygame.display.flip()

while True :
    for event in pygame.event.get():
        if event.type==QUIT:
            pygame.quit()
```

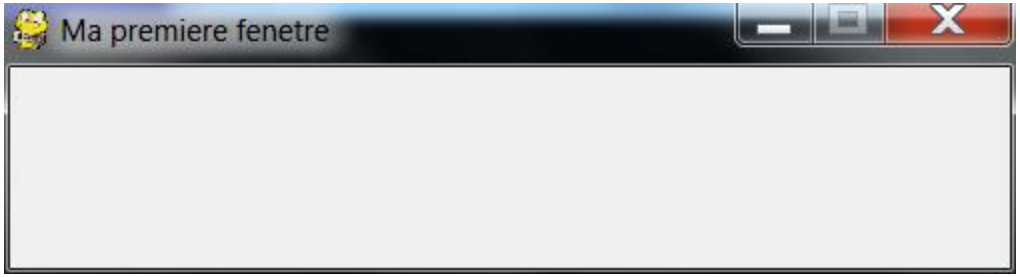


http://www.pygame.org/news.html
https://fr.wikibooks.org/wiki/Pygame/Introduction_%C3%A0_Pygame

Copyright : Yantra Technologies 2004-2022

17 - Environnement Graphique : PYGAME
-> le plus petit programme graphique

```
import pygame
from pygame.locals import *

blue = 0,0,255
grisclaire=[240,240,240]

pygame.init()
fenetre_pygame = pygame.display.set_mode((500,100))
pygame.display.set_caption('Ma premiere fenetre')
fenetre_pygame.fill(grisclaire)

pygame.display.flip()

while True :
    for event in pygame.event.get():
        if event.type==QUIT:
            pygame.quit()
```



Copyright : Yantra Technologies 2004-2022

17 - Environnement Graphique : PYGAME -> Widget

Pas de widget en pygame obligé de se les faire ou d'utiliser des bibliothèques (Button, slider...)

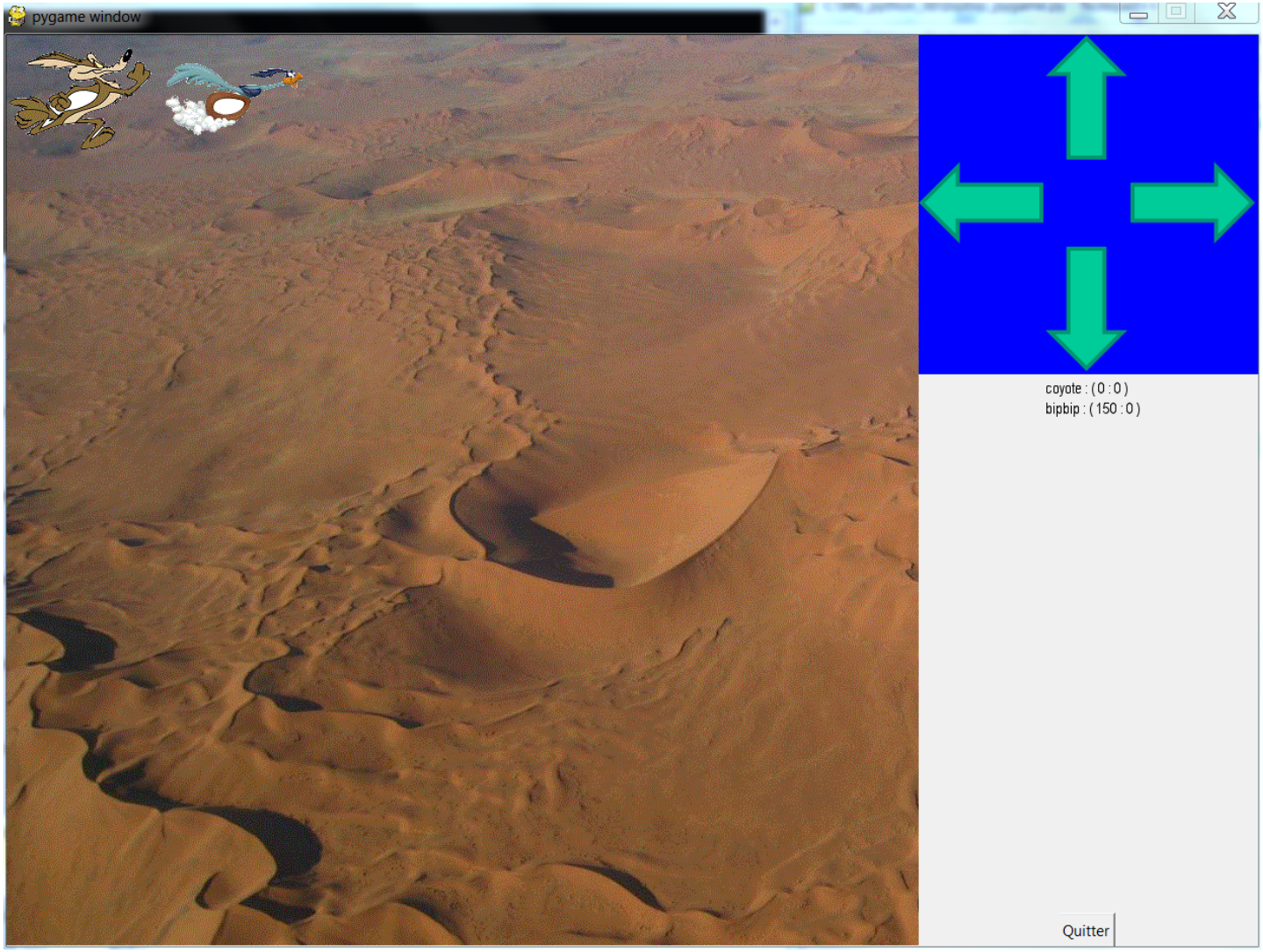
Version 1.5
Copyright : Yantra Technologies 2004-2022

17 - Environnement Graphique : PYGAME -> pygame.draw

pygame.draw.rect	dessiner une forme de rectangle
pygame.draw.polygon	dessiner une forme avec un nombre quelconque de côtés
pygame.draw.circle	dessiner un cercle autour d'un point
pygame.draw.ellipse	dessiner une forme ronde dans un rectangle
pygame.draw.arc	dessiner une coupe partielle d'une ellipse
pygame.draw.line	dessiner un segment de droite
pygame.draw.lines	dessiner plusieurs segments de ligne contigus
pygame.draw.aaline	tracer des lignes fines avec anti-aliasing
pygame.draw.aalines	dessiner une séquence connectée de l'anticrénelage de lignes

Version 1.5
Copyright : Yantra Technologies 2004-2022

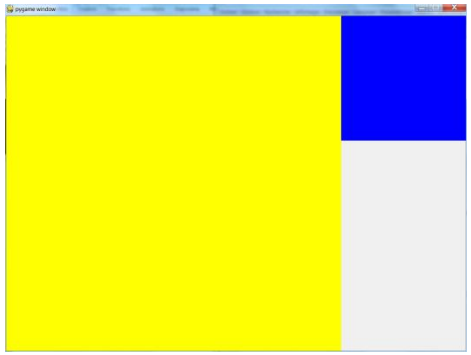
17 - Environnement Graphique : PYGAME -> Exemple : Bip Bip et Coyote



Copyright : Yantra Technologies 2004-2022

17 - Environnement Graphique : PYGAME -> Exemple : Cadre

