



جامعة عبد المالك السعدي
Université Abdelmalek Essaâdi

Université Abdelmalek Essaâdi

Faculté des Sciences - Tétouan

Département Informatique

Interface Homme Machine

Pr. Abdoun Otman

Département Informatique,
Faculté des Sciences,
Université Abdelmalek Essaâdi, Tétouan

Plan du cours

1. Introduction aux IHM
2. Principes ergonomiques des IHM : Règles et standards
3. Sciences cognitives, ergonomie et guides de style

Que veut dire par IHM ?

- ✓ IHM
 - ✓ Interface Homme ↔ Machine
 - ✓ Interactions Homme ↔ Machine
- ✓ Mais aussi
 - ✓ Communication Homme ↔ Machine
 - ✓ Dialogue Homme ↔ Machine
 - ✓ Interaction Personne ↔ Machine
- ✓ Finalement
 - ✓ Interface Humain ↔ Machine
 - ✓ Interactions Humain ↔ Machine



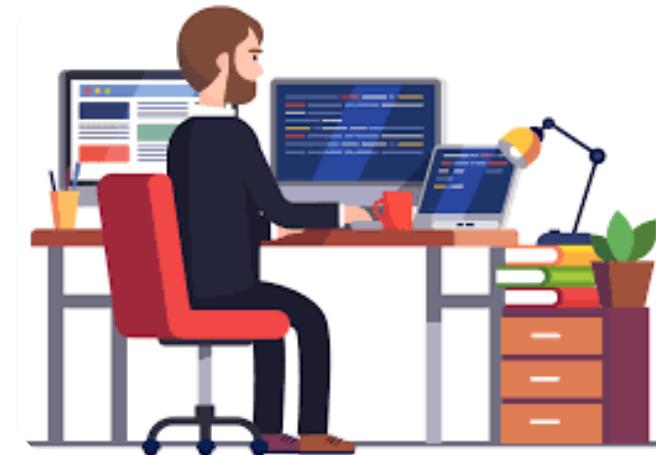
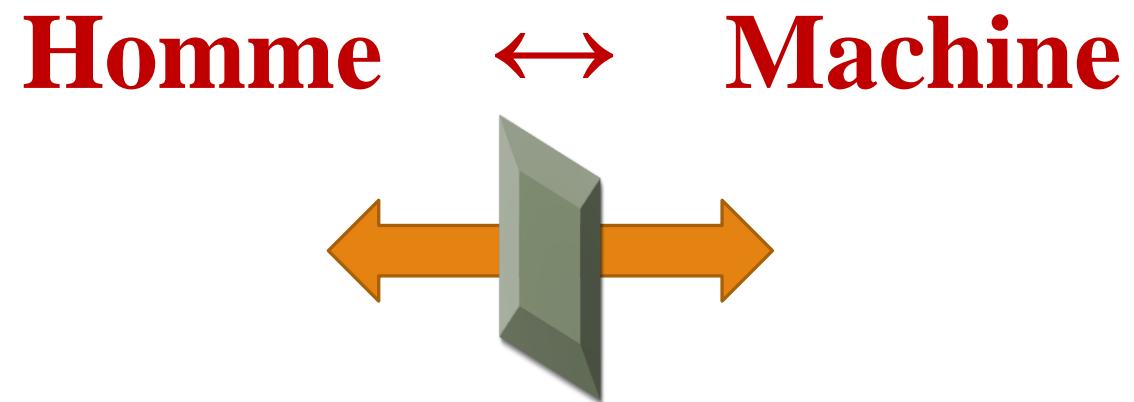
Définition IHM

INTERFACE HUMAIN - MACHINE

ensemble des dispositifs matériels et logiciels permettant à un utilisateur humain d'interagir avec un système numérique interactif

INTERACTION HUMAIN - MACHINE

ensemble des actions permettant la communication entre un système numérique interactif et son utilisateur humain



The user First !?

APPROCHE TECHNOCENTRÉE

- centrée sur la machine et ses possibilités
- l'utilisateur doit s'adapter à la machine
- point de vue concepteur

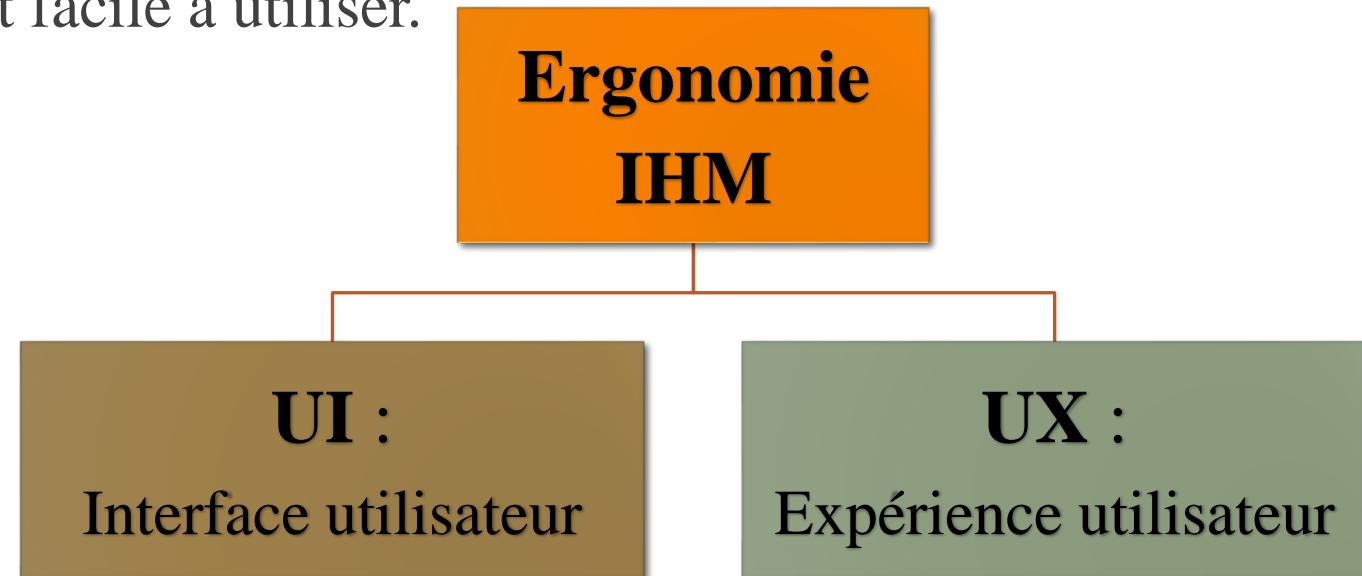
APPROCHE ANTHROPOCENTRÉE

- centrée sur l'humain et ses besoins
- la machine doit s'adapter à l'utilisateur
- point de vue utilisateur



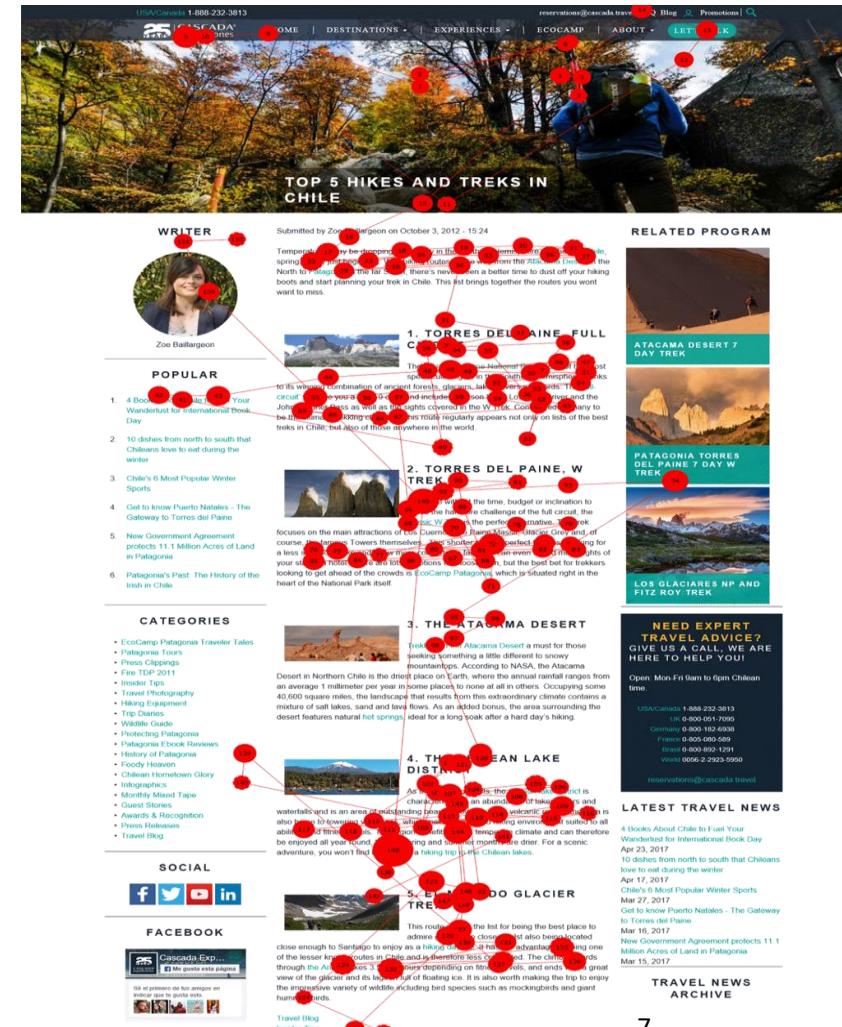
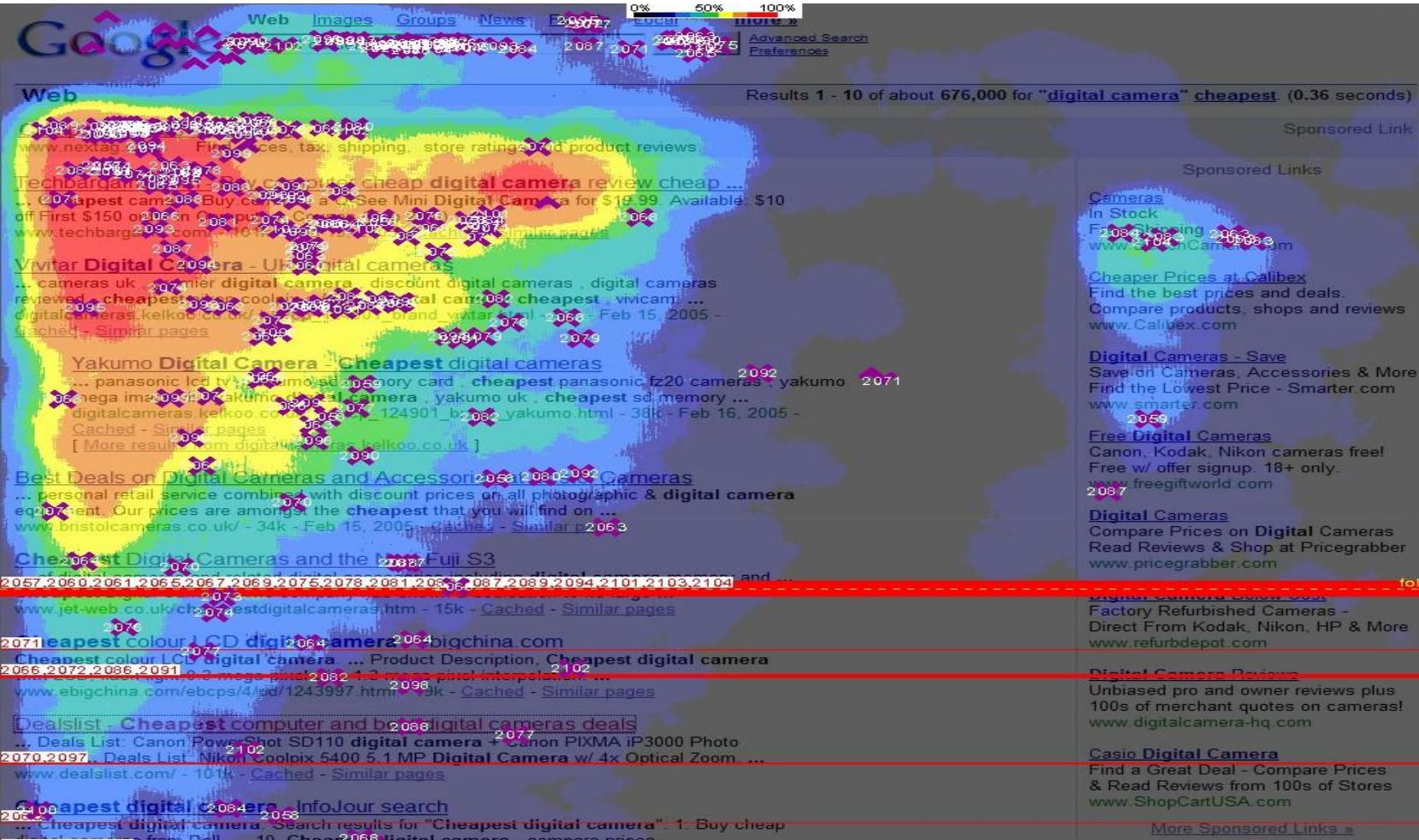
Ergonomie des IHMs

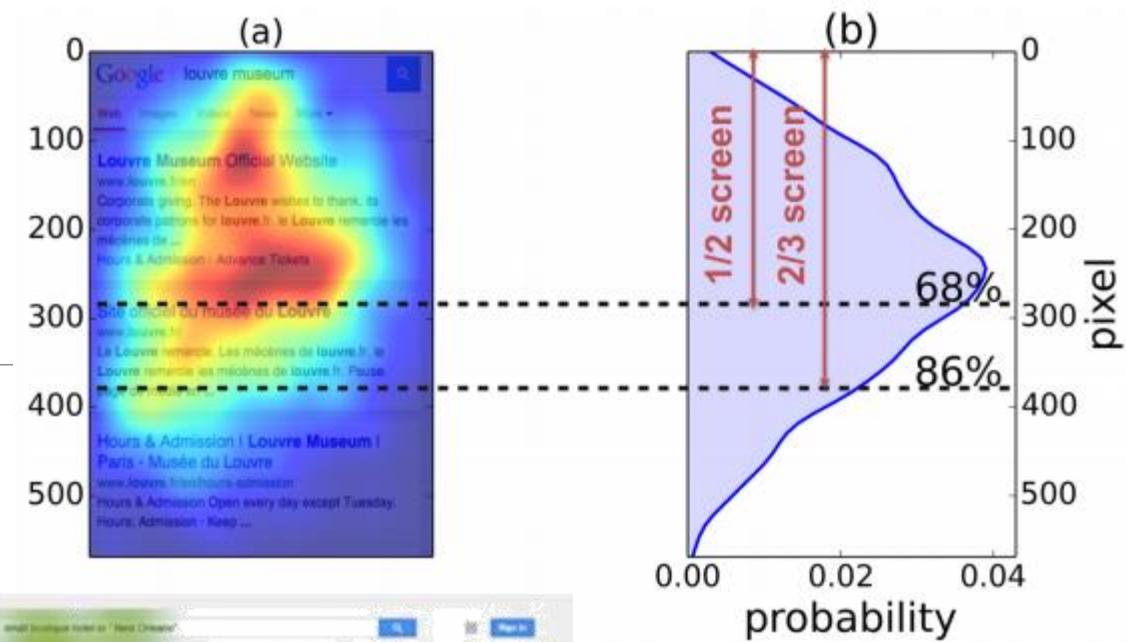
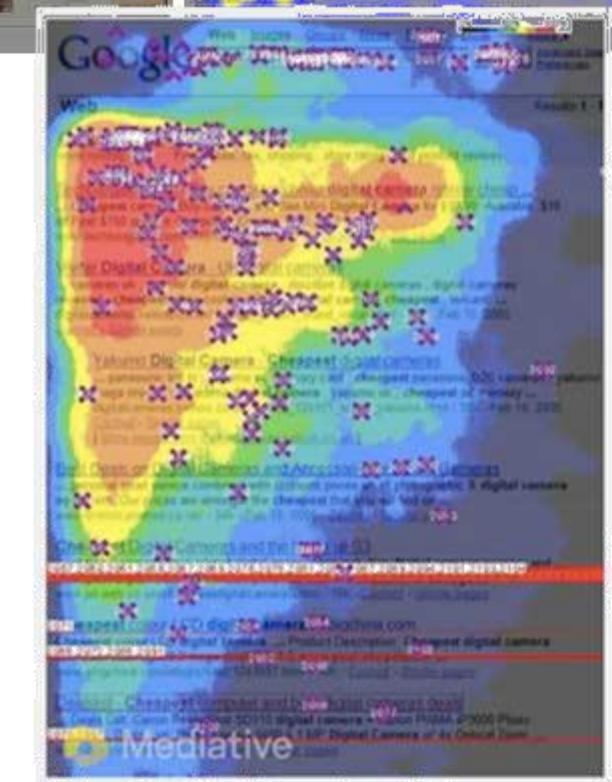
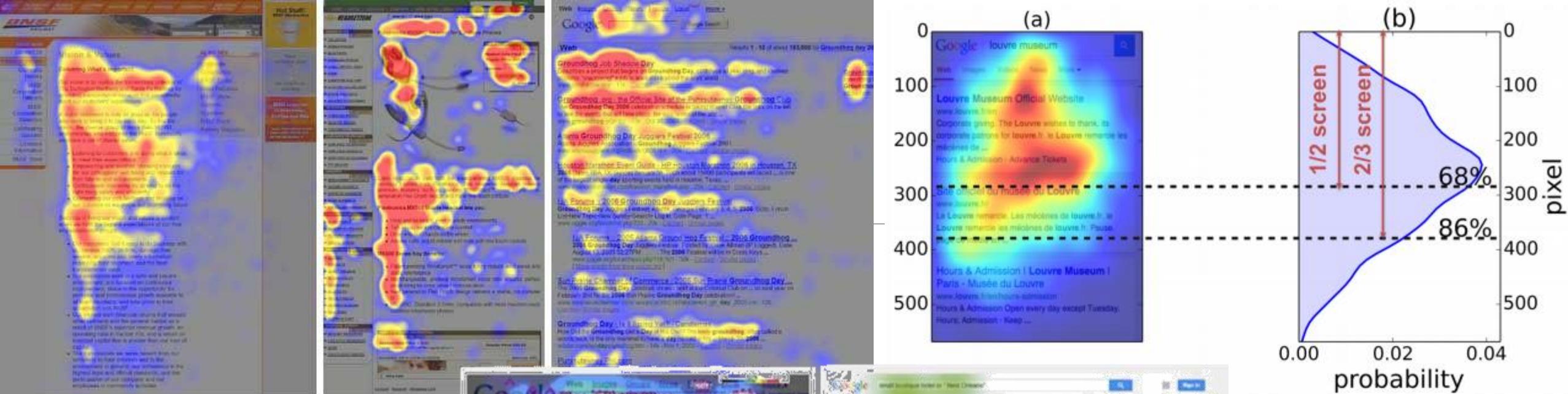
- Ergonomie des IHMs est une spécialité qui vise à rendre un Interface/Service maniable et facile à utiliser.



- La conception d'IHM ergonomiques est fondamentale à la réussite des projets de développement logiciel, notamment web et mobile.

Ergonomie des IHMs





Ergonomie des IHMs :

> *30 concepts-clés de l'utilisabilité d'IHM*

-
- | | |
|--|--|
| <ol style="list-style-type: none"> 1. Le principe des 7±2 éléments 2. La règle des 2 secondes 3. La règle des 3 clics 4. La loi de Pareto (20/80) 5. Les 8 règles d'or de la conception 6. La loi de Fitts 7. La pyramide inversée 8. Satisfaction 9. Le syndrome de l'oisillon 10. Banner blindness 11. Les effets Cliché et Zeigarnik 12. Les lois de la Gestalt 13. L'effet d'autoréférence 14. L'eye-tracking (ou oculométrie) 15. Le pli (fold) | <ol style="list-style-type: none"> 16. La zone fovéale 17. Les annotations 18. Dégradation élégante 19. La granularité 20. Les zones sensibles 21. La lisibilité perceptive 22. La navigation en démineur 23. La navigation mystère 24. La cohérence visuelle 25. L'enrichissement progressif 26. La lisibilité cognitive 27. La conception centrée utilisateur 28. La vigilance 29. Le design intuitif (Walk-Up-And-Use) 30. Les Wireframes |
|--|--|

C/C:

La conception d'IHM ergonomiques (**ergonomie IHM**), centrée sur les notions d'interface utilisateur (UI) et d'expérience utilisateur (UX), est fondamentale à la réussite des projets de développement logiciel, notamment web et mobile.

IHM : Ergonomie

Ergonomie vs design

- Design : dépendant des modes
- Ergonomie : constant (aux évolutions de la science près)

Idées reçues sur l'ergonomie



C'est facile



C'est une opération esthétique de l'écran



◦ nécessite une approche précoce, méthodique, itérative, expérimentale



C'est seulement une affaire de goût, de bon sens, d'intuition



◦ des règles à respecter, qui ont des sources scientifiques



Il existe une méthode miracle

◦ pas de solution clé en main



◦ des points de repères théoriques et expérimentaux, des savoir-faire,



◦ des questionnements



◦ des équilibres à trouver, des compromis à faire



IHM : Standard et Règles

Aux caractéristiques de l'utilisateur

- Différences physiques (âge, handicap)
- Connaissances et expérience (novice, expert, professionnel)
 - Dans le domaine de la tâche
 - En informatique, sur le logiciel
- Caractéristiques psychologiques
 - Visuel/auditif, logique/intuitif, analytique/synthétique...
- Caractéristiques socioculturelles
 - Format des dates 12.10.2021, des nombres décimaux 17,42 / 17.42
 - Sens d'écriture
 - Signification des icônes, des couleurs



The screenshot shows a Google search results page for the query "Steel industry development". It highlights the difference between the English search results (top) and the Arabic search results (bottom). The English results focus on international developments like the United States and China, while the Arabic results focus on local developments in Saudi Arabia.

IHM : Standard et Règles (2)

Au contexte

- grand public (proposer une prise en main immédiate)
- loisirs (rendre le produit attrayant)
- industrie (augmenter la productivité)
- systèmes critiques (assurer un risque zéro)

Caractéristiques de la tâche

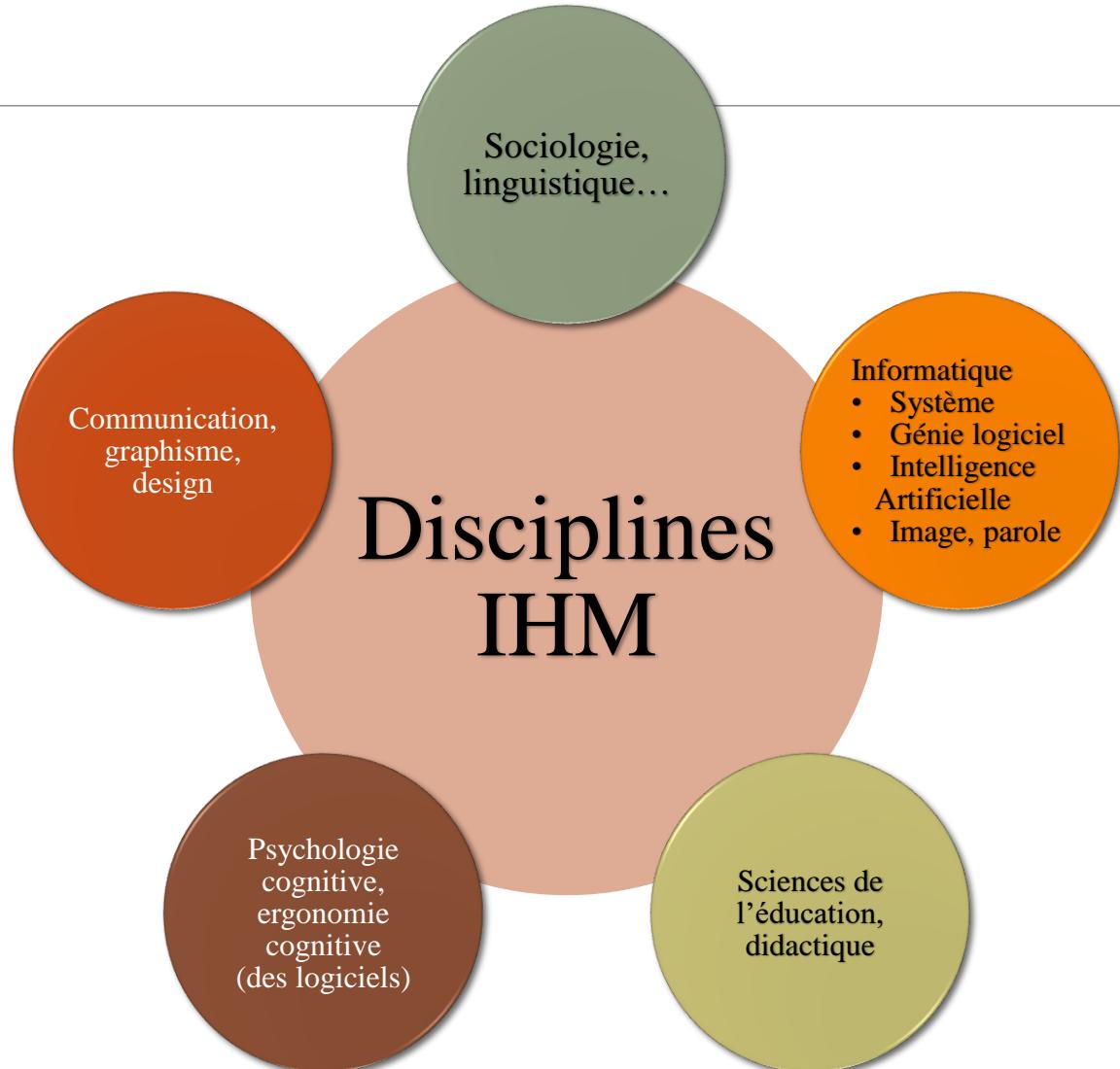
- usage occasionnel, régulier, quotidien, tâche répétitive
- sensible aux modifications de l'environnement, risquée,
- contrainte par le temps...

Contraintes techniques

- plateforme
- mémoire, bande passante
- écran, capteurs, effecteurs
- réutilisation de code ancien

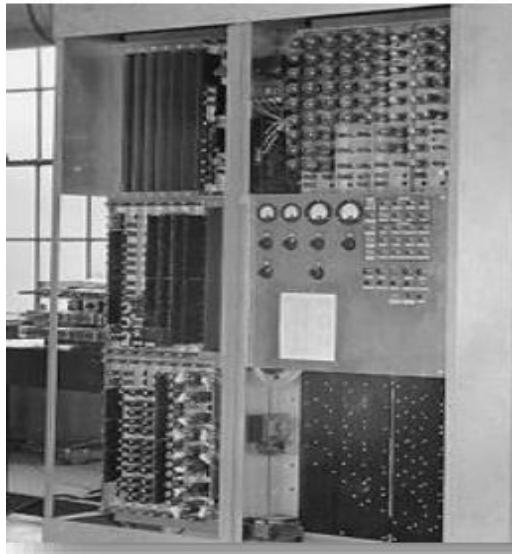
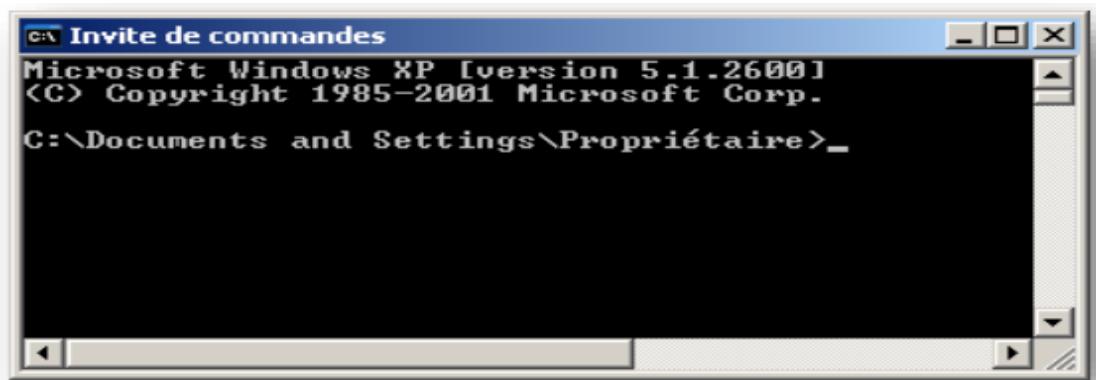


IHM : Domaine pluridisciplinaire



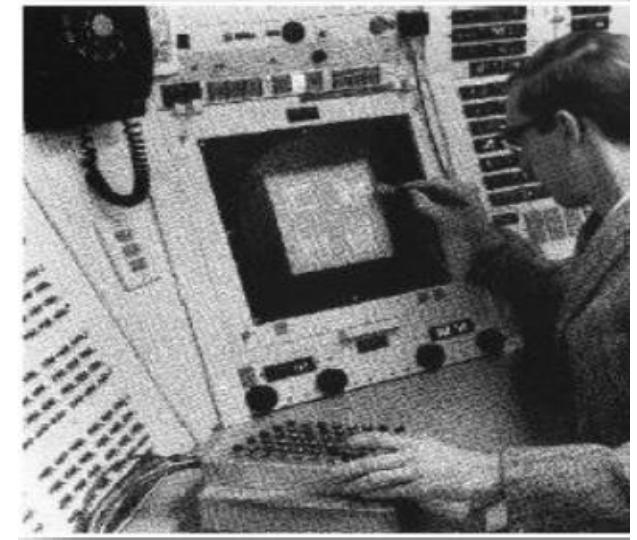
Historique, 1945-1970

- Dispositifs d'entrée-sortie limités:
 - perforateurs/lecteurs de cartes
 - tableaux de bord (voyants)
 - Imprimantes
- Langages de commandes



Historique : 1970 Ordinateurs modernes

- Nouveaux dispositifs d'entrée-sortie
 - 1963 : écran graphique et stylo optique
 - 1968 : première souris
 - 1980 : applications grand public
- Manipulation directe



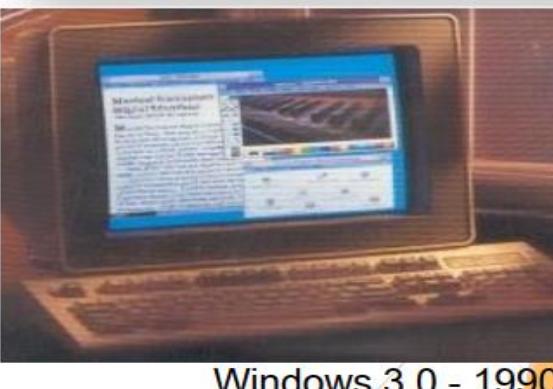
Xerox 8010 Star - 1981



Apple Lisa - 1982



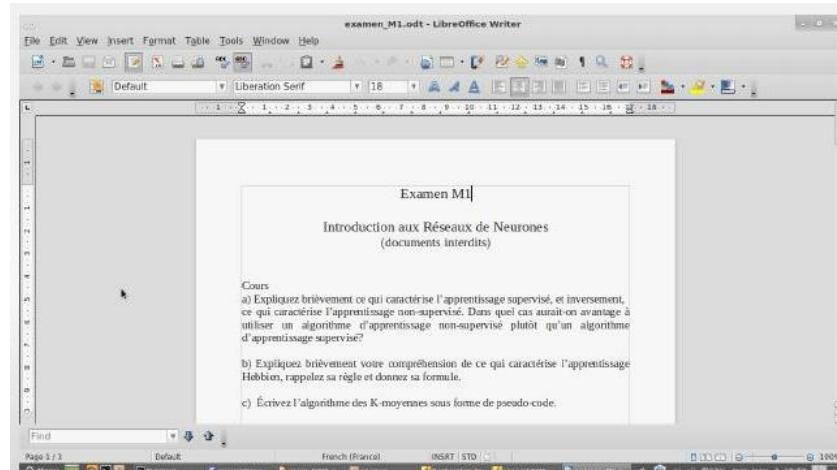
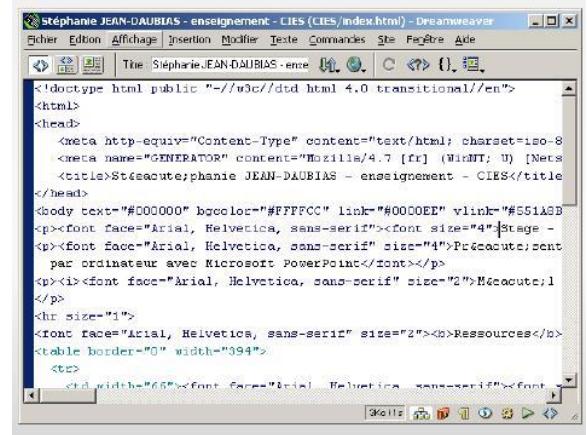
Macintosh - 1984



Windows 3.0 - 1990

Évolution des interfaces

- Systèmes plus conviviaux, faciles à comprendre et à utiliser,
- Interfaces graphiques
- manipulation directe
- action directe pour les objets représentés à l'écran
- WYSIWYG
- What You See Is What You Get
- ACAI : Achage Conforme A l'Impression



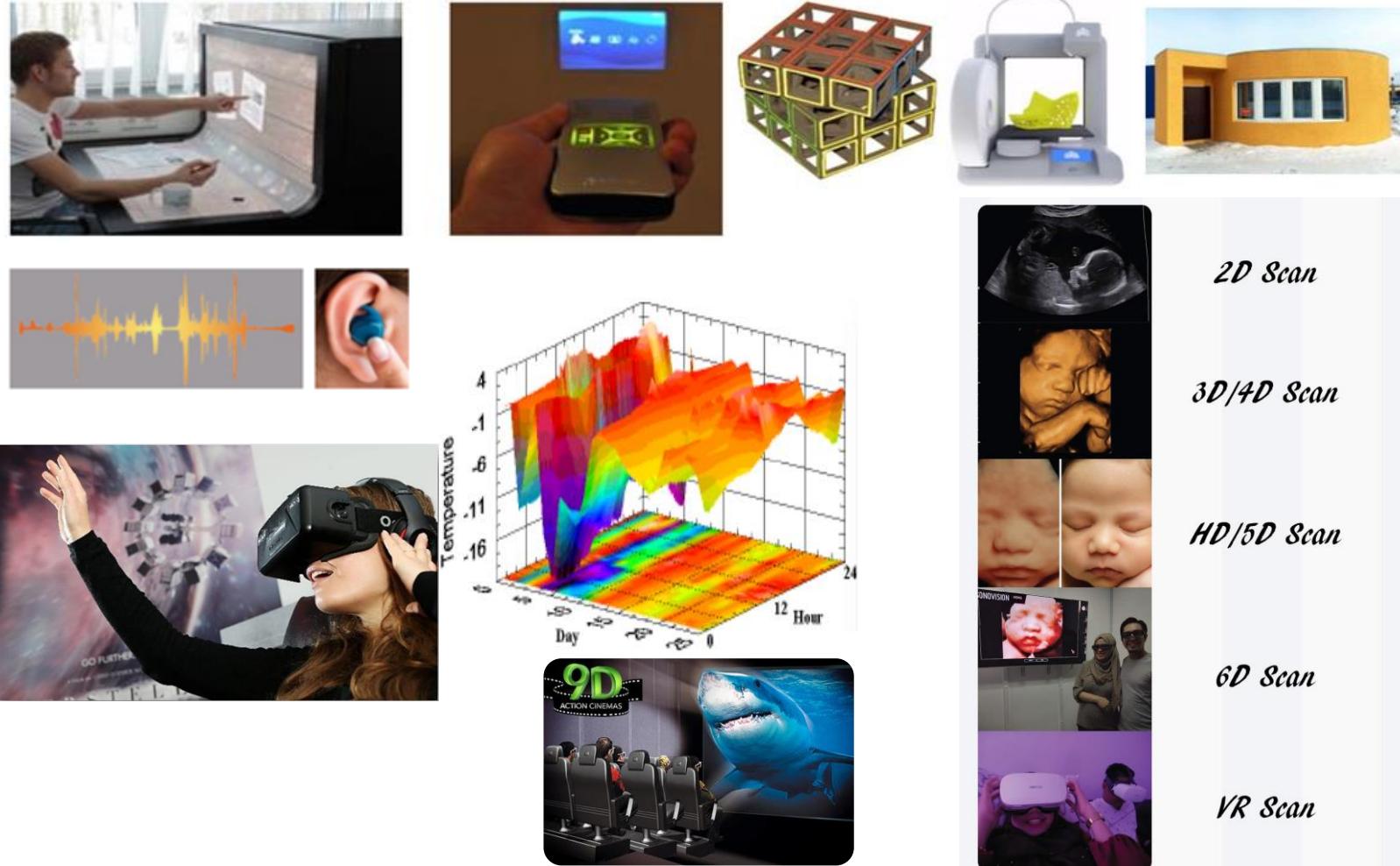
Évolution des interfaces

- Dispositifs de sortie :

 - écrans, cf. visualisation d'informations
 - Vision 3D,
 - Imprimantes 3D,
 - Son: synthèse vocale, son spatialisé,
 - retour tactile, retour de force,

- Dispositifs d'entrée :

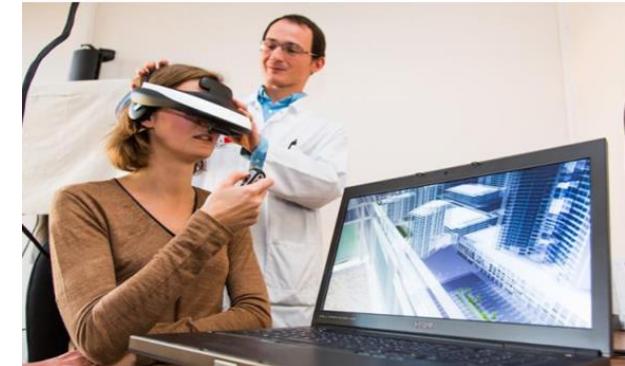
 - clavier (azerty, Dvorak)
 - souris, trackball, joystick, pavé tactile,
 - 2D: écran tactile, crayons optiques
 - 3D: capteurs de position et de direction.



Évolution des interfaces

- Réalité virtuelle:

- simulation d'un environnement dans lequel le sujet a l'impression d'évoluer (avatar),
- immersion dans un monde 3D,



- Réalité augmentée:

- superposition de l'image d'un modèle virtuel sur une image de la réalité
- en temps réel,
- le virtuel est intégré dans le réel,



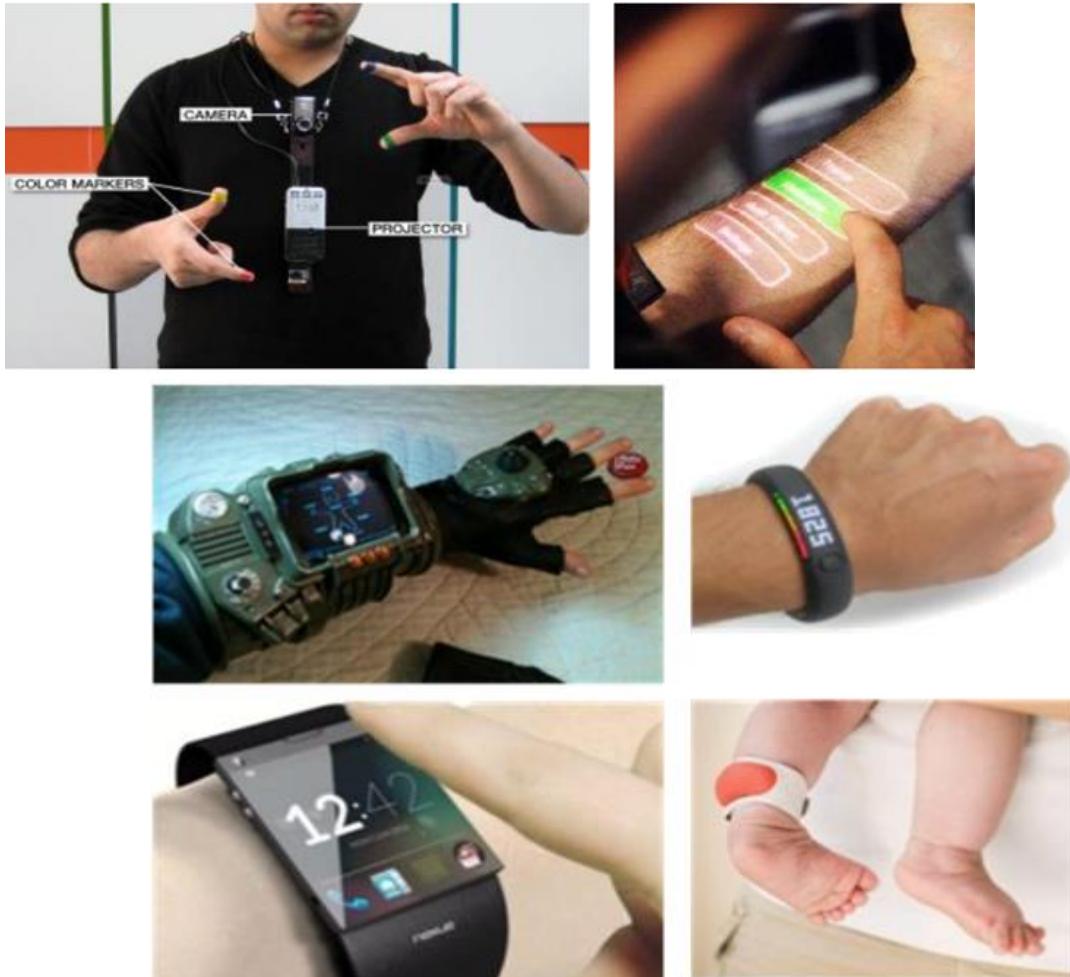
- Réalité diminuée:

- suppression d'un élément de l'image sur une image de la réalité en temps réel

- Réalité mixte:

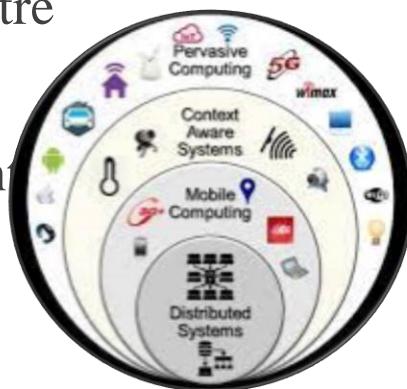
Évolution des interfaces

- Interfaces tangibles: association d'objets réels et numériques pour une interaction plus simple et intuitive,
- Réalité cliquable ,
- Manipulation virtuelle du monde réel: désignation d'une zone du monde réel par un geste .



Environnements pervasifs, ubiquitaires

- Environnement pervasif : des objets communicants qui se reconnaissent, se localisent, interagissent entre eux (transfert d'information, synchronisation des données), sans action de l'utilisateur, à tout moment



- Environnement ubiquitaire : pervasif, mobile
- Système interactif collaboratif présentiel/distante :

- tableau blanc interactif ou table tactile multipoint
- éditeurs partagés
- intégrant des moyens de communication



Retour à la réalité

- Écran, clavier, souris,



Plan du cours

1. Introduction aux IHM
2. Principes ergonomiques des IHM : Règles et standards
3. Sciences cognitives, ergonomie et guides de style

Questions

Plan 1

1. Introduction aux IHM
2. Principes ergonomiques des IHM : Règles et standards
3. Sciences cognitives, ergonomie et guides de style
4. Présentation des technologies du Web : HTML/XHTML
5. Développement en PHP et interaction en AJAX
6. Manipuler les interfaces graphiques avec XML/XSLT
7. Programmation d'interface homme-machine en langage Java : AWT
8. Les composants Java avancés : Swing
9. Développement Web en J2EE : Servlet & JSP

Plan 1

1. Introduction aux IHM
2. Principes ergonomiques des IHM : Règles et standards
3. Sciences cognitives, ergonomie et guides de style
4. ~~Présentation des technologies du Web : HTML/XHTML~~
5. ~~Développement en PHP et interaction en AJAX~~
6. Manipuler les interfaces graphiques avec XML/XSLT
7. ~~Programmation d'interface homme-machine en langage Java : AWT~~
8. ~~Les composants Java avancés : Swing~~
9. ~~Développement Web en J2EE : Servlet & JSP~~

Plan 2

1. Introduction aux IHM : Définition et Historique
2. Principes ergonomiques des IHM : Règles et standards
3. Sciences cognitives, ergonomie et guides de style
4. **Programmation sur Python**
5. **Data Management en Python:**
 - a. Sérialisation avec **PICKLE** et **JSON**
 - b. Stockage avec **SQLITE3**
 - c. Mapping en **XML/XSLT**
6. Interface graphique pour Python :
 - a. **TKINTER** pour Tk
 - b. **PYQT** pour Qt
 - c. **WXPYTHON** pour wxWidgets
7. Application Web sur Python : **DJANGO, FLASK,...**
8. IHM challenge : Smart IHM, Soya, SCADA

Plan V1

1. Introduction aux Interfaces Homme Machine (IHM)
2. Principes ergonomiques des IHM : Règles et standards
3. Sciences cognitives, ergonomie et guides de style
4. ~~Présentation des technologies du Web : HTML/XHTML~~
5. ~~Développement en PHP et interaction en AJAX~~
6. Manipuler les interfaces graphiques avec XML/XSLT
7. ~~Programmation d'interface homme-machine en langage Java : AWT~~
8. ~~Les composants Java avancés : Swing~~
9. ~~Développement Web en J2EE : Servlet & JSP~~

Plan

- 1. Introduction aux IHM : Définition et Historique**
- 2. Principes ergonomiques des IHM : Règles et standards**
- 3. Sciences cognitives, ergonomie et guides de style**
- 4. Programmation sur Python**
- 5. Data Management en Python**
- 6. Application Web sur Python**
- 7. IHM challenge : Smart IHM, Soya, SCADA**

Programmation de base sur Python



Programmation de base sur Python : Kernel



Python 3.10.0

Release Date: Oct. 4, 2021



Programmation de base sur Python : Getting started by using ?!