```
import pygame
from pygame.locals import *
noir=[0,0,0]
blanc=[255,255,255]
vert=[0,255,0]
bleue=[0,0,255]
jaune =[255,255,0]
bleucyan=[0,255,255]
grisclaire=[240,240,240]
taille=150
rmax=6
tmax=taille*rmax
pygame.init()
fenetre_pygame = pygame.display.set_mode((tmax+125*2+85,tmax))
fenetre_pygame.fill(grisclaire)
#-> insérer définition des images
rectangle= pygame.draw.rect(fenetre_pygame, bleue,[tmax,0,125*2+85,125*2+85] ,0)
dessert= pygame.draw.rect(fenetre_pygame, jaune,[0,0,tmax,tmax] ,0)
#-> affichage des images
rbhaut=pygame.Rect(tmax+125,0,83,125)
rbbas=pygame.Rect(tmax+125,125+84,83,125)
rbgauche=pygame.Rect(tmax,125,125,83)
rbdroite=pygame.Rect(tmax+125+84,125,125,83)
#-> insérer définition des boutons
pygame.display.flip()
pygame.key.set_repeat(1,20)
#-> insérer son
while True:
    for event in pygame.event.get():
        if event.type==QUIT:
            pygame.quit()
#-> gestion evenement
```



Version 1.5
Copyright : Yantra Technologies 2004-2022

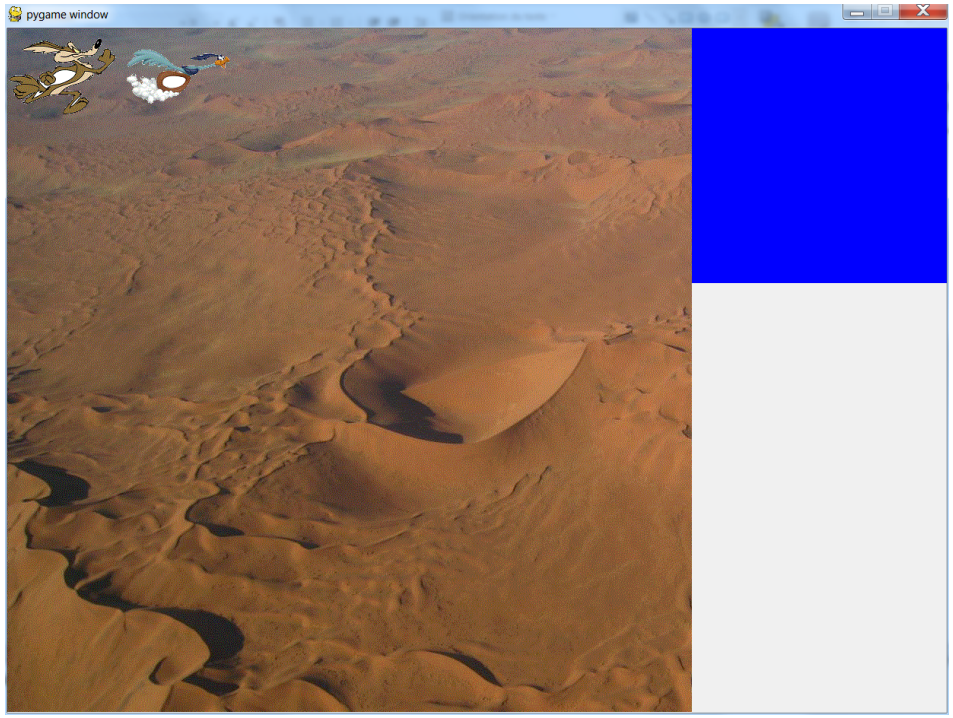
17 - Environnement Graphique : PYGAME -> Exemple : Définition des images

```
fdesert=pygame.image.load("C:\My_python_lib\desert900.png")

fcoyote_droite=pygame.image.load("C:\My_python_lib\coyote_min_droite.png")
fbipbip_droite=pygame.image.load("C:\My_python_lib\\bipbip_min_droite.png")
fcoyote_gauche=pygame.image.load("C:\My_python_lib\\coyote_min_gauche.png")
fbipbip_gauche=pygame.image.load("C:\My_python_lib\\bipbip_min_gauche.png")
fcoyote_haut=pygame.image.load("C:\My_python_lib\\coyote_min_haut.png")
fbipbip_haut=pygame.image.load("C:\My_python_lib\\bipbip_min_haut.png")
fcoyote_bas=pygame.image.load("C:\My_python_lib\\coyote_min_bas.png")
fbipbip_bas=pygame.image.load("C:\My_python_lib\\bipbip_min_bas.png")

fbgauche=pygame.image.load("C:\My_python_lib\\gauche.png")
fbdroite=pygame.image.load("C:\My_python_lib\\droite.png")
fbhaut=pygame.image.load("C:\My_python_lib\\haut.png")
fbbas=pygame.image.load("C:\My_python_lib\\bas.png")
```

Version 1.5
Copyright : Yantra Technologies 2004-2022

17 - Environnement Graphique : PYGAME -> Exemple : Affichage des images

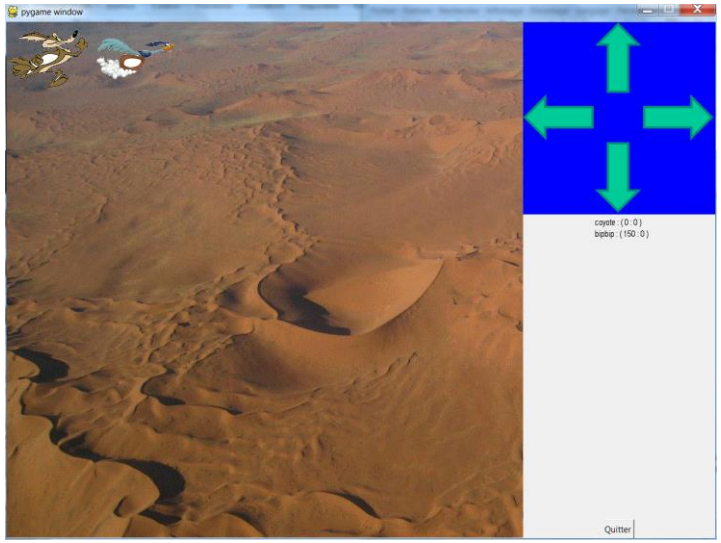
```
xb=taille
yb=xc=yc=0

rectangle= pygame.draw.rect(fenetre_pygame, bleue,[tmax,0,125*2+85,125*2+85] ,0)
dessert= pygame.draw.rect(fenetre_pygame, jaune,[0,0,tmax,tmax] ,0)

fenetre_pygame.blit(fdesert,(0,0))
fenetre_pygame.blit(fcoyote_droite,(xc,yc))
fenetre_pygame.blit(fbipbip_droite,(xb,yb))
position_fbipbip = fbipbip_droite.get_rect()
```



Copyright : Yantra Technologies 2004-2022

17 - Environnement Graphique : PYGAME
-> Exemple : Insérer définition des boutons

```
fenetre_pygame.blit(fbhaut,(tmax+125,0))
fenetre_pygame.blit(fbbas,(tmax+125,125+84))
fenetre_pygame.blit(fbgauche,(tmax,125))
fenetre_pygame.blit(fbdroite,(tmax+125+84,125))

my_font = pygame.font.SysFont('Arial', 15)

fenetre_pygame.blit(my_font.render("coyote : ( {0} : {1} )".format(xc,yc), True,noir),
(tmax+125,125*2+84+5))
fenetre_pygame.blit(my_font.render("bipbip : ( {0} : {1} )".format(xb,yb), True,noir),
(tmax+125,125*2+84+5+20))

fqui=pygame.image.load("C:\My_python_lib\\bouton_quitter.png")
fenetre_pygame.blit(fqui,(tmax+125+84/2-28,tmax-33))
rbquit=pygame.Rect(tmax+125+84/2-28,tmax-33,56,33)
```



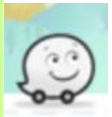
Copyright : Yantra Technologies 2004-2022

17 - Environnement Graphique : PYGAME -> Exemple :

```
while continuer:
        for event in pygame.event.get():
                if event.type==QUIT:
                        pygame.quit()
                        continuer=0
                if event.type == pygame.MOUSEBUTTONDOWN:
                        if rbbas.collidepoint(event.pos) :
                                event_bas()
                        if rbhaut.collidepoint(event.pos) :
                                event_haut()
                        if rbdroite.collidepoint(event.pos) :
                                event_droite()
                        if rbgauche.collidepoint(event.pos) :
                                event_gauche()
                        if rbquit.collidepoint(event.pos) :
                                fenetre_pygame.quit()
                if event.type == KEYDOWN :
                        if event.key == K_DOWN :
                                event_bas()
                        if event.key == K_UP:
                                event_haut()
                        if event.key == K_RIGHT:
                                event_droite()
                        if event.key == K_LEFT:
                                event_gauche()
```


Version 1.5
Copyright : Yantra Technologies 2004-2022

17 - Environnement Graphique : PYGAME
-> Exemple : Fonction event : bas, haut



```
def event_bas():
  global my_font,fenetre_pygame,yb,xb,xc,yc,pygame
  fenetre_pygame.blit(my_font.render("bipbip : ( {0} : {1} )".format(xb,yb), True,grisclaire), (tmax+125,125*2+84+5+20))
  fenetre_pygame.blit(my_font.render("coyote : ( {0} : {1} )".format(xc,yc), True,grisclaire), (tmax+125,125*2+84+5))
  bas()
  fenetre_pygame.blit(fdesert, (0,0))
  fenetre_pygame.blit(fbipbip_bas, (xb,yb))
  fenetre_pygame.blit(fcoyote_bas,(xc,yc))
  fenetre_pygame.blit(my_font.render("bipbip : ( {0} : {1} )".format(xb,yb), True,noir), (tmax+125,125*2+84+5+20))
  fenetre_pygame.blit(my_font.render("coyote : ( {0} : {1} )".format(xc,yc), True,noir), (tmax+125,125*2+84+5))
   pygame.display.flip()
```