The slide features a large title 'Getting started with Python' in a serif font. To the right of the title is the Python logo, which consists of two interlocking snakes, one blue and one yellow. Below the title is a dark rectangular box with a light gray border, resembling a terminal window. Inside this window, there are several horizontal bars of varying colors (blue, green, white, orange, purple) of different lengths, creating a visual representation of data or code. The background of the slide is white with a subtle shadow effect at the bottom.

Getting started with
Python

O. ABDOUN

M2I & MQL : INTERFACE HOMME MACHINE - IHM

29

C:\Windows\system32\cmd.exe

Microsoft Windows [version 10.0.16299.1127]

(c) 2017 Microsoft Corporation. Tous droits réservés.

C:\Users\User>

C:\Users\User> PY

Python 3.10.0 (tags/v3.10.0:b494f59, Oct 4 2021, 19:00:18) [MSC v.1929 64 bit (AMD64)] on win32

Type "help", "copyright", "credits" or "license" for more information.

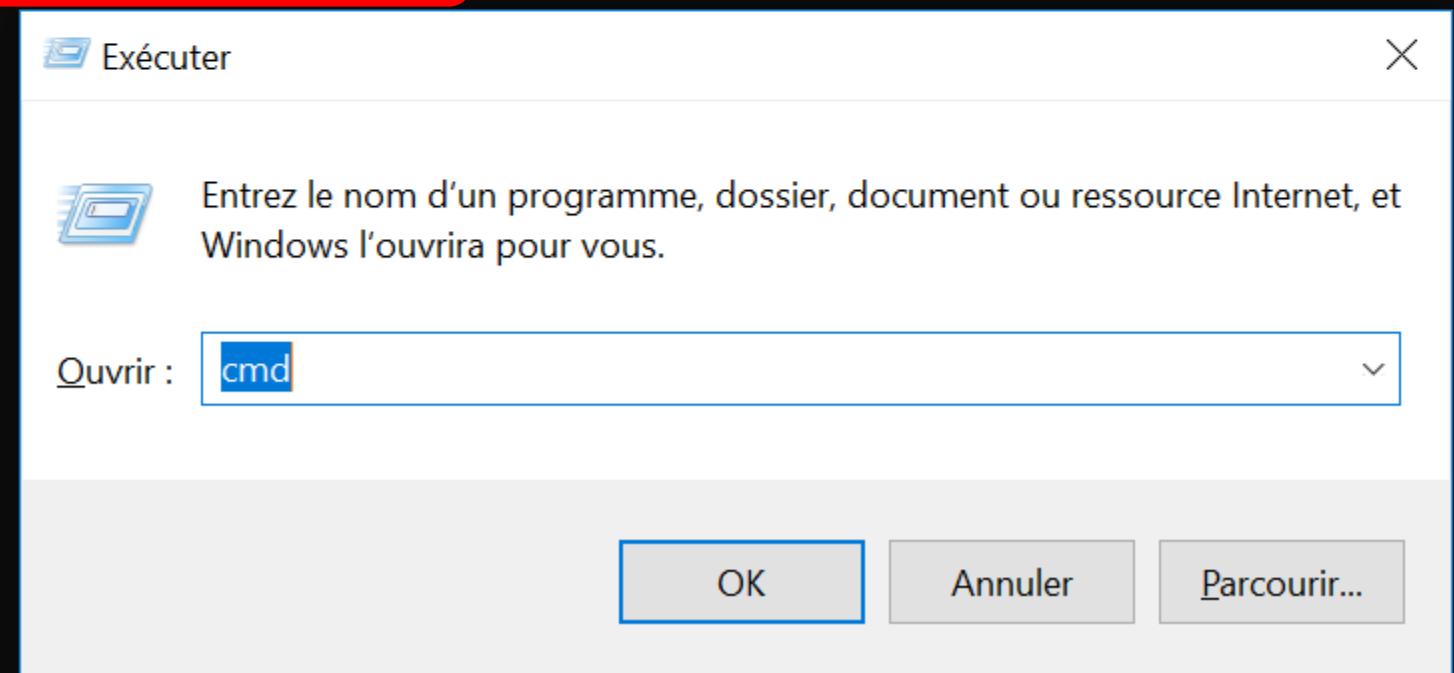
>>> print("Bonjour chers étudiants M2I - MQL <->")

Bonjour chers étudiants M2I - MQL <->

>>>

>>> exit()

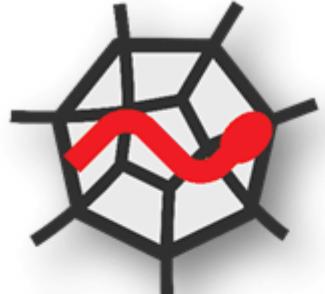
C:\Users\User>



Programmation de base sur Python : IDE



IDLE



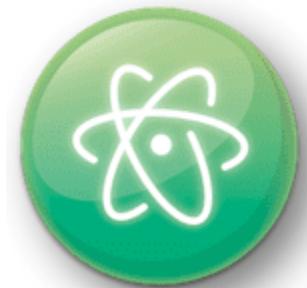
SPYDER



PyCharm



eric



Atom



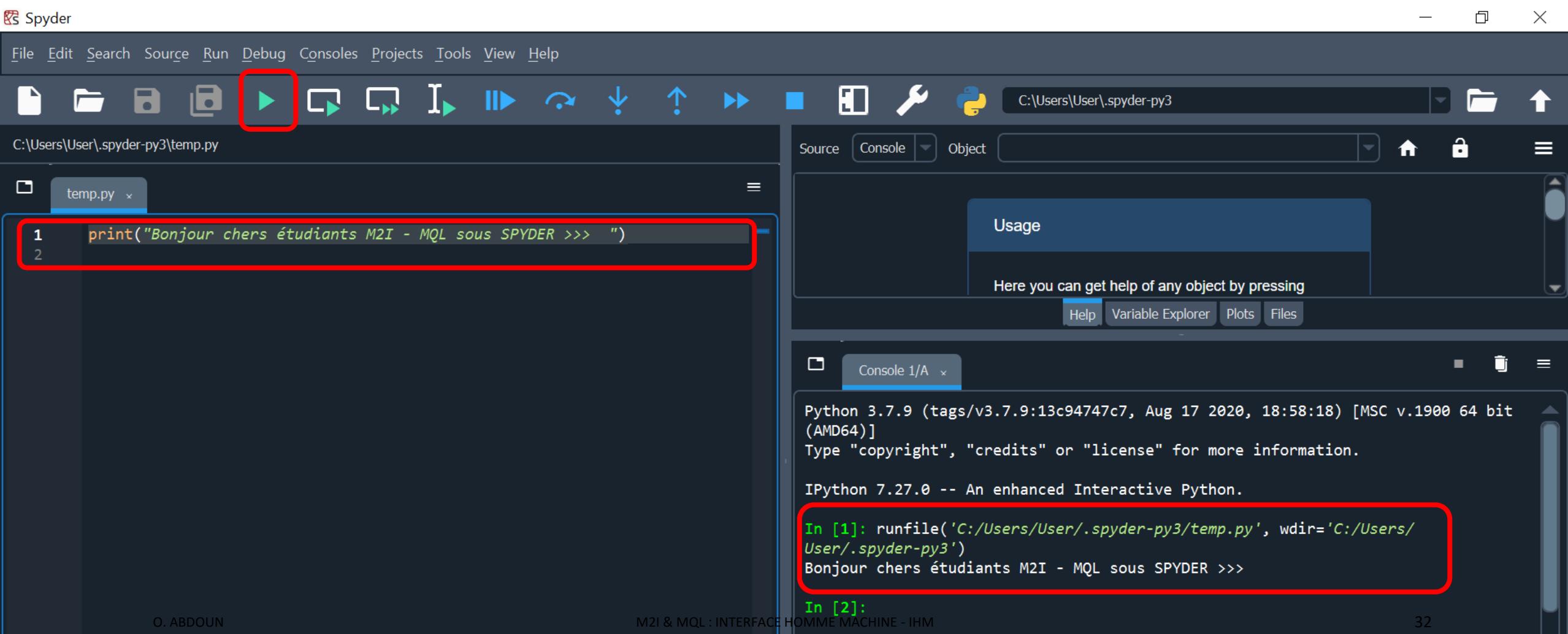
jupyter



Anaconda



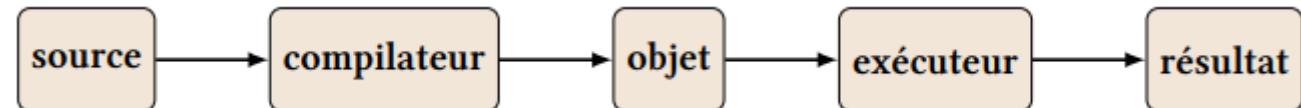
Programmation de base sur Python : Getting started with SPYDER



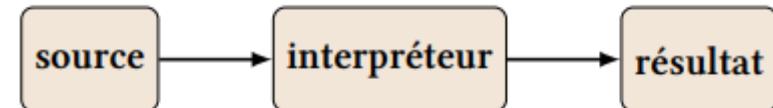
Programmation de base sur Python : Getting started : Compile/Run

Deux techniques de production des programmes :

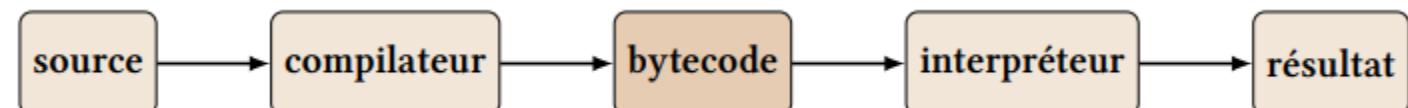
1. La **compilation** est la traduction du *source* en langage *objet*. La compilation est contraignante mais offre au final une grande vitesse d'exécution.



2. L'**interprétation** chaque ligne du *source* analysé est traduite au fur et à mesure en instructions directement exécutées. Aucun programme objet n'est généré.



3. Python mixte les deux méthodes : l'**interprétation du bytecode compilé**



Pour **exécuter un programme**, Python charge le fichier source .py en mémoire vive, en fait l'analyse produit le bytecode et enfin l'exécute. Python sauvegarde le bytecode produit (en .pyo ou .pyc) et le recharge en cas de besoin afin de compiler les modules.

Programmation de base sur Python :

PYTHON CALCULATOR

Programmation de base sur Python : Python Calculator

- ✓ En Python, un commentaire commence par le caractère # et s'étend jusqu'à la fin de la ligne :

c'est mon commentaire

- ✓ Python est un langage faiblement **typé**, le type d'une variable dépend de son contenu :

- `n=20` # n est un entier
- `ch='MQL & M2I'` # ch est une chaîne de caractères

- ✓ Pour avoir des informations sur un objet, exemple son type : **type(ch)**

```
n=20
print("n est de",type(n), ': ', n)

ch='MQL & M2I'
print('ch est de',type(ch), ': ',ch)
```

```
In [14]: runfile('C:/Users/User/.spyder-py3/temp.py', wdir='C:/Users/
User/.spyder-py3')
n est de <class 'int'> : 20
ch est de <class 'str'> : MQL & M2I
```

Python Calculator : Opérateurs de comparaison

Opérateur	Signification
==	Est égal à
!=	Est différent de
>=	Est plus grand ou égal que
<=	Est plus petit ou égal que
>	Est plus grand que
<	Est plus petit que
Opérateurs	
AND	[Condition 1] and [Condition 2] Si les deux conditions sont vraies => True
OR	[Condition 1] and [Condition 2] Si une des conditions est vraie => True
NOT	Not [Condition] Si la condition est fausse => True

```
a = 5
b = 6

print (a == b)    #Est ce que a est égal à b
print (a != b)   #Est ce que a est différent de b
print (a >= b)   #Est ce que a est supérieur ou égal à b
print (a <= b)   #Est ce que a est inférieur ou égal à b
print (a > b)    #Est ce que a est supérieur à b
print (a < b)    #Est ce que a est inférieur à b
print ("")
print (True and True)  #Opérateur "and"
print (True and False)
print (False and False)
print ("")
print (True or True)   #Opérateur "or"
print (True or False)
print (False or False)
print ("")
print not True        #Opérateur "not"
print not False
```

Python Calculator : Opérateurs de comparaison

- ✓ Test d'identité : **is, is not**

```
n=20
print('n : ', n)
print(n is 20)
print(type(n) is not int)
```

```
n : 20
True
False
```

- ✓ Appartenance à une séquence : **in, not in**

```
ch='MQL & M2I'
print('ch est de',type(ch), ': ',ch)

print('o' in ch)

print('I' not in ch)
```

```
ch est de <class 'str'> :  MQL & M2I
False
False
```

Python Calculator : Affectation

```

>>> v = 4 # affectation simple
>>> v += 2 # affectation augmentée. Idem à v = v + 2 si v est déjà référencé
>>> v
6
>>> c = d = 8 # d reçoit 8 puis c reçoit d (c et d sont des alias)
>>> c, d
(8, 8)
>>> e, f = 2.7, 5.1 # affectation parallèle d'un tuple
>>> (e, f)
(2.7, 5.1)
>>> g, h = ['G', 'H'] # affectation parallèle d'une liste
>>> [g, h]
['G', 'H']
>>> a = 3
>>> a, b = a + 2, a * 2 # toutes les expressions sont évaluées avant la première affectation
>>> a, b
(5, 6)

```

✓ Affectation : =

- ✓ Pour les Variables, **c = d** (**c** référencera le même objet que **d**)
- ✓ Pour les Listes, la copie se fait par **c = d[:]**
- ✓ Pour les Dictionnaires, la méthode **copy**, **c = d.copy()**.

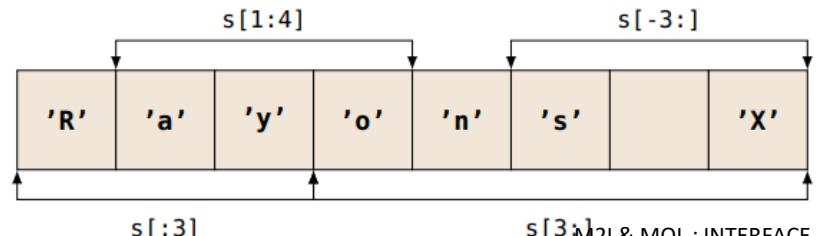
Python Calculator : Chaines de caractères

- ✓ C'est un type de données non modifiable **str**, dont une fois créée en mémoire, ne pourra plus être changée ; toute transformation résultera en la création d'une nouvelle valeur distincte.

<code>s[0]</code>	<code>s[1]</code>	<code>s[2]</code>	<code>s[3]</code>	<code>s[4]</code>	<code>s[5]</code>	<code>s[6]</code>	<code>s[7]</code>
'R'	'a'	'y'	'o'	'n'	's'		'x'
<code>s[-8]</code>	<code>s[-7]</code>	<code>s[-6]</code>	<code>s[-5]</code>	<code>s[-4]</code>	<code>s[-3]</code>	<code>s[-2]</code>	<code>s[-1]</code>

```
>>> s = "Rayon X"    # len(s) ==> 7
>>> s[0]
'R'
>>> s[2]
'y'
>>> s[-1]
'X'
>>> s[-3]
'n'
>>> s[1:4]           # de l'index 1 compris à 4 non compris
'ay'
>>> s[-2:]          # de l'index -2 compris à la fin
' x'
>>> s[:3]            # du début à l'index 3 non compris
'Ray'
>>> s[3:]            # de l'index 3 compris à la fin
'on X'
>>> s[::2]            # du début à la fin, de 2 en 2
'RynX'
>>> s[::-1]          # de la fin au début (retournement)
'X noyaR'
```

- ✓ Extraction de sous-chaînes. L'opérateur **[]** avec des indexes séparés par le caractère **:** permet d'extraire des sous-chaînes d'une chaîne.



Python Calculator : Chaines de caractères

✓ Méthodes de test de l'état d'une chaîne :

- ✓ **upper()** et **islower()** : retournent True si la chaîne ne contient respectivement que des majuscules/minuscules
- ✓ **istitle()** : retourne True si seule la première lettre de chaque mot de la chaîne est en majuscule
- ✓ **isalnum()**, **isalpha()**, **isdigit()** et **isspace()** : retournent True si la chaîne ne contient respectivement que des caractères alphanumériques, alphabétiques, numériques ou des espaces
- ✓ **startswith(prefix[, start[, stop]])** et **endswith(suffix[, start[, stop]])** : testent si la sous chaîne définie par **start** et **stop** commence respectivement par **prefix** ou finit par **suffix**

```
ch='Deux Masters Qualité Ingen'
print('ch est de',type(ch), ': ',ch)

print(ch.isupper())
print(ch.istitle())
print(ch.isalpha())
print(ch.startswith('Deux'))
print(ch.startswith('Un'))
print(ch.endswith('2I'))
print(ch.endswith('Ingen'))
```

```
ch est de <class 'str'> : Deux Masters Qualité Ingen
False
True
False
True
False
False
True
```

Python Calculator : Chaines de caractères

- ✓ Méthodes retournant une nouvelle chaîne :
 - ✓ **lower()**, **upper()**, **capitalize()** et **swapcase()** : retournent respectivement une chaîne en minuscule, en majuscule, en minuscule commençant par une majuscule, ou en casse inversée
 - ✓ **center(width[, fillchar])**, **ljust(width[, fillchar])** et **rjust(width[, fillchar])** : retournent respectivement une chaîne centrée, justifiée à gauche ou à droite, complétée par le caractère *fillchar* (ou par l'espace par défaut)
 - ✓ **zfill(width)** : complète *ch* à gauche avec des *O* jusqu'à une longueur maximale de *width*
 - ✓ **strip([chars])**, **lstrip([chars])** et **rstrip([chars])** : suppriment toutes les combinaisons de *chars* (ou l'espace par défaut) respectivement au début et en fin, au début, ou en fin d'une chaîne
 - ✓ **find(sub[, start[, stop]])** : renvoie l'index de la chaîne *sub* dans la sous-chaîne start à *stop*, sinon renvoie -1. **rfind()** effectue le même travail en commençant par la fin. **index()** et **rindex()** font de même mais produisent une erreur (exception) si la chaîne n'est pas trouvée
 - ✓ **replace(old, new[, count])** : remplace *count* instances (toutes par défaut) de *old* par *new*
 - ✓ **split(seps[, maxsplit])** : découpe la chaîne en *maxsplit* morceaux (tous par défaut). **rsplit()** effectue la même chose en commençant par la fin et **striplines()** effectue ce travail avec les caractères de fin de ligne
 - ✓ **join(seq)** : concatène les chaînes du conteneur *seq* en intercalant entre chaque élément la chaîne sur laquelle la méthode est appliquée

Python Calculator : Input/Output

- ✓ La saisie au clavier : la fonction standard **input()**

```
i=input("Donner i : ")
print('i est ',type(i),' : ',i)
```

```
Donner i : 10
i est <class 'str'> : 10
```

- ✓ La fonction **input()** effectue une saisie en mode texte (la valeur rentrée est une chaîne), dont on peut ensuite changer le type (on dit aussi « **transtyper** » ou **cast** en anglais) :

```
j=int(input("Donner j : "))
print('j est ',type(j),' : ',j)
```

```
Donner j : 20
j est <class 'int'> : 20
```

La fonction **print()** reste indispensable aux affichages dans les scripts :

Séquence	Signification
\\"	affiche un antislash
\'	apostrophe
\"	guillemet
\a	sonnerie (<i>bip</i>)
\b	retour arrière
\f	saut de page
\n	saut de ligne
\r	retour en début de ligne
\t	tabulation horizontale
\v	tabulation verticale

Programmation de base sur Python :

Contrôle du flux d'instructions

Contrôle du flux d'instructions : Instructions composées

- ✓ Une instruction composée contient :
 - ✓ une **ligne d'en-tête** terminée par **deux-points** :
 - ✓ un **bloc d'instructions** indenté par rapport à la ligne d'en-tête : **quatre espaces** par indentation

```
Note = 13

if Note < 10 :
    print('NV')
else :
    print('Validé')
    if Note > 12 :
        print(' Avec Mention ')
```

```
In [85]: runfile('C:/Users/User/.spyder-py3/temp.py', wdir='C:/Users/
User/.spyder-py3')
Validé
Avec Mention
```

Contrôle du flux d'instructions : Contrôler une alternative

- ✓ Un contrôle alternative **simple** est composé de **if** - [elif] - [else] :

```
>>> x = 5
>>> if x < 0:
...     print("x est négatif")
... elif x % 2 != 0:
...     print("x est positif et impair")
...     print ("ce qui est bien aussi!")
... else:
...     print("x n'est pas négatif et est pair")
...
x est positif et impair
ce qui est bien aussi!
```

```
A = 5
B = 10

if A < B :
    Min = A
else :
    Min = B

print('Minimum entre ',A,' et ',B,' : ',Min)

Minimum entre  5   et  10  :  5
```

- ✓ Un contrôle alternative **compacte** :

```
Min = A if A < B else B
```

Contrôle du flux d'instructions : Boucles

BOUCLE **FOR**

```
for i in range(a,b):
    instruction(s)
```

Exécute les instructions de la boucle pour les valeurs entières *i* suivantes :

a, a+1, a+2, ... jusqu'à la valeur b-1

```
n = 5
fact = 1

for i in range(1,n+1) :
    fact *= i
print('Factoriel de ',n,' est ',fact)
```

Factoriel de 5 est 120

BOUCLE **WHILE**

```
while test :
    instruction(s)
```

Exécute les instructions de la boucle tant que **test** est vrai.

```
n = 5
Sigma = 0

for i in range(1,n+1) :
    Sigma += i
print('Somme de 1 à ',n,' est ',Sigma)
```

Somme de 1 à 5 est 15

Contrôle du flux d'instructions : Ruptures de séquences

INTERROMPRE UNE BOUCLE : **BREAK**

```
for i in range(1,10) :  
    if i == 5:  
        break  
    print(i, end=' ')
```

1 2 3 4

COURT-CIRCUITER UNE BOUCLE : **CONTINUE**

```
for i in range(1,10) :  
    if i == 5:  
        continue  
    print(i, end=' ')
```

1 2 3 4 6 7 8 9

Contrôle du flux d'instructions :

Boucle avec **else**

- ✓ Les boucles **while** et **for** peuvent posséder une clause **else** qui ne s'exécute que si la boucle se termine normalement, c'est-à-dire sans interruption :

WHILE – ELSE

```
y = int(input("Entrez un entier positif : "))
while not(y > 0) :
    y = int(input('Entrez un entier positif, S.V.P. : '))

x = y // 2
while x > 1:
    if y % x == 0:
        print(x, "a pour facteur", y)
        break # voici l'interruption !
    x -= 1
else :
    print(y, "est premier.")
```

FOR - ELSE

```
>>> entiers = [2, 11, 8, 7, 5]
>>> cible = int(input("Entrez un entier : "))
Entrez un entier : 7
>>> for entier in entiers:
...     if cible == entier:
...         print(cible, 'est dans la séquence', entiers)
...         break # voici l'interruption!
... else:
...     print(cible, "n'est pas dans la séquence", entiers)
...
7 est dans la séquence [2, 11, 8, 7, 5]
>>> cible = int(input("Entrez un entier : "))
Entrez un entier : 6
>>>
>>> for entier in entiers:
...     if cible == entier:
...         print(cible, 'est dans la séquence', entiers)
...         break # voici l'interruption!
... else:
...     print(cible, "est absent de la séquence", entiers)
...
6 est absent de la séquence [2, 11, 8, 7, 5]
```

Programmation de base sur Python :

CONTENEURS STANDARD

Conteneurs standard

Séquences

- ✓ Une séquence est un conteneur ordonné d'éléments indexés par des entiers indiquant leur position dans le conteneur.
- ✓ Python dispose de trois types prédéfinis de séquences :
 - ✓ **Les chaînes**
 - ✓ **Les listes**
 - ✓ **Les tuples**

Conteneurs standard

Liste

- ✓ Une liste est une collection ordonnée et modifiable d'éléments éventuellement hétérogènes.
- ✓ Les éléments de la liste sont séparés par des **virgules**, et entourés de crochets.

```
Nom = ['E1', 'E2', 'E3']

Note = [13, 15, 8]

Liste = [Nom, Note]

for lis in Liste :
    for Val in lis :
        print(Val,end='\t')
print('\n')
```

```
E1  E2  E3
13  15  8
```

```
Jour = ['Lundi', 'Mardi', 'Mercredi', 'Jeudi', 'Vendredi', 'Samedi', 'Dimanche']

print(Jour)      ['Lundi', 'Mardi', 'Mercredi', 'Jeudi', 'Vendredi', 'Samedi', 'Dimanche']
print(Jour[0])   Lundi
print(Jour[-1]) Dimanche
```

Conteneurs standard

Liste : Initialisation

- ✓ Utilisation de la répétition, de l'itérateur d'entiers range() et de l'opérateur d'appartenance (in)

```
>>> truc = []
>>> machin = [0.0] * 3
>>> truc
[]
>>> machin
[0.0, 0.0, 0.0]

>>> liste_1 = list(range(4))
>>> liste_1
[0, 1, 2, 3]
>>> liste_2 = list(range(4, 8))
>>> liste_2
[4, 5, 6, 7]

>>> liste_1, liste_2, liste_3
(True, False, True)
```

Conteneurs standard

Liste : Méthodes

```
>>> nombres = [17, 38, 10, 25, 72]
>>> nombres.sort()
>>> nombres
[10, 17, 25, 38, 72]
```

```
>>> nombres.append(12)
>>> nombres.reverse()
>>> nombres
[12, 72, 38, 25, 17, 10]
```

```
>>> nombres.remove(38)
>>> nombres
[12, 72, 25, 17, 10]
>>> nombres.index(17)
3
```

```
>>> nombres[0] = 11
>>> nombres[1:3] = [14, 17, 2]
>>> nombres
[11, 14, 17, 2, 17, 10]
```

```
>>> nombres.pop()
10
>>> nombres
[11, 14, 17, 2, 17]
>>> nombres.count(17)
2
```

```
>>> nombres.extend([1, 2, 3])
>>> nombres
[11, 14, 17, 2, 17, 1, 2, 3]
```

Conteneurs standard

Tableaux en Python

- ✓ Un tableau **t** est une structure de données qui permet de stocker des éléments de manière ordonnée.
Chaque élément **x** appartenant à **t** est accessible par son indice (le numéro de la case de **t** dans laquelle est stockée **x**).
- ✓ En python, on définit un tableau en énumérant ses valeurs entre deux crochets. Par exemple, pour créer le tableau **t1** qui contient les valeurs **5**, **7** et **3**, on écrit :

t1 = [5, 7, 3]

- ✓ Pour créer un tableau **t2** composé de **10** cases toutes initialisées à **0**, on écrit :

t2 = [0]*10 équivalent à : **t1 = [0,0,0,0,0,0,0,0,0,0]**

- ✓ Pour accéder au contenu de la case numéro **i** du tableau **t**, on écrit : **t[i]**. Dont, la première case d'un tableau est la case numéro **0**
- ✓ Pour obtenir la longueur du tableau **t**, on écrit :

len(t)

Conteneurs standard

Tableaux en Python : Particularités

- ✓ Pour afficher un tableau **t**, on écrit :

print(t)

- ✓ On peut mélanger les types de données dans un tableau **t** (déconseillé) :

t = [5,"a",[5.2,3]]

- ✓ Copie de tableau

- ✓ Lorsqu'on écrit **t1 = t2**, on a simplement donné l'alias **t2** au tableau **t1** (**t1** et **t2** pointent vers les mêmes cases mémoires).
- ✓ Lorsqu'on change une valeur de **t1**, cette valeur sera donc également changée dans **t2** (et vice versa).

```
>>> t1=[3,9,6]
>>> print(t1)
[3, 9, 6]
>>> t2=t1
>>> print(t2)
[3, 9, 6]
>>> t2[0]=4
>>> print(t2)
[4, 9, 6]
>>> print(t1)
[4, 9, 6]
```

Conteneurs standard

Tableaux en Python : Copie

- ✓ Pour créer une véritable copie **t2** du tableau **t1**, on copie les valeurs case par case :

```
>>> t1=[3,9,6]
>>> print(t1)
[3, 9, 6]
>>> taille=len(t1)
>>> print(taille)
3
>>> t2=[0]*taille
>>> print(t2)
[0, 0, 0]
>>> for i in range(0, taille):
        t2[i]=t1[i]

>>> print(t2)
[3, 9, 6]
>>> t2[0]=4
>>> print(t2)
[4, 9, 6]
>>> print(t1)
[3, 9, 6]
```

Conteneurs standard

Tableaux à 2 dimensions

- ✓ Une matrice peut se coder comme un "tableau de tableaux". Par exemple, pour coder en Python la matrice suivante :

1	2
3	4
5	6

- ✓ On écrit :

m = [[1,2],[3,4],[5,6]]

- ✓ Pour accéder au contenu de la case de la ligne i et colonne j, on écrit :

m[i][j]

- ✓ Par exemple, si on veut afficher la valeur **6** de la matrice ci-dessus, on écrit :

print(m[2][1])

Conteneurs standard

Tableaux à 2 dimensions

- ✓ Pour initialiser créer une matrice de taille arbitraire, on utilisera une boucle.
- ✓ Par exemple, pour initialiser une matrice de 4 lignes, 6 colonnes, et qui ne contient que des zéros, on écrit :

```
>>> nb_lignes=4
>>> nb_colonnes=6
>>> m = [0]*nb_lignes
>>> for i in range(0,nb_lignes):
    m[i] = [0]*nb_colonnes

>>> print(m)
[[0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0]]
```

Conteneurs standard

Tuples

- ✓ Un tuple est une collection ordonnée et non modifiable d'éléments éventuellement hétérogènes.
- ✓ Éléments séparés par des virgules, et entourés de parenthèses.

```
>>> mon_tuple = ('a', 2, [1, 3])
```

- L'indexage des tuples s'utilisent comme celui des listes ;
- le parcours des tuples est plus rapide que celui des listes ;
- ils sont utiles pour définir des constantes.

Conteneurs standard

Les opérations des objets de type séquentiel

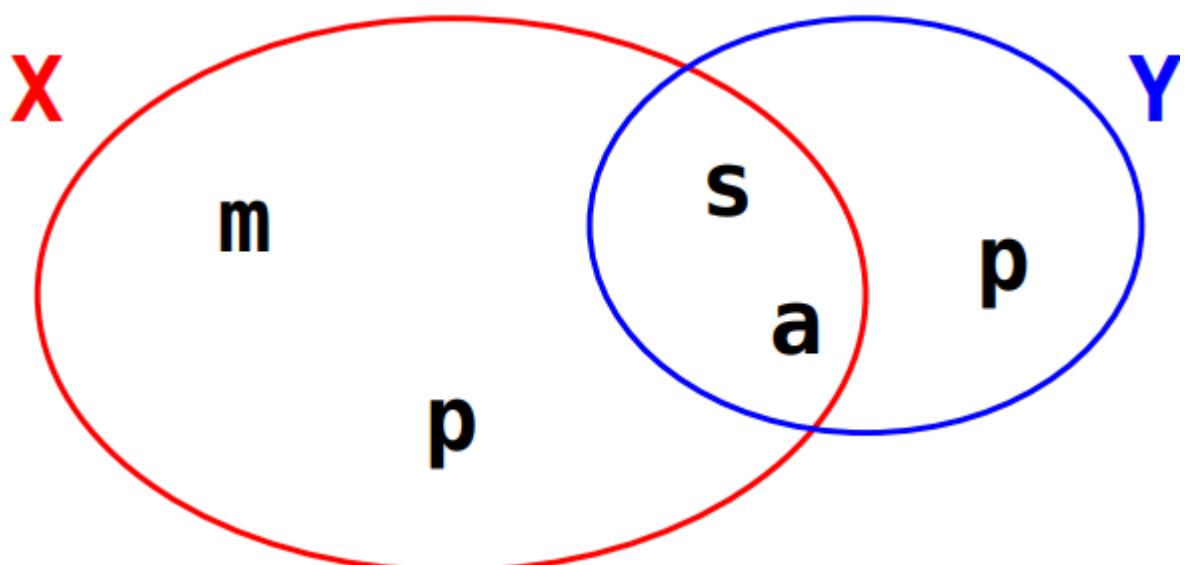
- ✓ Les types prédéfinis de séquences Python (chaîne, liste et tuple) ont en commun les opérations résumées dans le tableau suivant où s et t désignent deux séquences du même type et i, j et k des entiers :

l'opération	son effet
<code>x in s</code>	True si s contient x, False sinon
<code>x not in s</code>	True si s ne contient pas x, False sinon
<code>s + t</code>	concaténation de s et t
<code>s * n, n * s</code>	n copies (superficielles) concaténées de s
<code>s[i]</code>	i ^e élément de s (à partir de 0)
<code>s[i:j]</code>	tranche de s de i (inclus) à j (exclu)
<code>s[i:j:k]</code>	tranche de s de i à j avec un pas de k
<code>len(s)</code>	longueur de s
<code>max(s), min(s)</code>	plus grand, plus petit élément de s
<code>s.index(i)</code>	indice de la 1 ^{re} occurrence de i dans s
<code>s.count(i)</code>	nombre d'occurrences de i dans s

Conteneurs standard

Les ensembles

- ✓ Un ensemble est une collection itérable non ordonnée d'éléments hachables uniques , il existe deux types:
 - ✓ les ensembles modifiables : **set**
 - ✓ les ensembles non modifiables : **frozenset**



```
>>> X = set('spam')
>>> Y = set('pass')
>>> X
{'s', 'p', 'm', 'a'}
>>> Y # pas de duplication : qu'un seul 's'
{'s', 'p', 'a'}
>>> 'p' in X
True
>>> 'm' in Y
False
>>> X - Y # ensemble des éléments de X qui ne sont pas dans Y
{'m'}
>>> Y - X # ensemble des éléments de Y qui ne sont pas dans X
set()
>>> X ^ Y # ensemble des éléments qui sont soit dans X soit dans Y
{'m'}
>>> X | Y # union
{'s', 'p', 'm', 'a'}
>>> X & Y # intersection
{'s', 'p', 'a'}
```

Programmation de base sur Python :

FONCTIONS ET ESPACES DE NOMS

Fonctions et espaces de noms

KeyWord : def

- ✓ La définition d'une fonction se fait par le mot clé **def** suivi de l'identificateur de la fonction

```
def nom_fct (arg1,arg2,...,argn):
    <instruction>
    return <valeur>
```

- ✓ Passage des Arguments :

```
import math
def surfaceVolumeSphere(r) :
    surf = 4.0 * math.pi * r**2
    vol = surf * r/3
    return surf, vol
# programme principal
rayon = float(input('Rayon : '))
s, v = surfaceVolumeSphere(rayon)
print("Sphère de surface {:.g} et de volume {:.g}".format(s,v))
```

```
Rayon : 2
Sphère de surface 50.2655 et de volume 33.510321638291124
```

```
def division (x, y):
    return x//y, x%y

Q, R = division (9, 2)

print('Q = ', Q, 'R = ', R)
```

Q = 4 R = 1

Fonctions et espaces de noms

Les formes Lambda

- ✓ Les Lisp ont été ajouté à Python. Avec le mot-clé « **lambda** », de petites fonctions anonymes peuvent être créées. Elles sont limitées syntaxiquement à une expression unique.

Exemple : **f=lambda x,y,z: x+y+z**

f(2,3,4)

ce qui donne : 9

```
f=lambda x,y,z: x*y*z  
print(f(2,3,4))
```

24

Fonctions et espaces de noms

Passage d'une fonction en paramètre

- ✓ Puisque en Python une variable peut référencer une fonction, on peut transmettre une fonction comme paramètre :

```
>>> def f(x):
...     return 2*x+1
...
>>> def g(x):
...     return x//2
...
>>> def h(fonc, x):
...     return fonc(x)
...
>>> h(f, 3)
7
>>> h(g, 4)
2
```

Fonctions et espaces de noms

Paramètres avec valeur par défaut

```
def Master(nom, prenom, mast="QL", semestre="S3"):  
    print(prenom, nom, ", Master : ", mast, ", Semestre :", semestre)  
  
Master("Oumarou", "Fouziya", "2I", "S1 - S3")  
  
Master("Mkhechen", "Bilal", "2I")  
  
Master("Elattq", "Nadi")
```

```
Fouziya Oumarou , Master : 2I , Semestre : S1 - S3  
Bilal Mkhechen , Master : 2I , Semestre : S3  
Nadi Elattq , Master : QL , Semestre : S3
```

Fonctions et espaces de noms

Fonctions et espaces de noms

PASSAGE D'UN TUPLE DE VALEURS

```
def somme(*args) :
    """Renvoie la somme du tuple <args>."""
    resultat = 0
    for nombre in args :
        resultat += nombre
    return resultat

# Exemples d'appel :
print(somme(23)) # 23
print(somme(23, 42, 13)) # 78

    def somme(a, b, c) :
        return a+b+c

# Exemple d'appel :
elements = (2, 4, 6)
print(somme(*elements)) # 12
```

PASSAGE D'UN DICTIONNAIRE : KWARGS

```
def unDict(**kwargs) :
    return kwargs

# Exemples d'appels
## par des paramètres nommés :
print(unDict(a=23, b=42)) # {'a' : 23, 'b' : 42}

## en fournissant un dictionnaire :
mots = {'d' : 85, 'e' : 14, 'f' : 9}
print(unDict(**mots)) # {'e' : 14, 'd' : 85, 'f' : 9}
```

Fonctions et espaces de noms

Espaces de noms

- ✓ Résolution des noms : règle « LGI » :

Interne

Noms prédéfinis : open, len,...

Global

Noms affectés à la base d'un module

Noms déclarés global dans une fonction ou un module

Local

Noms affectés dans une fonction ou un module

```
def fonc(y) : # y et z sont affectés dans fonc => locaux
    global x # permet de modifier x ligne suivante
    x = x + 2
    z = x + y
    return z

x = 99
print(fonc(1)) # 102
print(x) # 101
```

Programmation de base sur Python :

MODULES ET PACKAGES

Modules et packages :

Module

- ✓ C'est un fichier script Python permettant de définir des éléments de programme réutilisables.
- ✓ Ce mécanisme permet d'élaborer efficacement des bibliothèques de fonctions ou de classes :
 - ✓ Réutilisation du code ;
 - ✓ La documentation et les tests peuvent être intégrés au module ;
 - ✓ Réalisation de services ou de données partagés ;
 - ✓ Partition de l'espace de noms du `sys`
- ✓ Chaque module a sa propre table de symboles privée, qui est utilisée comme table de symbole globale par toutes les fonctions définies dans le module. Ainsi, l'auteur d'un module peut utiliser des variables globales dans le module sans s'inquiéter des désaccords accidentels avec les variables globales d'un utilisateur.

Modules et packages :

Module

- ✓ L'instruction **import <nom_module>** donne accès à l'ensemble des définitions du module importé en utilisant le nom du module comme espace de nom:

```
import tkinter
print("Version TKINTER : ",tkinter.TkVersion) Version TKINTER : 8.6
```

- ✓ L'instruction **from <nom_module> import nom1, nom2,...** donne accès directement à une sélection choisie de noms définis dans le module:

```
from math import cos, pi
print(" Le cosinus de pi (",pi,") est : ",cos(pi))
```

```
Le cosinus de pi ( 3.141592653589793 ) est : -1.0
```

- ✓ Si **nom1, nom2,...** est remplacé par *****, permet d'obtenir alors une **copie de tous les noms** définis à la racine du module.

Modules et packages :

Module

- ✓ ***random*** : fonctions permettant de travailler avec des valeurs aléatoires
- ✓ ***math*** : toutes les fonctions utiles pour les opérations mathématiques (cosinus,sinus,exp,etc.)
- ✓ ***sys*** : fonctions systèmes os : fonctions permettant d'interagir avec le système d'exploitation
- ✓ ***time*** : fonctions permettant de travailler avec le temps
- ✓ ***calendar*** : fonctions de calendrier
- ✓ ***profile*** : fonctions permettant d'analyser l'exécution des fonctions
- ✓ ***urllib2*** : fonctions permettant de récupérer des informations sur internet
- ✓ ***re*** : fonctions permettant de travailler sur des expressions régulières

Modules et packages :

Module : Random

```
# Entier aléatoire dans l'intervalle [1,10]
from random import randint
print("un entier dans [1,10] :")
print(randint(1,10))

print("Dix entiers dans [1,10] :")
for i in range(0,10):
    print(randint(1,10))

# Réel aléatoire dans l'intervalle [1,10]
from random import uniform
print("un réel dans [1,10] :")
print(uniform(1,10))

print("Dix réels dans [1,10] :")
for i in range(0,10):
    print(uniform(1,10))
```

```
>>>
un entier dans [1,10] :
7

Dix entiers dans [1,10] :
9
10
10
1
9
6
4
7
2
9

un réel dans [1,10] :
2.044407339209075

Dix réels dans [1,10] :
7.7196867882233455
5.169837282939778
1.4407085060701377
8.75850218769779
8.295436977033223
2.11400487460969
8.45403339823567
3.5033880658645304
2.0713147209483065
5.828660809242383
```

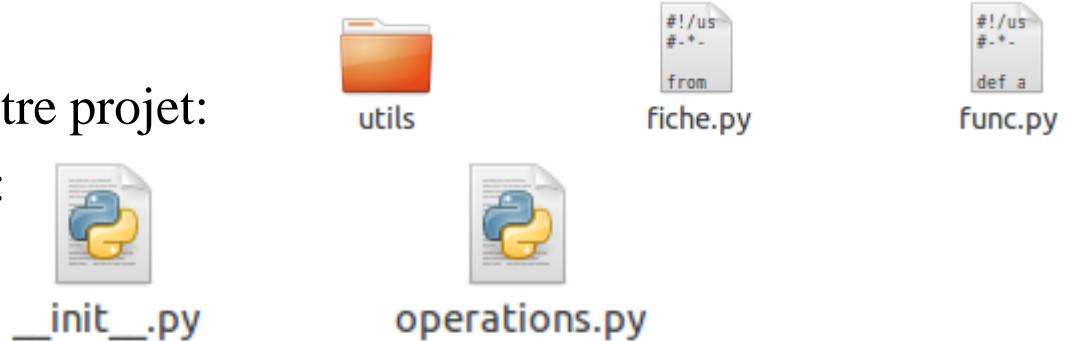
Modules et packages : Packages

- ✓ C'est une organisation en une arborescence de répertoires

- ✓ Pour être reconnu comme un package valide, chaque répertoire du paquet doit posséder un fichier `__init__` qui peut soit être vide soit contenir du code d'initialisation.

Modules et packages : Packages : Création

- ✓ Il faut créer, dans le même dossier du programme, un dossier portant le nom de votre package.
- ✓ Dans notre exemple, nous le nommerons "**utils**".
- ✓ Dans ce dossier, créons le fichier suivant: **`__init__.py`**, cela indique à python qu'il s'agit d'un package . Ce fichier peut être vide, seule sa présence est importante.
- ✓ Ensuite créons un fichier toujours dans ce répertoire **utils** que nous nommerons par exemple "**`operations.py`**"
- ✓ Contenu du dossier de votre projet:
- ✓ Contenu du dossier **utils** :



Programmation de base sur Python :

POO sous Python

POO sous Python : Avant propos

- ✓ Encapsulation : Structure de données va ensemble avec ses procédures de traitement
`liste.append(element)`
- ✓ Héritage: afin d'éviter la duplication de code
`fichier`
`fichierZip`
`fichierTexte`
- ✓ Surcharge : Un même nom pour différentes fonctions (distinguées par contexte - ou signature)
`mafonction(a,b,c) / mafonction(a)`
- ✓ ça n'existe pas en Python ! Un seul nom admis !

POO sous Python : Introduction

- ✓ Python offre la possibilité de programmer en Orienté objet.
- ✓ Le mécanisme de classe en Python est un mélange des mécanismes de classes de C++ et de Modula-3.
- ✓ Ex : Comme en C++, tous les membres d'une classe sont publics, et toutes les fonctions membres sont virtuelles. Il n'y a pas de constructeur ni de destructeur particulier.
- ✓ Comme en Modula-3, il n'y a pas de raccourcis pour faire référence aux membres d'un objet à partir de ses méthodes.