```
def event_haut():
  global my_font,fenetre_pygame,yb,xb,xc,yc,pygame
  fenetre_pygame.blit(my_font.render("bipbip : ( {0} : {1} )".format(xb,yb), True,grisclaire), (tmax+125,125*2+84+5+20))
  fenetre_pygame.blit(my_font.render("coyote : ( {0} : {1} )".format(xc,yc), True,grisclaire), (tmax+125,125*2+84+5))
  haut()
  fenetre_pygame.blit(fdesert, (0,0))
  fenetre_pygame.blit(fbipbip_haut, (xb,yb))
  fenetre_pygame.blit(fcoyote_haut,(xc,yc))
  fenetre_pygame.blit(my_font.render("bipbip : ( {0} : {1} )".format(xb,yb), True,noir), (tmax+125,125*2+84+5+20))
   fenetre_pygame.blit(my_font.render("coyote : ( {0} : {1} )".format(xc,yc), True,noir), (tmax+125,125*2+84+5))
   pygame.display.flip()
```


Version 1.5
Copyright : Yantra Technologies 2004-2022

17 - Environnement Graphique : PYGAME
-> Exemple : Fonction event : gauche, droite

```
def event_gauche():
  fenetre_pygame.blit(my_font.render("bipbip : ( {0} : {1} )".format(xb,yb), True,grisclaire), (tmax+125,125*2+84+5+20))
  fenetre_pygame.blit(my_font.render("coyote : ( {0} : {1} )".format(xc,yc), True,grisclaire), (tmax+125,125*2+84+5))
  gauche()
  fenetre_pygame.blit(fdesert, (0,0))
  fenetre_pygame.blit(fbipbip_gauche, (xb,yb))
  fenetre_pygame.blit(fcoyote_gauche,(xc,yc))
  fenetre_pygame.blit(my_font.render("bipbip : ( {0} : {1} )".format(xb,yb), True,noir), (tmax+125,125*2+84+5+20))
  fenetre_pygame.blit(my_font.render("coyote : ( {0} : {1} )".format(xc,yc), True,noir), (tmax+125,125*2+84+5))
  pygame.display.flip()
```

```
def event_droite():
  global my_font,fenetre_pygame,yb,xb,xc,yc,pygame
  fenetre_pygame.blit(my_font.render("bipbip : ( {0} : {1} )".format(xb,yb), True,grisclaire), (tmax+125,125*2+84+5+20))
  fenetre_pygame.blit(my_font.render("coyote : ( {0} : {1} )".format(xc,yc), True,grisclaire), (tmax+125,125*2+84+5))
  droite()
  fenetre_pygame.blit(fdesert, (0,0))
  fenetre_pygame.blit(fbipbip_droite, (xb,yb))
  fenetre_pygame.blit(fcoyote_droite,(xc,yc))
  fenetre_pygame.blit(my_font.render("bipbip : ( {0} : {1} )".format(xb,yb), True,noir), (tmax+125,125*2+84+5+20))
  fenetre_pygame.blit(my_font.render("coyote : ( {0} : {1} )".format(xc,yc), True,noir), (tmax+125,125*2+84+5))
  pygame.display.flip()
```


Version 1.5
Copyright : Yantra Technologies 2004-2022

17 - Environnement Graphique : PYGAME
-> Exemple : Fonction bas, haut, gauche droite

```
def bas():
        global yb,xb,xc,yc,taille
        yb=yb+taille*2
        yc=yc+taille
        if ( xb == xc and yb == yc):
                yb=yb+taille
        if ( yb >= tmax ):
                yb = 0
        if ( yc >= tmax ):
                yc = 0
                if ( xb == xc and yb == yc):
                        yb=yb+taille

        ))
```

```
def haut():
        global yb,xb,xc,yc,taille
        yb=yb-taille*2
        yc=yc-taille
        if ( xb == xc and yb == yc) :
                yb=yb-taille
        if ( yb < 0 ):
                yb = tmax-150
        if ( yc < 0 ):
                yc = tmax-150
                if ( xb == xc and yb == yc):
                        yb=yb-taille
```

```
def gauche():
        global yb,xb,xc,yc,taille
        xb=xb-taille*2
        xc=xc-taille
        if ( xb == xc and yb == yc) :
                xb=xb-taille
        if ( xb < 0 ):
                xb = tmax-150
        if ( xc < 0 ):
                xc = tmax-150
                if ( xb == xc and yb == yc) :
                        xb=xb-taille
```

```
def droite():
        global yb,xb,xc,yc,taille
        xb=xb+taille*2
        xc=xc+taille
        if ( xb == xc and yb == yc):
                xb=xb+taille
        if ( xb >=tmax ):
                xb = 0
        if ( xc >=tmax ):
                xc = 0
                if ( xb == xc and yb == yc):
                        xb=xb+taille
```


Version 1.5
Copyright : Yantra Technologies 2004-2022

17 - Environnement Graphique : PYGAME -> Exemple : Son

```
import winsound

fname = "C:\\My_python_lib\\4074.wav"
winsound.PlaySound(fname, winsound.SND_FILENAME | winsound.SND_ASYNC |
winsound.SND_NOSTOP | winsound.SND_LOOP)
```


Version 1.5
Copyright : Yantra Technologies 2004-2022

18 - Application Web - Simple

```python
#!d:/programs/python/python39/python
from math import sin, cos, tan, pi