POO sous Python : Syntaxe

- ✓ La forme la plus simple de définition de classe ressemble à ceci :

```
class Nomclasse :  
    <instruction1>  
    ...  
    <instructionN>
```

- ✓ Dans la pratique, les instructions à l'intérieur de la définition de classe seront souvent des définitions de fonctions, mais d'autres instructions sont acceptées.
- ✓ L'objet classe permet deux types d'opération : La référenciation des attributs et l'instanciation.
- ✓ La référence d'attributs : *Nomclasse.i* où *i* est un attribut de *Nomclasse* est une références d'attribut valide.
- ✓ L'instanciation : création d'une instance (objet) d'une classe *Nomclasse*
x = Nomclasse() affecte la nouvelle instance à la variable *x*

POO sous Python : Class – Création d'objets

- ✓ Les class permettent de générer des objets indépendants et pouvant interagir avec le programme principal.
- ✓ Une class peut comprendre plusieurs attributs qui peuvent être appelés dans le programme principal.

```
class Exemple:  
    Nom_class = "Exemple"  
    Nombre_attributs = 3  
    Type_de_variable = "Objet"  
  
a = Exemple()  
b = Exemple()  
b.Nom_class = "Super class"  
  
print(a.Nom_class, b.Nom_class)  
print(a.Nombre_attributs, b.Nombre_attributs)  
print(a.Type_de_variable, b.Type_de_variable)  
  
=>>>  
('Exemple', 'Super class')  
(3, 3)  
('Objet', 'Objet')  
=>>>
```

POO sous Python : Class – Attributs

- ✓ Une variable définie *au niveau d'une classe* (comme **x** dans la classe **C**) est appelé **attribut de classe** et est partagée par tous les objets instances de cette classe.
- ✓ Une variable définie *au niveau d'un objet* (comme **y** dans l'objet **a**) est appelée **attribut d'instance** et est liée uniquement à l'objet pour lequel elle est définie.

```
class C :
    """Documentation de la classe C."""
    x = 23 # attribut de classe
    y = 'un string'
    z = [1, x, 3, y]

# a est un objet de la classe C (ou une instance)
a = C()
# Affichage des attributs de l'instance a
print(a.x)
print(a.y)
print(a.z)
```

```
class C :
    """Documentation de la classe C."""
    x = 23 # attribut de classe

    a = C() # a est un objet de la classe C (ou une instance)
    a.x # 23 : affiche la valeur de l'attribut de l'instance a
    a.x = 12 # modifie son attribut d'instance (attention...)
    a.x # 12
    C.x # 23 mais l'attribut de classe est inchangé
    C.z = 6 # z : nouvel attribut de classe
    a.y = 44 # y : nouvel attribut de l'instance a
    b = C() # b est un autre objet de la classe C (une autre instance)
    b.x # 23 : b connaît bien son attribut de classe, mais...
    b.y # ... b n'a pas d'attribut y !
```

POO sous Python : Class – Initialisation

- ✓ Lorsqu'un objet est créé, il est possible d'initialiser certains attributs:

```
class Cube:  
    def __init__(self, arete):  
        self.surface_face = arete**2  
        self.surface_totale = self.surface_face * 6  
        self.volume = arete**3  
  
    Cube1 = Cube(10)  
    Cube2 = Cube(5)  
  
    print(Cube1.surface_face, Cube2.surface_face)  
    print(Cube1.surface_totale, Cube2.surface_totale)  
    print(Cube1.volume, Cube2.volume)
```

- ✓ Pour initialiser la class, vous devez nommer votre première fonction « **__init__** ». Le « **self** » correspond à votre objet (ici, « **Cube** ») et vous pouvez ajouter des arguments.

POO sous Python : Class – Méthodes

- ✓ Une méthode est appelée comme en C++ :

x.f()

- ✓ La fonction f définie dans MaClasse a été appelée sans paramètre alors que la définition en contenait un.
- ✓ Ca devrait normalement déclencher une exception. Mais il n'en est rien car la particularité des méthodes est que l'objet est passé comme premier argument de la fonction.
- ✓ Le plus souvent par convention appelé « **self** » pour améliorer la lisibilité du programme.
- ✓ L'appel **x.f()** est donc équivalent à **MaClasse.f(x)**.
- ✓ Il n'est pas nécessaire que la définition de la méthode soit comprise dans la définition de la classe.

```
def f1 (self, x, y) :  
    return min(x,x+y)  
  
class C :  
    f=f1
```

POO sous Python : Class – Méthodes

```

class Cube:
    nom = "Cube"

    def __init__(self, arete):
        self.surface_face = arete**2
        self.surface_totale = self.surface_face * 6
        self.volume = arete**3

    def help_me(self):
        print("Il y a 4 arguments:\nnom\nsurface_face\nsurface_totale\nvolume")

    def resumer(self):
        print("Le nom de l'element est %s"%(self.nom))
        print("La surface d'une face est %.2f"%(self.surface_face))
        print("La surface totale est %.2f"%(self.surface_totale))
        print("Le volume est %.2f"%(self.volume))

Cube1 = Cube(10)
Cube1.nom = "Cube1"
Cube2 = Cube(5)
Cube2.nom = "Cube2"

Cube1.help_me()
Cube1.resumer()
Cube2.resumer()

O.ABDOUN

```

```

>>>
Il y a 4 arguments:
nom
surface_face
surface_totale
volume
Le nom de l'element est Cube1
La surface d'une face est 100.00
La surface totale est 600.00
Le volume est 1000.00
Le nom de l'element est Cube2
La surface d'une face est 25.00
La surface totale est 150.00
Le volume est 125.00
>>>

```

POO sous Python : Méthodes prédéfinies

Méthodes	Surcharge	Appelée pour
<code>_init_</code>	Constructeur	Création d'objet : Classe()
<code>_del_</code>	Destructeur	Suppression d'objet
<code>_add_</code>	Opérateur '+'	X+Y
<code>_or_</code>	Opérateur 'l' (ou bit-à-bit)	X1Y
<code>_repr_</code>	Affichage, conversions	print X, 'X'
<code>_call_</code>	Appels de fonctions	X()
<code>_getattr_</code>	Qualification	X indéfini
<code>_getitem_</code>	Indication	X[cle], boucles for, tests in
<code>_setitem_</code>	Affectation par indice	X[cle] = valeur
<code>_getslice_</code>	Découpage	X[bas:haut]
<code>_len_</code>	Longueur	len(X), tests
<code>_cmp_</code>	Comparaison	X=Y,X<Y
<code>_radd_</code>	Opérateur '+' de côté droit	Non instance + X

POO sous Python : Exemple de surcharge

- ✓ Dans l'exemple suivant nous surchargeons l'opérateur d'addition pour le type Vecteur2D.
- ✓ Nous surchargeons également la méthode spéciale `__str__` utilisée pour l'affichage par `print()`

```
>>> class Vecteur2D:  
...     def __init__(self, x0, y0):  
...         self.x = x0  
...         self.y = y0  
...     def __add__(self, second): # addition vectorielle  
...         return Vecteur2D(self.x + second.x, self.y + second.y)  
...     def __str__(self):          # affichage d'un Vecteur2D  
...         return "Vecteur({:g}, {:g})".format(self.x, self.y)  
...  
>>> v1 = Vecteur2D(1.2, 2.3)  
>>> v2 = Vecteur2D(3.4, 4.5)  
>>>  
>>> print(v1 + v2)  
Vecteur(4.6, 6.8)
```

POO sous Python :

Méthode __init__(self)...

- ✓ Lorsqu'une class définit la méthode spéciale __init__(), l'instanciation de la classe appelle automatiquement cette méthode.
- ✓ La méthode __init__() peut prendre autant de paramètre qu'on le vaut, pour une flexibilité accrue...

```
class Complexe :  
    def __init__(self, partiereelle, partieimaginaire) :  
        self.r = partiereelle  
        self.i = partieimaginaire  
x = Complexe(3.0,-4.5)
```

POO sous Python : Exemple : SWITCH

```
-----CALCULATOR-----
Donner A : 2048
Donner B : 4
Donner l'opérateur (+,-,*,/) : /
Résultats : 512.0
```

```
class switch(object):
    value = None
    def __new__(class_, value):
        class_.value = value
        return True

def case(*args):
    return any((arg == switch.value for arg in args))

print("CALCULATOR".center(20, '-'))
a=int(input("Donner A : "))
b=int(input("Donner B : "))
op=input("Donner l'opérateur (+,-,*,/) : ")
print("\nRésultats : ",end='\t')
while switch(op):
    if case('+'):
        print(a+b)
        break
    if case('-'):
        print(a-b)
        break
    if case('*'):
        print(a*b)
        break
    if case('/'):
        print(a/b)
        break
    print('Erreur! Opérateur inconnu !!!!')
    break
```

POO sous Python : L'héritage

- ✓ L'héritage est le mécanisme qui permet de se servir d'une classe préexistante pour en créer une nouvelle qui possédera des fonctionnalités supplémentaires ou différentes. La syntaxe pour définir une classe dérive est la suivante :

Class Nomclassédérivée (*NomClassedeBase*):

<instruction1>

...

<instructionN>

- ✓ À la place d'un nom de classe de base, une expression est acceptée. Ce qui est utile lorsque la définition de la classe de base se trouve dans un autre module. **class NomClasseDerivee(*nommod.NomClasseDeBase*):**

- ✓ L'exécution d'une définition de classe dérivée se déroule comme une classe de base.
- ✓ Quand l'objet classe est construit, la classe de base est mémorisée.
- ✓ C'est ce qui permet de trouver facilement un attribut. Lorsqu'un attribut est référencé, il est recherché dans la classe dérivée s'il ne s'y trouve pas on le recherchera dans la classe de base.

POO sous Python : L'héritage

```
class Vehicule:

    def __init__(self):
        self._vitesse = 0

    @property
    def vitesse(self):
        return self._vitesse

    def accelerer(self, delta_vitesse):
        self._vitesse += delta_vitesse

    def decelerer(self, delta_vitesse):
        self._vitesse -= delta_vitesse

class Voiture(Vehicule):

    def klaxonner(self):
        print("tût tût !")
```

POO sous Python : Le polymorphisme

- ✓ Le polymorphisme est un mécanisme important dans la programmation objet. Il permet de modifier le comportement d'une classe fille par rapport à sa classe mère.
- ✓ Le polymorphisme permet d'utiliser l'héritage comme un mécanisme d'extension en adaptant le comportement des objets.

```
class Master:
    def ModuleWEB(self):
        print("Développement Web")

class M2I(Master):
    def ModuleWEB(self):
        print("Web Sémantique")

class MQL(Master):
    def ModuleWEB(self):
        print("PHP OO")
```

```
>>> E = Master()
>>> E.ModuleWEB()
Développement Web
>>> E = M2I()
>>> E.ModuleWEB()
Web Sémantique
>>> E = MQL()
>>> E.ModuleWEB()
PHP OO
```

```
In [2]: runfile('F:/FS/IHM/Exp/Poly.py', wdir='F:/FS/IHM/Exp')
Développement Web
Web Sémantique
PHP OO
```

POO sous Python : L'héritage multiple

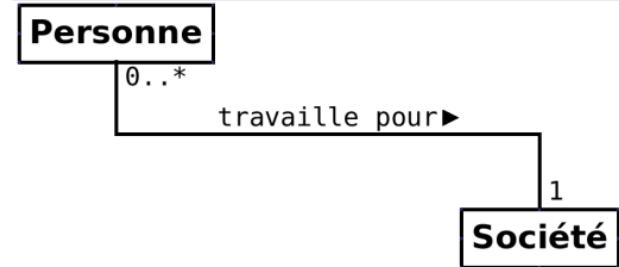
- ✓ Python supporte également l'héritage multiple. Une définition de classe avec plusieurs classes de base ressemble à :

Class Nomclassédérivée (*Base1, Base2, ..., BaseN*):
<instruction1>
...
<instructionN>

- ✓ Lorsqu'on référence un attribut, la recherche s'effectue en profondeur càd de gauche à droite?
- ✓ Donc si un attribut n'est pas trouvé dans *ClasseDerivee*, on le cherchera d'abord dans *Base1*, puis dans *Base2*, et ainsi de suite jusqu'à ce qu'on le trouve...

POO sous Python : Association

- Une association représente un lien unissant les instances de classe.
- Elle repose sur la relation « a-un » ou « utilise-un »



```

class Point :
    def __init__(self, x, y) :
        self.px, self.py = x, y

class Segment :
    """Classe conteneur utilisant la classe Point."""
    def __init__(self, x1, y1, x2, y2) :
        self.orig = Point(x1, y1)
        self.extrem = Point(x2, y2)

    def __str__(self) :
        return ("Segment : [({:g}, {:g}), ({:g}, {:g})]")
            .format(self.orig.px, self.orig.py,
                    self.extrem.px, self.extrem.py))

s = Segment(1.0, 2.0, 3.0, 4.0)
print(s) # Segment : [(1, 2), (3, 4)]
  
```

POO sous Python : Agrégation / Composition

- Une agrégation est une association non symétrique entre deux classes (l'agrégat et le composant).
- Une composition est un type particulier d'agrégation dans laquelle la vie des composants est liée à celle de l'agrégat.

```
adresse = Adresse(rue="Faculté des Science",
code_postal="93030", ville="Tétouan")
```

```
personne = Personne(prenom="Otman", nom="Abdoun",
adresse=adresse)
```

```
class Personne:

    def __init__(self, nom, prenom, adresse):
        self.nom = nom
        self.prenom = prenom
        self.adresse = adresse

class Adresse:

    def __init__(self, rue, code_postal, ville):
        self.rue = rue
        self.code_postal = code_postal
        self.ville = ville
```

Plan

- 1. Introduction aux IHM : Définition et Historique**
- 2. Principes ergonomiques des IHM : Règles et standards**
- 3. Sciences cognitives, ergonomie et guides de style**
- 4. Programmation sur Python**
- 5. Data Management en Python**
- 6. Application Web sur Python**
- 7. IHM challenge : Smart IHM, Soya, SCADA**

Gestion d'une base de données

Stockage en fichier

Sérialisation avec PICKLE et JSON

Stockage avec SQLITE3

Mapping XML/XSL

Gestion d'une base de données

Stockage en fichier

Gestion d'une base de données

Stockage en fichier ".txt"

- ✓ La prise en compte ou l'export d'un nombre important de données nécessite l'utilisation de fichier pour stocker les informations.
- ✓ La commande **open** pour ouvrir un document.
- ✓ La lecture se fait en utilisant l'argument '**r**' (*read*).
- ✓ L'instruction **.readlines()** retourne une liste des différentes lignes du fichier.

```
fichier = open("Doc.txt", 'r')
lignes = fichier.readlines()
fichier.close()

print(lignes)
```

Gestion d'une base de données

Stockage en fichier ".txt" : Lecture

```
fichier = open("M2IQL.txt", 'r')
lignes = fichier.readlines()
fichier.close()

print(lignes)
```

```
In [6]: runfile('F:/FS/IHM/Exp/Fichier.py', wdir='F:/FS/IHM/Exp')
['ABED-ECH-CHAIFI ASMAA \n', 'AHAJTAN KAMAR\n', '\n', 'AKDI MOAAD\n', 'AKHROUF
ISMAIL\n', 'ALLOUANE HATIM\n', 'AMAKRAN HAJAR\n', 'ANNA ZAKARIAE\n', 'AOUTAH
BILAL\n', 'ARAROU HAYAT\n', 'BEN HADDOU MOHAMED\n', 'BOGUHE ANGE STEPHANE
VICTORIEN\n', 'ECHCHERKI MOHAMED\n', 'EL ALLATI YASMINA\n', 'EL BIARI HAMID\n',
'EL BOUDALI NABIL\n', 'EL HAMDOUNI SOHAYA\n', 'EZ-ZAKKAR MOHAMMED\n', 'HICHOU
HALIMA\n', 'HLAL DRISS\n', 'JAMMAL SIHAM\n', "M'HARZI ALAOUI SAMYA\n", 'RECHAD
OUMAIMA\n', 'SOUNA SARA\n', 'OUAARA SARA\n', 'KHIEKIE NIATI DIESMER\n',
'ABDOURAOUF YOUSSEOUF\n']
```

Gestion d'une base de données

Stockage en fichier ".txt" : Écriture

- ✓ Pour écrire dans un fichier, nous utilisons la même commande (**open**) mais nous utilisons l'argument '**w**' (**write**).
- ✓ Attention, lorsque l'on ouvre un fichier de cette manière, on efface toutes les données du fichier.

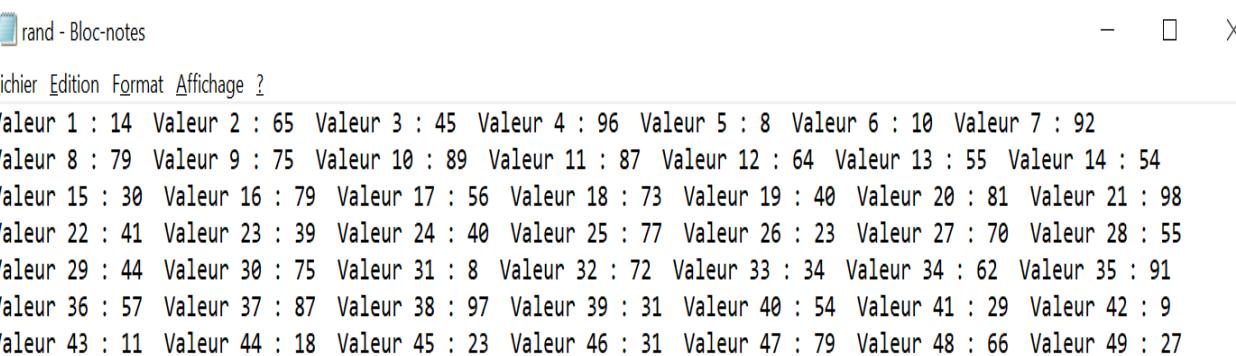
```
from random import randint

fichier = open("rand.txt", 'w')
for i in range(1, 50):
    fichier.write(" Valeur %i : %i "%(i, randint(1, 100)))
fichier.close()

fichier = open("rand.txt", 'r')
lignes = fichier.readlines()
fichier.close()

print(lignes)
```

```
In [18]: runfile('F:/FS/IHM/Exp/Fichier.py', wdir='F:/FS/IHM/Exp')
[' Valeur 1 : 14 Valeur 2 : 65 Valeur 3 : 45 Valeur 4 : 96 Valeur 5 : 8
Valeur 6 : 10 Valeur 7 : 92 Valeur 8 : 79 Valeur 9 : 75 Valeur 10 : 89
Valeur 11 : 87 Valeur 12 : 64 Valeur 13 : 55 Valeur 14 : 54 Valeur 15 : 30
Valeur 16 : 79 Valeur 17 : 56 Valeur 18 : 73 Valeur 19 : 40 Valeur 20 : 81
Valeur 21 : 98 Valeur 22 : 41 Valeur 23 : 39 Valeur 24 : 40 Valeur 25 : 77
Valeur 26 : 23 Valeur 27 : 70 Valeur 28 : 55 Valeur 29 : 44 Valeur 30 : 75
Valeur 31 : 8 Valeur 32 : 72 Valeur 33 : 34 Valeur 34 : 62 Valeur 35 : 91
Valeur 36 : 57 Valeur 37 : 87 Valeur 38 : 97 Valeur 39 : 31 Valeur 40 : 54
Valeur 41 : 29 Valeur 42 : 9 Valeur 43 : 11 Valeur 44 : 18 Valeur 45 : 23
Valeur 46 : 31 Valeur 47 : 79 Valeur 48 : 66 Valeur 49 : 27 ']
```



Gestion d'une base de données

Stockage en fichier ".txt" : Méthodes

Opération	Interprétation
<code>sortie = open('/tmp/spam', 'w')</code>	crée un fichier de sortie ('w' => écriture)
<code>entre = open('donnee', 'r')</code>	ouvre un fichier en entrée ('r' => lecture)
<code>s = entre.read()</code>	lit le fichier entier dans une chaîne
<code>s = entre.read(N)</code>	lit N octets (1 ou plus)
<code>s = entre.readline()</code>	lit la ligne suivante
<code>L = entre.readlines()</code>	lit le fichier dans une liste de lignes
<code>sortie.write(s)</code>	écrit s dans le fichier
<code>sortie.writelines(L)</code>	écrit toutes les lignes contenues pas L
<code>sortie.close()</code>	fermeture manuelle

Gestion d'une base de données

Stockage en fichier ".txt" : **with**

- ✓ Il existe une autre syntaxe plus courte qui permet de s'émanciper du problème de fermeture du fichier: le mot clé **with**.

```
f = open("truc.txt", "w", encoding='utf8')
s = 'MASTERS QL & 2I \n'
f.write(s)
l = [' IHM ', ' ML ', ' WEB ']
f.writelines(l)
f.close()
f2 = open("truc2.txt", "w", encoding='utf8')
print("Les Modules des deux masters", file=f2)
f2.close()

f = open("truc.txt", "r", encoding='utf8')
s = f.read() # lit tout le fichier --> chaîne
print(s)
f.close()
f = open("truc.txt", "r", encoding='utf8')
s = f.readlines() # lit tout le fichier --> liste de chaînes
print(s)
f.close()
```

```
with open("truc.txt", "w", encoding='utf8') as f:
    s = 'MASTERS QL & 2I \n'
    f.write(s)
    l = [' IHM ', ' ML ', ' WEB ']
    f.writelines(l)

with open("truc2.txt", "w", encoding='utf8') as f2:
    f2 = open("truc2.txt", "w", encoding='utf8')
    print("Les Modules des deux masters", file=f2)

with open("truc.txt", "r", encoding='utf8') as f:
    s = f.read()
    print(s)

with open("truc.txt", "r", encoding='utf8') as f:
    s = f.readlines()
    print(s)
```

Gestion d'une base de données

Dialoguer avec Excel en Python :
Openpyxl

Dialoguer avec Excel en Python : OPENPYXL

```
from openpyxl import Workbook
from openpyxl import load_workbook

#ouvrir le fichier excel sample.xlsx (le fichier excel doit
#se trouver dans le même répertoire que le fichier python)
wb = load_workbook(filename = "sample.xlsx")

#charger la feuille excel active dans une variable python
ws = wb.active

#on récupère la valeur de la case B2 puis on l'affiche
a = ws.cell(row=2, column=2).value
print(a)

#on écrit la valeur de la variable b dans la case B1
b = 425.5
ws.cell(row=1, column=2).value = b

#on enregistre le fichier sample.xlsx
wb.save("sample.xlsx")

#ouvrir un nouveau fichier excel
wb2 = Workbook()

#charger la feuille excel active dans une variable python
ws2 = wb2.active

#on écrit la valeur de la variable b dans la case A2
ws2.cell(row=1, column=2).value = b

#on enregistre le nouveau fichier excel en lui donnant le nom sample2.xlsx
wb2.save("sample2.xlsx")
```

Gestion d'une base de données

Sérialisation avec PICKLE / JSON

Gestion d'une base de données

Le module Pickle

- ✓ Le module pickle permet de sauvegarder dans un fichier, au format binaire, n'importe quel objet Python.
- ✓ C'est un module étonnant qui peut prendre presque n'importe quel objet Python, et le convertir en une représentation sous forme de chaîne de caractères; ce processus s'appelle **pickling**.
- ✓ Si vous avez un objet x, et un objet fichier f ouvert en écriture, la voie la plus simple de "pickler" l'objet prend seulement une ligne de code:
pickle.dump(x, f)

```
>>> import pickle
>>> import string
>>> L = list(string.ascii_letters)
>>> print(L)
['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm', 'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v', 'w',
 'x', 'y', 'z', 'A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K', 'L', 'M', 'N', 'O', 'P', 'Q', 'R', 'S',
 'T', 'U', 'V', 'W', 'X', 'Y', 'Z']
>>> with open('mypicklefile', 'wb') as f1:
    pickle.dump(L, f1)
>>> with open('mypicklefile', 'r') as f1:
    f1.read()
```

. ABDOUN

€

\x03]q\x00(X\x01\x00\x00\x00aq\x01X\x01\x00\x00bq\x02X\x01\x00\x00\x00cq\x03X\x01\x00\x00\x00dq\x04X\x01\x00\x00\x00eq\x05X\x01\x00\x00\x00fq\x06X\x01\x00\x00\x00gq\x07X\x01\x00\x00\x00hq\x08X\x01\x00\x00\x00iq\x09X\x01\x00\x00\x00jq\x0AX\x01\x00\x00\x00kq\x0bX\x01\x00\x00\x00lq\x0cX\x01\x00\x00\x00mq\x0NX\x01\x00\x00\x00nq\x0eX\x01\x00\x00\x00oq\x0fX\x01\x00\x00\x00pq\x10X\x01\x00\x00\x00qq\x11X\x01\x00\x00\x00rq\x12X\x01\x00\x00\x00sq\x13X\x01\x00\x00\x00tq\x14X\x01\x00\x00\x00uq\x15X\x01\x00\x00\x00vq\x16X\x01\x00\x00\x00wq\x17X\x01\x00\x00\x00xq\x18X\x01\x00\x00\x00yq\x19X\x01\x00\x00\x00zq\x1aX\x01\x00\x00\x00Aq\x1bX\x01\x00\x00\x00Bq\x1cX\x01\x00\x00\x00Cq\x1dX\x01\x00\x00\x00Dq\x1eX\x01\x00\x00\x00Eq\x1fX\x01\x00\x00\x00Fq

X\x01\x00\x00\x00Gq!X\x01\x00\x00\x00Hq"X\x01\x00\x00\x00Iq#X\x01\x00\x00\x00Jq\$X\x01\x00\x00\x00Kq%X\x01\x00\x00\x00Lq&X\x01\x00\x00\x00Mq'\X\x01\x00\x00\x00Nq(X\x01\x00\x00\x00Oq)X\x01\x00\x00\x00Pq*X\x01\x00\x00\x00Qq+X\x01\x00\x00\x00Rq,X\x01\x00\x00\x00Sq-X\x01\x00\x00\x00Tq.X\x01\x00\x00\x00Uq/X\x01\x00\x00\x00Vq0X\x01\x00\x00\x00Wq1X\x01\x00\x00\x00Xq2X\x01\x00\x00\x00Yq3X\x01\x00\x00\x00Zq4e.'

Gestion d'une base de données

Le module Pickle : Unpickling

- ✓ Reconstruire l'objet à partir de sa représentation en chaîne de caractères s'appelle **unpickling**.
- ✓ Pour “**unpickler**” l'objet, si f est un objet fichier ouvert en lecture: **x = pickle.load(f)**
- ✓ Entre **pickling** et **unpickling**, la chaîne de caractères représentant l'objet a pu avoir été enregistrée dans un fichier, ou avoir été envoyée à une machine éloignée via une connexion réseau.
- ✓ Ça permet de stocker et de restaurer un objet Python tel quel sans aucune manipulation supplémentaire.

```
>>> OL = None
>>> print(OL)
None
>>> with open('mypicklefile', 'rb') as f1:
    OL = pickle.load(f1)
>>> type(OL)
<class 'list'>
>>> print(OL)
['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm', 'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v',
 'w', 'x', 'y', 'z', 'A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K', 'L', 'M', 'N', 'O', 'P', 'Q',
 'R', 'S', 'T', 'U', 'V', 'W', 'X', 'Y', 'Z']
>>> L == OL
True
```

Gestion d'une base de données

Le module Pickle

```
import pickle

data1 = {'a': [1, 2.0, 3, 4+6j],
         'b': ('string', u'Unicode string'),
         'c': None}

selfref_list = [1, 2, 3]
selfref_list.append(selfref_list)

output = open('data.pkl', 'wb')

pickle.dump(data1, output)
pickle.dump(selfref_list, output, -1)
output.close()
```

```
import pprint, pickle

pkl_file = open('data.pkl', 'rb')

data1 = pickle.load(pkl_file)
pprint.pprint(data1)

pkl_file.close()
```

```
In [9]: runfile('F:/FS/IHM/Exp/Umpickl.py', wdir='F:/FS/IHM/Exp')
{'a': [1, 2.0, 3, (4+6j)], 'b': ('string', 'Unicode string'), 'c': None}
```

Gestion d'une base de données

Stockage avec SQLITE3



Stockage avec SQLITE3 : SQLite ?

- ✓ Parmi les SGBD, *SQLite* qui utilise un sous-ensemble de SQL.
- ✓ Sa légèreté, le fait que les données se trouvent sur le terminal du client et non sur un serveur distant, en font un outil apprécié pour des applications personnelles ou encore dans l'embarqué.
- ✓ SQLite fait partie de la famille des SGBD dits « Relationnelles », car les données sont alors placées dans des tables et traitées comme des ensembles.
- ✓ En Python, le module sqlite3 permet de travailler avec ce moteur :

```
import sqlite3
```



Stockage avec SQLITE3: Connecter / Déconnecter

- ✓ Se **connecter** à une BDD en utilisant la méthode ***connect*** et en lui passant l'emplacement du fichier de stockage en paramètre. Si ce dernier n'existe pas, il est alors créé :

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

- ✓ Un objet retourné par la fonction, c'est de type **Connection** et permet de manipuler la base de données.
- ✓ Pour se **déconnecter**, il faut appeler la méthode ***close*** de notre objet Connection:

```
connexion.close()
```

Stockage avec SQLITE3 : Exécuter des requêtes

- ✓ Pour exécuter une requête, il faut passer par l'objet **Cursor**, récupéré en faisant appel à la méthode **cursor** de notre objet de type **Connection** :

curseur = connexion.cursor()

- ✓ Lorsque nous effectuons des modifications sur une table (insertion, modification ou encore suppression d'éléments), celles-ci ne sont pas automatiquement validées.
- ✓ Ainsi, sans validation, les modifications ne sont pas effectuées dans la base et ne sont donc pas visibles par les autres connexions.
- ✓ Pour cela, il nous faut donc utiliser la méthode **commit** de notre objet de type **Connection**.:

connexion.commit()

- ✓ En outre, si nous effectuons des modifications puis nous souhaitons finalement revenir à l'état du dernier commit, il suffit de faire appel à la méthode **rollback**, toujours de notre objet de type **Connection**:

connexion.rollback()

Stockage avec SQLITE3 : Exécuter des requêtes

- ✓ Pour exécuter une requête, il suffit de passer celle-ci à la méthode **execute**:

```
import sqlite3

connexion = sqlite3.connect("dbM2IQL.db")
curseur = connexion.cursor()

curseur.execute(''CREATE TABLE IF NOT EXISTS Etudiant(
    id INTEGER PRIMARY KEY AUTOINCREMENT UNIQUE,
    Nom TEXT,
    Master TEXT
)'''')
```

```
sqlite> .tables
Etudiant
sqlite> .schema Etudiant
CREATE TABLE Etudiant(
    id INTEGER PRIMARY KEY AUTOINCREMENT UNIQUE,
    Nom TEXT,
```

Stockage avec SQLITE3 : Exécuter des requêtes

- ✓ Pour exécuter plusieurs requêtes, comme pour ajouter des éléments à une table par exemple, nous pouvons faire appel plusieurs fois à la méthode **execute**:

```
donnees = [("El Fahssi", "MQL"), ("Ktiti Mohamed", "MQL"), ("Amakran Hajar", "M2I")]
#Exécutions multiples
for donnee in donnees:
    curseur.execute(''INSERT INTO Etudiant (Nom, Master) VALUES (?, ?)'', donnee)
connexion.commit()
```

```
sqlite> select * from Etudiant
...> ;
1|El Fahssi|MQL
2|Ktiti Mohamed|MQL
3|Amakran Hajar|M2I
sqlite>
```

Stockage avec SQLITE3 : Exécuter un script

```
import sqlite3

connexion = sqlite3.connect("dbM2IQL.db")
curseur = connexion.cursor()

curseur.execute(''DROP TABLE Etudiant''')

curseur.execute(''CREATE TABLE IF NOT EXISTS Etudiant(
    id INTEGER PRIMARY KEY AUTOINCREMENT UNIQUE,
    Nom TEXT,
    Master TEXT
)''')

curseur.execute("""DELETE FROM Etudiant""")

donnees = [("El Fahssi", "MQL"), ("Ktiti Mohamed", "MQL"), ("Amakran Hajar", "M2I")]
#Exécutions multiples
for donnee in donnees:
    curseur.execute(''INSERT INTO Etudiant (Nom, Master) VALUES (?, ?)'', donnee)
connexion.commit()

donnee=("M2I", )
curseur.execute("SELECT * FROM Etudiant WHERE Master = ?", donnee)
result = curseur.fetchone()
while result:
    print(result)
    result = curseur.fetchone()
```

```
sqlite> select * from Etudiant
...> ;
1|El Fahssi|MQL
2|Ktiti Mohamed|MQL
3|Amakran Hajar|M2I
sqlite>
```

```
In [48]: runfile('C:/Users/User/Desktop/db/Sqlite.py',
wdir='C:/Users/User/Desktop/db')
(3, 'Amakran Hajar', 'M2I')
```

Stockage avec SQLITE3 : Exécuter un script

```

print("\n M2 - Liste des étudiants : MQL et M2I - \n")
curseur.execute("SELECT * FROM Etudiant")
resultats = curseur.fetchall()
for resultat in resultats:
    print(resultat)

if curseur.arraysize == 1 :
    print("\nREQUETTE OK  ++++ ")
else:
    print("\nREQUETTE KO  ---- ")

```

```

M2 - Liste des étudiants : MQL et M2I -
(1, 'El Fahssi', 'MQL')
(2, 'Ktiti Mohamed', 'MQL')
(3, 'Amakran Hajar', 'M2I')

REQUETTE OK  ++++

```

- ✓ Le nombre de résultats prend par défaut la valeur de l'attribut **arraysize** du curseur.
- ✓ S'il n'y a pas de résultat, la liste renvoyée est vide

Stockage avec SQLITE3: En transaction ou pas

- ✓ Pour savoir si des modifications ont été apportées sans être validées, il suffit de récupérer la valeur de l'attribut **in_transaction** de l'objet de type **Connection**.
- ✓ En effet, celui-ci vaut **True** si c'est le cas et **False** sinon:

```
#modifications...
print(connexion.in_transaction) #Affiche "True"
connexion.commit()
print(connexion.in_transaction) #Affiche "False"
```

Stockage avec SQLITE3: Total des modifs

- ✓ Ensuite, pour être au courant du nombre de modifications (ajouts, mises à jour ou suppressions) apportées depuis notre connexion à la base, il suffit de récupérer la valeur de l'attribut **total_changes** de notre objet de type Connection.

```
print(connexion.total_changes)
```

Stockage avec SQLITE3: Utiliser des clefs étrangères

- ✓ Dès que le nombre de tables augmente, il est souvent primordial de les lier à l'aide de clefs étrangères.
- ✓ Avec sqlite3, les *clefs étrangères ne sont pas activées de base*. Il faut donc y remédier avec la requête adéquate

```
curseur.execute("PRAGMA foreign_keys = ON") #Active les clés étrangères
```

- ✓ Puis il faut relier les tableaux, relier la table Etudiant avec CF

```

import sqlite3
connexion = sqlite3.connect("dbM2IQL.db")
curseur = connexion.cursor()
curseur.execute("PRAGMA foreign_keys = ON")
curseur.executescript("""
    CREATE TABLE IF NOT EXISTS Etudiant(
        id INTEGER PRIMARY KEY,
        Nom TEXT,
        Master TEXT);

    CREATE TABLE IF NOT EXISTS CF(
        id_cf INTEGER PRIMARY KEY ,
        fk_etudiant INTEGER NOT NULL,
        note INTEGER,
        FOREIGN KEY(fk_etudiant) REFERENCES Etudiant(id)
        ON DELETE CASCADE);
        DELETE FROM Etudiant;
""")
donnees_Etudiant = [(1, "El Fahssi", "MQL"), (2, "Ktiti Mohamed", "MQL"),
                     (3, "Amakran Hajar", "M2I")]
donnees_CF = [(1, 1, 10),(2, 2, 17), (3, 3, 15)]
curseur.executemany("INSERT INTO Etudiant (id, Nom, Master) VALUES (?, ?, ?)", donnees_Etudiant)
curseur.executemany("INSERT INTO CF (id_cf,fk_etudiant, note) VALUES (?, ?, ?)", donnees_CF)
connexion.commit()

for etudiant in curseur.execute("SELECT * FROM Etudiant"):
    print("Etudiants :", etudiant)
for notecf in curseur.execute("SELECT * FROM CF"):
    print("Notes CF :", notecf)
connexion.close()
    
```

```

Etudiants : (1, 'El Fahssi', 'MQL')
Etudiants : (2, 'Ktiti Mohamed', 'MQL')
Etudiants : (3, 'Amakran Hajar', 'M2I')
Notes CF : (1, 1, 10)
Notes CF : (2, 2, 17)
Notes CF : (3, 3, 15)
    
```

Stockage avec SQLITE3 : Requêtes Jointure

- ✓ Récupérer la meilleure note:

```
sqlite> SELECT e.Nom, c.note FROM Etudiant as e INNER JOIN
...>      CF as c ON e.id = c.fk_etudiant
...> ;
El Fahssi|10
Ktiti Mohamed|17
Amakran Hajar|15
sqlite>
```

```
import sqlite3
connexion = sqlite3.connect("dbM2IQL.db")
curseur = connexion.cursor()

curseur.execute("""SELECT e.Nom, c.note FROM Etudiant as e INNER JOIN
CF as c ON e.id = c.fk_etudiant
ORDER BY c.note DESC LIMIT 1""")
print(curseur.fetchone())
```

```
In [72]: runfile('C:/Users/User/Desktop/db/NoteMeilleur.py',
kdir='C:/Users/User/Desktop/db')
('Ktiti Mohamed', 17)
```



About

Download

Blog

Docs

GitHub

Gitter

Stats

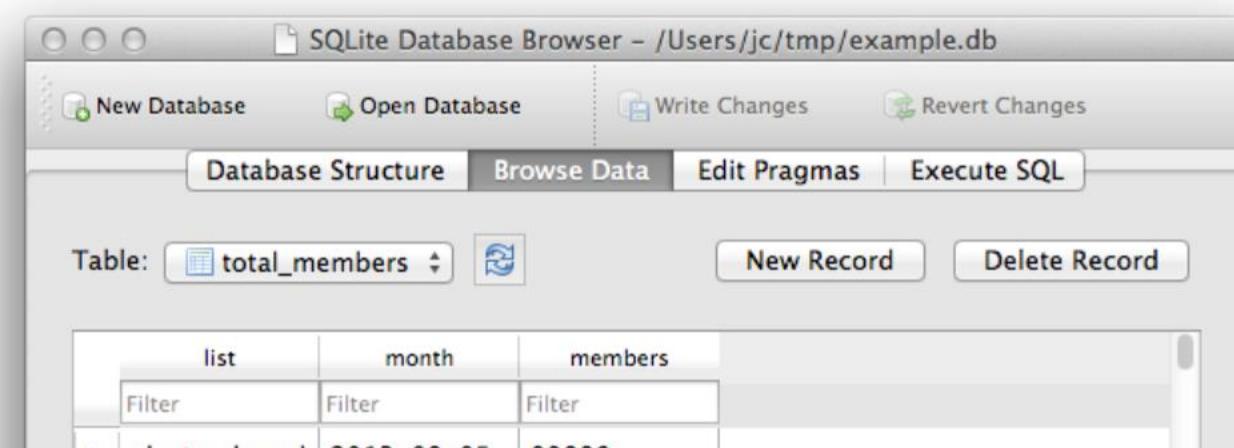
Patreon

DBHub.io

DB Browser for SQLite

The Official home of the DB Browser for SQLite

Screenshot



Manipulation de la base de données: Tool - DB Browser for SQLite

The screenshot shows the DB Browser for SQLite application interface. The title bar indicates the database is located at F:\FS\Mes Cours\IHM\Exp\db\dbM2IQL.db. The menu bar includes Fichier, Édition, Vue, Outils, and Aide. The toolbar features icons for Nouvelle Base de Données, Ouvrir une Base de Données, and Enregistrer les modifications. Below the toolbar, tabs for Structure de la Base de Données, Parcourir les données, Éditer les Pragmas, and Exécuter le SQL are visible, with 'Parcourir les données' selected. A dropdown menu labeled 'Table : CF' is open, showing three filter options: Filtre, Filtre, and Filtre. The main area displays a table with the following data:

	id_cf	fk_etudiant	note
1	1	1	10
2	2	2	17
3	3	3	15

Une base de données **XML** est un format de fichier qui permet de stocker quasiment n'importe quel type d'information de façon **structurée** et **hiérarchisée**.

Gestion d'une base de données

Mapping XML/XSL

Mapping XML/XSL: About XML & XSL !?

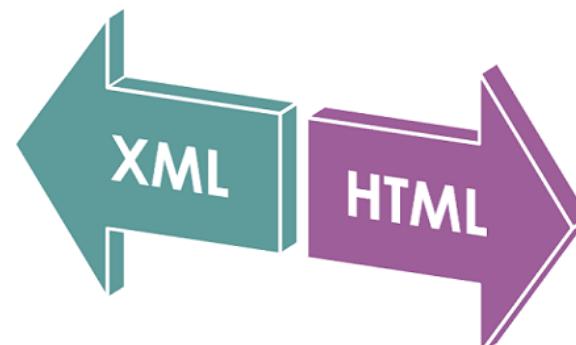
- ✓ XML est l'abréviation de Extensible Markup Language est donc un langage de balises comme le HTML;
- ✓ XML n'est pas un langage de programmation ni de MAPPING;
- ✓ XML est simplement une méthode pour représenter les données. Celles-ci sont écrites entre des balises ou sous forme d'attributs, et l'ensemble est écrit sous forme d'un ARBRE. Il ne fait rien d'autre !
- ✓ Le XML n'est que de l'information encodée entre des balises. Il faudra d'autres éléments, comme par exemple un fichier XSL, pour que le navigateur puisse Comprendre vos balises et Afficher ce fichier sous une forme plus conviviale.

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<Etudiants>
  - <personne>
    - <nom_complet>
      <prenom>Ahmed</prenom>
      <nom>Sadik</nom>
    </nom_complet>
    <email>sadik@gmail.com</email>
  - <adresse>
    <rue>Rue DRADEB</rue>
    <ville>Tanger</ville>
    <pays>Maroc</pays>
  </adresse>
</personne>
- <personne>
  - <nom_complet>
    <prenom>Samira</prenom>
    <nom>Souni</nom>
  </nom_complet>
  <email>samira@hotmail.com</email>
- <adresse>
  <rue>Rue NAHDA</rue>
  <ville>Larache</ville>
  <pays>Maroc</pays>
</adresse>
</personne>
</Etudiants>
```

Mapping XML/XSL: XML # HTML

- ✓ Le XML et le HTML partagent des caractéristiques communes:
 - ✓ Ils fonctionnent avec des Balises.
 - ✓ Ils sont indépendants de la plateforme.
 - ✓ Ils sont en mode texte [plain text].

- ✓ Le HTML et le XML sont différents en de très nombreux points:
 - ✓ Le XML décrit, structure, échange des données tandis que le HTML ne fait qu'afficher des données.
 - ✓ Le XML est extensible et permet de créer ses propres balises en fonction des données traitées. En HTML, les balises sont prédéfinies et donc figées.



Mapping XML/XSL: Entête et encodage

- ✓ Les documents XML doivent posséder une première ligne qui est de la forme :

```
<?xml version="1.0" encoding="ISO-8859-1"?>
```

Cette ligne définit le numéro de version (ici 1.0), et l'encodage des caractères.

- ✓ Pour l'encodage:
 - ✓ **ISO-8859-1** : C'est le plus simple pour écrire un document en Français car les lettres accentuées ne prennent qu'un caractère (octet).
 - ✓ **UTF-8** : Format international qui permet d'écrire dans n'importe quelle langue.

Mapping XML/XSL:

Entête et encodage : Exemple

- ✓ Premier Exemple:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<ExempleXML>
    <message> Exemple de contenu XML (é, è, à, á, ...) </message>
</ExempleXML>
```

Mapping XML/XSL: La syntaxe du XML

- ✓ Le XML est un langage de balises [Markup Language].
- ✓ Les balises sont sensibles au majuscules et minuscules [case sensitive].
- ✓ Toute balise ouverte doit impérativement être fermée.
- ✓ Les balises doivent être correctement imbriquées.
- ✓ Commentaires : <!-- -->
- ✓ Tout document XML doit comporter une racine

```
<parents>
    <enfants>
        <petits_enfants> ... </petits_enfants>
    </enfants>
</parents>
```

Mapping XML/XSL: Exemple XML 1

```
<?xml version="1.0" encoding="UTF-8"?>
<racine nom="FS-Tétouan">
    <masters>
        <master1>M2I</master1 >
        <master2>MQL</master2 >
    </masters>
</racine>
```

racine "FS-Tétouan"

- masters M2I
- master1 M2I
- master2 MQL

Mapping XML/XSL: Exemple XML 2

✓ Générer un fichier XML, qui permet de structurer les informations étudiants sous forme de l'arbre suivant:

✓ Etudiant i

✓ Nom complet

✓ Nom

✓ Prenom

✓ Adresse

✓ Rue

✓ Ville

✓ Num

✓ Pays

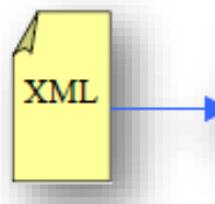
✓ Email

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
- <Etudiants>
  - <personne>
    - <nom_complet>
      <prenom>Ahmed</prenom>
      <nom>Sadik</nom>
    </nom_complet>
    <email>sadik@gmail.com</email>
    - <adresse>
      <rue>Rue DRADEB</rue>
      <ville>Tanger</ville>
      <pays>Maroc</pays>
    </adresse>
  </personne>
  - <personne>
    - <nom_complet>
      <prenom>Samira</prenom>
      <nom>Souni</nom>
    </nom_complet>
    <email>samira@hotmail.com</email>
    - <adresse>
      <rue>Rue NAHDA</rue>
      <ville>Larache</ville>
      <pays>Maroc</pays>
    </adresse>
  </personne>
</Etudiants>
```

Mapping XML/XSL: DTD

- ✓ Le **DTD** ou **Document Type Declaration** ou encore **Document Type Definition** est *l'ensemble des règles et des propriétés que doit suivre le document XML.*
- ✓ Ces règles définissent généralement le **nom** et le **contenu** de chaque **balise** et le contexte dans lequel elles doivent exister.