# Construction de l'en-tête du tableau avec les titres de colonnes :
print("""
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="fr" lang="fr" dir="ltr">
<head>
<title>Exemple convertion degres en radians</title>
<style type="text/css">
table, th, td {
  border: 1px solid black;
}
</style>
</head>
<body>""")

print("""<table>
<tr><td>Angle</td><td>Sinus</td><td>Cosinus</td><td>Tangente</td></tr>""")


for angle in range(0,62,10):
    # conversion des degrés en radians :
    aRad = angle * pi / 180
    # construction d'une ligne de tableau, en exploitant le formatage des
    # chaînes de caractères pour fignoler l'affichage :
    print ("""<tr><td>%s</td><td>%8.7f</td><td>%8.7f</td><td>%8.7g</td></tr>"""\
          % (angle, sin(aRad), cos(aRad), tan(aRad)))

print ("""</table>\n""")

print ("""</body>\n</html>\n""")
```

https://fr.wikibooks.org/wiki/Programmation_Python/Applications_web#Un_serveur_web_en_pur_Python_!

Version 1.5
Copyright : Yantra Technologies 2004-2022

18 - Application Web - Simple

Exemple installation pour que le serveur apache interprète du python :

Version Wamp sous windows :

1) Installer python

2) Intsaller mod_wsgi :

```
set "MOD_WSGI_APACHE_ROOTDIR=C:\wamp64\bin\apache\apache2.4.33"

pip install mod_wsgi

mod_wsgi-express.exe module-config
mod_wsgi-express.exe module-location
```

3) Modifier httpd.conf du serveur apache

```
<Directory "C:/wamp64/www/py/">
        Options +ExecCGI
        AddHandler cgi-script .py
        Allowoverride all
        Require local
</Directory>

LoadFile "d:/programs/python/python39/python39.dll"
WSGIPythonHome "d:/programs/python/python39"
WSGIScriptAlias /myapp "C:/wamp64/www/py/theTest.py"
LoadModule wsgi_module "d:/programs/python/python39/lib/site-packages/mod_wsgi/server/mod_wsgi.cp39-win_amd64.pyd"
```

https://pypi.org/project/mod-wsgi/

Version 1.5
Copyright : Yantra Technologies 2004-2022

18 - Application Web - Simple

Angle	Sinus	Cosinus	Tangente
0	0.0000000	1.0000000	0
10	0.1736482	0.9848078	0.176327
20	0.3420201	0.9396926	0.3639702
30	0.5000000	0.8660254	0.5773503
40	0.6427876	0.7660444	0.8390996
50	0.7660444	0.6427876	1.191754
60	0.8660254	0.5000000	1.732051

Version 1.5
Copyright : Yantra Technologies 2004-2022

18 - Application Web - Frameworks

Liste de frameworks Python

Les frameworks (web) suivants ont été écrits en Python :

- Bottle — licence MIT
- CherryPy - open source
- Django — open source
- Flask — licence BSD
- Grok (en) — open source
- Karrigell — open source
- Pylons (en) — open source
- Pyramid — licence BSD
- TurboGears — open source
- Tornado (framework) (en) — licence Apache
- Twisted — licence MIT
- Web2py — open source
- Zope — open source

https://geekflare.com/fr/best-python-frameworks/

Version 1.5
Copyright : Yantra Technologies 2004-2022

19 - Le Réseau

- Socket
- Socket & Thread

https://python.doctor/page-reseaux-sockets-python-port
https://python.antoinepernot.fr/cours.php?course=chap7
https://python.developpez.com/cours/TutoSwinnen/?page=page_20

Version 1.5
Copyright : Yantra Technologies 2004-2022

19 - Le Réseau -> Socket : Serveur

```
#!/usr/bin/env python
# coding: utf-8

import socket
print("Serveur : lancement socket")
socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
socket.bind(('', 15555))
print( "Serveur : Boucle")
while True:
        socket.listen(5)
        client, address = socket.accept()
        print ("adresse {} connecte ".format( address ) )
        response = client.recv(255)
        if response != "":
            print("reponse : {} ".format(response))
print( "Serveur : Close")
client.close()
stock.close()
```

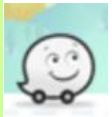

Version 1.5
Copyright : Yantra Technologies 2004-2022

19 - Le Réseau -> Socket : Serveur

```
>>> import serveur
Serveur : lancement socket
Serveur : Boucle
```


Copyright : Yantra Technologies 2004-2022

19 - Le Réseau -> Socket : Client



```
#!/usr/bin/env python
# coding: utf-8

import socket

hote = "localhost"
port = 15555
print("Client : connection socket")
socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
socket.connect((hote, port))
print ("Client : Connection port {}".format(port))
msg="Bonjour de David"
socket.send(msg.encode('utf-8'))

print ("Client : Close")
socket.close()
```


Version 1.5
Copyright : Yantra Technologies 2004-2022

19 - Le Réseau -> Socket : Client

```
>>> import client
Client : connection socket
Client : Connection port 15555
Client : Close
```

```
>>> import serveur
Serveur : lancement socket
Serveur : Boucle
adresse ('127.0.0.1', 55856) connecte
reponse : b'Bonjour de David'
```


Version 1.5
Copyright : Yantra Technologies 2004-2022

19 - Le Réseau -> Socket & Thread : Serveur

```
#!/usr/bin/env python
# coding: utf-8
import socket
import threading

class ClientThread(threading.Thread):

    def __init__(self, ip, port, clientsocket):
        print ("ClientThread : initialisation")
        threading.Thread.__init__(self)
        self.ip = ip
        self.port = port
        self.clientsocket = clientsocket
        print("[+] Nouveau thread pour %s %s" % (self.ip, self.port, ))

    def run(self):
        print ("ClientThread : execution")
        print("ClientThread : Connection de %s %s" % (self.ip, self.port, ))
        r = self.clientsocket.recv(2048)
        print("ClientThread : Ouverture du fichier: ", r, "...")
        fp = open(r, 'rb')
        self.clientsocket.send(fp.read())
        print("ClientThread : Client déconnecté...")
```


Version 1.5
Copyright : Yantra Technologies 2004-2022

19 - Le Réseau -> Socket & Thread : Serveur