```
<!ELEMENT Baliese_parent(element1,...)>
<!ELEMENT element1 (#PCDATA)>
<!ELEMENT ..... (#PCDATA)>
```



```
<?xml version="1.0"?>
<!DOCTYPE personne [
    <!ELEMENT personne (#PCDATA)>
]>
<personne> Ahmed Sami </personne>
```

DTD



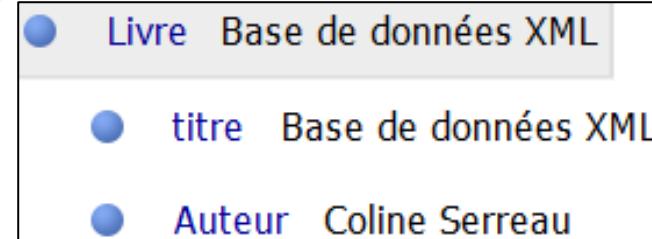
Mapping XML/XSL: DTD

- ✓ L'ajout d'une définition DTD se fait par la création par la clause DOCTYPE et comportant :
 - ✓ Le nom de la racine du document
 - ✓ La DTD proprement dite (= déclaration locale) ou le nom/URI du fichier dans laquelle elle est stockée (= déclaration externe)

Ex. de déclaration locale :

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE Livre [
  <!ELEMENT Livre (titre, Auteur)>
  <!ELEMENT titre (#PCDATA)>
  <!ELEMENT Auteur (#PCDATA)>
]>
<Livre>
  <titre>Base de données XML</titre>
  <Auteur>Coline Serreau</Auteur>
</Livre>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE Livre [
  <!ELEMENT Livre (titre, Auteur)>
  <!ELEMENT titre (#PCDATA)>
  <!ELEMENT Auteur (#PCDATA)>
]>
<Livre>
  <titre>Base de données XML</titre>
  <Auteur>Coline Serreau</Auteur>
</Livre>
```



Ex. de déclaration externe :

```
<!DOCTYPE root SYSTEM "F:\XML\exemples\choice.dtd">
```

Mapping XML/XSL: DTD

✓ Autres usages d'une DTD

- ✓ Donner des valeurs par défaut aux attributs
- ✓ Définir des contraintes d'unicité d'identification – attributs **ID**
- ✓ Définir des contraintes de référence – attributs **IDREF**

✓ Déclaration des types d'éléments par leur contenu :

- ✓ Textuel :

```
<!ELEMENT page (#PCDATA)>
```

Syntaxe

```
<!ELEMENT nom modèleDeContenu >
```

- ✓ Groupe d'éléments :

```
<!ELEMENT adresse (numero, rue, code, ville)>
```

– *nom* : nom de l'élément

– *modèleDeContenu* : expression définissant le contenu autorisé dans l'élément

Mapping XML/XSL: DTD

✓ Connecteurs :

```
<!ELEMENT contact (telephone | mail)>
<!ELEMENT identité (prénom+, surnom?, nom) >
<!ELEMENT expérience (stage|emploi)>
<!ELEMENT diplôme ( (intitulé, année) | 
(année, compétences, stage?) + )>
<!ELEMENT parcours (#PCDATA | diplôme)*>
<!ELEMENT ville EMPTY>
<!ELEMENT About ANY>
```

✓ Déclaration des attributs :

- ✓ Nom de l'élément
- ✓ Nom de l'attribut
- ✓ Type de l'attribut (ID, IDREF, CDATA, etc.)
- ✓ Valeur par défaut / présence facultative ou obligatoire (#REQUIRED, #IMPLIED)

Opérateurs d'expressions régulières

Sémantique	Opérateur
Enchaînement	..., ...
Choix
Zéro ou 1	...?
Zéro ou plus	...*
Un ou plus	...+
Groupe	(...)
un et un seul	nom de l'élément

Opérateur d'expressions régulières

```
<!-- choix parmi deux -->
<!ELEMENT mobile (train | avion)>
<!-- séquence d'éléments -->
<!ELEMENT adresse (numéro, voie, ville)>
<!-- Au moins un élément -->
<!ELEMENT carnetDAdresse (carteDeVisite+)>
<!-- séquence d'éléments dont le dernier peut
apparaître 0 ou plusieurs fois -->
<!ELEMENT carteDeVisite (prénom, nom, organisme,
adresse, tel*)>
```

Mapping XML/XSL: DTD : Exemple 1 (Etudiants)

```
<!ELEMENT etudiant(Nom_complet+, Adresse+, Email)>  
  
<!ELEMENT Nom_cpmplet (Nom, prenom)>  
  
  <!ELEMENT Nom (#PCDATA)>  
  
  <!ELEMENT prenom (#PCDATA)>  
  
<!ELEMENT Email (#PCDATA)>  
  
<!ELEMENT Adresse (Rue, Ville, Pays)>  
  
  <!ELEMENT Rue (#PCDATA)>  
  
  <!ELEMENT Ville (#PCDATA)>  
  
  <!ELEMENT Pays (#PCDATA)>
```

Mapping XML/XSL: DTD : Exemple 2 (Livres)

livres.dtd

```
<!ELEMENT liste_livres (livre+)
<!ELEMENT livre (titre, auteur+, éditeur,
description?, prix)
<!ELEMENT titre (#PCDATA)
<!ELEMENT auteur (#PCDATA)
<!ELEMENT éditeur (#PCDATA)
<!ELEMENT description (#PCDATA)
<!ELEMENT prix (#PCDATA)>
```

```
<?xml version="1.0" encoding="iso-8859-1" ?>
<!DOCTYPE liste_livres SYSTEM "livres.dtd">
<liste_livres>
  <livre>
    <titre>Comprendre XSLT</titre>
    <auteur>Bernd Amann</auteur>
    <auteur>Philippe Rigaux</auteur>
    <éditeur>O'Reilly</éditeur>
    <description>
      Le livre sur le langage XSLT.
    </description>
    <prix>350 Dh</prix>
  </livre>
  <livre>
    <titre>Learning XML</titre>
    <auteur>Erik T. Ray</auteur>
    <éditeur>O'Reilly</éditeur>
    <prix>400 Dh</prix>
  </livre>
</liste_livres>
```

Mapping XML/XSL: DTD : Déclaration d'un Attribut

- Associer une liste d'attributs à un élément
- Schéma de la déclaration

```
<!ATTLIST nomElément
    nomAttribut type Contrainte
    ...
    nomAttribut type Contrainte } Liste de
                                triplets
>
```

Types d'attributs

Description	Type
Texte	CDATA
Type énuméré	(v1 v2 ...)
Définition d'identifiant unique	ID
Référence à identifieur	IDREF
un mot sans espace	NMOKEN

- Exemple

```
<!ATTLIST ville
    nom CDATA #IMPLIED deux
    id ID      #REQUIRED attributs
>
```

Mapping XML/XSL: DTD : Contraintes d'occurrence

Type de contrainte	Expression de la contrainte
Val par défaut d'un type énuméré	'val'
Val par défaut	#DEFAULT val
Obligatoire	#REQUIRED
Non obligatoire	#IMPLIED
Valeur constante	#FIXED val

```

<!ATTLIST voie
      type (rue | boulevard | place) 'rue'
>
<!ATTLIST ville
      codepostal CDATA #IMPLIED
      nom CDATA #IMPLIED
      id ID #REQUIRED
>

```

Mapping XML/XSL: DTD par Attributs : Exemple

```
<!ELEMENT personne( Nom_complet+, Adresse+,>
<!ATTLIST personne
          Email CDATA #REQUIRED>
<!ELEMENT Nom_cpmplet EMPTY >
<!ATTLIST Nom_cpmplet
          Nom CDATA #REQUIRED
          Prenom CDATA #REQUIRED>
<!ELEMENT Adresse EMPTY>
<!ATTLIST Adresse
          Rue CDATA #REQUIRED
          Ville CDATA #REQUIRED
          Pays (Maroc | France) #REQUIRED >
```

Mapping XML/XSL: DTD : Cardinalités

Relations en relationnelle :

- ✓ 1-1 \Leftrightarrow 1-n: Relation imbrication
- ✓ 1-n \Leftrightarrow 1-n : Relation plus complexe
- ✓ **Imbrication**
- ✓ **Référence(ID IDREF)**

Aucun	Si aucune cardinalité n'est précisée, l'élément doit apparaître une et une seule fois.
*	Avec cette cardinalité, l'élément peut ne pas apparaître ou apparaître plusieurs fois
+	Avec cette cardinalité, l'élément doit apparaître soit une fois, soit plusieurs fois.
?	Avec cette cardinalité, l'élément peut apparaître une seule fois ou ne pas apparaître du tout.

Mapping XML/XSL: DTD : Contraintes référentielles

- ✓ Un attribut **ID** doit identifier de manière unique un élément au sein d'un document considéré (contrainte d'unicité)
 - ✓ Contrainte d'unicité (clé primaire en SGBD)
 - ✓ L'élément est identifié de manière unique
- ✓ Un attribut **IDREF** est contraint à prendre la valeur d'un attribut ID existant dans le document (contrainte de référence)
 - ✓ Contrainte d'intégrité (clé étrangère en SGBD)
 - ✓ Référence un élément identifié (ID)

Mapping XML/XSL: DTD par Attributs : Exemple

```
<grossite>
  <fournisseur id="f1">
    <nom>machin</nom>
  </fournisseur>
  <fournisseur id="f2">
    <nom>sansos</nom>
  </fournisseur>

  <produit id="p1">
    <nom>carotte</nom>
  </produit>
  <produit id="p2">
    <nom>pomme</nom>
  </produit>
```

```
<commande id="c1">
  <client>truc</client>
  <ligne qte="2"
    refF="f1"
    refP="p1"/>
  <ligne qte="10"
    refF="f2"
    refP="p2"/>
</commande>
</grossite>
```

```
<!ATTLIST fournisseur
  id ID #REQUIRED
>
<!ATTLIST produit
  id ID #REQUIRED
>
<!ATTLIST ligne
  refF IDREF #REQUIRED
  refP IDREF #REQUIRED
  qte CDATA #REQUIRED
>
```

Mapping XML/XSL: DTD : Relations (Imbrication)

```
<!ELEMENT Facture( Lignedecomande+)>
```

```
<!ATTLIST Facture
```

```
    Nom_clt CDATA #REQUIRED
```

```
    Dat_cmd CDATA #REQUIRED>
```

```
<!ELEMENT Lignedecomande EMPTY >
```

```
<!ATTLIST Lignedecomande
```

```
    Descrp_prdt CDATA #REQUIRED
```

```
    Qte_cmd CDATA #REQUIRED
```

```
    PU_prd CDATA #REQUIRED>
```

Mapping XML/XSL: DTD : Relations (Par Référence)

```
<!ELEMENT Gest_Facture(Facture+, produit+)>
<!ELEMENT Facture( Lignedecomande+)>
<!ATTLIST Facture
      Nom_clt CDATA #REQUIRED
      Dat_cmd CDATA #REQUIRED>
<!ELEMENT Lignedecomande EMPTY >
<!ATTLIST Lignedecomande
      Id_prd IDREF #REQUIRED
      Qte_cmd CDATA #REQUIRED >
<!ELEMENT produit EMPTY >
<!ATTLIST produit
      Id_prd ID #REQUIRED
      Descrp_prdt CDATA #REQUIRED
      PU_prd CDATA #REQUIRED>
```

Mapping XML/XSL: XSL

- ✓ Le XSL est donc le complément indispensable pour l'affichage du XML.
- ✓ Le XSL est en quelque sorte le langage de feuille de style du XML.
- ✓ Le XSL ne permet pas uniquement l'affichage de XML. Il permet aussi :
 - ✓ Sélectionner une partie des éléments XML.
 - ✓ Trier des éléments XML.
 - ✓ Filtrer des éléments XML en fonction de certains critères.
 - ✓ Choisir des éléments.
 - ✓ Retenir des éléments par des tests conditionnels.

Mapping XML/XSL: XSL : Structure

```
<?xml version="1.0"?>
<xsl:stylesheet xmlns:xsl = "http://www.w3.org/TR/WD-xsl">
<xsl:template match="/">
<html>
  <body>
    Diverses balises Html et XSL.....
  </body>
</html>
</xsl:template>
</xsl:stylesheet>
```

On ajoutera donc dans le fichier XML :

```
<?xml-stylesheet type="text/xsl" href="nom_du_fichier_xsl.xsl"?>
```

Mapping XML/XSL: XSL (Balise Afficher)

- ✓ Pour l'affichage de la valeur d'un élément:

```
<xsl:value-of select="Chemin/Nom_sous_element"/>
```

- ✓ Pour afficher tout les données:

```
<xsl:for-each select="Chemin">  
.....  
</xsl:for-each>
```

- ✓ Pour trier l'affichage selon un élément ASC:

```
order-by="+nom_element"
```

- ✓ Pour trier l'affichage selon un élément DESC:

```
order-by="-nom_element"
```

Mapping XML/XSL: XSL (Balise Filtrage)

- ✓ Le langage XSL permet aussi de filtrer les données du fichier XML associé selon des critères comme *égal*, *pas égal*, *plus grand que*, *plus petit que*:

select="chemin_d'accès[balise='xxx']«

- ✓ Les opérateurs possibles sont :
 - = pour égal.
 - != pour différent (non égal).
 - > pour plus grand que.
 - < pour plus petit que.

Mapping XML/XSL: XSL (Balise Choix)

- ✓ La balise qui permet d'effectuer un choix dans les données du fichier XML:

```
<xsl:if match=".#[balise='xxx']">  
    balises Html.....  
</xsl:if>
```

Mapping XML/XSL: XSL (Balise Choix conditionnel)

- ✓ La balise qui permet d'effectuer un choix conditionnel dans les données du fichier XML:

```
<xsl:choose>
```

```
<xsl:when test=".#[balise='valeur']">
```

```
.....
```

```
</xsl:when>
```

```
<xsl:otherwise>
```

```
.....
```

```
</xsl:otherwise>
```

```
</xsl:choose>
```

Mapping XML/XSL:

XSL : Exemple d'illustration

La gestion des commandes

Les Produits:	
Designation	Prix
PC	2500 DH
Clavier	100 DH
Souris	20 DH
Ecran	1000 DH
SCanner	200 DH

Les clients:	
Nom	Prenom
AFKIR	Marouan
ABED	Abdelkader

Les produits commandés par le premier client séparés en fonction de commande:

Designation	Prix unitaire	Quantité commandée
Commande 1:		
PC	2500 Dh	10 unité(s)
Clavier	100 Dh	1 unité(s)
Ecran	1000 Dh	3 unité(s)
Commande 2:		
Souris	20 Dh	1 unité(s)
Clavier	100 Dh	5 unité(s)
Ecran	1000 Dh	13 unité(s)

Mapping XML/XSL: Analyseurs – Parseur XML

- ✓ Parseur XML = Outil logiciel permettant de parcourir le document et d'en extraire les informations

Parseurs non-validants (*well-formed*) et Parseurs validants (*DTD*)

- ✓ Deux approches : **DOM** (approche objet) et **SAX** (approche événement)

- DOM se base sur la structure du document est explicitement représentée en mémoire sous forme d'une hiérarchie d'objets
- Fonctionnement DOM : parcours du document à partir de la racine. Création d'un objet à chaque élément rencontré, avec attachement à l'objet le contenant
- DOM fait un chargement entier du document préalablement à tout traitement.

- SAX parcourt le document et transmet des événements au fur et à mesure qu'il rencontre des éléments tels que :
- Balise d'ouverture d'un élément;
 - Balise de fermeture d'un élément;
 - Contenu textuel d'un élément;
 - Entités;
 - Erreurs d'interprétation

Mapping XML/XSL: Mapping XML by Python

```
<?xml version="1.0" encoding="UTF-8"?>
<users>
  - <user data-id="101">
    <nom>ALALLATI Yassmina</nom>
    <master>Master M2I</master>
  </user>
  - <user data-id="102">
    <nom>AHAJTAN Kama</nom>
    <master>Master M2I</master>
  </user>
  - <user data-id="103">
    <nom>ANNA Zakariae</nom>
    <master>Master MQL</master>
  </user>
  - <user data-id="104">
    <nom>AKDI Moaad</nom>
    <master>Master M2I</master>
  </user>
  - <user data-id="105">
    <nom>AKHROUFI Ismail</nom>
    <master>Master MQL</master>
  </user>
  - <user data-id="106">
    <nom>ELATTAQ Nadia</nom>
    <master>Master MQL</master>
  </user>
</users>
```

- ✓ Pour lire un fichier XML :

```
from lxml import etree
tree = etree.parse("data.xml")
for user in tree.xpath("/users/user/nom"):
    print(user.text)
```

- ✓ Afficher les attributs des balises:

```
from lxml import etree
tree = etree.parse("data.xml")
for user in tree.xpath("/users/user"):
    print(user.get("data-id"))
```

ALALLATI Yassmina
AHAJTAN Kama
ANNA Zakariae
AKDI Moaad
AKHROUFI Ismail
ELATTAQ Nadia

101
102
103
104
105
106

Mapping XML/XSL: Mapping XML by Python

- ✓ Affiner l'affichage en proposant :

```
from lxml import etree

tree = etree.parse("data.xml")
for user in tree.xpath("/users/user[master='Master M2I']/nom"):
    print(user.text)
```

ALALLATI Yassmina
AHAJTAN Kama
AKDI Moaad

Mapping XML/XSL: Créer un XML en Python

```
from lxml import etree

users = etree.Element("etudiants")
user = etree.SubElement(users, "etudiant")
user.set("data-id", "11")
nom = etree.SubElement(user, "nom")
nom.text = "BOUZIKI Abderrahman"
metier = etree.SubElement(user, "master")
metier.text = "Master MQL"
print(etree.tostring(users, pretty_print=True))

tree = etree.ElementTree(users)
tree.write("data1.xml")
```



```
<?xml version="1.0"?>
- <etudiants>
  - <etudiant data-id="11">
    <nom>BOUZIKI Abderrahman</nom>
    <master>Master MQL</master>
  </etudiant>
</etudiants>
```

Mapping XML/XSL: Créer un XML - Liste

```
from lxml import etree

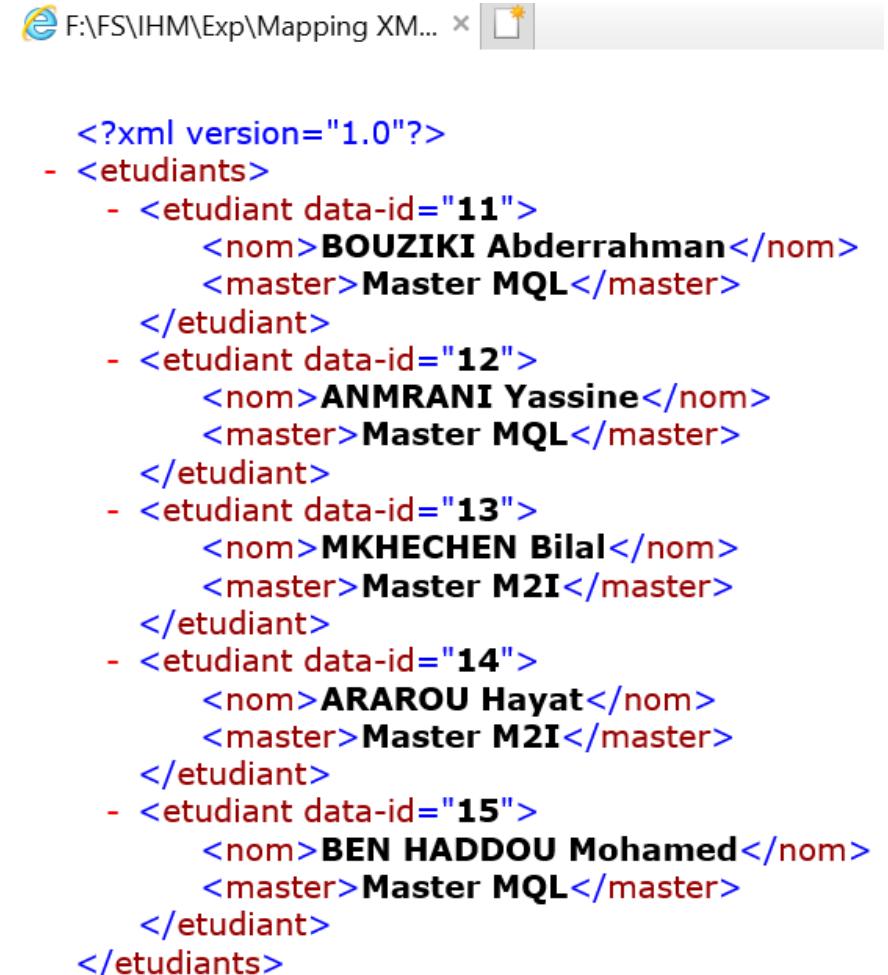
users = etree.Element("etudiants")

users_data = [
    ("11", "BOUZIKI Abderrahman", "Master MQL"),
    ("12", "ANMRANI Yassine", "Master MQL"),
    ("13", "MKHECHEN Bilal", "Master M2I"),
    ("14", "ARAROU Hayat", "Master M2I"),
    ("15", "BEN HADDOU Mohamed", "Master MQL"),
]

for user_data in users_data:
    user = etree.SubElement(users, "etudiant")
    user.set("data-id", user_data[0])
    nom = etree.SubElement(user, "nom")
    nom.text = user_data[1]
    metier = etree.SubElement(user, "master")
    metier.text = user_data[2]

print(etree.tostring(users, pretty_print=True))

tree = etree.ElementTree(users)
tree.write("ListeM.xml")
```



```
<?xml version="1.0"?>
- <etudiants>
  - <etudiant data-id="11">
    <nom>BOUZIKI Abderrahman</nom>
    <master>Master MQL</master>
  </etudiant>
  - <etudiant data-id="12">
    <nom>ANMRANI Yassine</nom>
    <master>Master MQL</master>
  </etudiant>
  - <etudiant data-id="13">
    <nom>MKHECHEN Bilal</nom>
    <master>Master M2I</master>
  </etudiant>
  - <etudiant data-id="14">
    <nom>ARAROU Hayat</nom>
    <master>Master M2I</master>
  </etudiant>
  - <etudiant data-id="15">
    <nom>BEN HADDOU Mohamed</nom>
    <master>Master MQL</master>
  </etudiant>
</etudiants>
```

Mapping XML/XSL: Mapping XML by Python

<code>addnext(element)</code>	: Ajoute un élément en tant que frère juste après l'élément
<code>addprevious(element)</code>	: Ajoute un élément en tant que frère juste avant l'élément
<code>append(element)</code>	: Ajoute un sous-élément à la fin de l'élément
<code>clear()</code>	: Supprime tous les sous-éléments
<code>extends(elements)</code>	: Etend les éléments passé en paramètre
<code>find(path)</code>	: Recherche le premier sous-élément qui correspond au tag/path
<code>findall(path)</code>	: Recherche tous les sous-éléments qui correspondent au tag/path
<code>findtext(path)</code>	: Trouve le texte du premier sous-élément qui correspond au tag/path
<code>get(key)</code>	: Recupère l'attribut d'un élément
<code>getchildren()</code>	: Retourne tous les enfants d'un élément ! attention déprécié pour <i>List(element)</i>
<code>getnext()</code>	: Retourne le prochain élément frère
<code>getparent()</code>	: Retourne le parent de l'élément
<code>getprevious()</code>	: Retourne l'élément frère précédent
<code>index(child)</code>	: Trouve la position de l'élément
<code>insert(index)</code>	: Insère un sous-élément à la position indiquée
<code>items()</code>	: Retourne les attributs d'un élément (dans un ordre aléatoire)
<code>keys()</code>	: Retourne une liste des noms des attributs
<code>remove(element)</code>	: Supprime l'élément passé en paramètre
<code>replace(el1, el2)</code>	: Remplace el1 par el2
<code>set(key, value)</code>	: Crée un attribut avec une valeur
<code>values()</code>	: Retourne les valeurs des attributs
<code>xpath(path)</code>	: Evaluate une expression xpath

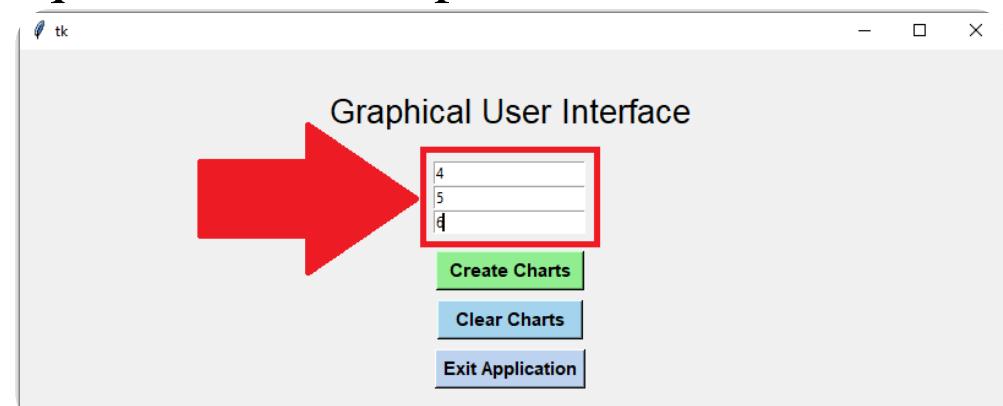
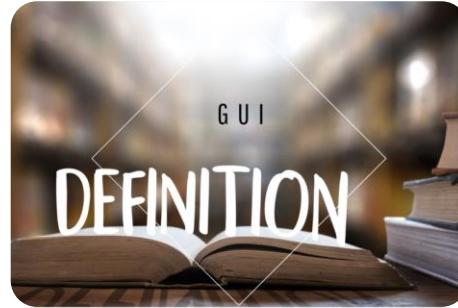
Interface graphique pour Python

- GUI : INTERFACE GRAPHIQUE

Interface graphique pour Python

GUI : Interface graphique

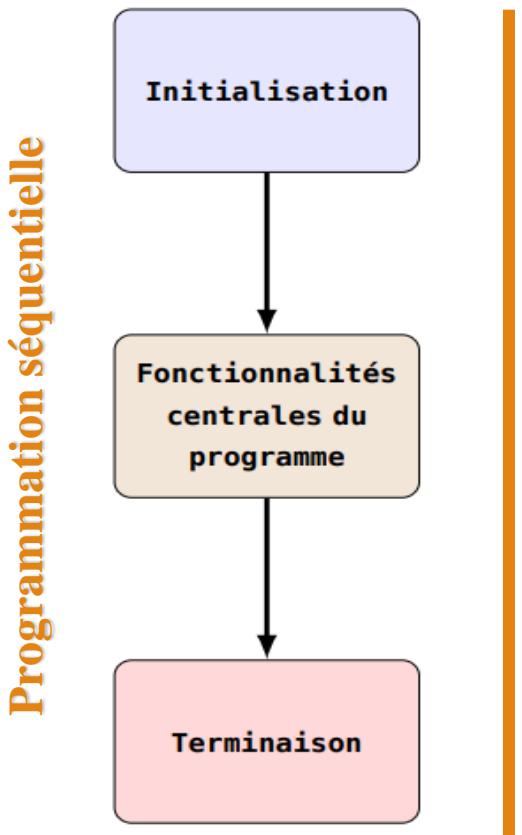
- ✓ Une interface graphique est un outil permettant d'interagir avec un programme informatique.
- ✓ C'est un objet très complexe qui est à l'interface de nombreux aspects .
- ✓ La plupart des langages disposent de bibliothèques pour prendre en charge une partie de cette complexité.
- ✓ Le programmeur doit simplement penser aux éléments graphiques à mettre en place (cadre, menu, boutons).
- ✓ Il doit également penser au fond de son programme c'est-à-dire les traitements qu'il doit faire.



Interface graphique pour Python

GUI : Interface graphique

- ✓ Parmi les styles de programmation :

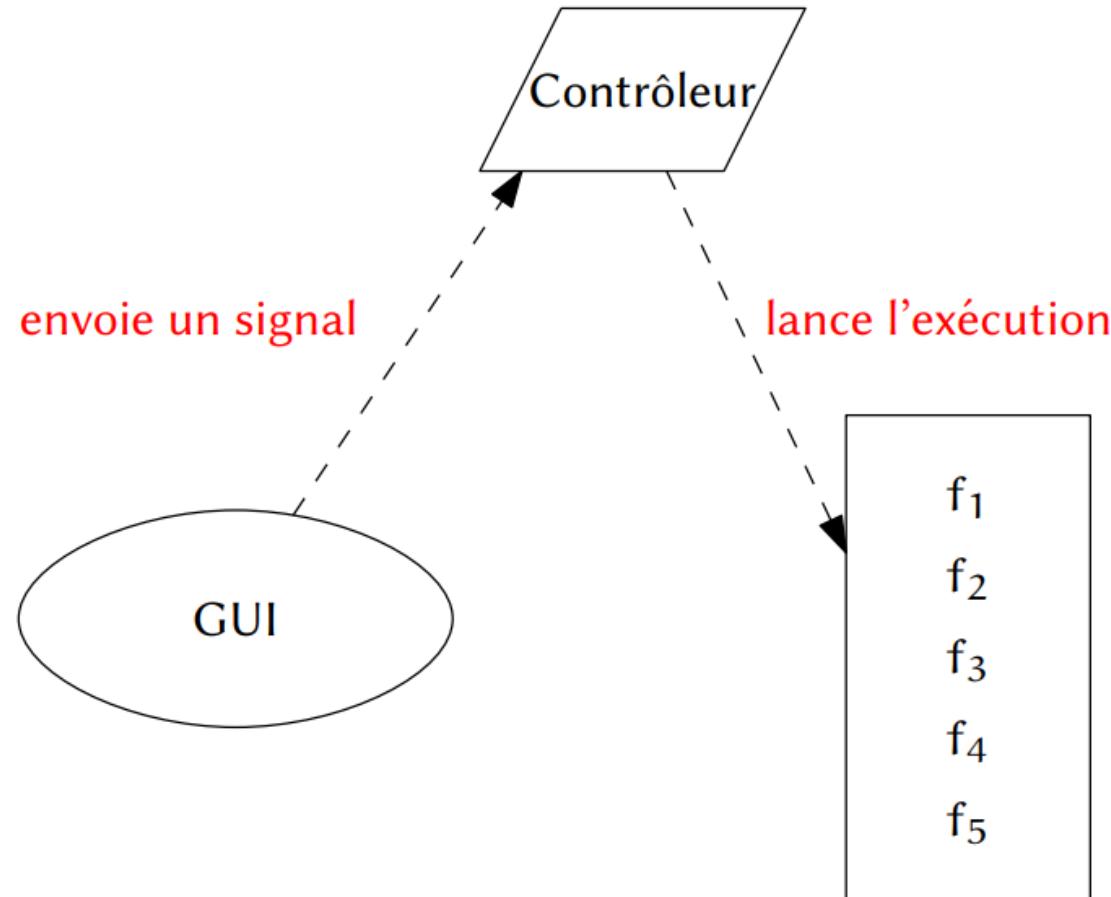


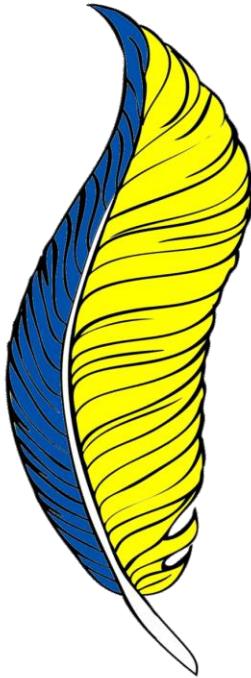
Programmation pilotée par
une boucle d'évènements

Interface graphique pour Python

GUI : Interface graphique

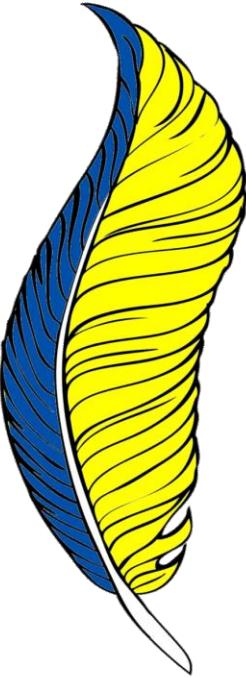
- ✓ La Programmation par événements se base sur :
 - ✓ Envoi de signaux
 - ✓ Un contrôleur s'occupe d'envoyer les signaux au bon destinataire
- ✓ L'architecture Modèle - Vue – Contrôleur s'articule sur :
 - ✓ *Signaux* : Ils sont déclenchés par des instances extérieures au programme (clique de l'utilisateur, branchement d'un périphérique, . . .).
 - ✓ *Vue* : Compose l'interface utilisateur, ainsi les Différents éléments d'interaction (bouton, champs de texte, labels,...)





Interface graphique pour Python

- TKINTER POUR TK
- PYQT POUR QT
- WXPYTHON POUR WXWIDGETS
- PYGTK POUR GTK+

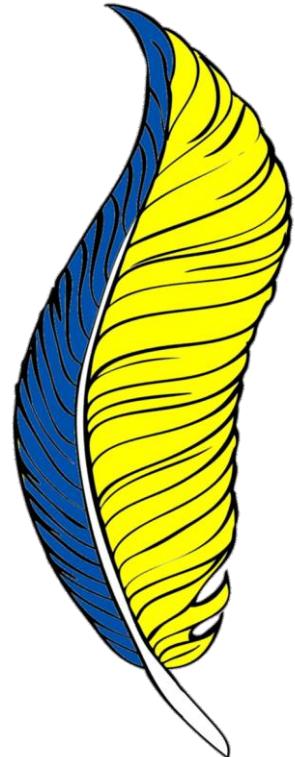


Interface graphique pour Python

TKINTER pour Tk

Interface graphique pour Python TKinter :

- ✓ Tkinter est un module intégré dans Python
- ✓ C'est un module composé de plusieurs éléments graphiques produisant une représentation graphique
- ✓ Parmi les avantages de Tkinter est sa portabilité sur les OS



Interface graphique pour Python

TKinter : Création d'une fenêtre

- ✓ Importation du module tkinter :

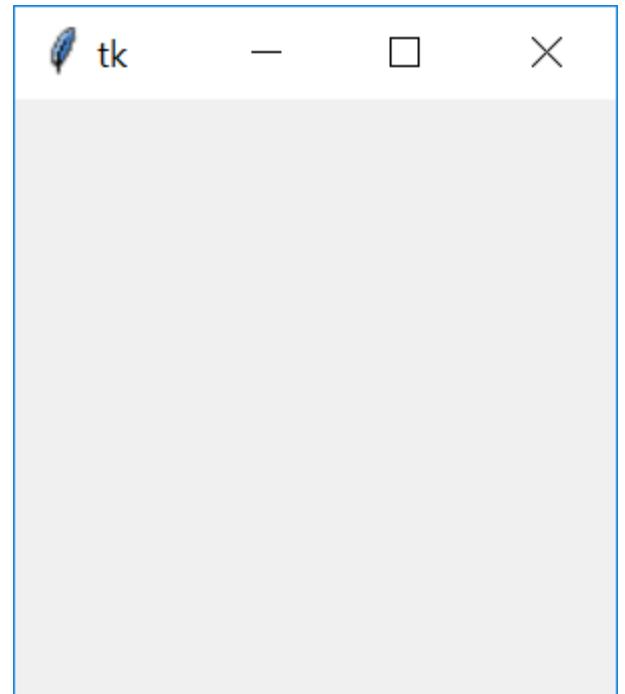
```
from tkinter import *           import tkinter
```

- ✓ Pour la création et l'affichage de la fenêtre principale, on utilise la classe **Tk** importée du module **tkinter**, on crée un objet qui va représenter la fenêtre principale de notre application:

```
from tkinter import *           import tkinter
fenetre = Tk()                fenetre = tkinter.Tk()
```

- ✓ Pour l'affichage de la fenêtre créée, on utilise la méthode **mainloop()** tel que :

```
fenetre.mainloop()
```



Interface graphique pour Python TKinter : Personnaliser la fenêtre

- ✓ Pour ajouter un titre à la fenêtre principale, on utilise la méthode `title()`.

```
fenetre.title("Ma plateforme")
```

- ✓ Afin de définir un logo personnalisé à la fenêtre principale, on utilise la méthode `iconbitmap()`.

- ✓ Pour configurer un aspect tel que la couleur de l'arrière-plan, on utilise la méthode `config()`, contenant comme argument le paramètre `bg` :

- ✓ `bg = 'blue'` ou `bg = 'green'`
- ✓ `bg = '#808000'` pour la couleur jaune,

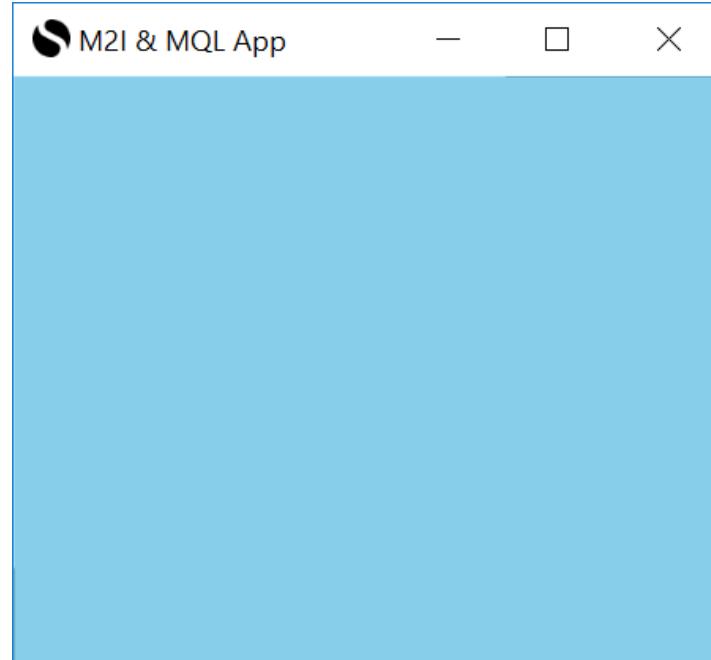
- ✓ Pour personnaliser la taille d'affichage par défaut de la fenêtre principale, on utilise la méthode `geometry()`, en ayant comme attribut la largeur x la hauteur en pixels.

```
fenetre.geometry("640x480")
```

- ✓ Il y a la possibilité de limiter les dimensions d'affichage de la fenêtre principale :

```
fenetre.maxsize(700,500)
```

```
fenetre.minsize(200,200)
```



Interface graphique pour Python TKinter : Exemple fenêtre

```
from tkinter import *

fenetre = Tk()

fenetre.title("M2I & MQL App")

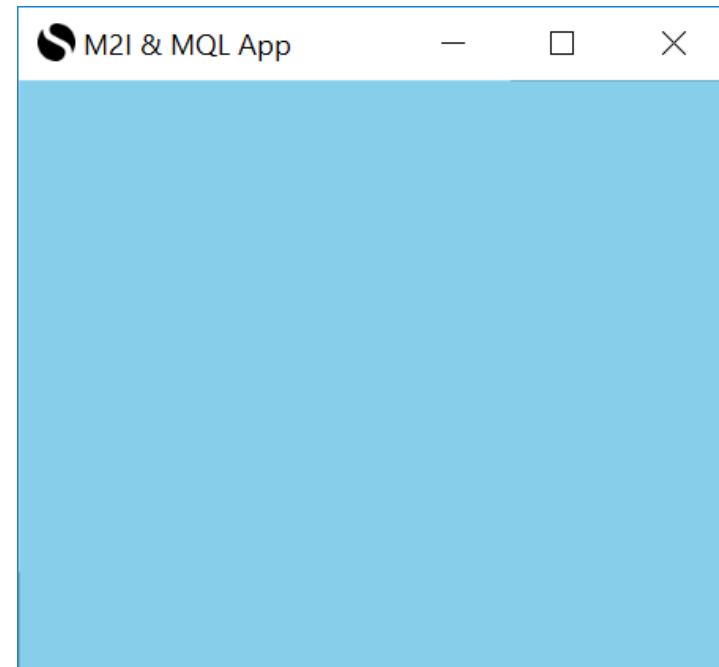
fenetre.iconbitmap("Logo.ico")

fenetre.config(bg = "#87CEEB")

fenetre.geometry("300x250")

fenetre.maxsize(800,600)
fenetre.minsize(300,200)

fenetre.mainloop()
```



Interface graphique pour Python TKinter : Les Widgets de Tkinter

Widget (*Window & gadget*) les composants graphiques de base d'une bibliothèque.

- ✓ **Tk** : fenêtre de plus haut niveau
- ✓ **Frame** contenant pour organiser d'autres widgets
- ✓ **Label** zone de message
- ✓ **Button** bouton d'action avec texte ou image
- ✓ **Message** zone d'affichage multi-lignes
- ✓ **Entry** zone de saisie
- ✓ **Checkbutton** bouton à deux états
- ✓ **Radiobutton** bouton à deux états exclusifs par groupe de boutons
- ✓ **Scale** glissière à plusieurs positions
- ✓ **PhotoImage** sert à placer des images (GIF et PPM/PGM) sur des widgets
- ✓ **BitmapImage** sert à placer des bitmaps sur des widgets
- ✓ **Menu** menu déroulant associé à un Menubutton
- ✓ **Menubutton** bouton ouvrant un menu d'options
- ✓ **Scrollbar** ascenseur
- ✓ **Listbox** liste à sélection pour des textes
- ✓ **Text** édition de texte simple ou multi-lignes
- ✓ **Canvas** zone de dessins graphiques ou de photos
- ✓ **OptionMenu** liste déroulante
- ✓ **ScrolledText** widget Text avec ascenseur
- ✓ **PanedWindow** interface à onglets
- ✓ **LabelFrame** contenant pour organiser d'autres widgets, avec un cadre et un titre
- ✓ **Spinbox** un widget de sélection multiple

Interface graphique pour Python

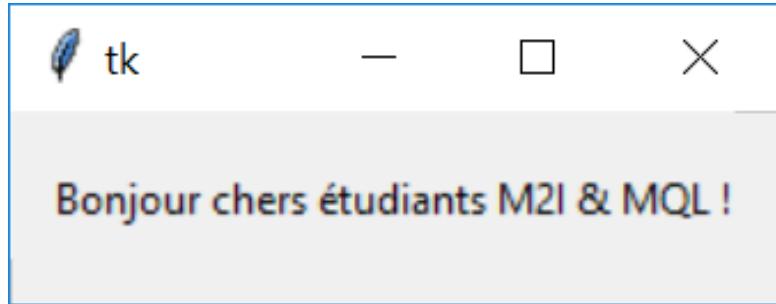
TKinter : Personnaliser la fenêtre

- ✓ Chacun de ces widgets comporte des paramètres permettant de personnaliser leurs **couleurs** de l'arrière-plan ou du **texte**, leurs **dimensions**, la **police** de leurs caractères... etc.
- ✓ Ainsi, pour les organiser et les placer, on utilise une des méthodes suivantes :
 - ✓ **·pack()** qui permet d'organiser et placer des widgets en blocs.
 - ✓ **·grid()** qui permet d'organiser et placer des widgets en étant comme des cases de tableau.
 - ✓ **·place()** qui permet d'organiser et placer des widgets dans un endroit précisé

Interface graphique pour Python

TKinter : Widget - Label

- ✓ La création d'un widget affichant un simple message textuel **Label**



- ✓ Le positionnement du label : **widget.pack()**
- ✓ Le lancement de la boucle d'événements : **widget.mainloop()**

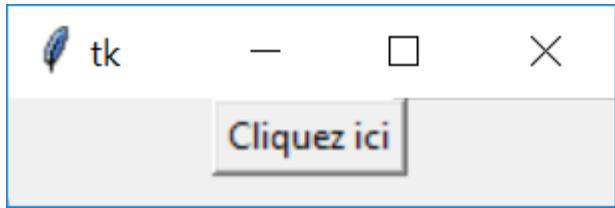
```
import tkinter

widget = tkinter.Label(None, text='\nBonjour chers étudiants M2I & MQL !')
widget.pack()
widget.mainloop()
```

Interface graphique pour Python

TKinter : Widget - Buttons

- ✓ L'ajouter d'un bouton « Button » :



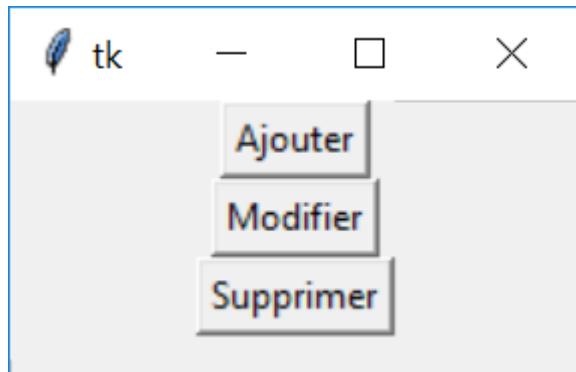
```
from tkinter import *

fenetre = Tk()

bouton1 = Button (fenetre, text = "Cliquez ici")
bouton1.pack()

fenetre.mainloop()
```

- ✓ Ajout d'un widget « Frame » pour créer une zone sur laquelle on peut placer, grouper et organiser d'autres widgets:



```
from tkinter import *

fenetre = Tk()

cadre1 = Frame(fenetre)
cadre1.pack()

bouton1 = Button (cadre1, text = "Ajouter")
bouton2 = Button (cadre1, text = "Modifier")
bouton3 = Button (cadre1, text = "Supprimer")
bouton1.pack()
bouton2.pack()
bouton3.pack()

fenetre.mainloop()
```

Interface graphique pour Python TKinter : Widget - Saisie / Choise

- ✓ Ajout une zone de saisie : **entrée1 = Entry (fenetre)**
- ✓ Ajouter des cases à cocher « Checkbutton » : **case_cocheri = Checkbutton (fenetre, text = "choix i")**
- ✓ Ajouter des boutons radio « RadioButton » : **case_cocheri = Radiobutton (fenetre, text = "choix i")**
- ✓ Ajouter des listes « Listbox » :
liste1 = Listbox (fenetre)
liste1.insert(1, "valeur 1")
- ✓ Ajouter Spinbox, qui sert à ajouter un champ de saisie d'utilisateur, ou bien, à ajouter un bouton sur lequel l'utilisateur peut choisir une valeur dans une intervalle précise, et sa syntaxe s'écrit sous la forme:
spinbox1 = Spinbox(fenetre, from_=1, to=10, increment=1)
- ✓ Ajouter une barre de menu « Menu » à travers le widget « Menubutton » sert à configurer une barre de menu sur laquelle l'utilisateur peut choisir des options, et sa syntaxe s'écrit sous la forme :

```
menu1 = Menu(fenetre)
menu1.add_cascade(label="Fichier")
menu1.add_cascade(label="Aide")
fenetre.config(menu = menu1)
```

Interface graphique pour Python Tkinter : Exemple Widgets

```
from tkinter import *

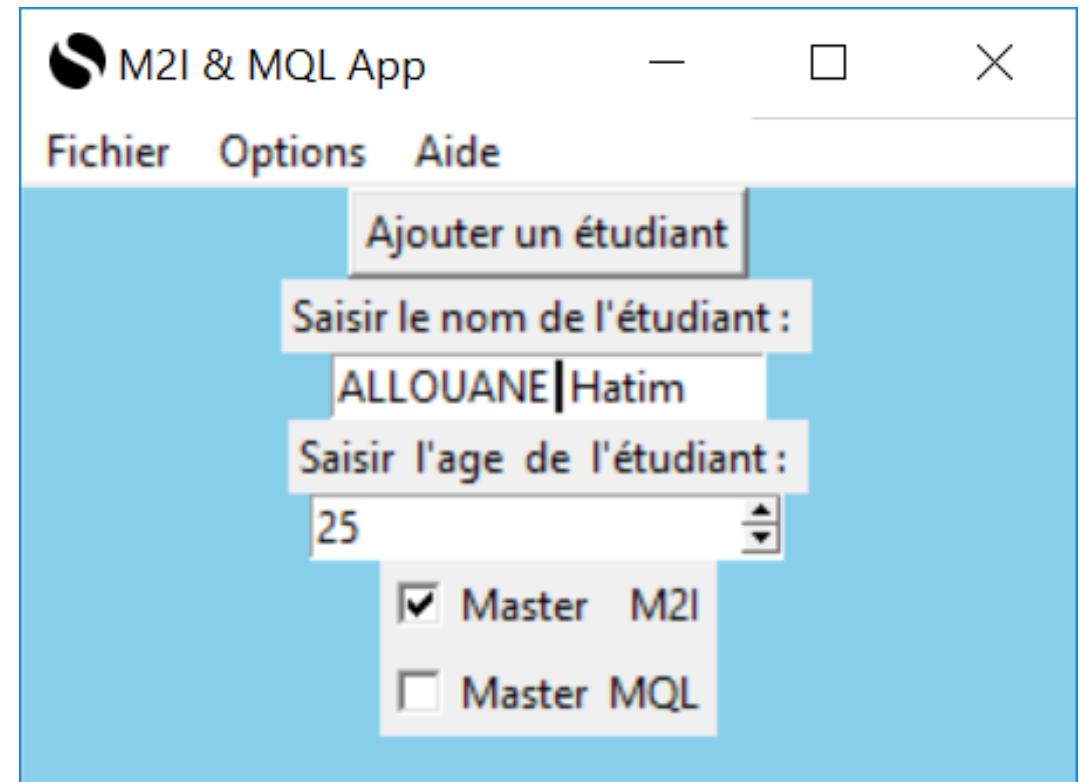
fenetre = Tk()

fenetre.title("M2I & MQL App")
fenetre.iconbitmap("Logo.ico")
fenetre.config(bg = "#87CEEB")
fenetre.geometry("300x250")

menu1 = Menu(fenetre)
menu1.add_cascade(label="Fichier")
menu1.add_cascade(label="Options")
menu1.add_cascade(label="Aide")
fenetre.config(menu = menu1)

cadre1 = Frame(fenetre)
cadre1.pack()
texte1 = Label (fenetre, text = "Saisir le nom de l'étudiant : ")
texte1.pack()
entrée1 = Entry (fenetre)
entrée1.pack()
texte2 = Label (fenetre, text = "Saisir l'age de l'étudiant : ")
texte2.pack()
spinbox1 = Spinbox(fenetre, from_=20, to=30, increment=1)
spinbox1.pack()
case_cocher1 = Checkbutton (fenetre, text = "Master M2I")
case_cocher2 = Checkbutton (fenetre, text = "Master MQL")
case_cocher1.pack()
case_cocher2.pack()
bouton1 = Button (cadre1, text = "Ajouter un étudiant")
bouton1.pack()

fenetre.mainloop()
```



Interface graphique pour Python

TKinter : Notion CallBack

- ✓ La programmation d'interface graphique passe par une boucle principale chargée de traiter les différents événements qui se produisent.
- ✓ Cette boucle est généralement gérée directement par la bibliothèque d'interface graphique utilisée, il faut donc pouvoir spécifier à cette bibliothèque quelles fonctions doivent être appelées dans quels cas.
- ✓ Ces fonctions sont nommées des **callbacks** (*rappels*), appelées directement par la bibliothèque d'interface graphique lorsque des événements spécifiques se produisent.

<Button-1>	: Click gauche
<Button-2>	: Click milieu
<Button-3>	: Click droit
<Double-Button-1>	: Double click droit
<Double-Button-2>	: Double click gauche
<KeyPress>	: Pression sur une touche
<KeyPress-a>	: Pression sur la touche A (minuscule)
<KeyPress-A>	: Pression sur la touche A (majuscule)
<Return>	: Pression sur la touche entrée
<Escape>	: Touche Echap
<Up>	: Pression sur la flèche directionnelle haut
<Down>	: Pression sur la flèche directionnelle bas
<ButtonRelease>	: Lorsque qu'on relâche le click
<Motion>	: Mouvement de la souris
<B1-Motion>	: Mouvement de la souris avec click gauche
<Enter>	: Entrée du curseur dans un widget
<Leave>	: Sortie du curseur dans un widget
<Configure>	: Redimensionnement de la fenêtre
<Map> <Unmap>	: Ouverture et iconification de la fenêtre
<MouseWheel>	: Utilisation de la roulette

Interface graphique pour Python calculator App

```

from tkinter import *
from math import *

def evaluer(event) :
    chaine.configure(text = '=> ' + str(eval(entree.get())))

fenetre = Tk()

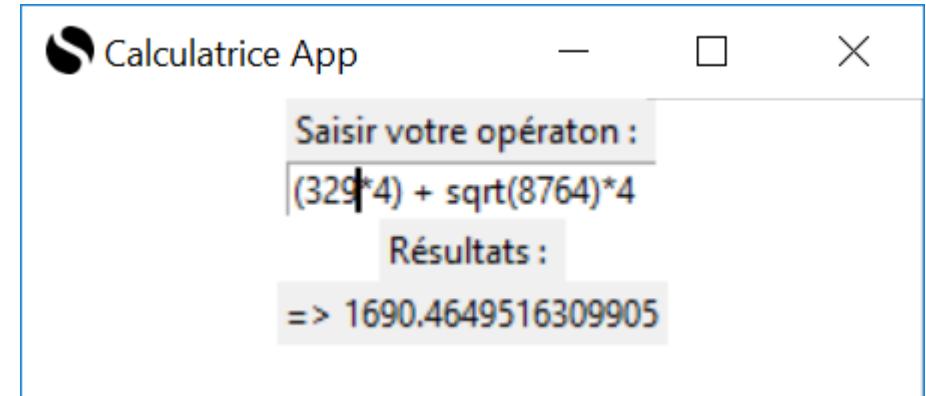
fenetre.title("Calculatrice App")
fenetre.iconbitmap("Logo.ico")
fenetre.config(bg = "White")
fenetre.geometry("300x100")

entree = Entry(fenetre)
entree.bind('<Return>', evaluer)
chaine = Label(fenetre)
texte1 = Label (fenetre, text = "Saisir votre opératon : ")
texte2 = Label (fenetre, text = "Résultats : ")

texte1.pack()
entree.pack()
texte2.pack()
chaine.pack()

fenetre.mainloop()

```



Interface graphique pour Python

Tkinter/Exp 2 : Mapping Sqlite3/XML

M2I & MQL Add Student

Fichier Options Aide

Id :	<input type="text"/>
Nom :	<input type="text"/>
Prénom :	<input type="text"/>
Master :	<input type="text"/>

Add > DB Add > XML Select Clear

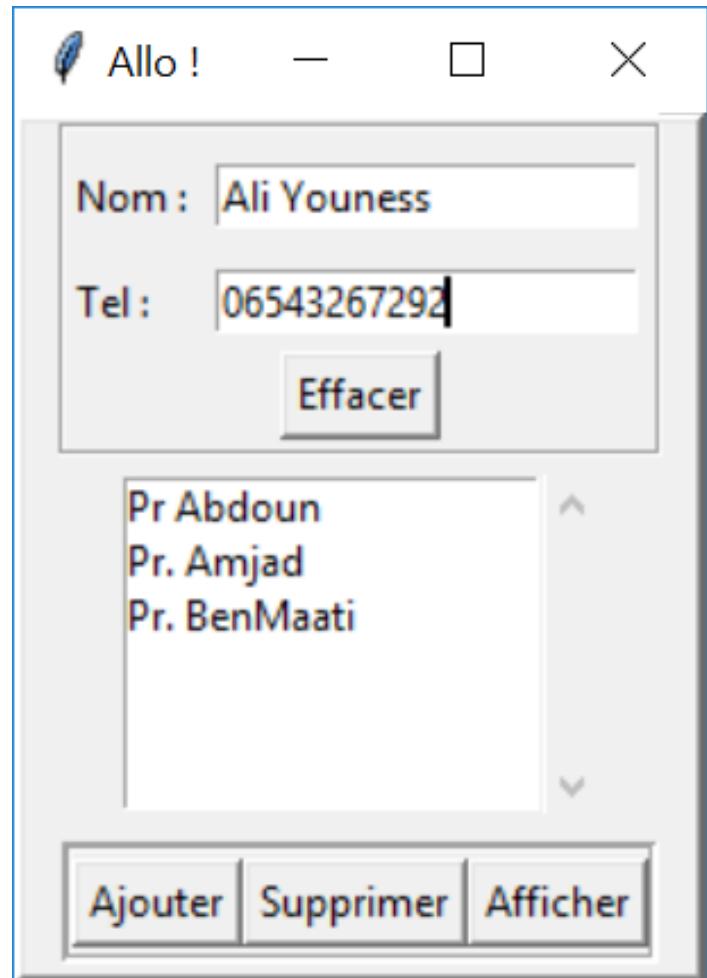
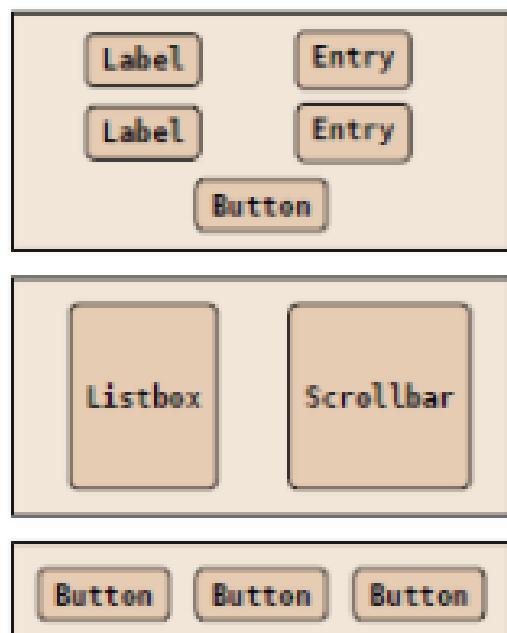
Liste des étudiants : MQL et M2I

Identifiant	Nom	Prénom	Master
1	El Fahssi	Ikram	Master MQL
2	Ktiti	Mohamed	Master MQL
3	Amakran	Hajar	Master M2I
4	Esafi	Khalid	Master M2I

Interface graphique pour Python

Tkinter/Exp 3 : Annuaire Téléphonique

- ✓ Pour la conception graphique, on aura 3 zones :
- ✓ Une zone supérieure, dédiée à l'affichage ;
- ✓ Une zone médiane est une liste alphabétique ordonnée ;
- ✓ Une zone inférieure est formée de boutons de gestion





Interface graphique pour Python

PYQT POUR QT

Interface graphique pour Python PYQT pour Qt : Module QT

- ✓ Qt est une API orientée objet développée en C++ qui offre des composants :
 - ✓ d'interface graphique (widgets)
 - ✓ d'accès aux données
 - ✓ de connexions réseaux
 - ✓ d'analyse XML, ...
- ✓ Sous Python et pour avoir un binding avec le Module QT, deux frameworks GUI Qt sont disponibles :
 - ✓ Framework **PySide**
 - ✓ Framework **PyQt**
- ✓ Étant donné que les deux sont des liaisons de la même bibliothèque QT (mêmes widgets et code), **la différence ?!**



Interface graphique pour Python **PYQT pour Qt : PyQt # PySide**

- ✓ PyQt a été créé par Riverbank Computing en tant que liaison Python pour Qt. Pendant longtemps, PyQt était la seule liaison Python disponible.
- ✓ Cependant, en 2009, un différend est survenu concernant la licence sous laquelle PyQt devrait être publié, entre les créateurs de PyQt et les créateurs de Qt (Nokia).
- ✓ Comme les groupes n'ont pas pu se mettre d'accord, une nouvelle liaison Python, PySide, est née.
- ✓ PySide (pour Qt4) a été publié sous licence LGPL (comme Qt), tandis que PyQt a été publié sous licence GPL:
 - ✓ La licence LGPL vous permet de distribuer du code sans avoir à partager votre code source. Cela permet à une personne de développer des applications commerciales avec **PySide**.
 - ✓ La licence GPL vous empêche cependant de retenir le code source dans un programme distribué pour **PyQt**.



Interface graphique pour Python **PYQT pour Qt : What is PyQt, exactly ?**

- ✓ PyQt est un ensemble de liaisons Python pour le Framework d'application Qt.
- ✓ PyQt6 prend en charge Qt v6, PyQt5 prend en charge Qt v5 et PyQt4 prend en charge Qt v4.
- ✓ Les liaisons sont implémentées sous la forme d'un ensemble de modules Python et contiennent plus de 1 000 classes.
- ✓ PyQt rassemble le framework d'application multiplateforme Qt C++ et le langage interpréte multiplateforme Python.
- ✓ **Qt** est plus qu'une boîte à outils **GUI**. Il comprend des abstractions de sockets réseau, des threads, Unicode, des expressions régulières, des bases de données SQL, SVG, OpenGL, XML, un navigateur Web entièrement fonctionnel, un système d'aide, un cadre multimédia, ainsi qu'une riche collection de widgets GUI.
- ✓ **PyQt** inclut également **Qt Designer**, un concepteur d'interface utilisateur graphique.
- ✓ **PyQt** est capable de générer du code Python à partir de **Qt Designer**.



Interface graphique pour Python PYQT pour Qt : So ! PyQt6 new one

- ✓ PyQt est un module libre de python qui permet de :
 - ✓ Lier le langage Python avec la bibliothèque Qt.
 - ✓ Créer des interfaces graphiques en Python.
- ✓ PyQt6 est la version utilisée actuellement par Python.

pip install PyQt6

For Examples, try with this link ^^^^

<https://wiki.python.org/moin/PyQt/SomeExistingApplications>



Interface graphique pour Python **PYQT pour Qt : About**

- ✓ PyQt5 offre un ensemble d'outils de conception et de réalisation d'applications tels que :
 - ✓ **Qt Creator** : IDE de Qt
 - ✓ **Qt Designer** : Outil graphique qui vous permet de créer des interfaces graphiques **QWidget**
- ✓ Pour installer le module PyQt5, il faut taper la commande : **pip install PyQt5**
- ✓ Pour installer les outils de PyQt5, il faut taper la commande : **pip install PyQt5-Tools**
- ✓ Après ces installations, les outils se trouvent dans le dossier suivant :

C:\ ... \Python3X\Lib\site-packages\qt5_applications\Qt\bin



Interface graphique pour Python PYQT pour Qt : Créer une interface

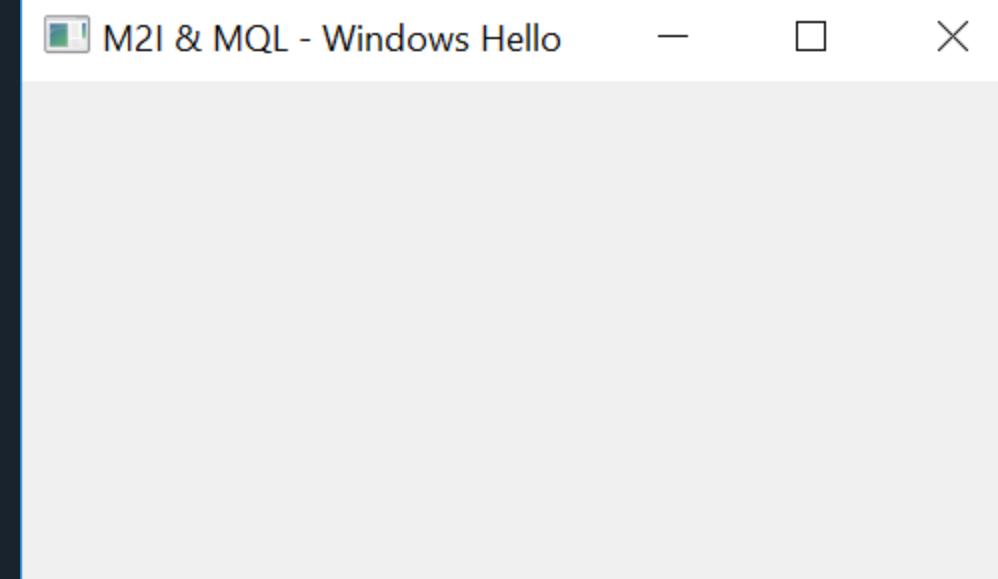
Pour créer une fenêtre graphique en PyQt5, on doit :

- ✓ Importer le module système : **sys**
- ✓ Importer la classe qui génère l'application : **QApplication** depuis le package **PyQt5.QtWidgets**
- ✓ Importer la classe **QWidget** depuis le package **PyQt5.QtWidgets**
- ✓ Création d'une instance de **Qapplication** avec **QApplication(sys.argv)**
- ✓ Créer une fenêtre avec la méthode **QWidget()**: **fen = QWidget()**
- ✓ Visualiser la fenêtre à l'aide de la méthode **show()** : **fen.show()**
- ✓ Exécuter l'application à l'aide de la méthode **exec_()** : **app.exec_()**



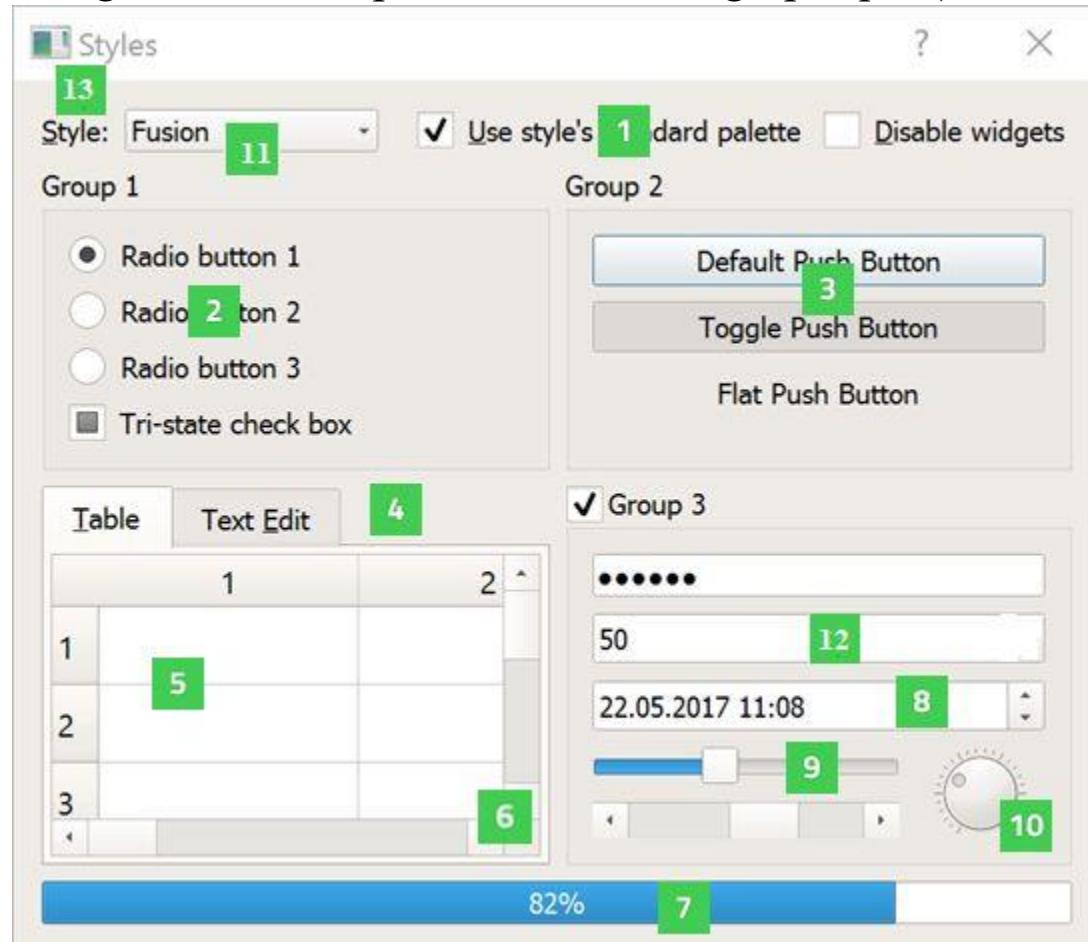
Interface graphique pour Python PYQT pour Qt : Windows Hello

```
# Création d'une fenêtre avec PyQt5
# importations nécessaire à la réalisation d'une interface graphique
import sys
from PyQt5.QtWidgets import QApplication, QWidget
# Création d'une instance de QApplication
app = QApplication(sys.argv)
# On crée une fenêtre à l'aide de l'objet QWidget
fen = QWidget()
# On donne un titre à la fenêtre
fen.setWindowTitle("M2I&MQL - Windows Hello")
# On fixe la taille de la fenêtre
fen.resize(500,250)
# On déplace le widget (fenêtre fen) à une position sur l'écran
fen.move(300, 200)
# On visualise la fenêtre
fen.show()
# Exécution de l'application
app.exec_()
```



Interface graphique pour Python PYQT pour Qt : Notion de Widget

- ✓ Le Widget est un composant d'interface graphique (Bouton, Label, zone de texte, Barre de défilement, Liste déroulante, ...)



QCheckBox (1) fournit une case à cocher avec une étiquette de texte.

QRadioButton (2) fournit un bouton radio avec un texte ou une étiquette pixmap.

QPushButton (3) fournit un bouton de commande.

QTabWidget (4) fournit une pile de widgets à onglets.

QTableWidget (5) offre une vue de table classique basée sur des éléments.

QScrollBar (6) fournit une barre de défilement vertical ou horizontal.

QProgressBar (7) fournit une barre de progression horizontale.

QDateTimeEdit (8) fournit un widget pour l'édition des dates et des heures.

QSlider (9) fournit un curseur vertical ou horizontal.

QDial (10) offre un contrôle de portée arrondi (comme un compteur de vitesse ou un potentiomètre).

QComboBox (11) fournit un bouton combiné et une liste pop-up (déroulante).

QLineEdit (12) fournit un widget pour entrer un texte.

QLabel (13) fournit un widget d'étiquette pour mettre un titre ou une image.

Interface graphique pour Python PYQT pour Qt : PYQT # TKINTER

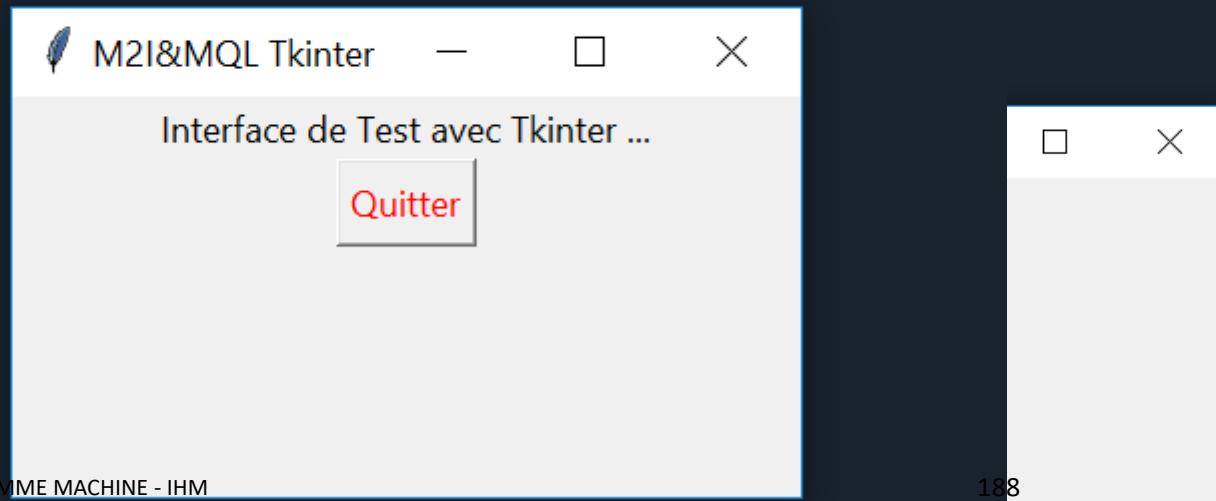
- ✓ Tkinter est la bibliothèque graphique libre d'origine pour le langage Python, permettant la création d'interfaces graphiques. L'interface est programmée puis exécutée.



- ✓ Tandis que, PyQt5 offre trois modules d'interface graphique
 1. L'interface est programmée puis exécutée dans un environnement Python (Compiler)
 2. L'interface peut être générée par un compilateur Python (Compiler)
 3. L'interface peut être créée par un programmeur Python (Avantage)
- ✓ Pour créer une application avec le module PyQt5 :
 - ✓ Installer ce module.
 - ✓ Ecrire le code correspondant à votre application.

```
import sys
from PyQt5 import QtWidgets,QtCore
from PyQt5.QtWidgets import QApplication, QWidget
app = QApplication(sys.argv)

import tkinter as tk
racine = tk.Tk()
racine.geometry("350x200")
racine.title("M2I&MQL Tkinter")
label = tk.Label(racine, text="Interface de Test avec Tkinter ...")
bouton = tk.Button(racine, text="Quitter", command=racine.quit)
bouton["fg"] = "red"
label.pack()
bouton.pack()
racine.mainloop()
```



```
from PyQt6.QtWidgets import QApplication, QWidget, QPushButton, QLabel
from PyQt6.QtGui import QIcon, QFont
import sys

class Window(QWidget):
    def __init__(self):
        super().__init__()
        self.setWindowTitle("PyQt6 Button - M2I & MQL")
        self.setWindowIcon(QIcon("qt.png"))
        self.setGeometry(500, 200, 500, 400)
        self.create_widgets()

    def create_widgets(self):
        btn = QPushButton("Cliquer Ici ", self)
        #btn.move(100,100)
        btn.setGeometry(100,100, 100,100)
        btn.setStyleSheet('background-color:red')
        btn.setIcon(QIcon("qt.png"))

        label = QLabel("Exp Label", self)
        label.move(100,220)
        label.setStyleSheet('color:green')
        label.setFont(QFont("Times New Roman", 15))

app = QApplication(sys.argv)
window = Window()
window.show()
sys.exit(app.exec())
```

Button & QLabel

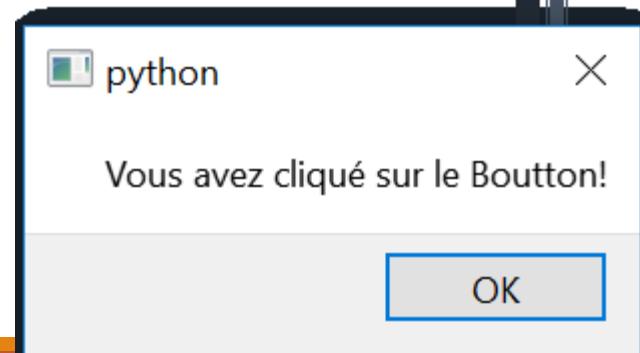
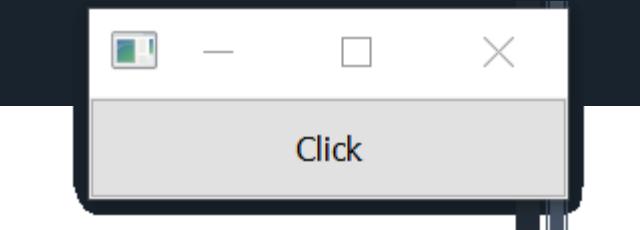


Interface graphique pour Python PYQT pour Qt : Signals / slots

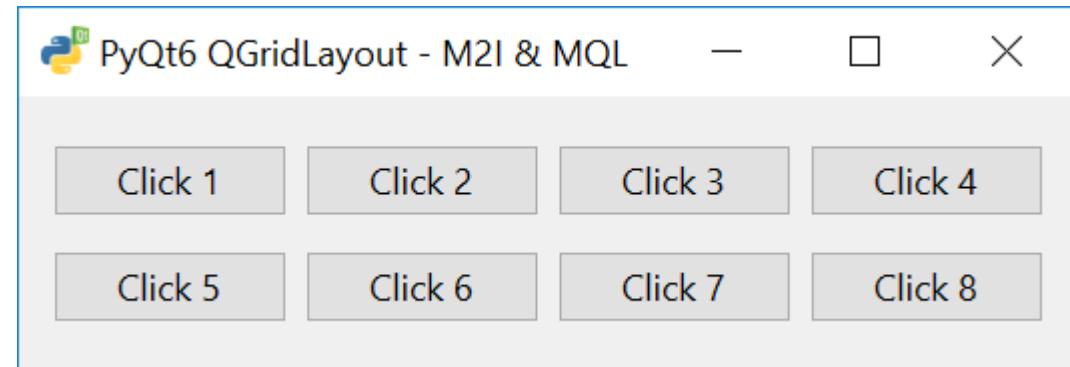
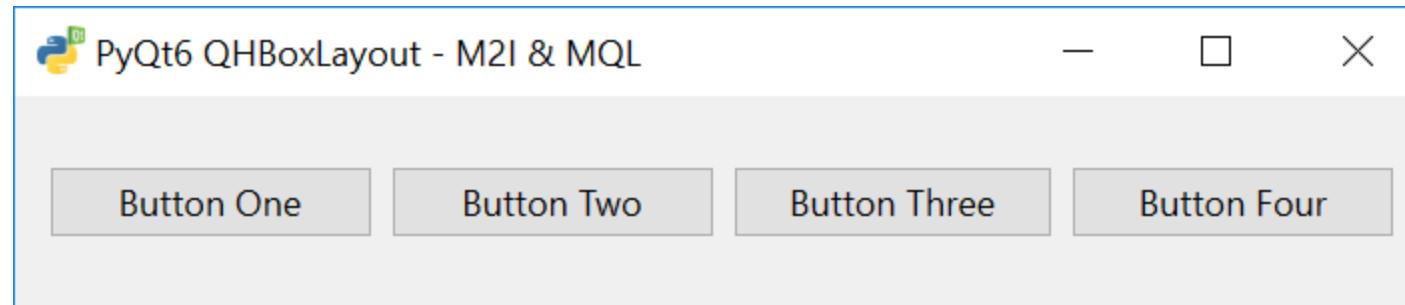
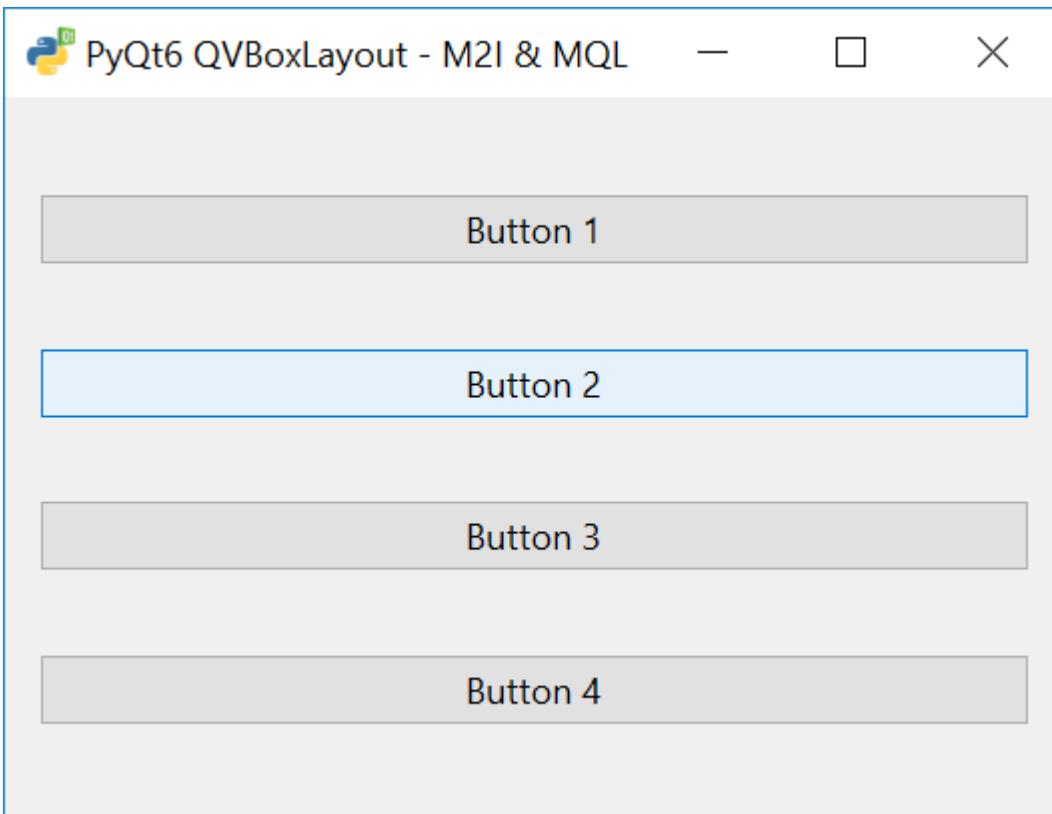
- ✓ Qt utilise un mécanisme appelé **signaux** pour vous permettre de réagir à des événements tels que l'utilisateur cliquant sur un bouton.
- ✓ ***button.clicked*** est un **signal**,
- ✓ ***.connect(...)*** est un **slot**, c'est une fonction qui est appelée lorsque le signal se produit.

```
from PyQt5.QtWidgets import *
app = QApplication([])
button = QPushButton('Click')
def on_button_clicked():
    alert = QMessageBox()
    alert.setText('Vous avez cliqué sur Le Boutton!')
    alert.exec()

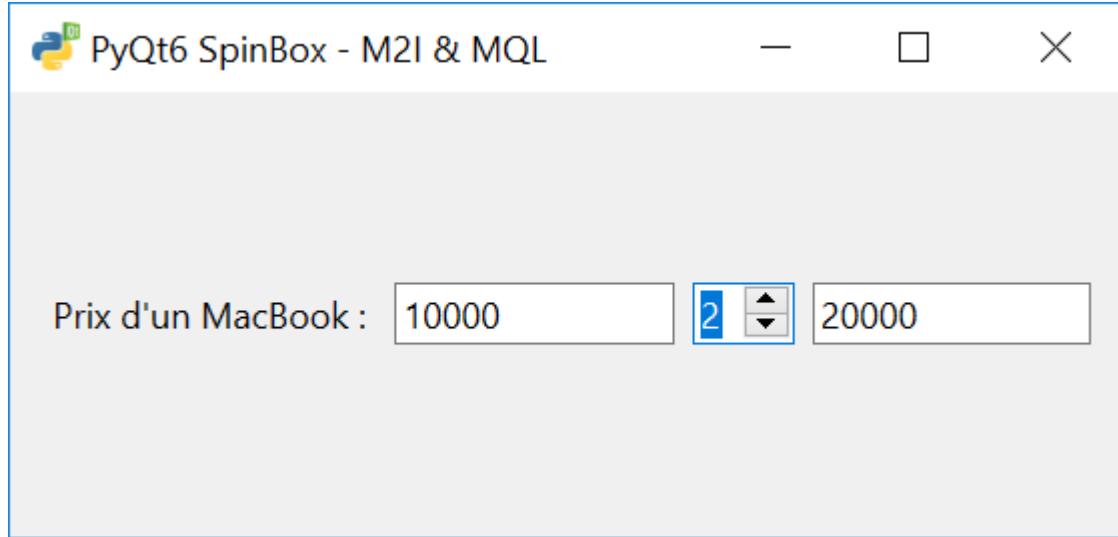
button.clicked.connect(on_button_clicked)
button.show()
app.exec()
```



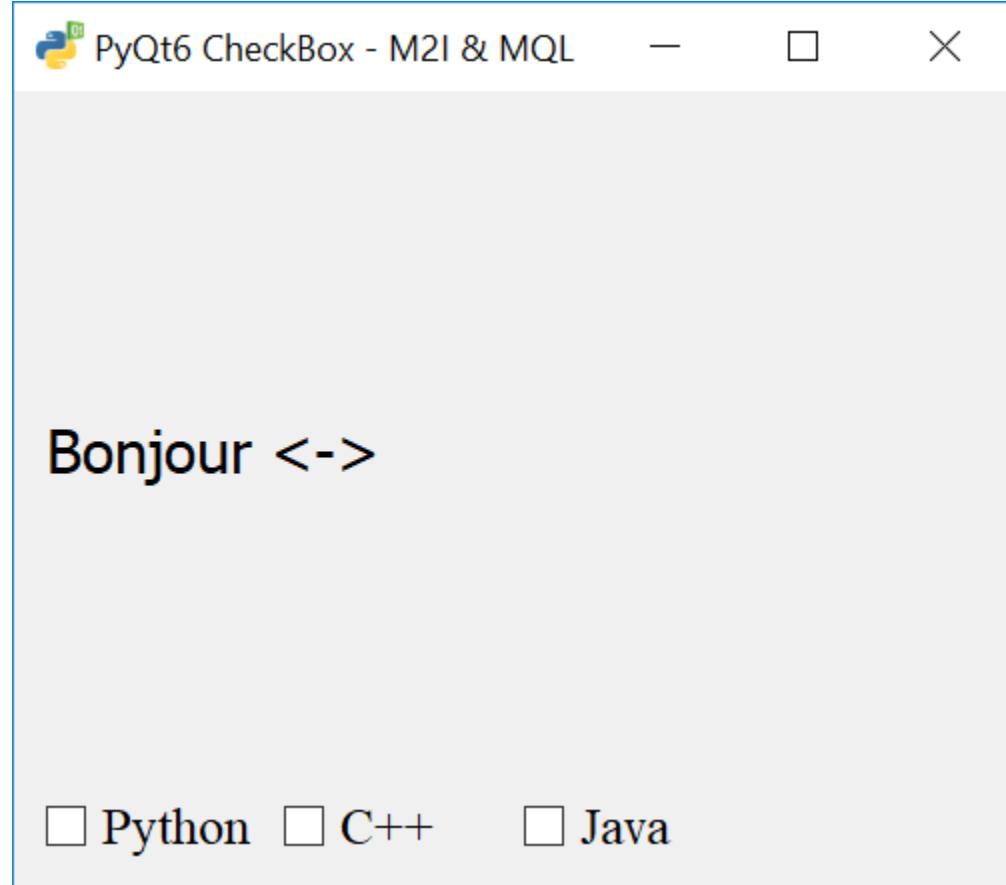
Interface graphique pour Python PYQT pour Qt : Layout Management



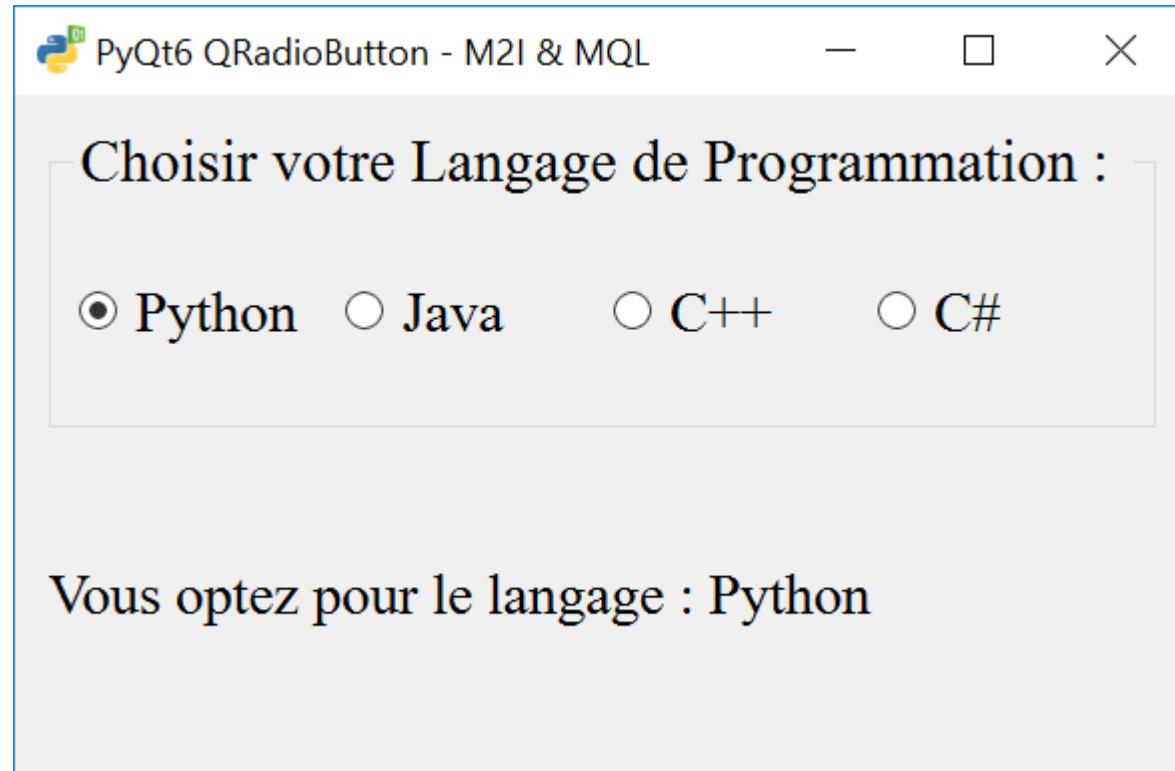
Interface graphique pour Python PYQT pour Qt : QSpinBox



Interface graphique pour Python PYQT pour Qt : QCheckBox



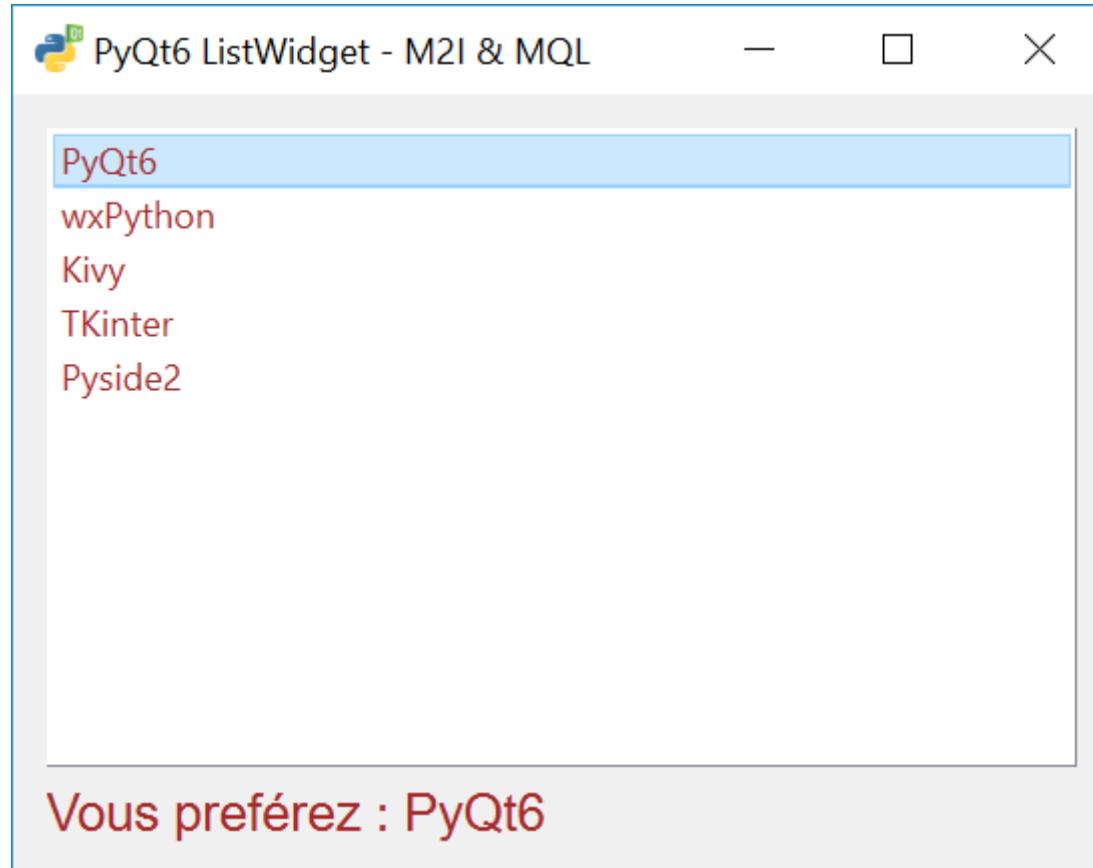
Interface graphique pour Python PYQT pour Qt : QRadioButton