```
tcpsock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
tcpsock.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
tcpsock.bind(("",15555))

print( "Serveur : Boucle")
while True:
    tcpsock.listen(10)
    print( "En écoute...")
    (clientsocket, (ip, port)) = tcpsock.accept()
    newthread = ClientThread(ip, port, clientsocket)
    newthread.start()
```


Version 1.5
Copyright : Yantra Technologies 2004-2022

19 - Le Réseau -> Socket & Thread : Serveur

```
>>> import serveur_thread
Serveur : Boucle
En écoute...
```


Copyright : Yantra Technologies 2004-2022

19 - Le Réseau -> Socket & Thread : Client

```
#!/usr/bin/env python
# coding: utf-8

import socket

def client():
    print("Client : Debut")
    s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    s.connect(("localhost", 15555))
    file_name = input("Entrer le nom du fichier que vous voulez récupérer: ")
    file_name_s = 'C:\\My_python_lib\\test\\%s.txt' % (file_name,)
    print ("Client: envoyer")
    s.send(file_name_s.encode())
    print ("Client : recevoir")
    r = s.recv(4096)
    file_name_s = 'C:\\My_python_lib\\test\\%s.res' % (file_name,)
    print (file_name_s)
    with open(file_name_s,'wb') as _file:
        _file.write(r)
    print("Le fichier a été correctement copié dans : %s." % file_name_s)
    print("Client : fermer")
    s.close()
```


Version 1.5
Copyright : Yantra Technologies 2004-2022

19 - Le Réseau -> Socket & Thread : Client

```
>>> import client_thread
>>> client_thread.client()
Client : Debut
Entrer le nom du fichier que vous voulez récupérer: test
Client: envoyer
Client : recevoir
C:\My_python_lib\test\test.res
Le fichier a été correctement copié dans : C:\My_python_lib\test\test.
res.
Client : fermer
```

```
>>> import serveur_thread
Serveur : Boucle
En écoute...
ClientThread : initialisation
[+] Nouveau thread pour 127.0.0.1 55896
ClientThread : execution
En écoute...
ClientThread : Connection de 127.0.0.1 55896
ClientThread : Ouverture du fichier:  b'C:\\My_python_lib\\test\\test.
txt' ...
ClientThread : Client déconnecté...
```


Version 1.5
Copyright : Yantra Technologies 2004-2022

20 –base de données : Exemple SQLITE

import sqlite3

- connexion

connexion = sqlite3.connect("basededonnees.db")

Ou en mémoire mais il n'y aura donc pas de persistance des données après la déconnexion

connexion = sqlite3.connect(":memory:") #BDD dans la RAM

- déconnection

connexion.close()

https://python.doctor/page-database-data-base-donnees-query-sql-mysql-postgre-sqlite
https://zestedesavoir.com/tutoriels/1294/des-bases-de-donnees-en-python-avec-sqlite3/fonctionnalites-de-base/#1-se-connecter-et-se-deconnecter

Version 1.5
Copyright : Yantra Technologies 2004-2022

20 –base de données : Exemple SQLITE

- Type de données

SQLite	Python
NULL	None
INTEGER	int
REAL	float
TEXT	str par défaut
BLOB	bytes

https://python.doctor/page-database-data-base-donnees-query-sql-mysql-postgre-sqlite
https://zestedesavoir.com/tutoriels/1294/des-bases-de-donnees-en-python-avec-sqlite3/fonctionnalites-de-base/#1-se-connecter-et-se-deconnecter

Version 1.5
Copyright : Yantra Technologies 2004-2022

20 –base de données : Exemple SQLITE

- Exécuter une commande
  - Valider ou annuler les modifications

Pour exécuter nos requêtes, nous avons besoin d'un objet Cursor, que nous récupéront avec la méthode cursor de notre objet de type Connection.

```
curseur = connexion.cursor()

#modifications....
connexion.commit()  #Validation des modifications

#modifications....
connexion.rollback()  #Retour à l'état du dernier commit, les modifications effectuées depuis sont perdues
```

https://python.doctor/page-database-data-base-donnees-query-sql-mysql-postgre-sqlite
https://zestedesavoir.com/tutoriels/1294/des-bases-de-donnees-en-python-avec-sqlite3/fonctionnalites-de-base/#1-se-connecter-et-se-deconnecter

Version 1.5
Copyright : Yantra Technologies 2004-2022

20 –base de données : Exemple SQLITE

- Exécuter une commande
  - Exécuter une requête

```
curseur.execute('''CREATE TABLE IF NOT EXISTS scores(
    id INTEGER PRIMARY KEY AUTOINCREMENT UNIQUE,
    pseudo TEXT,
    valeur INTEGER
)’‘’)

donnees = [("toto", 1000), ("tata", 750), ("titi", 500)]
#Exécutions multiples
for donnee in donnees:
    curseur.execute('''INSERT INTO scores (pseudo, valeur) VALUES (?, ?)''', donnee)
connexion.commit()
```

https://python.doctor/page-database-data-base-donnees-query-sql-mysql-postgre-sqlite
https://zestedesavoir.com/tutoriels/1294/des-bases-de-donnees-en-python-avec-sqlite3/fonctionnalites-de-base/#1-se-connecter-et-se-deconnecter

Version 1.5
Copyright : Yantra Technologies 2004-2022

20 –base de données : Exemple SQLITE

- Exécuter une commande
  - Exécuter une requête : multiple

```
donnees = [("toto", 1000), ("tata", 750), ("titi", 500)]
for donnee in donnees:
    curseur.execute('''INSERT INTO scores (pseudo, valeur) VALUES (?, ?)''', donnee)
connexion.commit()

donnees = [("toto", 1000), ("tata", 750), ("titi", 500)]
curseur.executemany("INSERT INTO scores (pseudo, valeur) VALUES (?, ?)", donnees)
connexion.commit()
```

https://python.doctor/page-database-data-base-donnees-query-sql-mysql-postgre-sqlite
https://zestedesavoir.com/tutoriels/1294/des-bases-de-donnees-en-python-avec-sqlite3/fonctionnalites-de-base/#1-se-connecter-et-se-deconnecter

Version 1.5
Copyright : Yantra Technologies 2004-2022

20 –base de données : Exemple SQLITE

- Exécuter une commande
  - Exécuter une requête : Script

```
curseur.executescript("""

    DROP TABLE IF EXISTS scores;

    CREATE TABLE IF NOT EXISTS scores(
    id INTEGER PRIMARY KEY AUTOINCREMENT UNIQUE,
    pseudo TEXT,
    valeur INTEGER);

    INSERT INTO scores(pseudo, valeur) VALUES ("toto", 1000);
    INSERT INTO scores(pseudo, valeur) VALUES ("tata", 750);
    INSERT INTO scores(pseudo, valeur) VALUES ("titi", 500)
""")
```

https://python.doctor/page-database-data-base-donnees-query-sql-mysql-postgre-sqlite
https://zestedesavoir.com/tutoriels/1294/des-bases-de-donnees-en-python-avec-sqlite3/fonctionnalites-de-base/#1-se-connecter-et-se-deconnecter

Copyright : Yantra Technologies 2004-2022

20 –base de données : Exemple SQLITE

identifiant	pseudo	valeur
1	"toto"	1000
2	"tata"	750
3	"titi"	500
4	"toto"	250
5	"tata"	150
6	"tete"	0

- Exécuter une commande
  - Exécuter une requête : parcourir

```
donnee = ("titi", )
curseur.execute("SELECT valeur FROM scores WHERE pseudo = ?", donnee)
print(curseur.fetchone())  #affiche "(500,)

print(curseur.arraysize)  #Affiche "1"
donnee = (400, )
curseur.execute("SELECT pseudo FROM scores WHERE valeur > ?", donnee)
print(curseur.fetchmany())  #Affiche "[('toto',)]"
print(curseur.fetchmany())  #Affiche "[('tata',)]"

curseur.execute("SELECT pseudo FROM scores WHERE valeur > ?", donnee)
print(curseur.fetchmany(2)) #Affiche "[('toto',), ('tata',)]“
```

https://python.doctor/page-database-data-base-donnees-query-sql-mysql-postgre-sqlite
https://zestedesavoir.com/tutoriels/1294/des-bases-de-donnees-en-python-avec-sqlite3/fonctionnalites-de-base/#1-se-connecter-et-se-deconnecter

Copyright : Yantra Technologies 2004-2022

20 –base de données : Exemple SQLITE

identifiant	pseudo	valeur
1	"toto"	1000
2	"tata"	750
3	"titi"	500
4	"toto"	250
5	"tata"	150
6	"tete"	0

- Exécuter une commande
  - Exécuter une requête : parcourir tous

```
curseur.execute("SELECT * FROM scores")
resultats = curseur.fetchall()
for resultat in resultats:
    print(resultat)
```

https://python.doctor/page-database-data-base-donnees-query-sql-mysql-postgre-sqlite
https://zestedesavoir.com/tutoriels/1294/des-bases-de-donnees-en-python-avec-sqlite3/fonctionnalites-de-base/#1-se-connecter-et-se-deconnecter

Version 1.5
Copyright : Yantra Technologies 2004-2022

20 – Base de données : Exemple MYSQL

- Installation

Soit ->

pip install mysql-connector-python

Soit ->

pip install mysql
pip install MySQL-python
pip install mysql-connector-python --allow-external mysql-connector-python

https://python.doctor/page-database-data-base-donnees-query-sql-mysql-postgre-sqlite
https://pynative.com/install-mysql-connector-python/

Version 1.5
Copyright : Yantra Technologies 2004-2022

20 –base de données : Exemple MYSQL

import mysql.connector

- connexion

connexion = mysql.connector.connect(host="localhost",user="root",password="XXX", database=" basededonnees ")

- déconnection

connexion.close()

https://python.doctor/page-database-data-base-donnees-query-sql-mysql-postgre-sqlite
https://zestedesavoir.com/tutoriels/1294/des-bases-de-donnees-en-python-avec-sqlite3/fonctionnalites-de-base/#1-se-connecter-et-se-deconnecter
https://python.developpez.com/cours/TutoSwinnen/?page=page_18#L16

Version 1.5
Copyright : Yantra Technologies 2004-2022

- Type de données

SQLite	Python
NULL	None
INTEGER	int
REAL	float
TEXT	str par défaut
BLOB	bytes

Version 1.5
Copyright : Yantra Technologies 2004-2022

20 –base de données : Exemple MYSQL

- Exécuter une commande
  - Valider ou annuler les modifications

Pour exécuter nos requêtes, nous avons besoin d'un objet Cursor, que nous récupéront avec la méthode cursor de notre objet de type Connection.

```
curseur = connexion.cursor()

#modifications....
connexion.commit()  #Validation des modifications

#modifications....
connexion.rollback()  #Retour à l'état du dernier commit, les modifications effectuées depuis sont perdues
```

https://python.doctor/page-database-data-base-donnees-query-sql-mysql-postgre-sqlite

Version 1.5
Copyright : Yantra Technologies 2004-2022

20 –base de données : Exemple MYSQL

- Exécuter une commande
  - Exécuter une requête

```
curseur.execute('''CREATE TABLE IF NOT EXISTS scores(
    id INTEGER PRIMARY KEY AUTOINCREMENT UNIQUE,
    pseudo TEXT,
    valeur INTEGER
)’‘’)

donnees = [("toto", 1000), ("tata", 750), ("titi", 500)]
#Exécutions multiples
for donnee in donnees:
    curseur.execute('''INSERT INTO scores (pseudo, valeur) VALUES (?, ?)''', donnee)
connexion.commit()
```

https://python.doctor/page-database-data-base-donnees-query-sql-mysql-postgre-sqlite

Version 1.5
Copyright : Yantra Technologies 2004-2022

- Exécuter une commande
  - Exécuter une requête : multiple