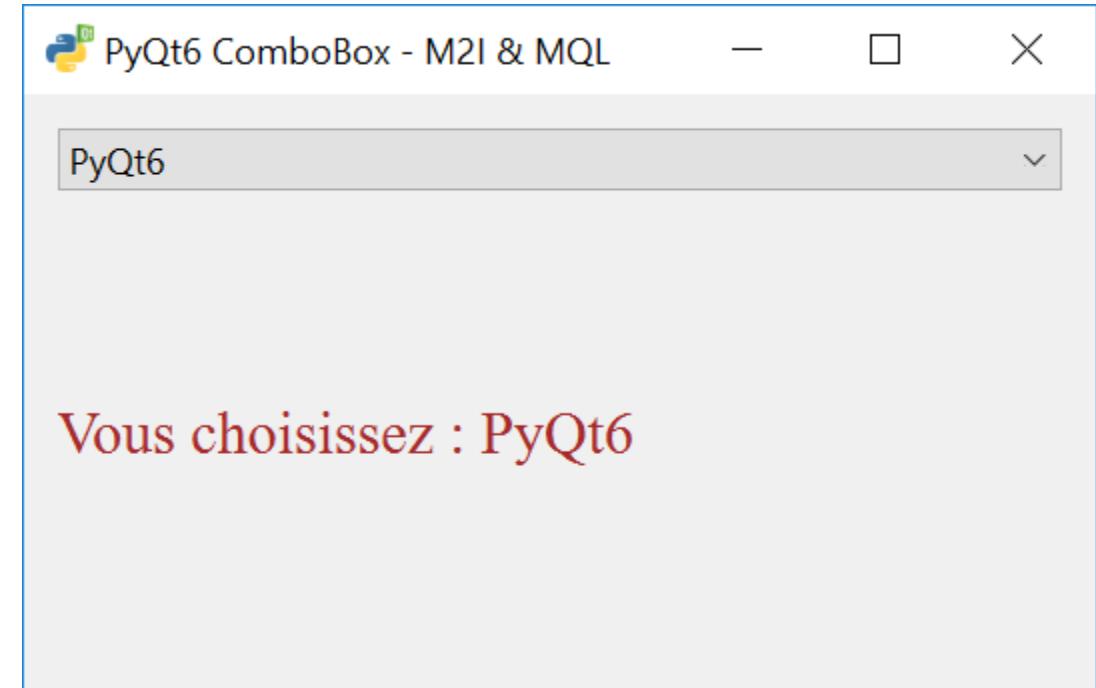
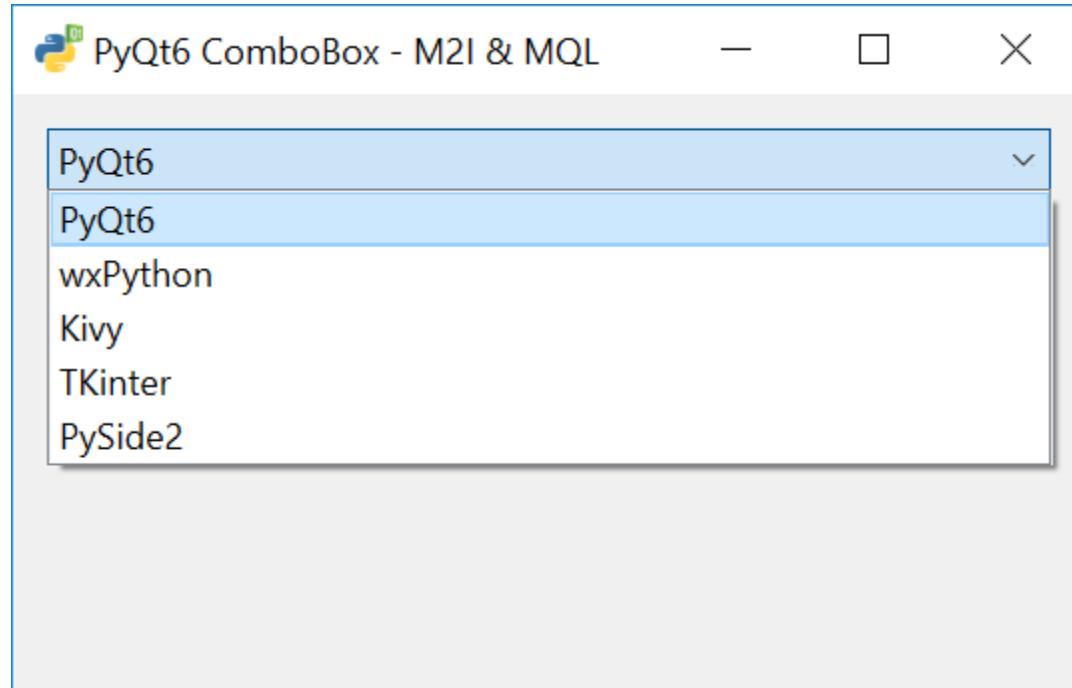
Interface graphique pour Python PYQT pour Qt : QTable

1	2	3
1	Prénom	Nom
2	Ahmed	Bakkali
3	Samira	Diani

Interface graphique pour Python PYQT pour Qt : QListWidget

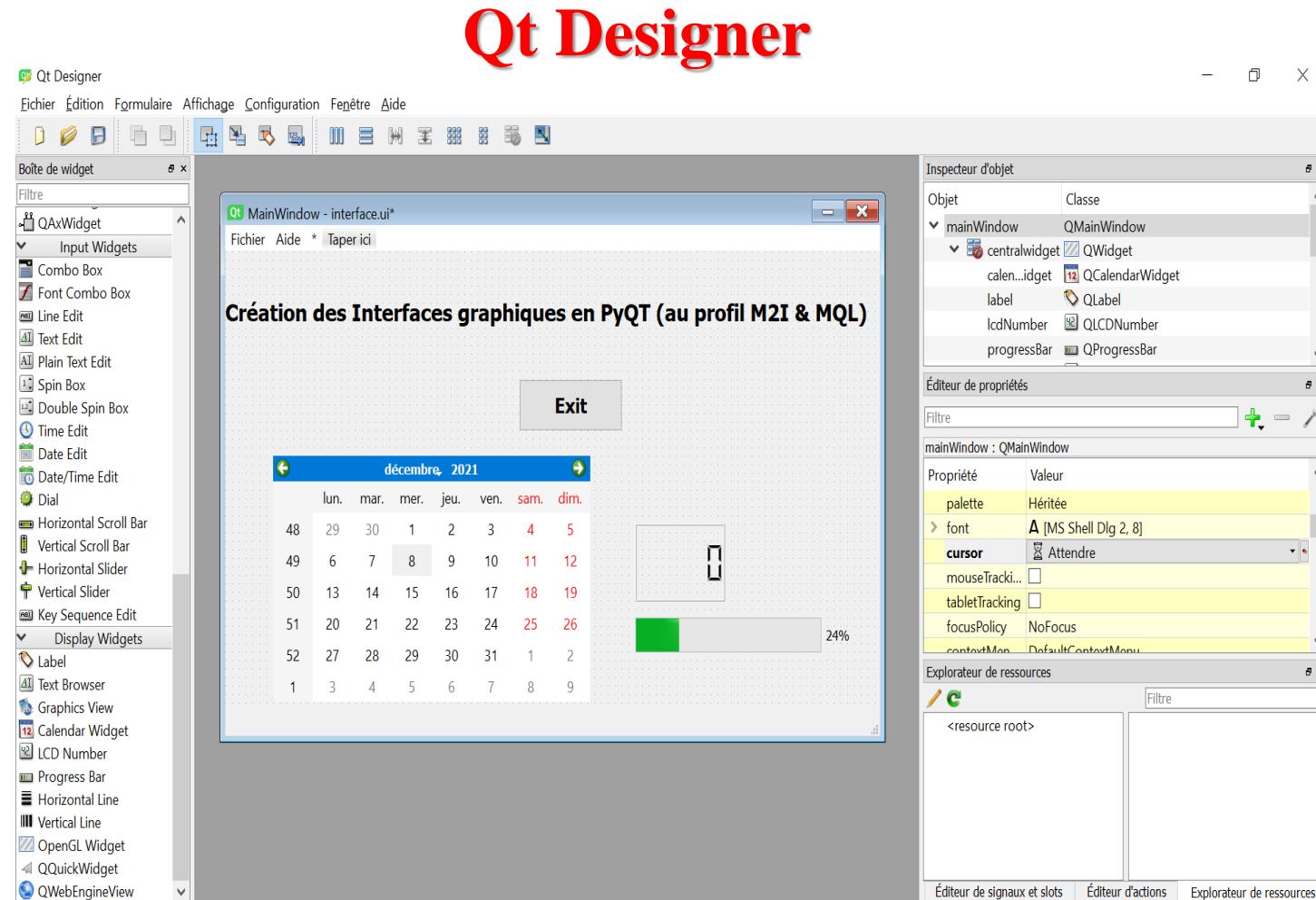


Interface graphique pour Python PYQT pour Qt : QCombiBox

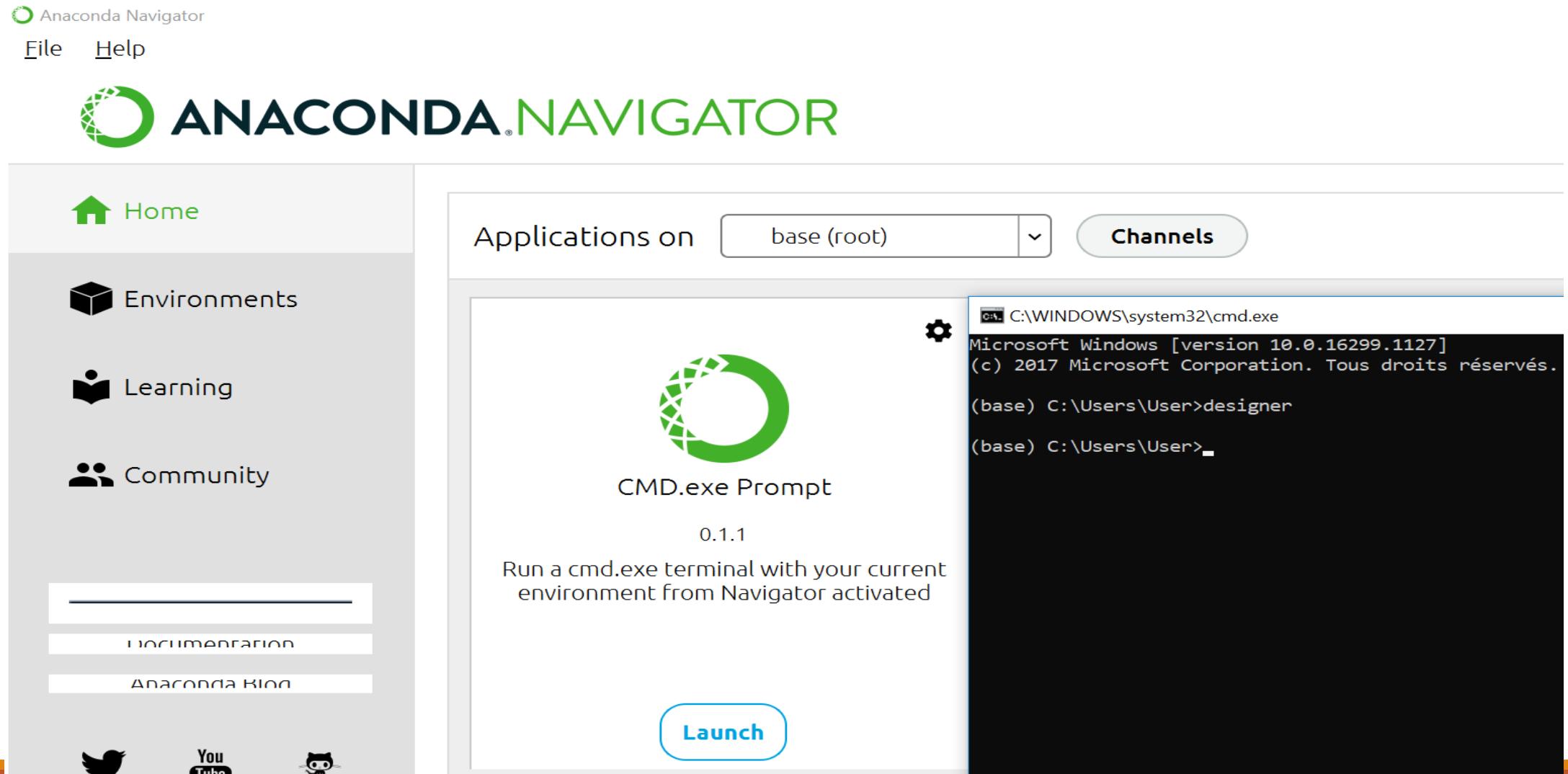


Interface graphique pour Python PYQT pour Qt : Difficultés

- ✓ Si on va créer des grandes applications où les **interfaces sont plus compliquées**:
 - ✓ le **travail** peut devenir un **peu lourd** en programmation.
 - ✓ **Pénible**, si on va créer ces interfaces par écriture d'un code.
- ✓ Pour cela il vaut mieux d'utiliser un outil de conception et de création d'interfaces graphiques GUI pour faciliter le travail.
- ✓ Deux fichier à générer :
 - ✓ **Nom_fichier.ui** : Créer l'interface par l'outil **Qt Designer**
 - ✓ **Nom_fichier.py** : Appeler le fichier de l'interface GUI dans le code python crée par l'éditeur Python (Spider).



Interface graphique pour Python PYQT pour Qt : Run QtDesigner

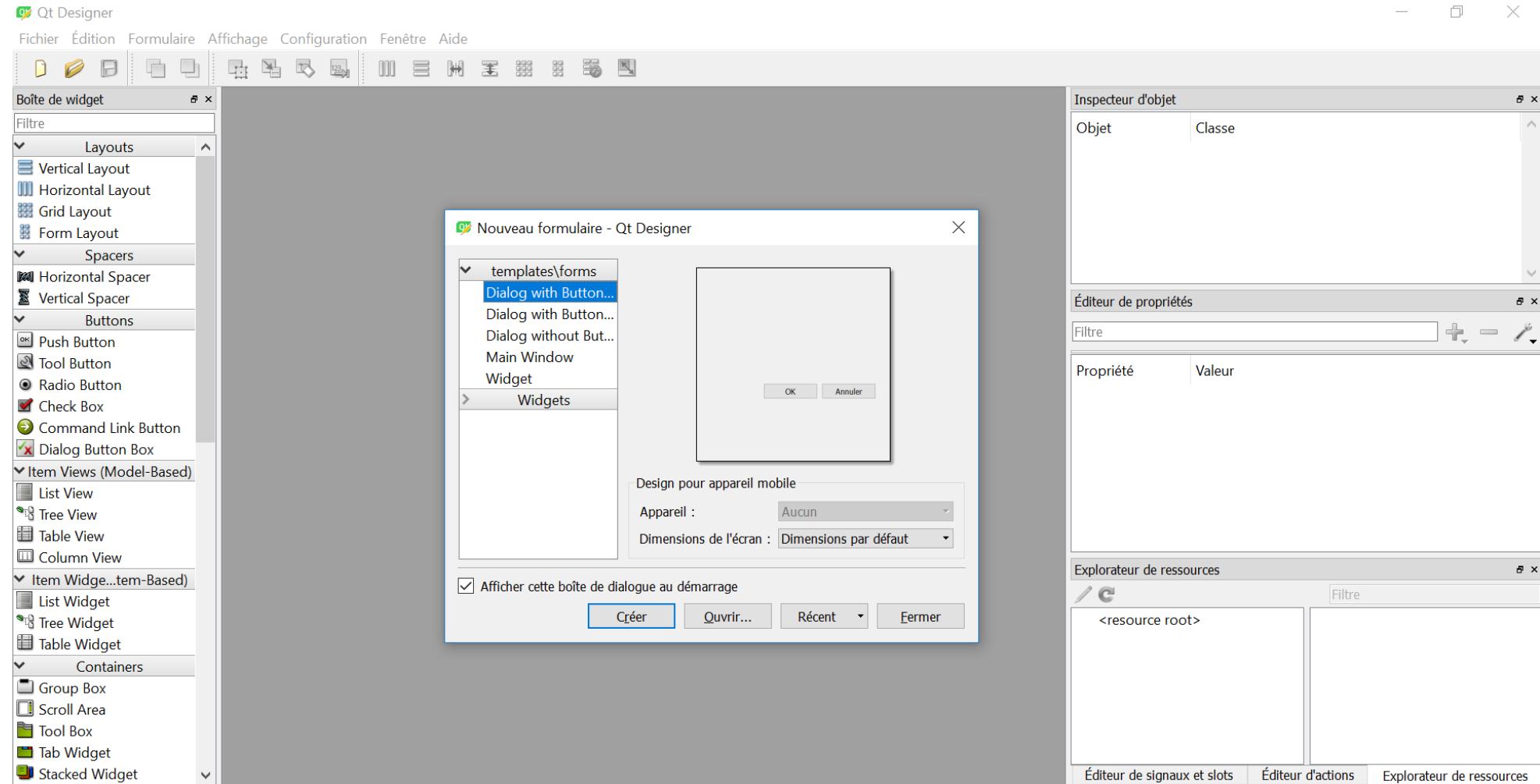


The screenshot shows the Anaconda Navigator interface. On the left, there's a sidebar with icons for Home, Environments, Learning, and Community. Below these are links for Documentación and Anaconda RIO. At the bottom are social media links for Twitter, YouTube, and GitHub.

The main area displays the "CMD.exe Prompt" application. It features a green circular logo and the text "CMD.exe Prompt" followed by "0.1.1". A subtext says "Run a cmd.exe terminal with your current environment from Navigator activated". A "Launch" button is at the bottom. To the right is a terminal window titled "C:\WINDOWS\system32\cmd.exe" showing the Windows version and a command prompt "(base) C:\Users\User>".

At the top of the main area, there's a header with "Applications on base (root)" and a "Channels" button.

Interface graphique pour Python PYQT pour Qt : That's QtDesigner

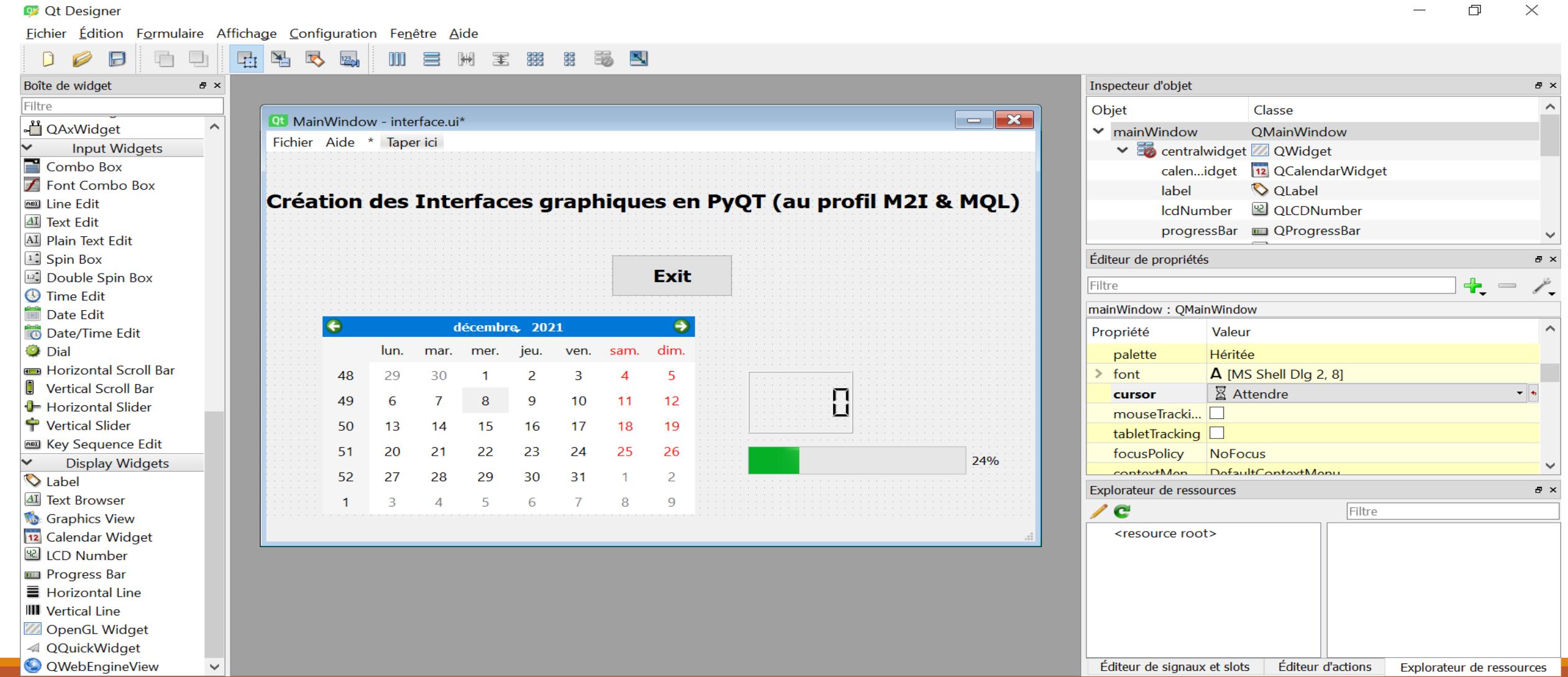


Interface graphique pour Python PYQT pour Qt : Présentation QtDesigner

- ✓ *Qt designer* permet la conception de l'application en insérant les objets nécessaires (Widgets) dans la fenêtre de l'application.
- ✓ Chaque objet possède un ensemble de propriétés permettant son paramétrage. (Nom, Taille, couleur, ...)
- ✓ Le fichier créé (interface graphique) possède l'extension “.*ui*” (User Interface).
- ✓ Après avoir créé l'interface graphique, nous pouvons maintenant écrire le code du programme (En utilisant la programmation événementielle) permettant d'agir sur les composants (Widgets) de la fenêtre.
- ✓ Avec PyQt, on utilise les deux termes suivants:
 - ✓ **Le signal** : L'événement réalisé sur un Widget (un clic, un appui de touche, ...)
 - ✓ **Le slot** : Le traitement réalisé suite au signal émis (Une fonction à définir).
- ✓ Nous obtenons à la fin un fichier d'interface (*.ui*) et un fichier de code (*.py*)



Interface graphique pour Python PYQT pour Qt : Exp1 - QtDesigner

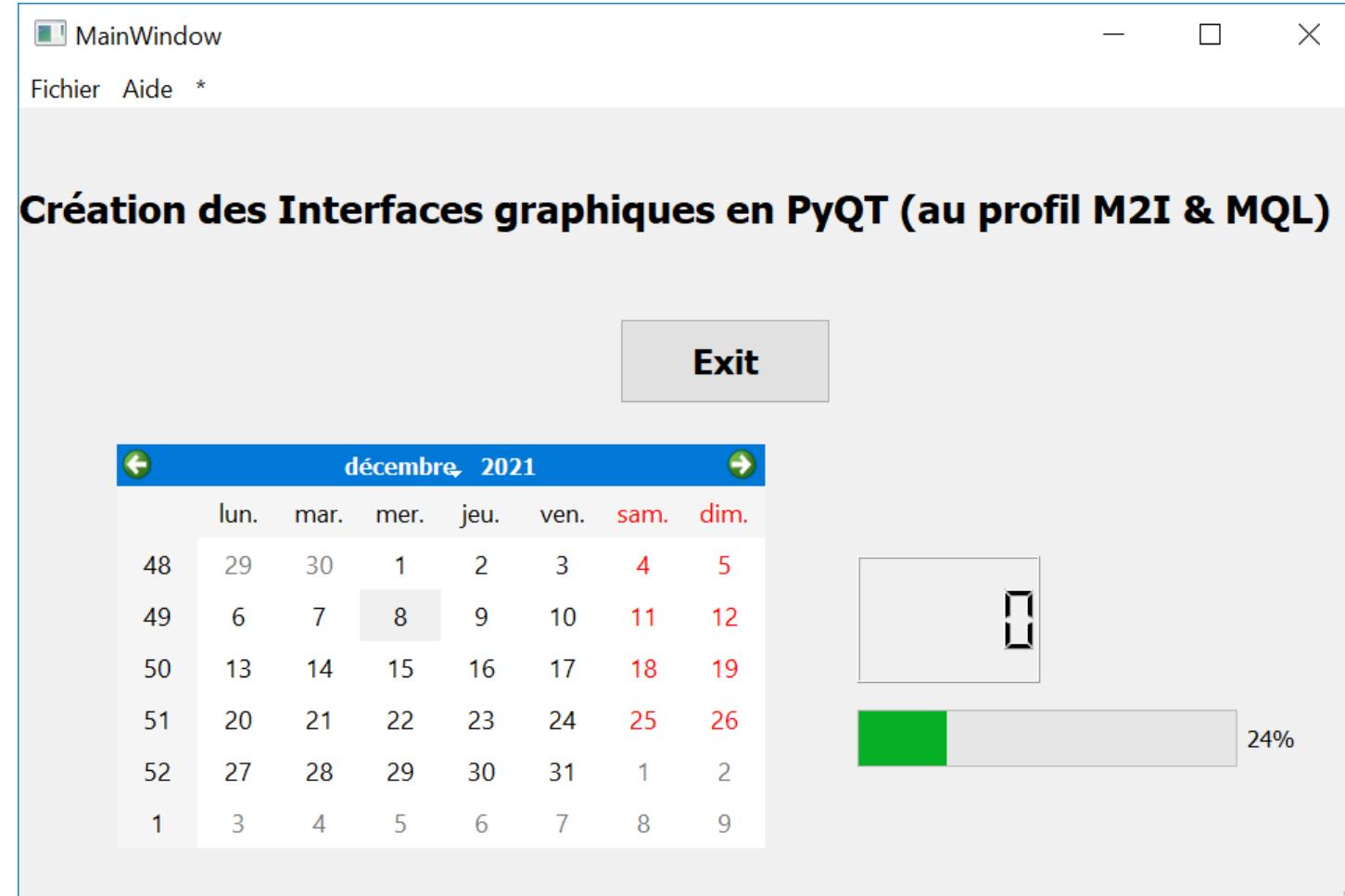


Interface graphique pour Python PYQT pour Qt : Exp1 – Run by Spider

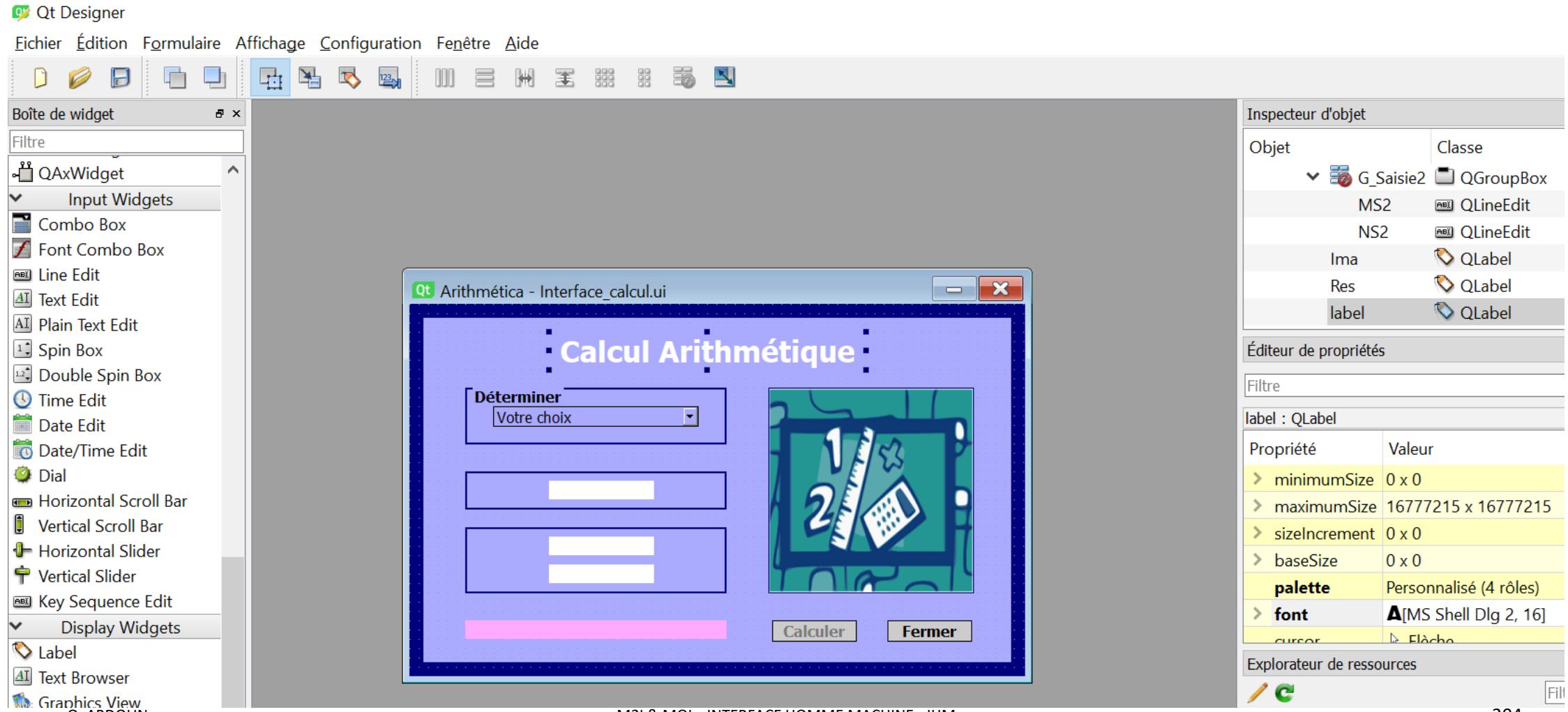
```
import sys
from PyQt5 import QtWidgets, uic

def Fermer():
    Fen.close()

App = QtWidgets.QApplication(sys.argv)
Fen=uic.loadUi("interface.ui")
Fen.show()
Fen.Fe.clicked.connect(Fermer)
App.exec_()
sys.exit()
```



Interface graphique pour Python PYQT pour Qt : Exp2 - Calculator



Interface graphique pour Python PYQT pour Qt : Exp2 - Calculator

Spyder (Python 3.9)

File Edit Search Source Run Debug Consoles Projects Tools View Help

F:\FS\IHM\Exp\PyQT\Exp 2\Calcul_Arith.py

PYQTTest.py X lancerexp1.py X Calcul_Arith.py X

```

44     N1=int(u.NS1.text())
45     elif Ind in list(range(1, 3)):
46         if u.NS2.text().isdigit()==False or u.MS2.text().isdigit()==False:
47             QtWidgets.QMessageBox.question(None, 'Attention', "Nombre 1 ou n
48             return None
49         else:
50             N2=int(u.NS2.text())
51             M2=int(u.MS2.text())
52             if Ind == 1:
53                 u.Res.setText(str(PGCD(N2,M2)))
54             elif Ind == 2:
55                 u.Res.setText(str(PPCM(N2,M2)))
56             elif Ind == 3:
57                 u.Res.setText(Fact_Premier(N1))
58             elif Ind == 4:
59                 u.Res.setText(Premier(N1))
60             elif Ind == 5:
61                 u.Res.setText(str(Fact(N1)))
62
63     def Afficher():
64         global Ind
65         Ind=u.Choix.currentIndex()
66         if Ind in list(range(1,3)):
67             u.G_Saisie1.hide()
68             u.G_Saisie2.show()
69         elif Ind in list(range(3,6)):
70             u.G_Saisie2.hide()
71             u.G_Saisie1.show()
72
73     u.Bouton.clicked.connect(Afficher)
74
75     u.Bouton.clicked.connect(lancerexp1)
76
77     u.Bouton.clicked.connect(calcul)
78
79     u.Bouton.clicked.connect(Fermer)
80
81     u.Bouton.clicked.connect(Determiner)
82
83     u.Bouton.clicked.connect(Afficher)
84
85     u.Bouton.clicked.connect(lancerexp1)
86
87     u.Bouton.clicked.connect(calcul)
88
89     u.Bouton.clicked.connect(Fermer)
90
91     u.Bouton.clicked.connect(Determiner)

```

Source Console Object

Usage

Here you can see the usage of the calculator. It can perform various arithmetic operations like PGCD, PPCM, Fact_Premier, Premier, Fact, Determiner, and Afficher.

Arithmétique

Calcul Arithmétique

Déterminer

Nombre premier

11

Calculer

Fermer

Nombre premier

In [1]: runfile('F:/FS/IHM/Exp 1')

Interface graphique pour Python PYQT pour Qt : DataBase – SQLite3

PyQT :: Liste Etudiants M2I & MQL

ID:	Numero identifiant Unique
Nom:	Nom de la personne
Prenom:	Prénom de la personne

Charger les données Insérer Supprimer

ID	NOM	PRENOM
1 200	ElHamdouni	Sohaya
2 201	Bouziki	Abderrahman
3 202	Bouguhe	Stephane
4 203	Baaza	Wafae



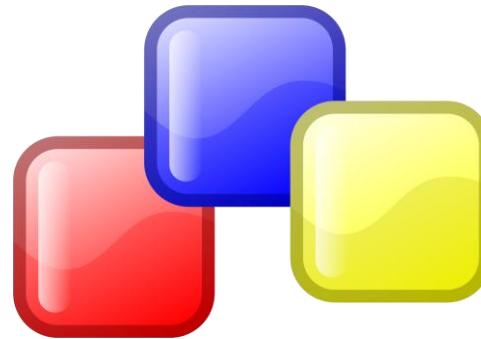
wxPython
The GUI Toolkit for Python

Interface graphique pour Python

wxPython pour wxWidget

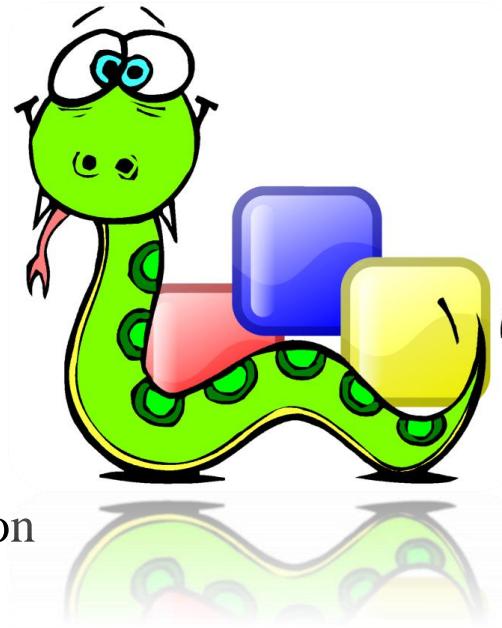
Interface graphique pour Python **wxPython : Module wxWidgets**

- ✓ **wxWidgets** est une bibliothèque multi-plateforme permettant à l'origine de créer des applications graphiques.
- ✓ C'est une bibliothèque **Opensource**.
- ✓ Sa richesse apporte des outils permettant d'aborder la quasi-totalité des domaines utiles en programmation :
 - ✓ Un système d'évènements très performant.
 - ✓ Contrôles principaux tels que menus, boutons, zones de texte et/ou de saisie (déroulantes ou non), cases à cocher, boutons radio, zones de liste, d' arborescence, onglets, barres d'outils, barres de progression, ...
 - ✓ Un système de sizers permettant la mise en place et le redimensionnement automatique des contrôles sur une fenêtre
 - ✓ Le support de fonctionnalités avancées telles que le copier/coller, les fichiers de configuration (classiques ou xml), le multi-threading, l'impression, l'internationalisation, les réseaux, ...



Interface graphique pour Python **wxPython : About !?**

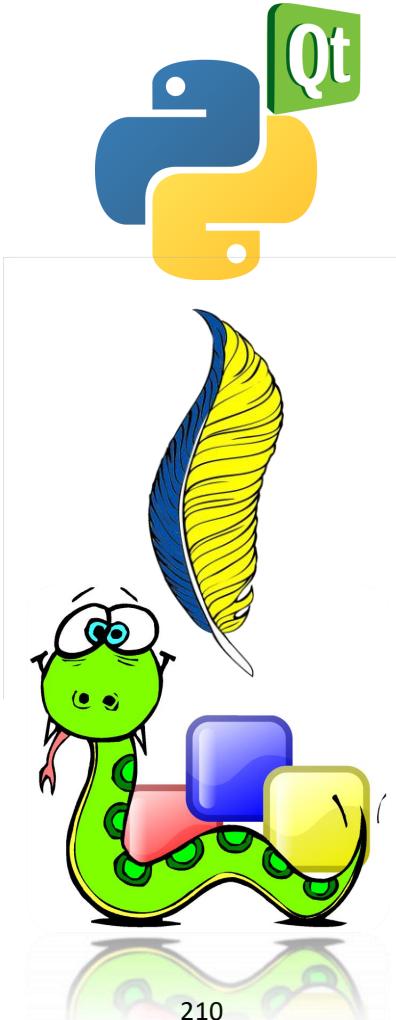
- ✓ **wxPython** est un wrapper autour de **wxWidget**, édité par *Robin Dunn*.
- ✓ **wxPython** est une bibliothèque d'interface graphique multiplateforme pour la création d'applications de bureau en langage de programmation python
- ✓ Avantages et inconvénients de **wxPython** :
 - ✓ **wxPython** a une grande bibliothèque de widgets pour les applications GUI
 - ✓ **wxPython** est free, open source et avec des licences permissives qui permettent son utilisation dans des produits commerciaux ainsi que dans freeware or sharewares.
 - ✓ **wxPython** dispose des constructeurs d'interface utilisateur pour la conception de votre application comme **wxDesigner** et **wxFombuilder**.
 - ✓ **wxPython** est très flexible



Interface graphique pour Python

wxPython : # PyQt # Tkinter

- ✓ **PyQt** est probablement le plus puissant, mais sa licence est compliquée. Si vous souhaitez vendre votre logiciel, vous devrez vous assurer d'avoir bien compris la licence QT ou être prêt à payer cher.
- ✓ Avec **Tkinter**, vous devrez créer votre interface graphique en tapant du code. Pourtant, **Tkinter** est plus facile à apprendre.
- ✓ **wxPython** pourrait être un bon compromis entre les deux. Il est gratuit, open source avec des licences permissives qui permettent son utilisation dans des produits commerciaux ainsi que dans des logiciels gratuits ou des partagiciels.



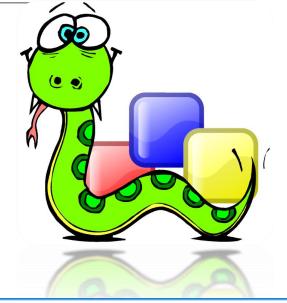
Let's start with the wxPython >>>

Interface graphique pour Python

wxPython : Install

- ✓ wxPython est une bibliothèque graphique portable, pour l'installer :

pip install wxpython



```
C:\WINDOWS\system32\cmd.exe
Microsoft Windows [version 10.0.16299.1127]
(c) 2017 Microsoft Corporation. Tous droits réservés.

(base) C:\Users\User>pip install wxpython
Collecting wxpython
  Downloading wxPython-4.1.1-cp39-cp39-win_amd64.whl (18.1 MB)
    |██████████| 18.1 MB 254 kB/s

Requirement already satisfied: numpy in c:\users\user\anaconda3\lib\site-packages (from wxpython) (1.20.3)
Requirement already satisfied: pillow in c:\users\user\anaconda3\lib\site-packages (from wxpython) (8.4.0)
Requirement already satisfied: six in c:\users\user\anaconda3\lib\site-packages (from wxpython) (1.16.0)
Installing collected packages: wxpython
Successfully installed wxpython-4.1.1
```



Interface graphique pour Python

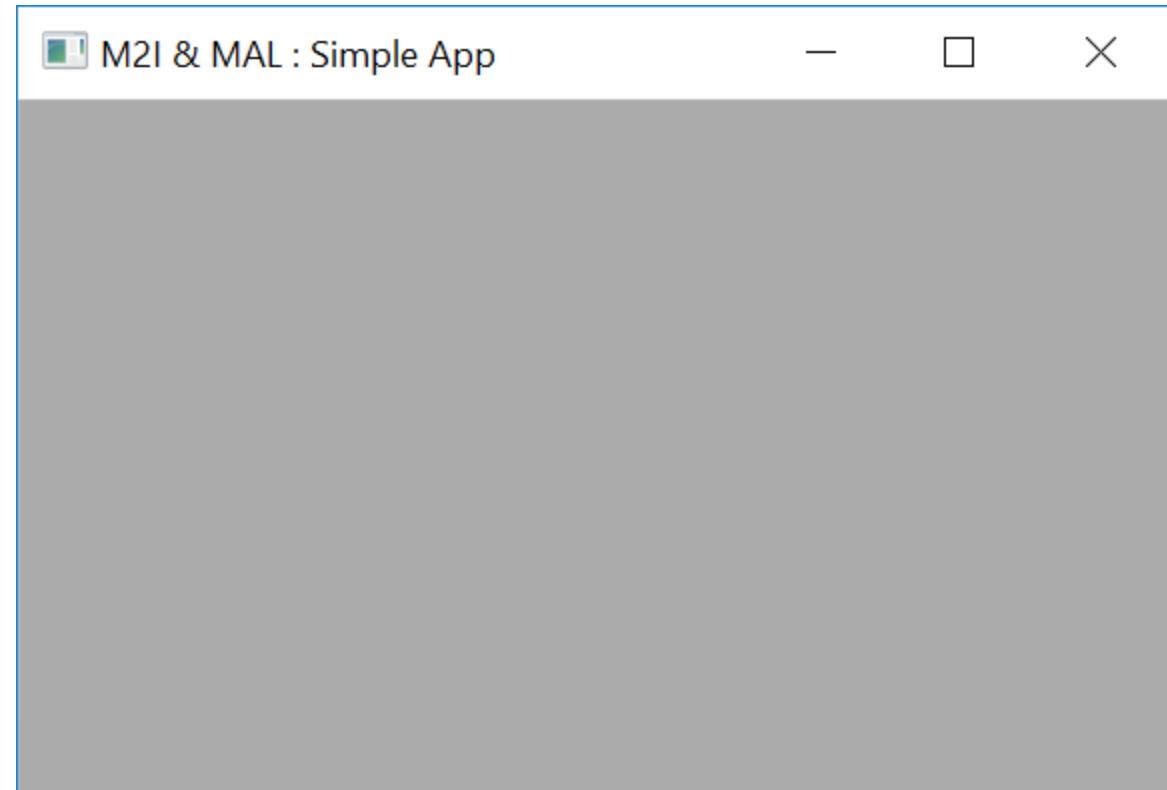
wxPython : Simple Window

```
import wx

app = wx.App()

frame = wx.Frame(None, title='M2I & MAL : Simple App')
frame.Show()

app.MainLoop()
```





Interface graphique pour Python wxPython : Size Window/Frame

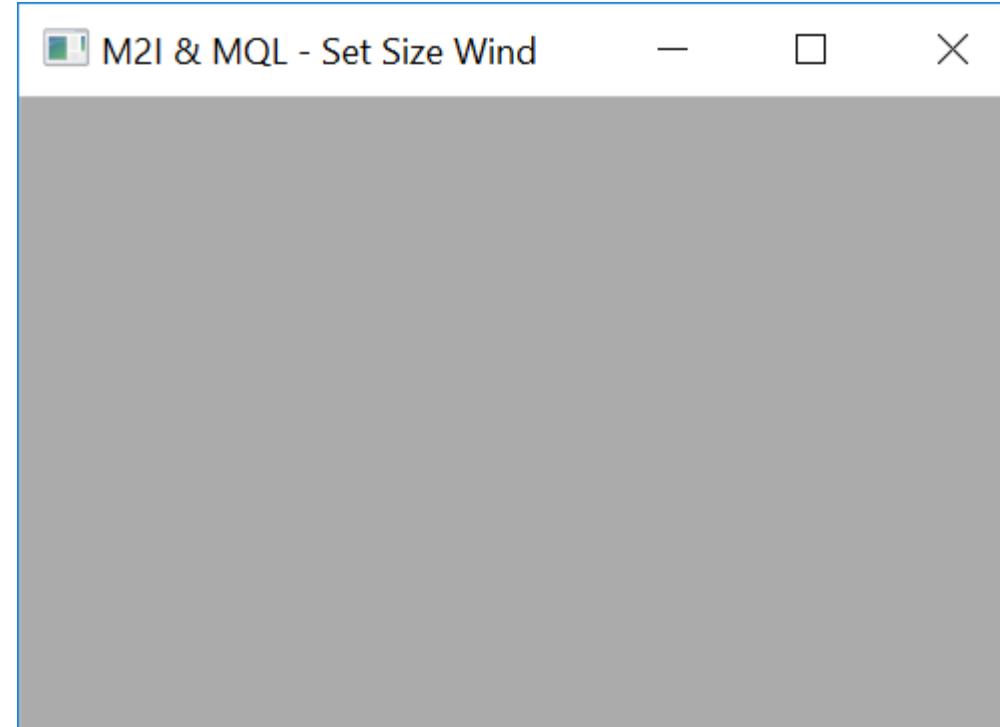
```
import wx

class Example(wx.Frame):
    def __init__(self, parent, title):
        super(Example, self).__init__(parent, title=title,
                                     size=(350, 250))

    def main():

        app = wx.App()
        ex = Example(None, title='M2I & MQL - Set Size Wind')
        ex.Show()
        app.MainLoop()

if __name__ == '__main__':
    main()
```





Interface graphique pour Python

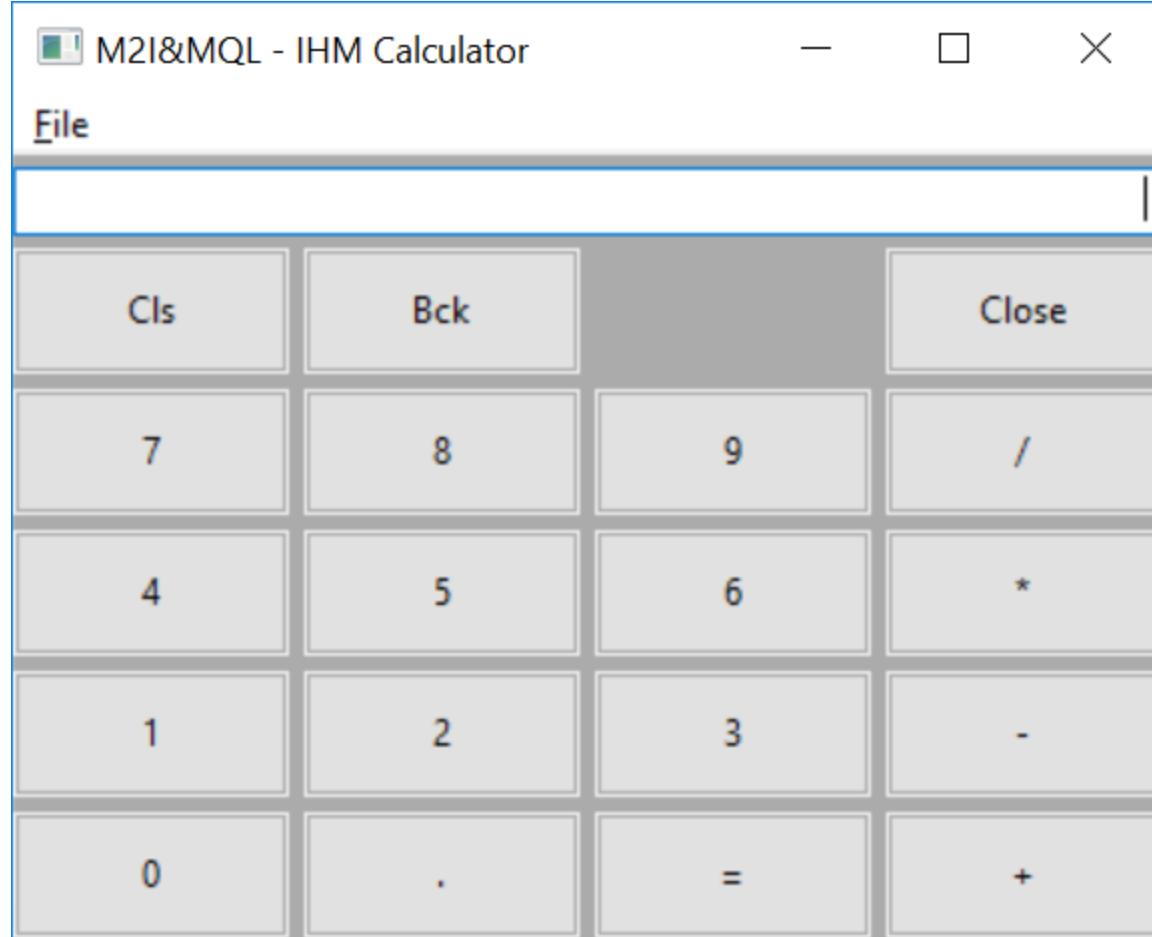
wxPython : Position Window/Frame

Method	Description
Move	Move the window to the given position
MoveXY	Move the window to the given position
SetPosition	Set the position of the window
SetPositionSize	Set the size of the window
def main():	
app = wx.App()	
ex = Example(None, title='M2I & MQL - Moving Wind')	
ex.Show()	
app.MainLoop()	
if __name__ == '__main__':	
main()	



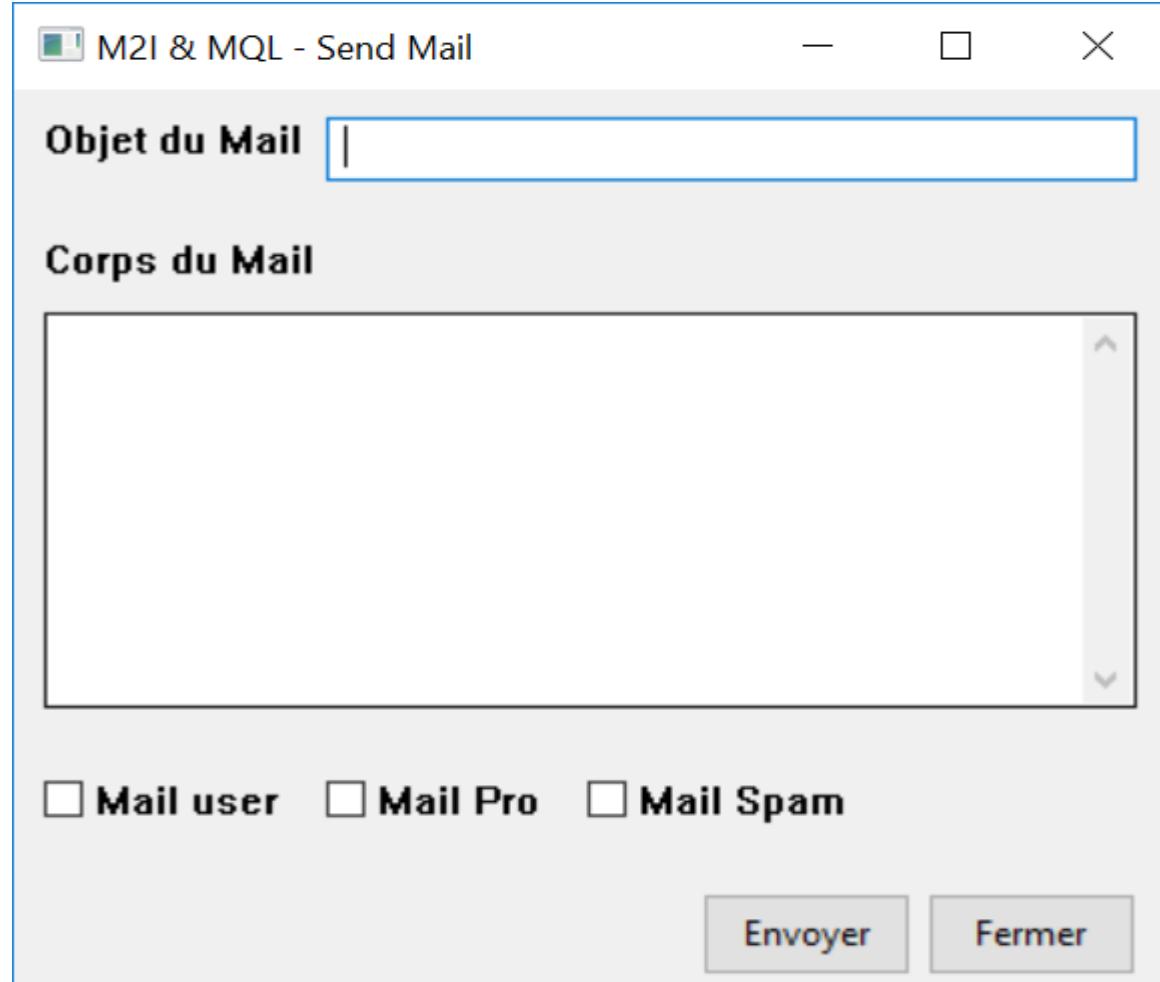


Interface graphique pour Python wxPython : Layout management 1



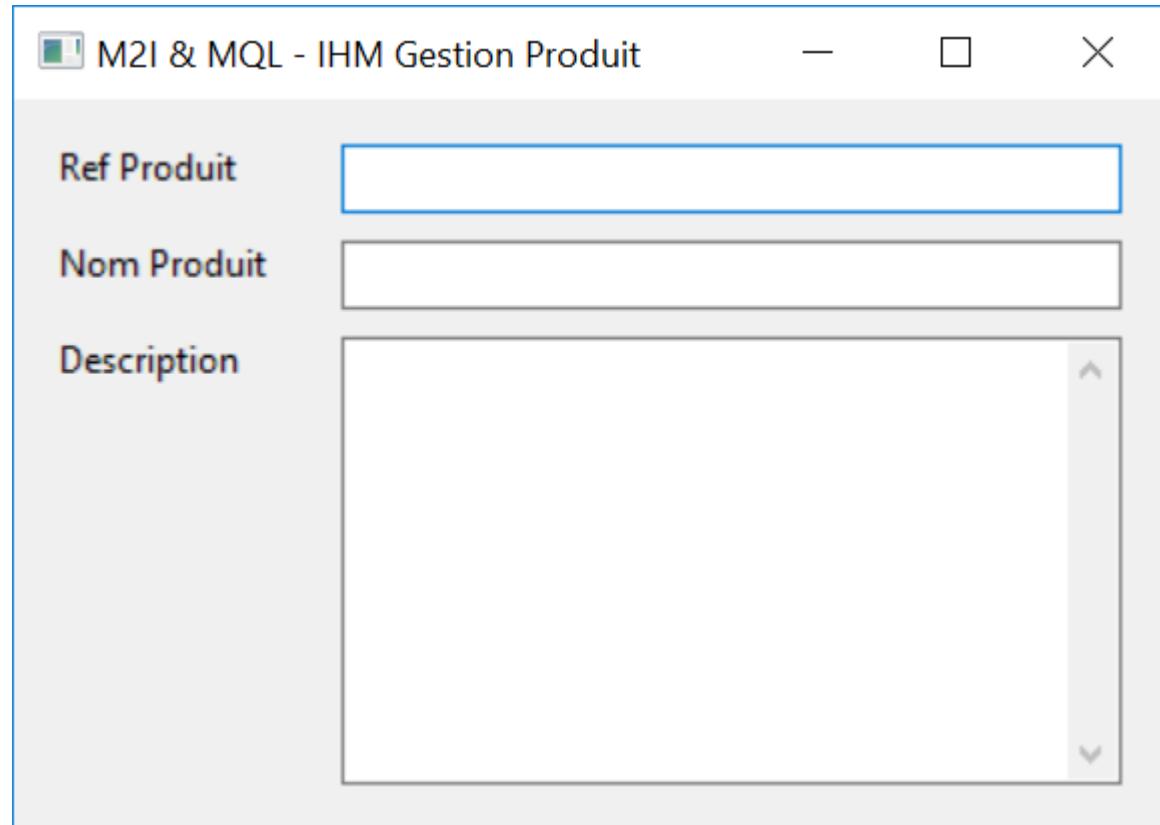


Interface graphique pour Python wxPython : Layout management 2





Interface graphique pour Python wxPython : Layout management 3

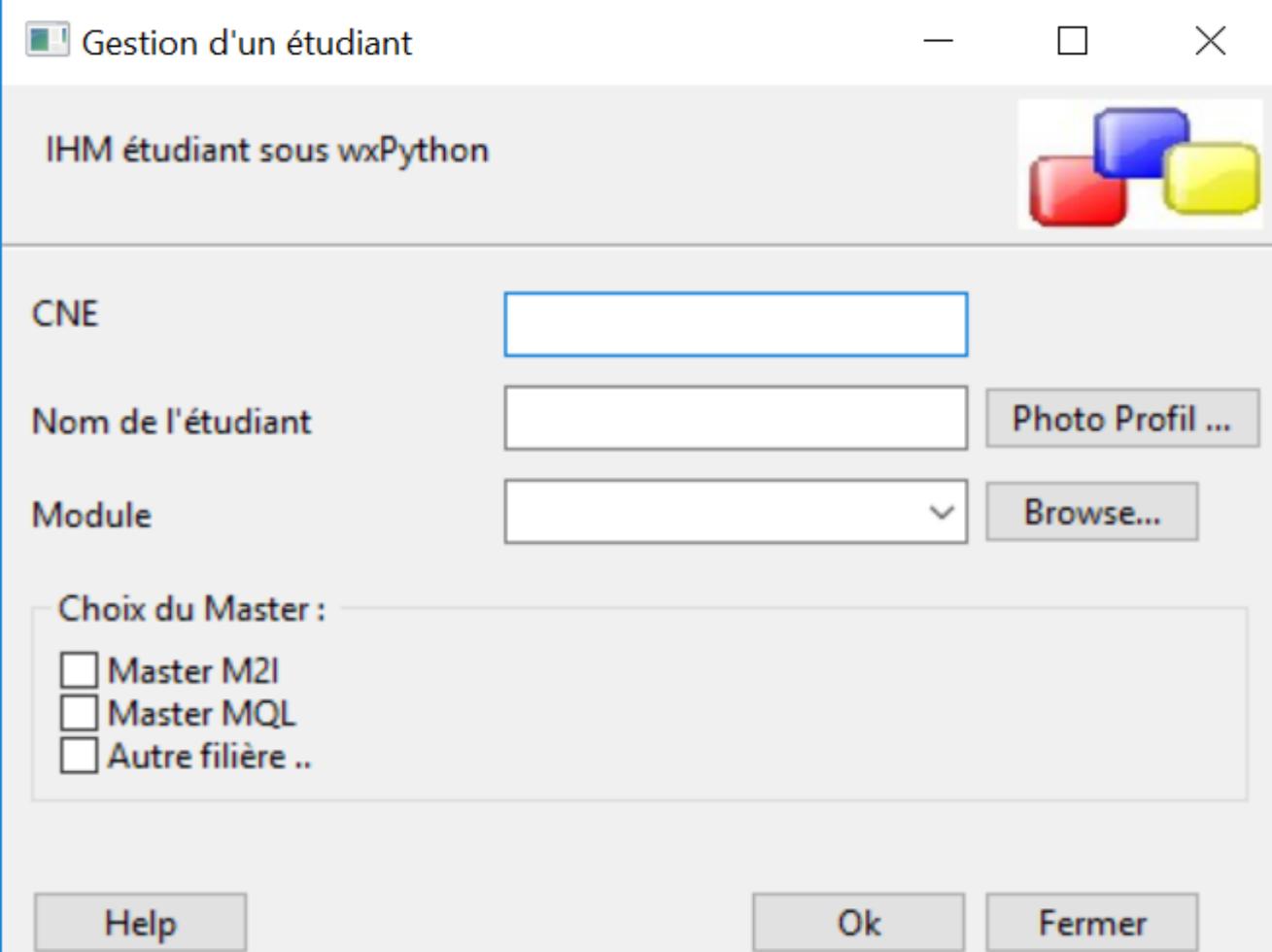




Interface graphique pour Python wxPython : Layout management 4

Gestion d'un étudiant

IHM étudiant sous wxPython



CNE

Nom de l'étudiant

Module

Choix du Master :

Master M2I
 Master MQL
 Autre filière ..

Help Ok Fermer

Interface graphique pour Python

wxPython : Events

- ✓ **Event loop** est une construction de programmation qui attend et distribue des événements ou des messages dans un programme.
- ✓ **Event loop** recherche à plusieurs reprises des événements à traiter. Un répartiteur est un processus qui mappe des événements à *event handlers*.
- ✓ **Event handlers** sont des méthodes qui réagissent aux événements.
- ✓ **Event object** est un objet associé à un **event**. Il s'agit généralement de la fenêtre ou d'un composant de fenêtre.
- ✓ **Event type** est un événement unique qui a été généré.
- ✓ **Event binder** est un objet qui lie un type d'événement à un **event handler**.

Interface graphique pour Python

wxPython : Test Events >>>

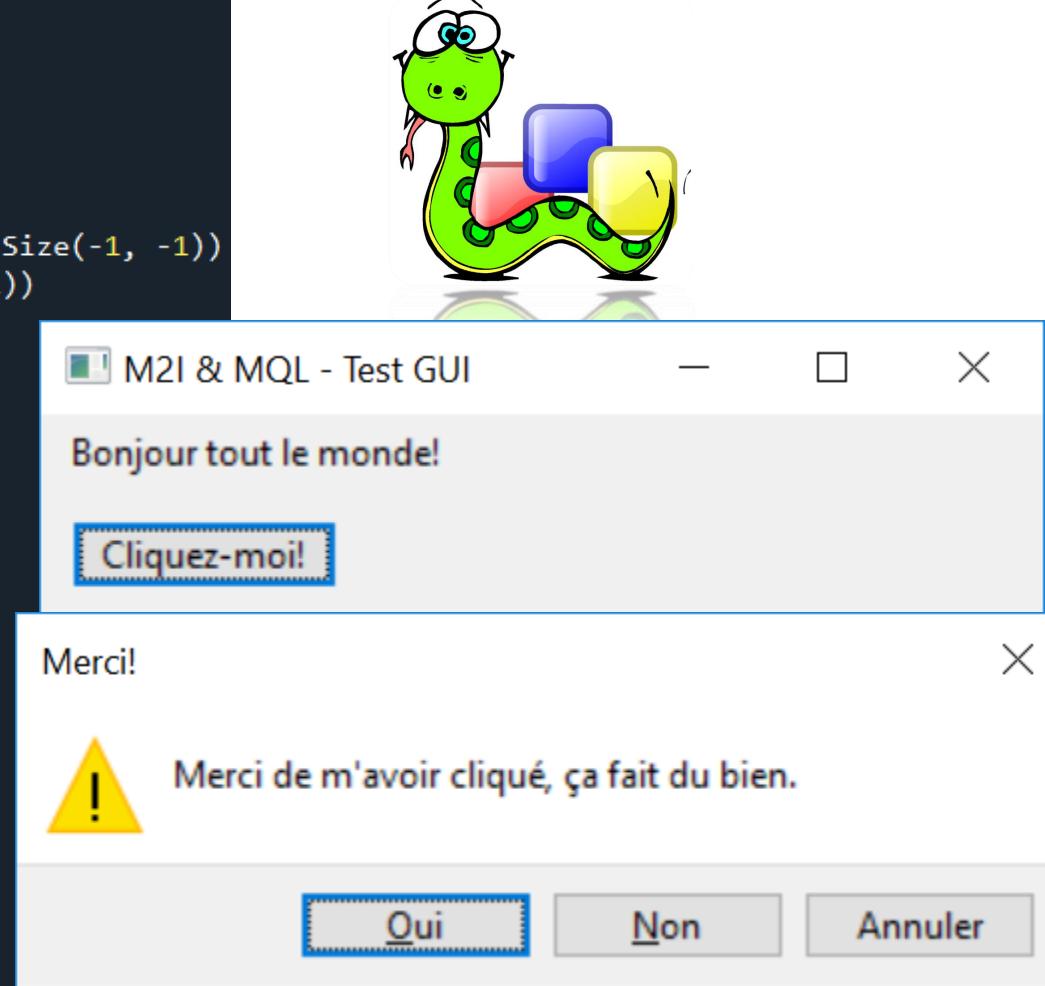
```
import wx

class TestFrame(wx.Frame):
    def __init__(self, parent, ID, title):
        wx.Frame.__init__(self, parent, -1, title, pos=(-1, -1), size=(350, 150))
        panel = wx.Panel(self, -1)
        texte = wx.StaticText(panel, -1, "Bonjour tout le monde!", wx.Point(10, 5), wx.Size(-1, -1))
        bouton = wx.Button(panel, -1, "Cliquez-moi!", wx.Point(10, 35), wx.Size(-1, -1))
        self.Bind(wx.EVT_BUTTON, self.creerDiag, bouton)

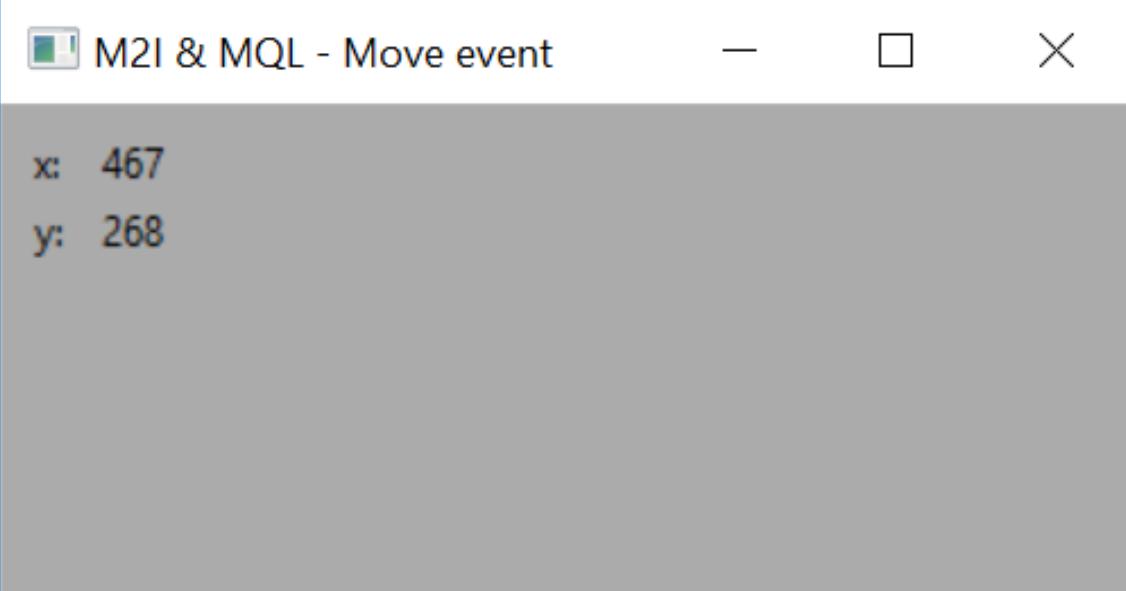
    def creerDiag(self, event):
        dlg = wx.MessageDialog(self, "Merci de m'avoir cliqué, ça fait du bien.", "Merci!", wx.ICON_EXCLAMATION | wx.YES_NO | wx.CANCEL)
        dlg.ShowModal()
        dlg.Destroy()

class TestApp(wx.App):
    def OnInit(self):
        frame = TestFrame(None, -1, "M2I & MQL - Test GUI")
        self.SetTopWindow(frame)
        frame.Show(True)
        return True

if __name__ == '__main__':
    app = TestApp(0)
    app.MainLoop()
```

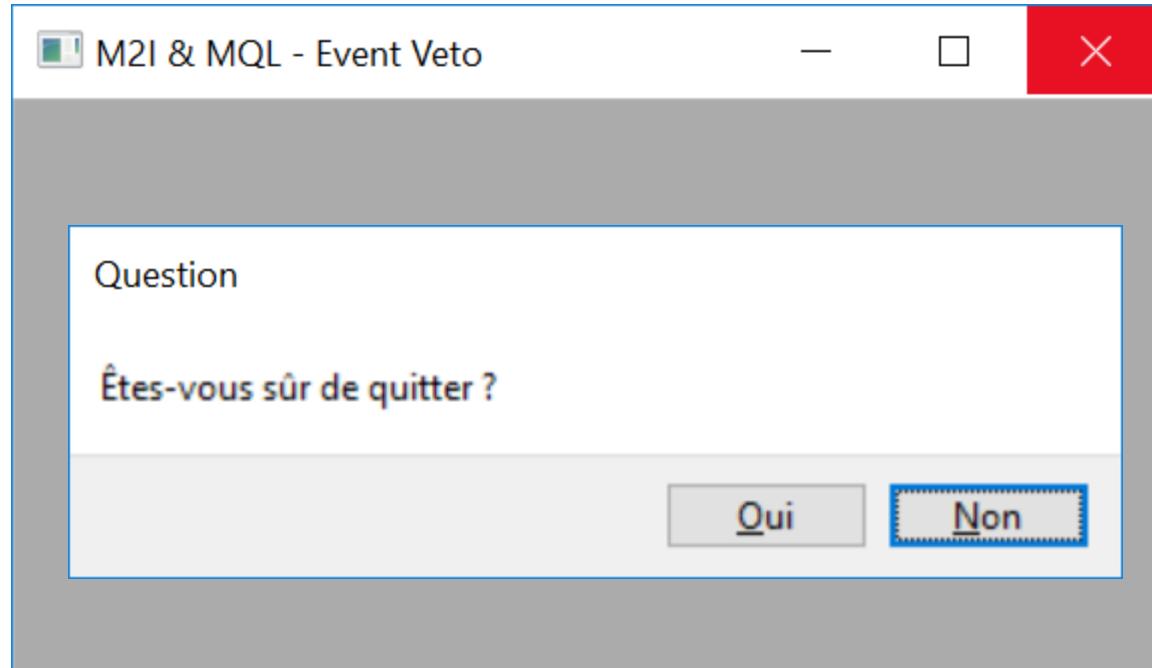


Interface graphique pour Python wxPython : MoveEvent



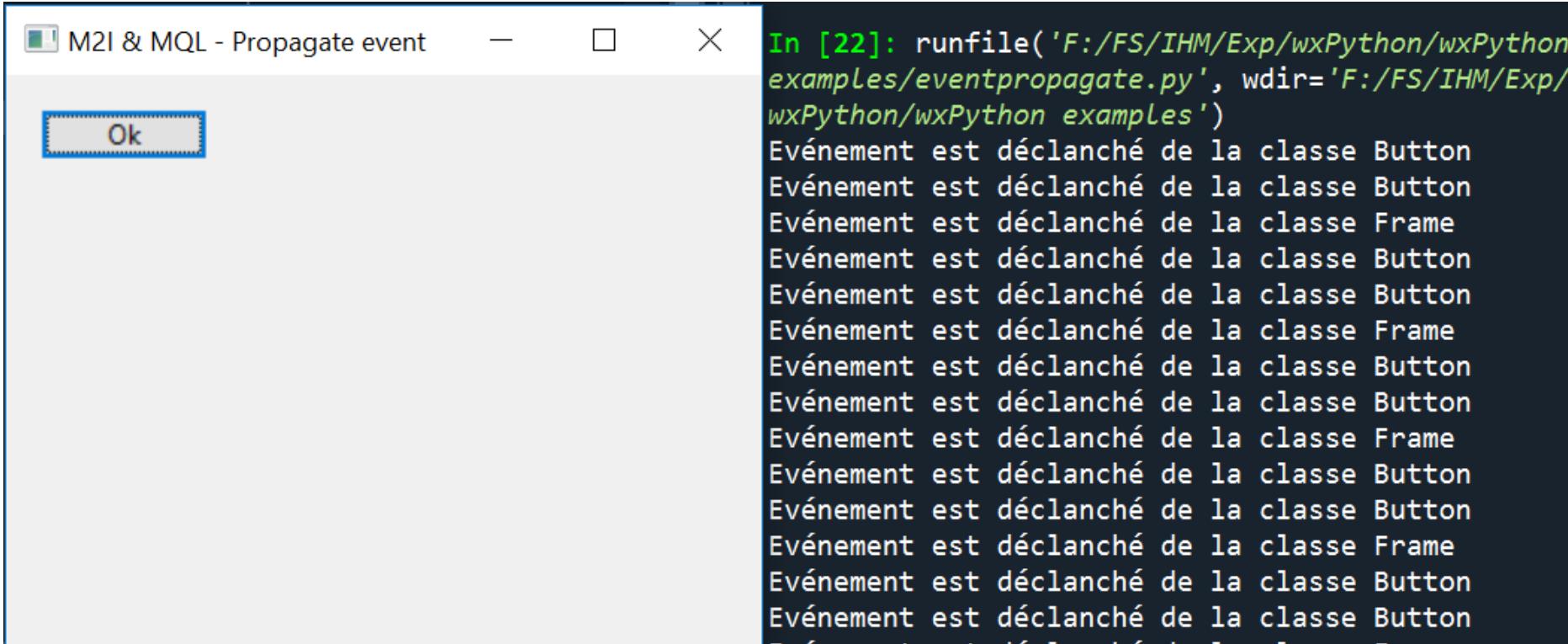
```
self.Bind(wx.EVT_MOVE, self.OnMove)
def OnMove(self, e):
    x, y = eGetPosition()
    self.st1.SetLabel(str(x))
    self.st2.SetLabel(str(y))
```

Interface graphique pour Python wxPython : Vetoing events



```
self.Bind(wx.EVT_CLOSE, self.OnCloseWindow)
def OnCloseWindow(self, e):
    dial = wx.MessageDialog(None, 'Êtes-vous sûr de quitter ?', 'Question',
                           wx.YES_NO | wx.NO_DEFAULT | wx.ICON_QUESTION)
```

Interface graphique pour Python wxPython : Propagation events

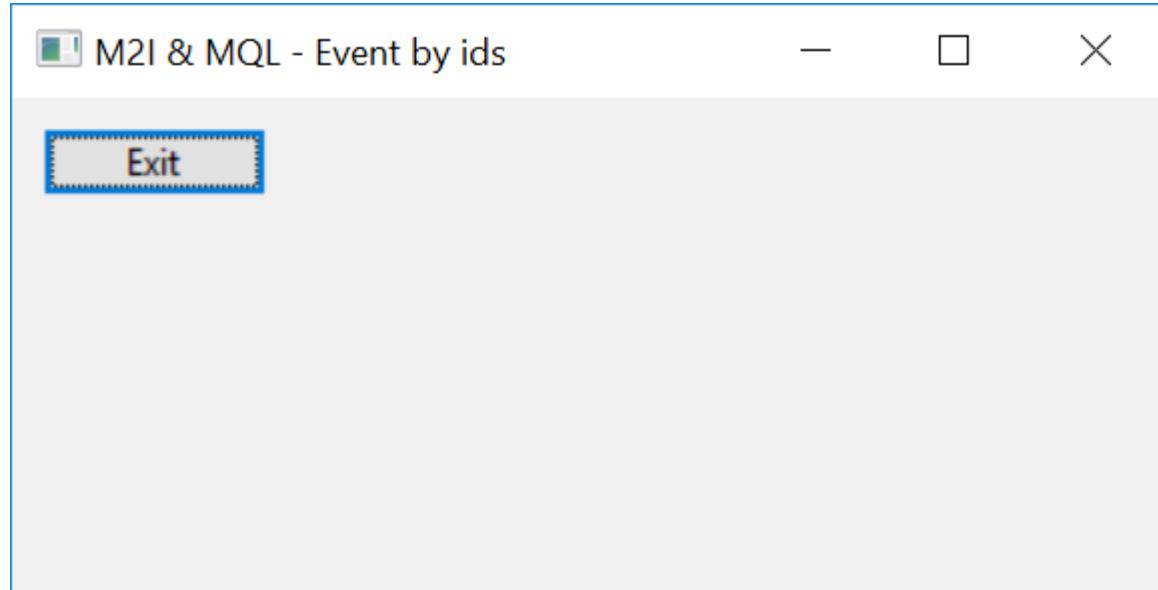


In [22]: runfile('F:/FS/IHM/Exp/wxPython/wxPython examples/eventpropagate.py', wdir='F:/FS/IHM/Exp/wxPython/wxPython examples')

Evénement est déclanché de la classe Button
Evénement est déclanché de la classe Button
Evénement est déclanché de la classe Frame
Evénement est déclanché de la classe Button
Evénement est déclanché de la classe Button
Evénement est déclanché de la classe Frame
Evénement est déclanché de la classe Button
Evénement est déclanché de la classe Button
Evénement est déclanché de la classe Frame
Evénement est déclanché de la classe Button
Evénement est déclanché de la classe Button
Evénement est déclanché de la classe Frame
Evénement est déclanché de la classe Button
Evénement est déclanché de la classe Button
Evénement est déclanché de la classe Button

```
        self.Bind(wx.EVT_BUTTON, self.OnButtonClicked)
def OnButtonClicked(self, e):
    print('Evénement est déclanché de La classe Button')
```

Interface graphique pour Python wxPython : Events with IDs



```
exitButton = wx.Button(pnl, wx.ID_ANY, 'Exit', (10, 10))

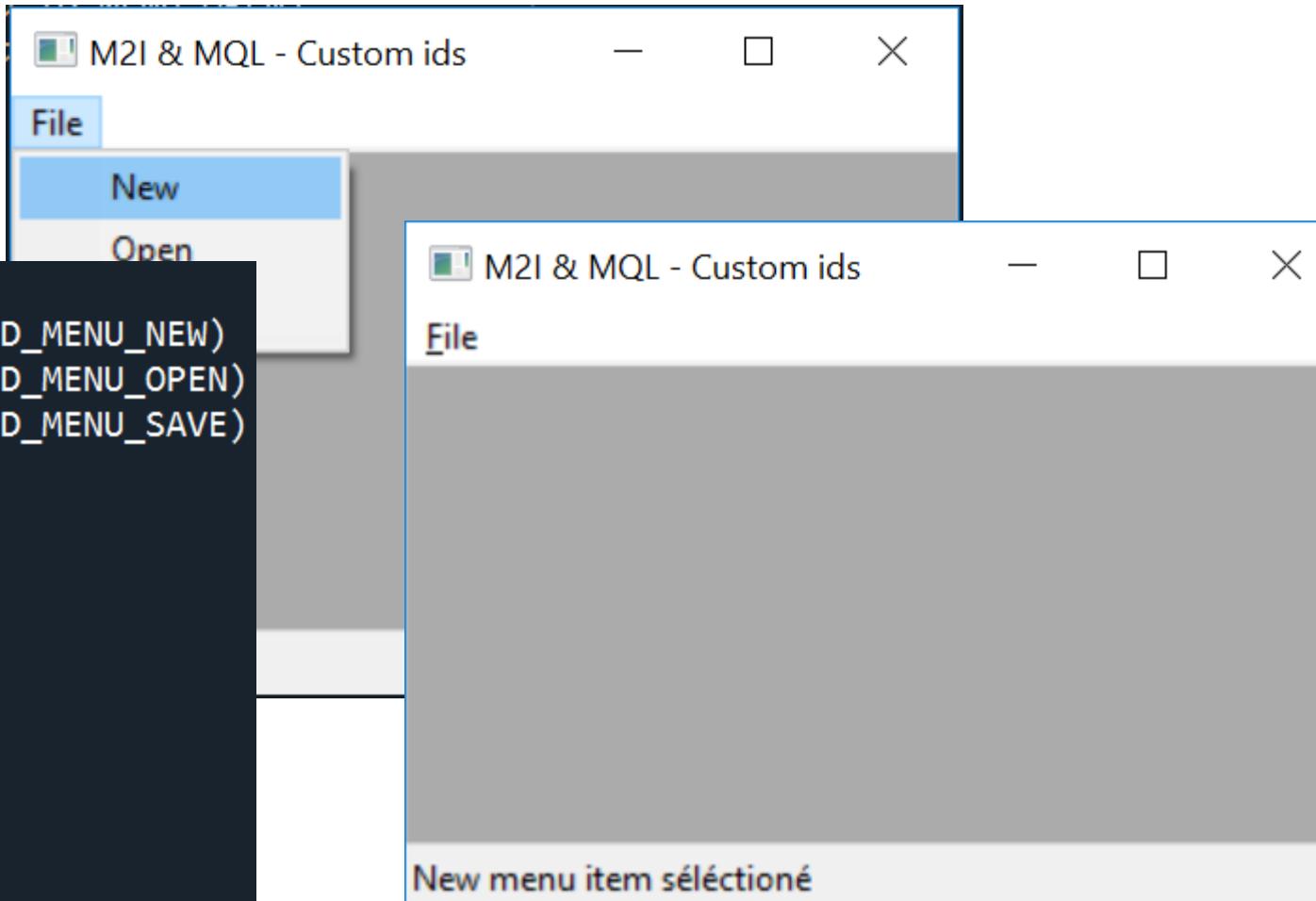
self.Bind(wx.EVT_BUTTON, self.OnExit, id=exitButton.GetId())
```

Interface graphique pour Python wxPython : ID d'événement personnalisés

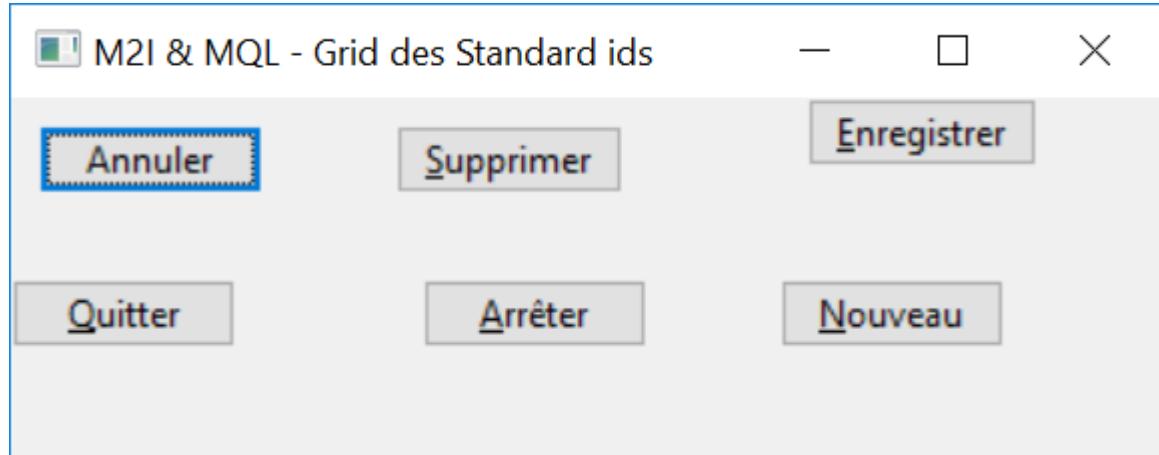
```

self.SetMenuBar(mb)
self.Bind(wx.EVT_MENU, self.DisplayMessage, id=ID_MENU_NEW)
self.Bind(wx.EVT_MENU, self.DisplayMessage, id=ID_MENU_OPEN)
self.Bind(wx.EVT_MENU, self.DisplayMessage, id=ID_MENU_SAVE)
def DisplayMessage(self, e):
    sb = self.GetStatusBar()
    eid = e.GetId()
    if eid == ID_MENU_NEW:
        msg = 'New menu item sélectionné'
    elif eid == ID_MENU_OPEN:
        msg = 'Open menu item sélectionné'
    elif eid == ID_MENU_SAVE:
        msg = 'Save menu item sélectionné'
    sb.SetStatusText(msg)

```



Interface graphique pour Python wxPython : Grid des ID prédéfinis



```
grid.AddMany([(wx.Button(pnl, wx.ID_CANCEL), 0, wx.TOP | wx.LEFT, 9),
              (wx.Button(pnl, wx.ID_DELETE), 0, wx.TOP, 9),
              (wx.Button(pnl, wx.ID_SAVE), 0, wx.LEFT, 9),
              (wx.Button(pnl, wx.ID_EXIT)),
              (wx.Button(pnl, wx.ID_STOP), 0, wx.LEFT, 9),
              (wx.Button(pnl, wx.ID_NEW))])
self.Bind(wx.EVT_BUTTON, self.OnQuitApp, id=wx.ID_EXIT)
```



```

import wx

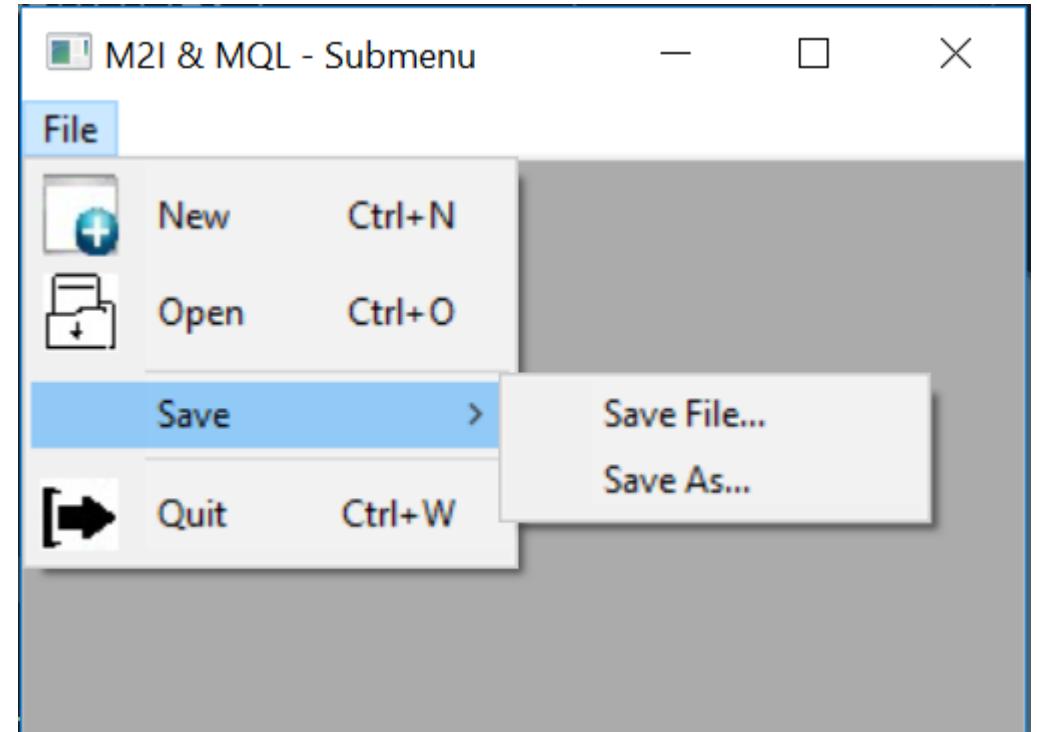
class Example(wx.Frame):
    def __init__(self, *args, **kwargs):
        super(Example, self).__init__(*args, **kwargs)
        self.InitUI()
    def InitUI(self):
        menubar = wx.MenuBar()
        fileMenu = wx.Menu()
        nmi = wx.MenuItem(fileMenu, wx.ID_NEW, '&New\tCtrl+N')
        nmi.SetBitmap(wx.Bitmap('new.jpg'))
        fileMenu.AppendItem(nmi)
        omi = wx.MenuItem(fileMenu, wx.ID_OPEN, '&Open\tCtrl+O')
        omi.SetBitmap(wx.Bitmap('open.jpg'))
        fileMenu.AppendItem(omi)
        fileMenu.AppendSeparator()
        smi = wx.Menu()
        smi.Append(wx.ID_ANY, 'Save File...')
        smi.Append(wx.ID_ANY, 'Save As...')
        fileMenu.AppendMenu(wx.ID_SAVE, 'Save', smi)
        fileMenu.AppendSeparator()
        qmi = wx.MenuItem(fileMenu, wx.ID_EXIT, '&Quit\tCtrl+W')
        qmi.SetBitmap(wx.Bitmap('quit.jpg'))
        fileMenu.AppendItem(qmi)
        self.Bind(wx.EVT_MENU, self.OnQuit, qmi)
        menubar.Append(fileMenu, '&File')
        self.SetMenuBar(menubar)
        self.SetSize((350, 250))
        selfSetTitle('M2I & MQL - Submenu')
        self.Centre()
    def OnQuit(self, e):
        self.Close()

def main():
    app = wx.App()
    ex = Example(None)
    ex.Show()
    app.MainLoop()

if __name__ == '__main__':
    main()

```

pour Python & SubMenu





Interface graphique pour Python wxPython : Multi-Windows-MDIFrame

```

import wx

class MDIFrame(wx.MDIParentFrame):
    def __init__(self):
        wx.MDIParentFrame.__init__(self, None, -1, "M2I & MQL - Multi Windows",
                                  size = (600,400))
        menu = wx.Menu()
        menu.Append(5000, "&New Window")
        menu.Append(5001, "&Exit")
        menubar = wx.MenuBar()
        menubar.Append(menu, "&File")

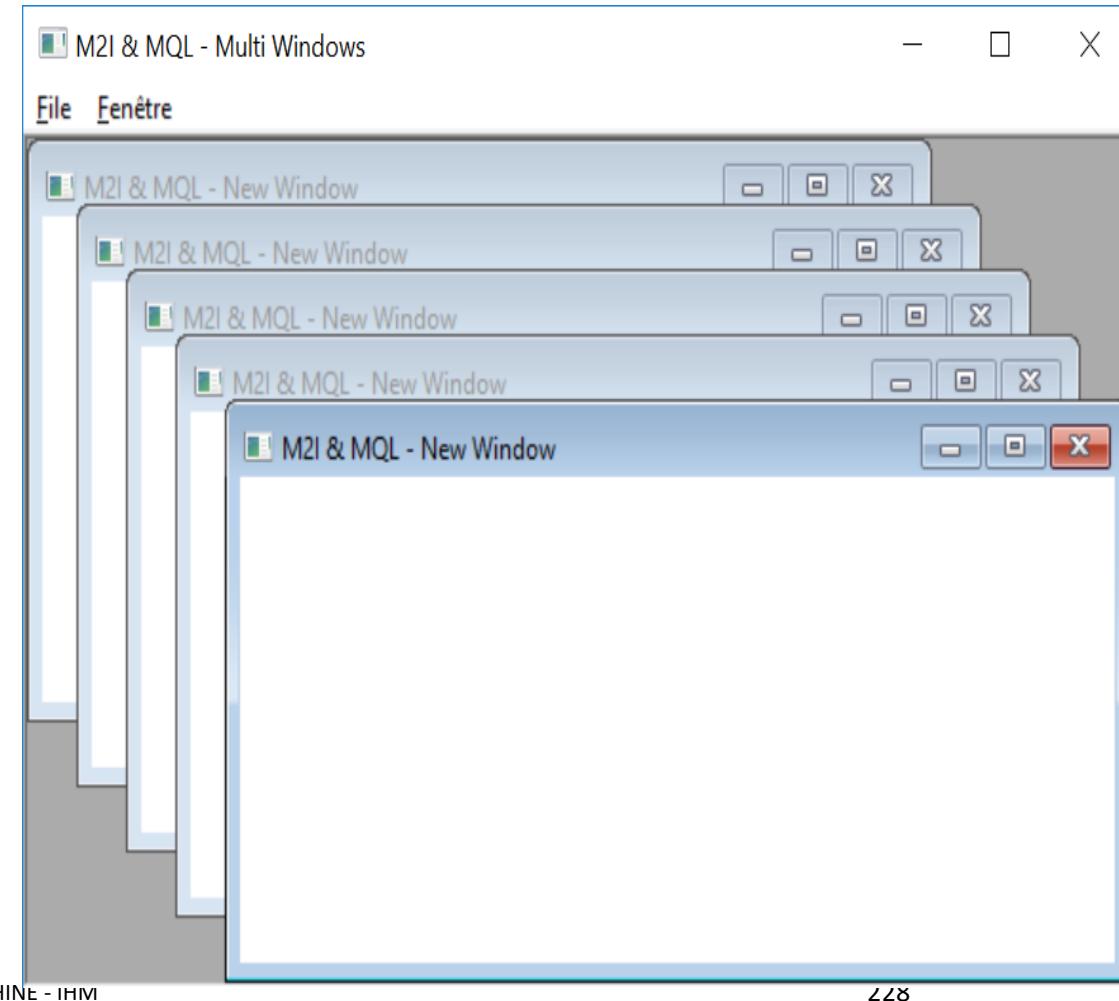
        self.SetMenuBar(menubar)
        self.Bind(wx.EVT_MENU, self.OnNewWindow, id = 5000)
        self.Bind(wx.EVT_MENU, self.OnExit, id = 5001)

    def OnExit(self, evt):
        self.Close(True)

    def OnNewWindow(self, evt):
        win = wx.MDIChildFrame(self, -1, "M2I & MQL - New Window")
        win.Show(True)

app = wx.App()
frame = MDIFrame()
frame.Show()
app.MainLoop()