```
donnees = [("toto", 1000), ("tata", 750), ("titi", 500)]
for donnee in donnees:
    curseur.execute('''INSERT INTO scores (pseudo, valeur) VALUES (?, ?)''', donnee)
connexion.commit()

donnees = [("toto", 1000), ("tata", 750), ("titi", 500)]
curseur.executemany("INSERT INTO scores (pseudo, valeur) VALUES (?, ?)", donnees)
connexion.commit()
```

https://python.doctor/page-database-data-base-donnees-query-sql-mysql-postgre-sqlite

Version 1.5
Copyright : Yantra Technologies 2004-2022

- Exécuter une commande
  - Exécuter une requête : Script

```
curseur.executescript("""

    DROP TABLE IF EXISTS scores;

    CREATE TABLE IF NOT EXISTS scores(
    id INTEGER PRIMARY KEY AUTOINCREMENT UNIQUE,
    pseudo TEXT,
    valeur INTEGER);

    INSERT INTO scores(pseudo, valeur) VALUES ("toto", 1000);
    INSERT INTO scores(pseudo, valeur) VALUES ("tata", 750);
    INSERT INTO scores(pseudo, valeur) VALUES ("titi", 500)
""")
```

https://python.doctor/page-database-data-base-donnees-query-sql-mysql-postgre-sqlite

Version 1.5
Copyright : Yantra Technologies 2004-2022

20 –base de données : Exemple MYSQL

identifiant	pseudo	valeur
1	"toto"	1000
2	"tata"	750
3	"titi"	500
4	"toto"	250
5	"tata"	150
6	"tete"	0

- Exécuter une commande
  - Exécuter une requête : parcourir

```
donnee = ("titi", )
curseur.execute("SELECT valeur FROM scores WHERE pseudo = ?", donnee)
print(curseur.fetchone())  #affiche "(500,)

print(curseur.arraysize)  #Affiche "1"
donnee = (400, )
curseur.execute("SELECT pseudo FROM scores WHERE valeur > ?", donnee)
print(curseur.fetchmany())  #Affiche "[('toto',)]"
print(curseur.fetchmany())  #Affiche "[('tata',)]"

curseur.execute("SELECT pseudo FROM scores WHERE valeur > ?", donnee)
print(curseur.fetchmany(2)) #Affiche "[('toto',), ('tata',)]“
```

https://python.doctor/page-database-data-base-donnees-query-sql-mysql-postgre-sqlite

Version 1.5
Copyright : Yantra Technologies 2004-2022

20 –base de données : Exemple MYSQL

identifiant	pseudo	valeur
1	"toto"	1000
2	"tata"	750
3	"titi"	500
4	"toto"	250
5	"tata"	150
6	"tete"	0

- Exécuter une commande
  - Exécuter une requête : parcourir tous

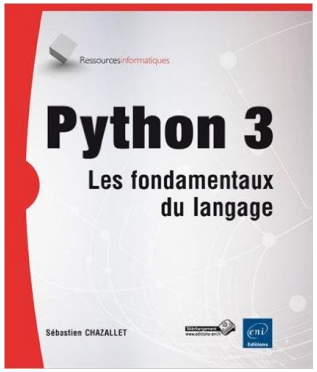
```
curseur.execute("SELECT * FROM scores")
resultats = curseur.fetchall()
for resultat in resultats:
    print(resultat)
```

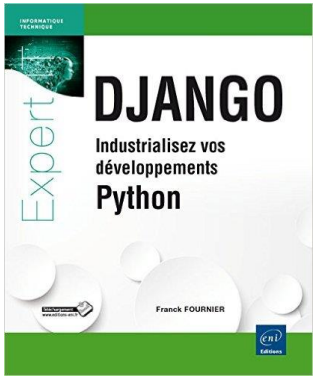
https://python.doctor/page-database-data-base-donnees-query-sql-mysql-postgre-sqlite

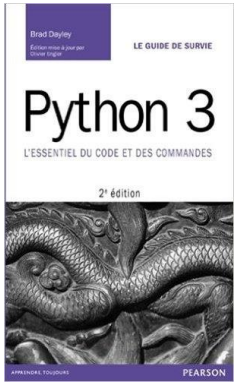
Version 1.5
Copyright : Yantra Technologies 2004-2022

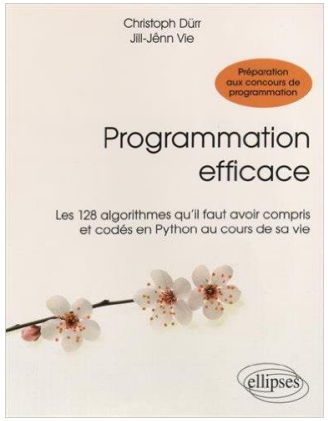
21 - Bibliographie

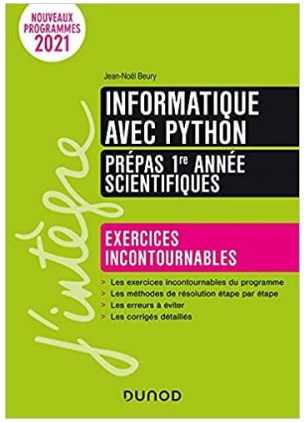
- https://docs.python.org/3/library/
- http://www.tutorialspoint.com/python/
- http://www.chicoree.fr/
- https://openclassrooms.com/courses/apprenez-a-programmer-en-python
- https://fr.wikibooks.org/wiki/Apprendre_%C3%A0_programmer_avec_Python
- **http://apprendre-python.com**











Copyright : Yantra Technologies 2004-2022