```





```

import wx
import wx.aui

class Mywin(wx.Frame):

    def __init__(self, parent, title):
        super(Mywin, self).__init__(parent, title = title, size = (300,300))

        self.mgr = wx.aui.AuiManager(self)

        pnl = wx.Panel(self)
        pbox = wx.BoxSizer(wx.HORIZONTAL)
        text1 = wx.TextCtrl(pnl, -1, "ICI, vous pouvez ajouter du commentaire",
                            style = wx.NO_BORDER | wx.TE_MULTILINE)
        pbox.Add(text1, 1, flag = wx.EXPAND)
        pnl.SetSizer(pbox)

        info1 = wx.aui.AuiPaneInfo().Bottom()
        self.mgr.AddPane(pnl, info1)
        panel = wx.Panel(self)
        text2 = wx.TextCtrl(panel, size = (300,200), style =  wx.NO_BORDER
                           | wx.TE_MULTILINE)
        box = wx.BoxSizer(wx.HORIZONTAL)
        box.Add(text2, 1, flag = wx.EXPAND)

        panel.SetSizerAndFit(box)
        self.mgr.Update()

        self.Bind(wx.EVT_CLOSE, self.OnClose)
        self.Centre()
        self.Show(True)

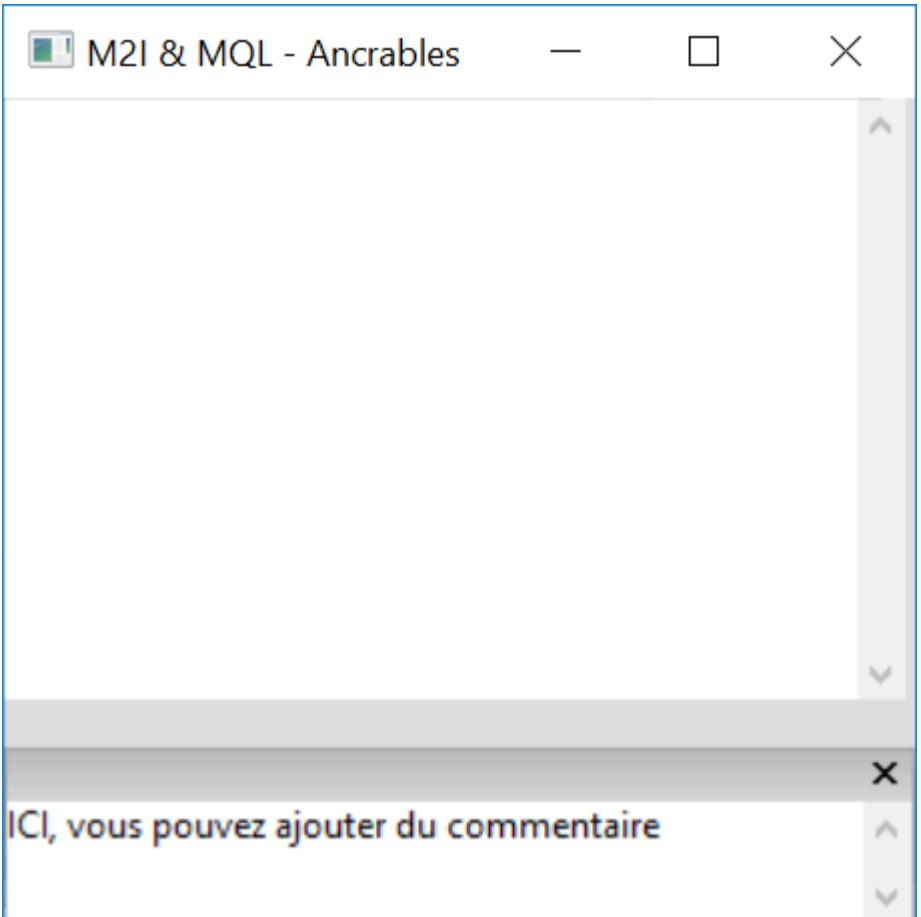
    def OnClose(self, event):
        self.mgr.UnInit()
        self.Destroy()

app = wx.App()
Mywin(None, "M2I & MQL - Ancrables ")
app.MainLoop()

```

pour Python

wx aui



Interface graphique pour Python wxPython : wxPython-Demo

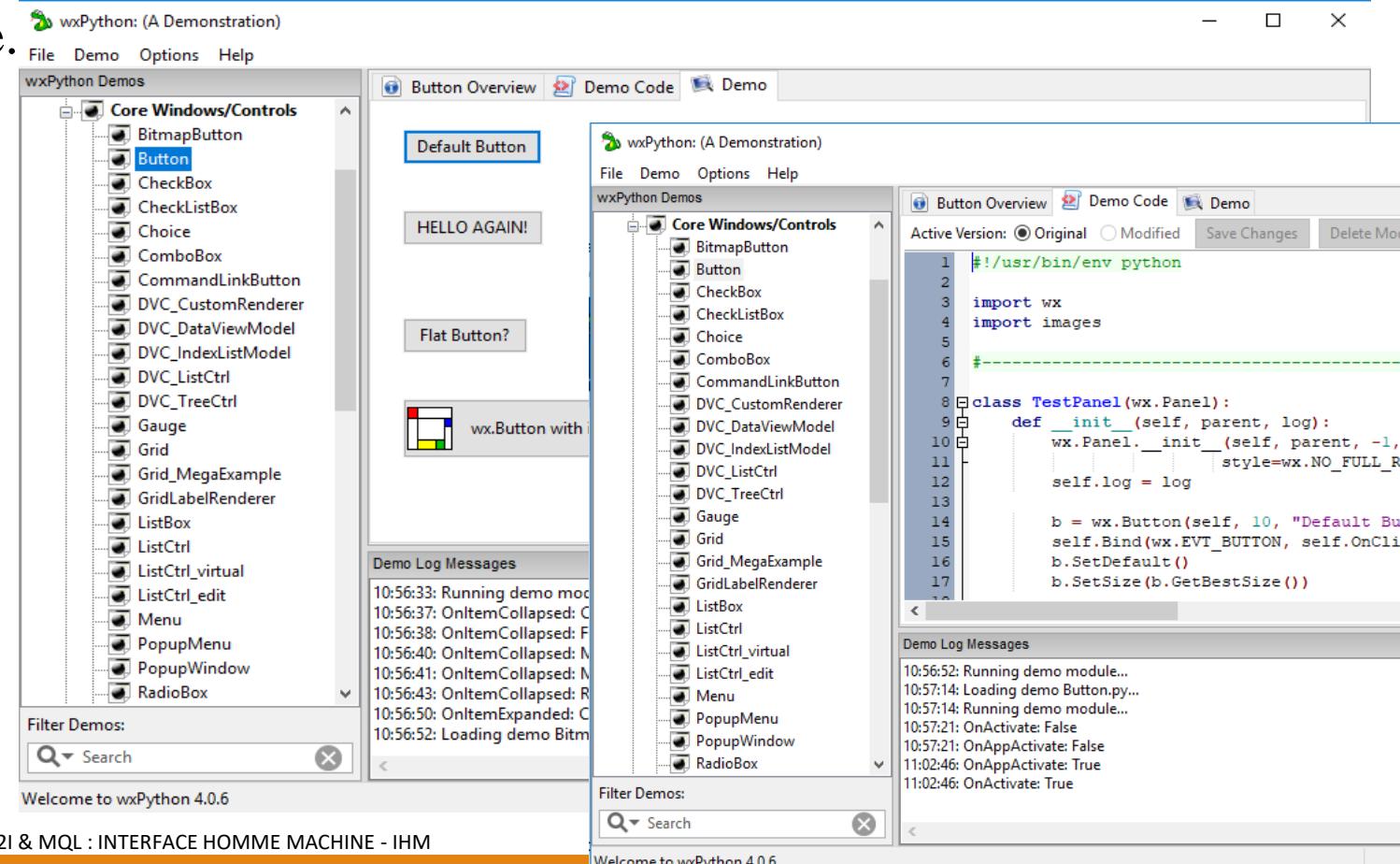
- ✓ L'un des gros intérêts de **wxPython** est l'application **wxPython-demo**, écrite en **Python** avec **wxPython** et permet de naviguer dans toutes les fonctionnalités de la bibliothèque, de voir des exemples des contrôles graphiques et d'accéder au code.

- ✓ To install it :

- ✓ Go to :

<https://github.com/wxWidgets/Phoenix>

- ✓ Téléchargez le **.zip**, puis Unzip.
 - ✓ Aller vers le dossier 'Demo' : **cd demo**
 - ✓ Run **wxPython-Demo** : **python demo.py**



Interface graphique pour Python **wxPython - Outils GUI Builder**

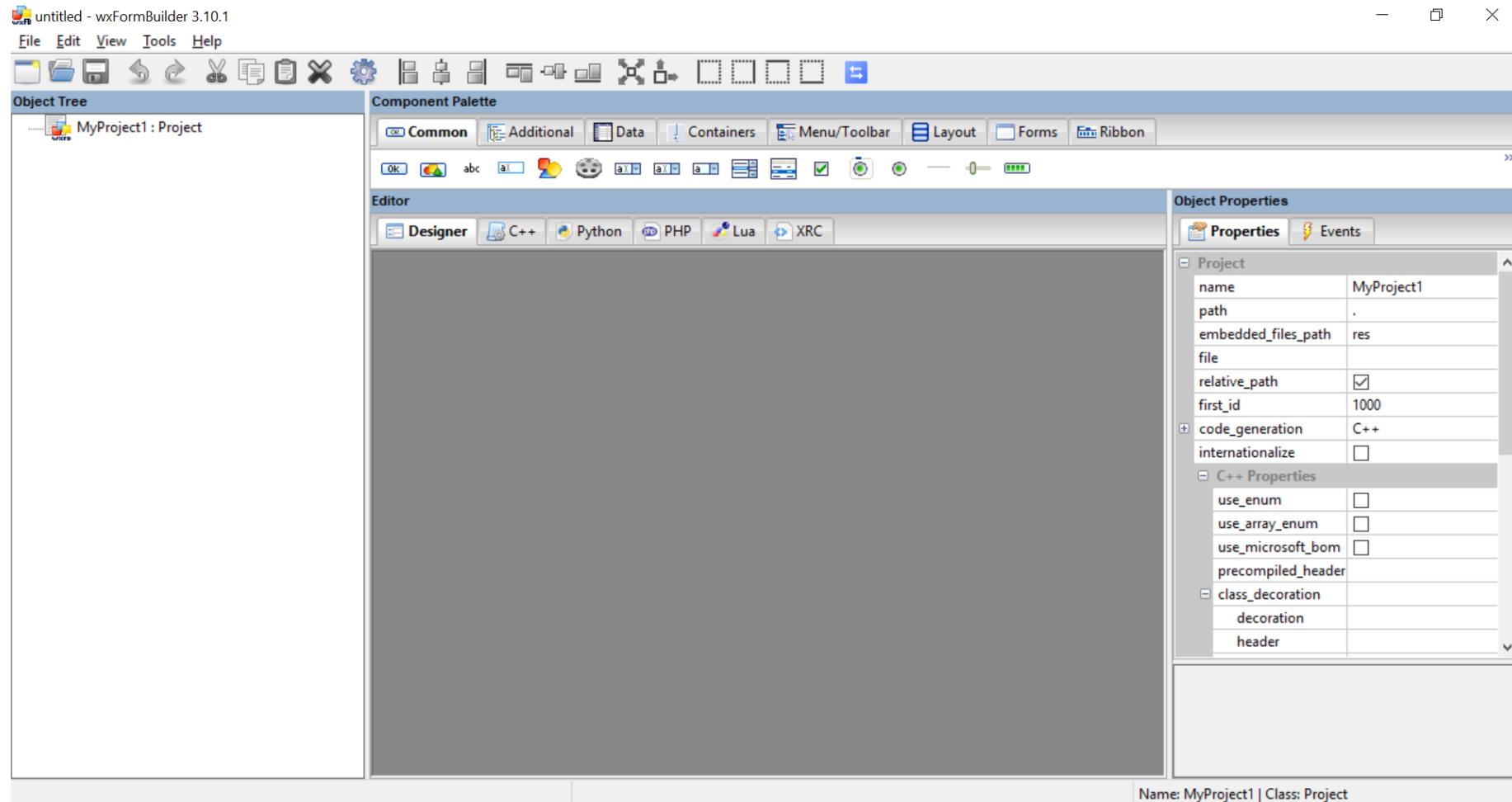
- ✓ Un outil de conception graphique est toujours pratique. De nombreux IDE développement GUI ciblés sur *wxPython* sont disponibles:
 - ✓ **wxFormBuilder**
 - ✓ wxDesigner
 - ✓ wxGlade
 - ✓ BoaConstructor
 - ✓ Gui2py
- ✓ **wxFormBuilder** est un constructeur d'interface graphique multiplateforme open source qui peut traduire la conception de l'interface graphique *wxWidget* au format Python, C++, PHP,
- ✓ **Get started with :**

<https://github.com/wxFormBuilder/wxFormBuilder>





Interface graphique pour Python wxPython - wxFormBuilder



ProjectTest - wxFormBuilder 3.10.1

File Edit View Tools Help

Object Tree

MyProjectTest : Project

MyFrameTest : Frame

- bSizer1 : wxBoxSizer
- gSizer1 : wxGridSizer
 - m_staticText2 : wxStaticText
 - m_textCtrl2 : wxTextCtrl
 - m_staticText3 : wxStaticText
 - m_textCtrl3 : wxTextCtrl
 - m_staticText4 : wxStaticText
 - m_comboBox1 : wxComboBox**
 - m_sdbSizer2 : wxStdDialogButtonSizer
- m_menuBar1 : wxMenuBar
 - m_menu1 : wxMenu
 - m_menu11 : submenu
 - m_menuItem1 : wxMenuItem
 - m_separator1 : separator
 - m_menuItem2 : wxMenuItem
 - m_menu2 : wxMenu
- m_statusBar1 : wxStatusBar

Component Palette

Common Additional Data Containers Menu/Toolbar Layout Forms Ribbon

Editor

Designer C++ Python PHP Lua XRC

M2I & MQL - Test Project

Nom

Prénom

Master Votre Master

Votre Master

Master M2I

Master MQL

Autre Master

OK Cancel

Object Properties

Properties Events

wxComboBox

name	m_comboBox1
value	Votre Master
choices	QL "Autre Master" ...
selection	-1

wxWindow

id	wxID_ANY
pos	-1; -1
size	-1; -1
minimum_size	-1; -1
maximum_size	-1; -1
font	-1; Default; ; Normal; N
fg	Window
bg	Window
window_name	
window_style	

choices
Contents of the Combo Box

Project Saved!

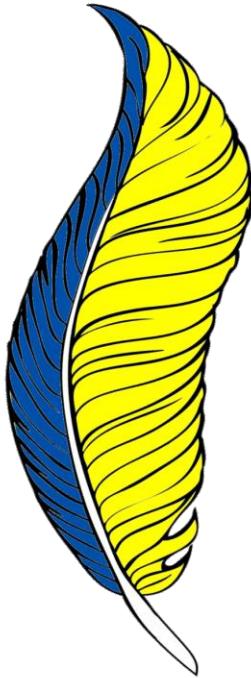
F:\FS\IHM\Exp\wxPython\wxFormBuilder\Test\ProjectTest.fbp

M2I & MQL : INTERFACE HOMME MACHINE - IHM

Name: m_comboBox1 | Class: wxComboBox

O. ABDOUN

233



Interface graphique pour Python

- ~~TKINTER POUR TK~~
- ~~PYQT POUR QT~~
- ~~WXPYTHON POUR WXWIDGETS~~
- PYGTK POUR GTK+

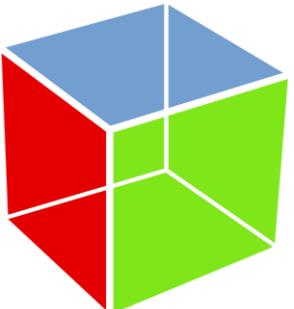


Interface graphique pour Python

PYGTK POUR GTK+

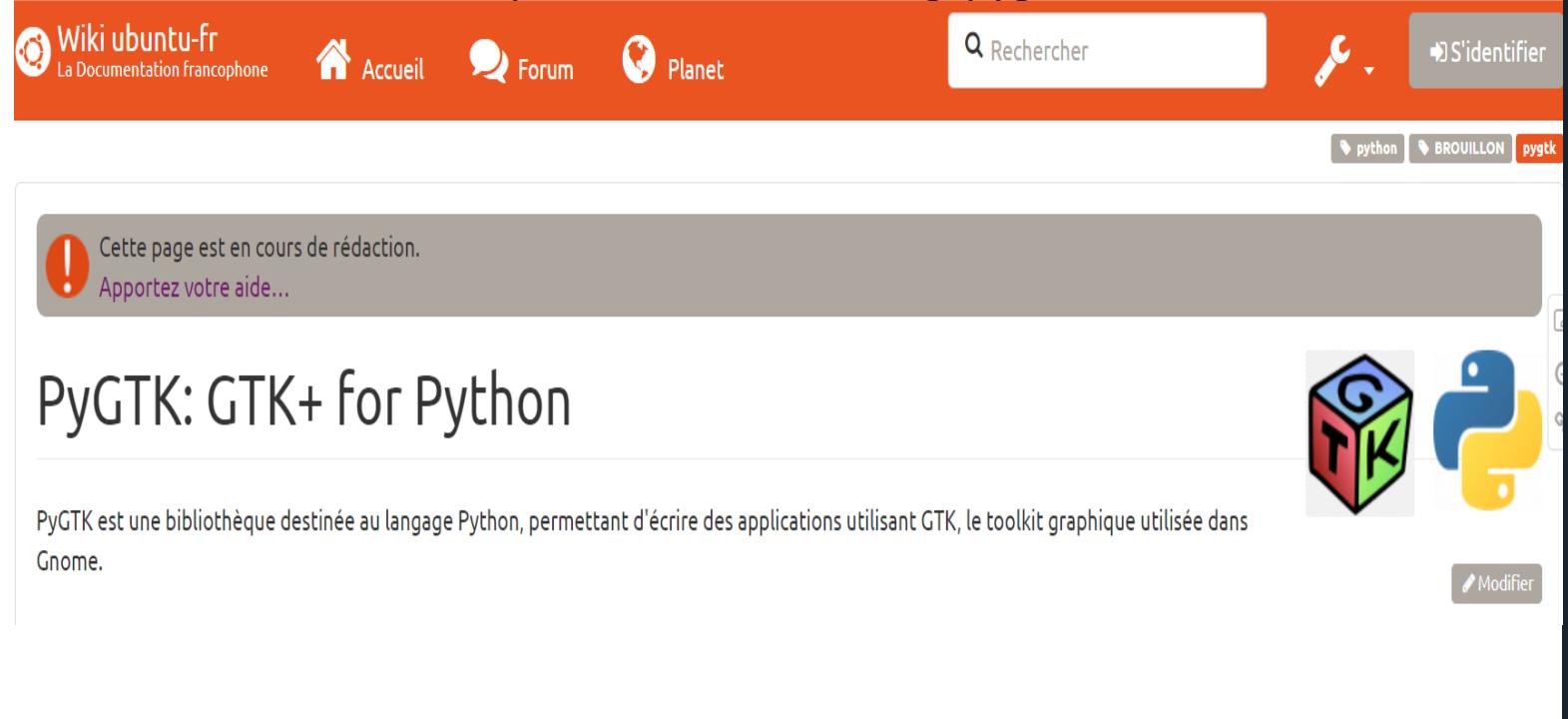
Interface graphique pour Python **pyGTK - About GTK and its bainds**

- ✓ **GTK** (GIMP Toolkit) est une bibliothèque de création d'interfaces graphiques. Elle est placée sous licence LGPL, et peut donc être utilisée pour le développement de logiciels libres, open-source, ou même commerciaux et non libres sans qu'il faille s'acquitter de quelconques droits d'auteur ou licences.
- ✓ **GTK** est principalement une interface de programmation d'application (API) orientée objet. Bien qu'écrite intégralement en C, elle est implémentée en utilisant le concept de classes et de fonctions de rappel (pointeurs sur fonctions).
- ✓ **PyGTK** est un module Python permettant la création d'interfaces graphiques utilisant la bibliothèque **GTK 2**.
- ✓ À partir de **GTK 3**, **PyGObject** remplace **PyGTK** dans ce rôle.



Interface graphique pour Python pyGTK – sous GTK2

<https://doc.ubuntu-fr.org/pygtk>



Wiki ubuntu-fr
La Documentation francophone

Accueil Forum Planet

Rechercher S'identifier

python Brouillon pygtk

Cette page est en cours de rédaction.
Apportez votre aide...

PyGTK: GTK+ for Python

PyGTK est une bibliothèque destinée au langage Python, permettant d'écrire des applications utilisant GTK, le toolkit graphique utilisée dans Gnome.

```
import gtk

def create_window():
    window = gtk.Window()
    window.set_default_size(200, 200)
    window.connect("destroy", gtk.main_quit)

    label = gtk.Label("Hello World")
    window.add(label)

    label.show()
    window.show()

create_window()
gtk.main()
```

Interface graphique pour Python PyGobject – sous GTK3



Search docs

Getting Started

Changelog

Bug Tracker / Git / Source

User Guide

Development Guide

Packaging Guide

Maintainer Guide

Further Resources

Contact

<https://pygobject.readthedocs.io/en/latest/>
Docs » Overview

[View page source](#)



PyGObject

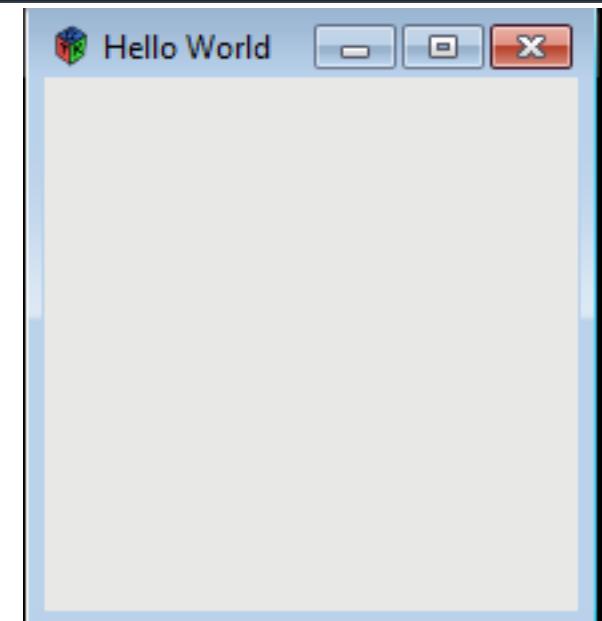
PyGObject is a Python package which provides bindings for [GObject](#) based libraries such as [GTK](#), [GStreamer](#), [WebKitGTK](#), [GLib](#), [GIO](#) and many more.

It supports Linux, Windows and macOS and works with [Python 3.7+](#) and [PyPy3](#). PyGObject, including this documentation, is licensed under the [LGPLv2.1+](#).

If you want to write a Python application for [GNOME](#) or a Python GUI application using GTK, then PyGObject is the way to go. For more information on specific libraries check out the "[Python GTK 3 Tutorial](#)" and the "[Python GI API Reference](#)".

```
import gi
gi.require_version("Gtk", "3.0")
from gi.repository import Gtk

window = Gtk.Window(title="Hello World")
window.show()
window.connect("destroy", Gtk.main_quit)
Gtk.main()
```





Interface graphique pour Python

Tkinter # PyQt # wxPython # PyGTK



- ✓ **wxPython** et **pyQt/pyside** sont probablement les plus faciles à utiliser sur toutes les plates-formes et seront parfaitement adaptés à la plupart des systèmes d'exploitation.
- ✓ Entre **wxPython** et **pyQt/pySide** : ils ont même avantage de l'outil de conception, mais il faut voir la question de Licence.
- ✓ **Tkinter** est soooo Light., surtout conçu pour tout développement mobile et sur toutes les plateformes.
- ✓ Éviter **PyGTK** à cause d'un problème de compatibilité multiplateforme. **GTK2** a une fuite de mémoire horrible sous **Windows**.

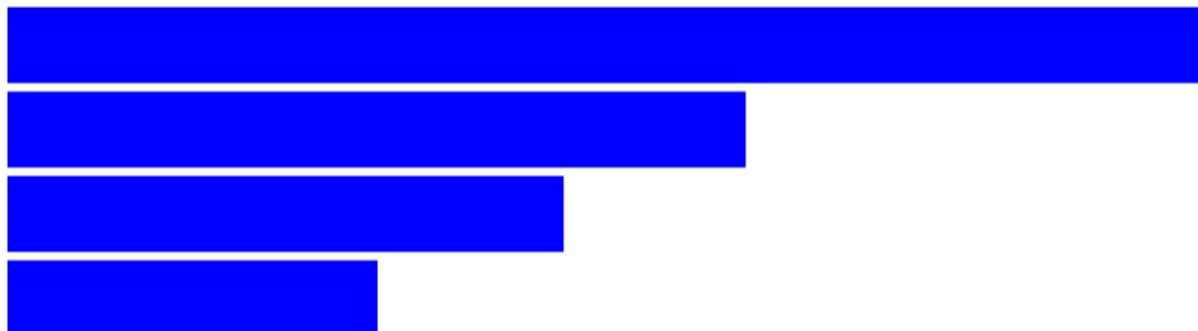
Des GUI en Python, c'est avec...

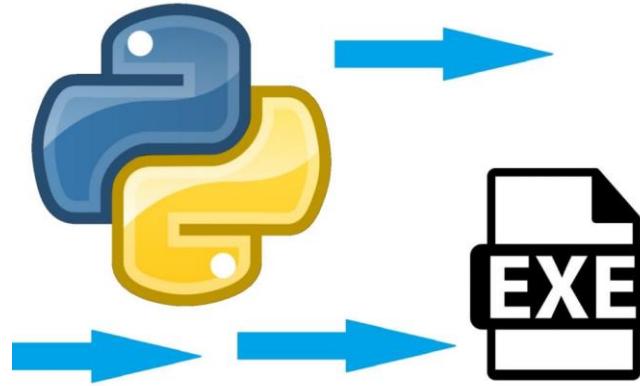
PyQt / PySide

wxPython

Tkinter

PyGTK





Distributing your Python application : Freezing

- PY2EXE
- CX_FREEZE

Interface graphique pour Python

Créer un exécutable : Py2exe or cx_freeze

- ✓ L'extension **Py2exe** ou **cx_freeze** permet de convertir un script Python en un programme exécutable (uniquement pour le système d'exploitation Windows). Ce programme exécutable (fichier avec l'extension .exe) peut ensuite être lancé sur un ordinateur où Python n'est pas installé.
- ✓ L'extension **Py2exe** ne fonction que pour la **version 2 de Python**, sous Windows. Pour la **version 3**, il existe des solutions comme **cx_freeze**

 **py2exe details**

Website	py2exe.org 
----------------	--

 **cx_Freeze details**

Website	cx-freeze.sourceforge.net 
----------------	--

Interface graphique pour Python

Créer un exécutable : Py2exe, how ?

F:\FS\IHM\Exp\py2exe\dist

Nom	
lib	
Lay1WindCalcul.exe	servés. 64.whl (166 kB) 261 kB/s
libcrypto-1_1-x64.dll	14 kB/s
libffi-8.dll	\users\user\anaconda3\
libssl-1_1-x64.dll	rs\user\anaconda3\lib\
MSVCP140.dll	
tcl86t.dll	
tk86t.dll	
wxbase315u_net_vc140_x64.dll	1.9.3-py3-none-any.whl da3a738db74308da41e23a
wxbase315u_vc140_x64.dll	cal\pip\cache\wheels\0 d78b5a3aecc
wxmsw315u_core_vc140_x64.dll	
wxmsw315u_html_vc140_x64.dll	.11.0.1
wxmsw315u_stc_vc140_x64.dll	

```
setup.py* x
from distutils.core import setup
import py2exe, sys, os

sys.argv.append('py2exe')

setup(
    options = {'py2exe': {'bundle_files': 1, 'compressed': True}},
    windows = [{ 'script': "Lay1WindCalcul.py" }],
    zipfile = None,
)
```



Interface graphique pour Python

Créer un exécutable : cx-Freeze, how ?

python : py2exe

```
Sélection C:\WINDOWS\system32\cmd.exe
```

```
Microsoft Windows [version 10.0.16299.1127]
(c) 2017 Microsoft Corporation. Tous droits réservés.

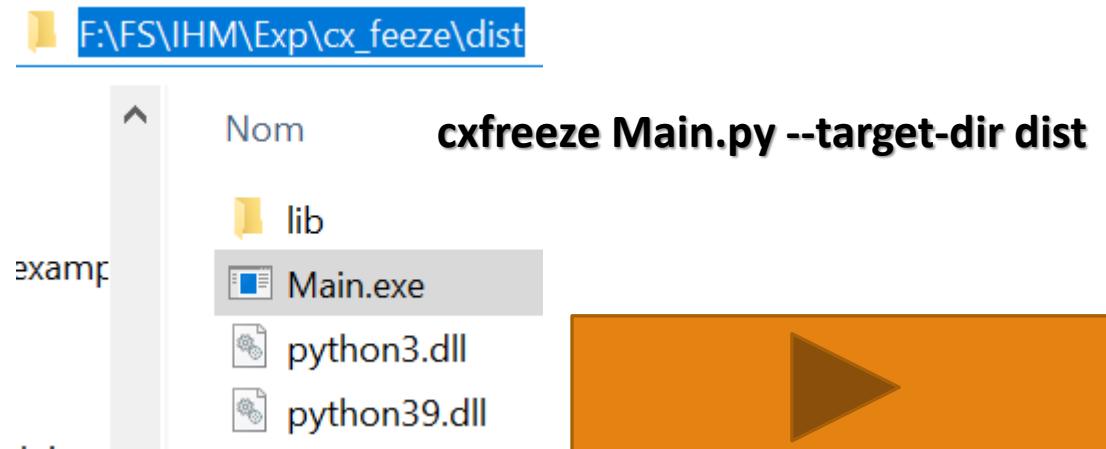
(base) C:\Users\User>pip install cx_freeze
Collecting cx_freeze
  Downloading cx_Freeze-6.9-cp39-cp39-win_amd64.whl (207 kB)
|██████████| 207 kB 273 kB/s
Collecting cx-logging>=3.0
  Downloading cx_Logging-3.0-cp39-cp39-win_amd64.whl (26 kB)
Requirement already satisfied: importlib-metadata>=4.3.1 in c:\users\user\anaconda3\lib\site-packages (from cx_freeze) (4.8.1)
Requirement already satisfied: zipp>=0.5 in c:\users\user\anaconda3\lib\site-packages (from importlib-metadata>=4.3.1->cx_freeze) (3.6.0)
Installing collected packages: cx-logging, cx-freeze
Successfully installed cx-freeze-6.9 cx-logging-3.0

(base) C:\Users\User>
```

```
setup.py x
```

```
from cx_Freeze import setup, Executable

setup(
    name = "IHM_students",
    version = "1",
    description = "M2I & MQL - IHM Students",
    executables = [Executable("main.py")],
)
```



Application Web sur Python

- BACKEND # FRONTEND

Application Web sur Python : Le développement Front-End

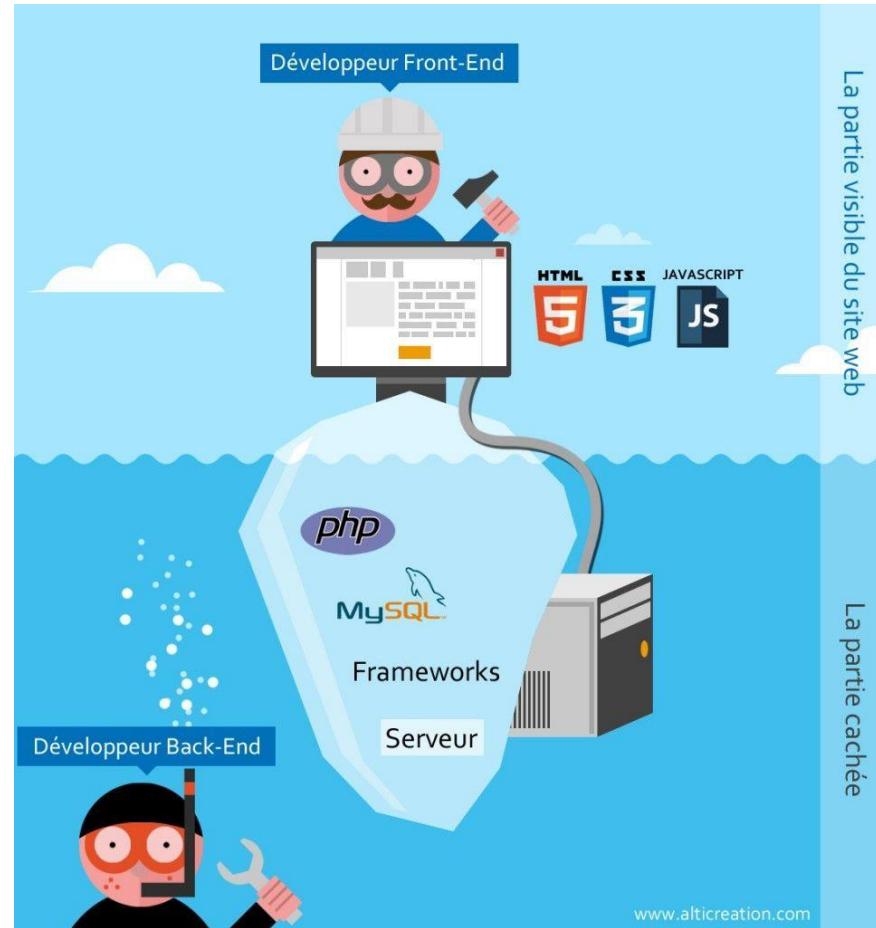
- ✓ Le terme front-end (*frontal*) désigne les éléments d'un site ou d'une application que les utilisateurs voient à l'écran et avec lesquels ils vont interagir.
- ✓ En général, le front-end se compose d'un design et d'un code HTML, CSS, JavaScript et jQuery càd des langages interprétés par le navigateur. C'est tout ce que les internautes vont voir sur un site internet.
- ✓ Mis à part les langages de base en front-end, les développeurs utilisent :
 - ✓ Frameworks : Bootstrap, Angular...,
 - ✓ Bibliothèques JavaScript (jQuery...)
 - ✓ Extensions CSS (Sass et LESS...)

Pour objectif de rendre le code plus facile à gérer grâce à divers outils et modèles compatibles avec les langages courants.

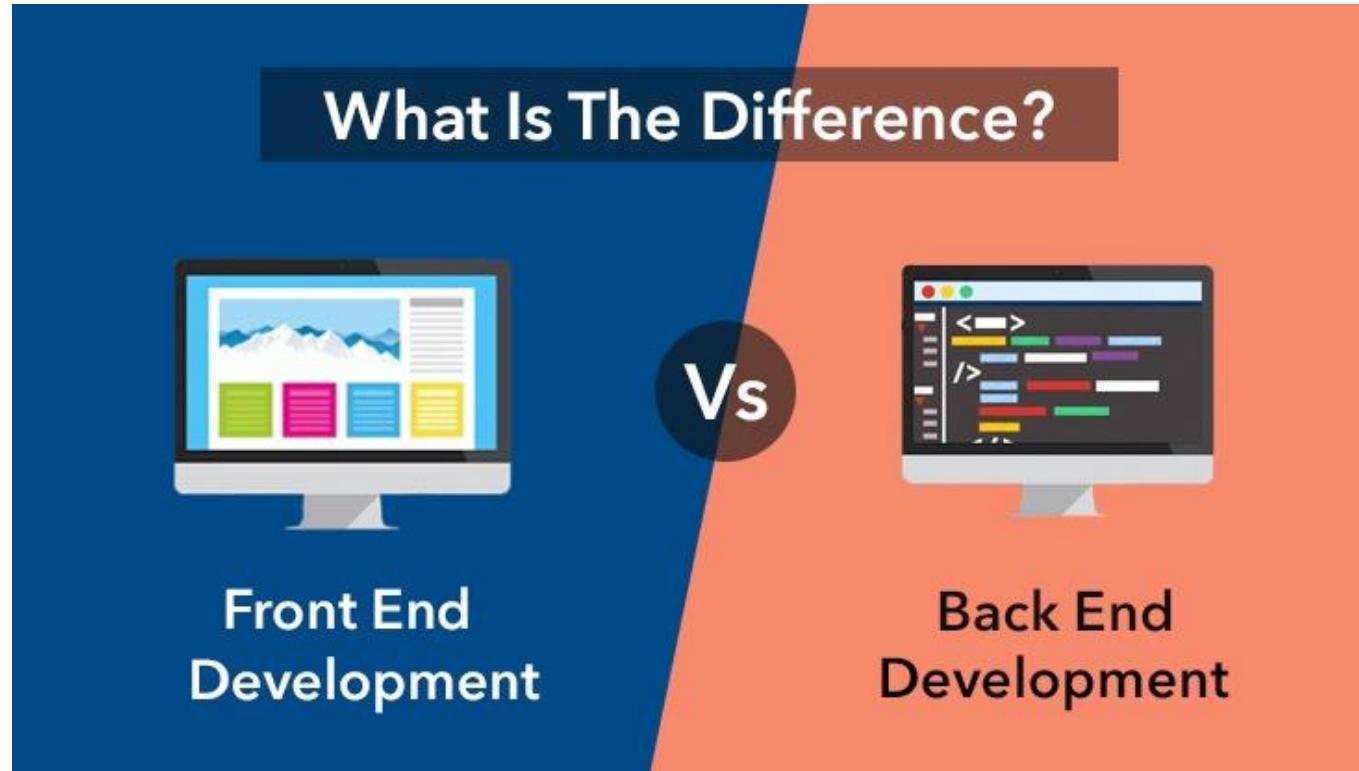


Application Web sur Python : Le développement Back-End

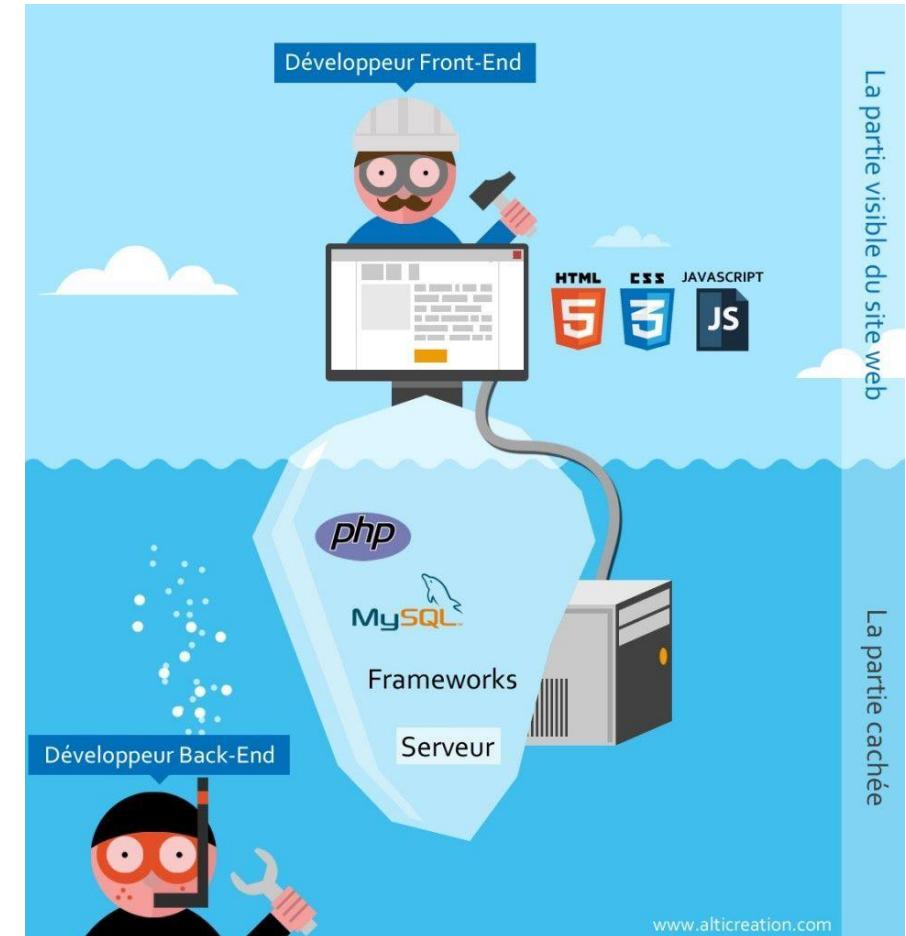
- ✓ Le développeur back-end (*dorsale*) programme la partie cachée d'une solution web, le côté serveur, administrateur d'une app ou d'un site web.
- ✓ C'est toute la partie du code qui gère les opérations côté serveur telles que la logique du serveur, les fonctions de base de données, et bien la sécurité,
- ✓ Les technologies backend sont en constante évolution :
 - ✓ Pour PHP : Symfony, Laravel, CakePHP
 - ✓ Pour Java : Spring, Hibernate, Maven, Struts
 - ✓ Pour Ruby : Ruby on Rails
 - ✓ Pour JavaScript : Node.js
 - ✓ Pour Python : Django, Pyramid, Flask



Application Web sur Python : BackEnd # FrontEnd

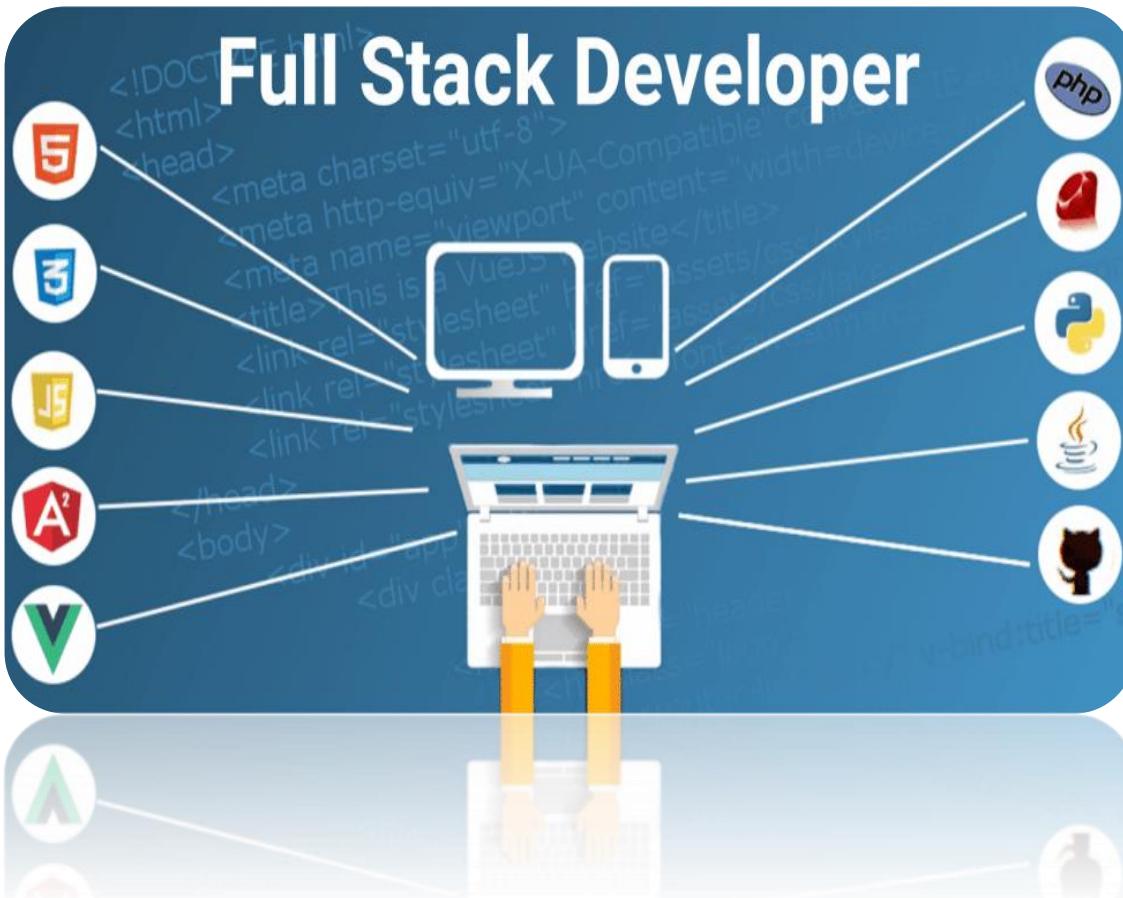


- ✓ Les notions de *front-end* et de *back-end* sont toutes les deux importantes dans la construction d'un projet digital.
- ✓ Le back-end et le front-end sont donc complémentaires.



Application Web sur Python : Full Stack Developer

- ✓ Un Full Stack Developer désigne un spécialiste qui maîtrise toutes les compétences du front-end et du back-end.
- ✓ Il sera alors autonome sur le développement complet d'un projet, du back au front-end
- ✓ Un profil très recherché par les entreprises et surtout les startups.
- ✓ Ce professionnel a une connaissance sur divers éléments, doit maîtriser la prise en compte de l'expérience utilisateur, l'intégration HTML ou encore la construction de l'architecture d'un site.
- ✓ Bien entendu, il doit également avoir une parfaite maîtrise des différents langages de programmation front-end et back-end.





Application Web sur Python

- DJANGO
- FLASK
- PYRAMID

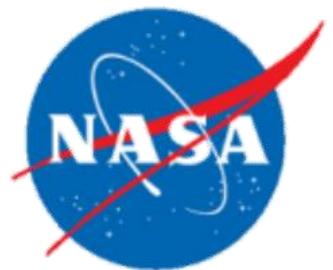


Application Web sur Python

DJANGO

Django Framework : About

- ✓ Django est un framework de développement Web en Python.
- ✓ ***Don't repeat yourself*** : Django permet le développement rapide de meilleures et plus performantes applications web, tout en conservant un code élégant. Par exemple :
 - ✓ L'administration d'un site sera automatiquement générée, et celle-ci est très facilement adaptable.
 - ✓ L'interaction avec une base de données se fait via un ensemble d'outils spécialisés et très pratiques. Il est donc inutile de perdre son temps à écrire directement des requêtes destinées à la base de données, car Django le fait automatiquement.
 - ✓ De plus, d'autres bibliothèques complètes et bien pensées sont disponibles, comme un espace membres, ou une bibliothèque permettant la traduction de votre application web en plusieurs langues.
- ✓ Django est devenu très populaire et est utilisé par des sociétés du monde entier :



Django Framework : Install

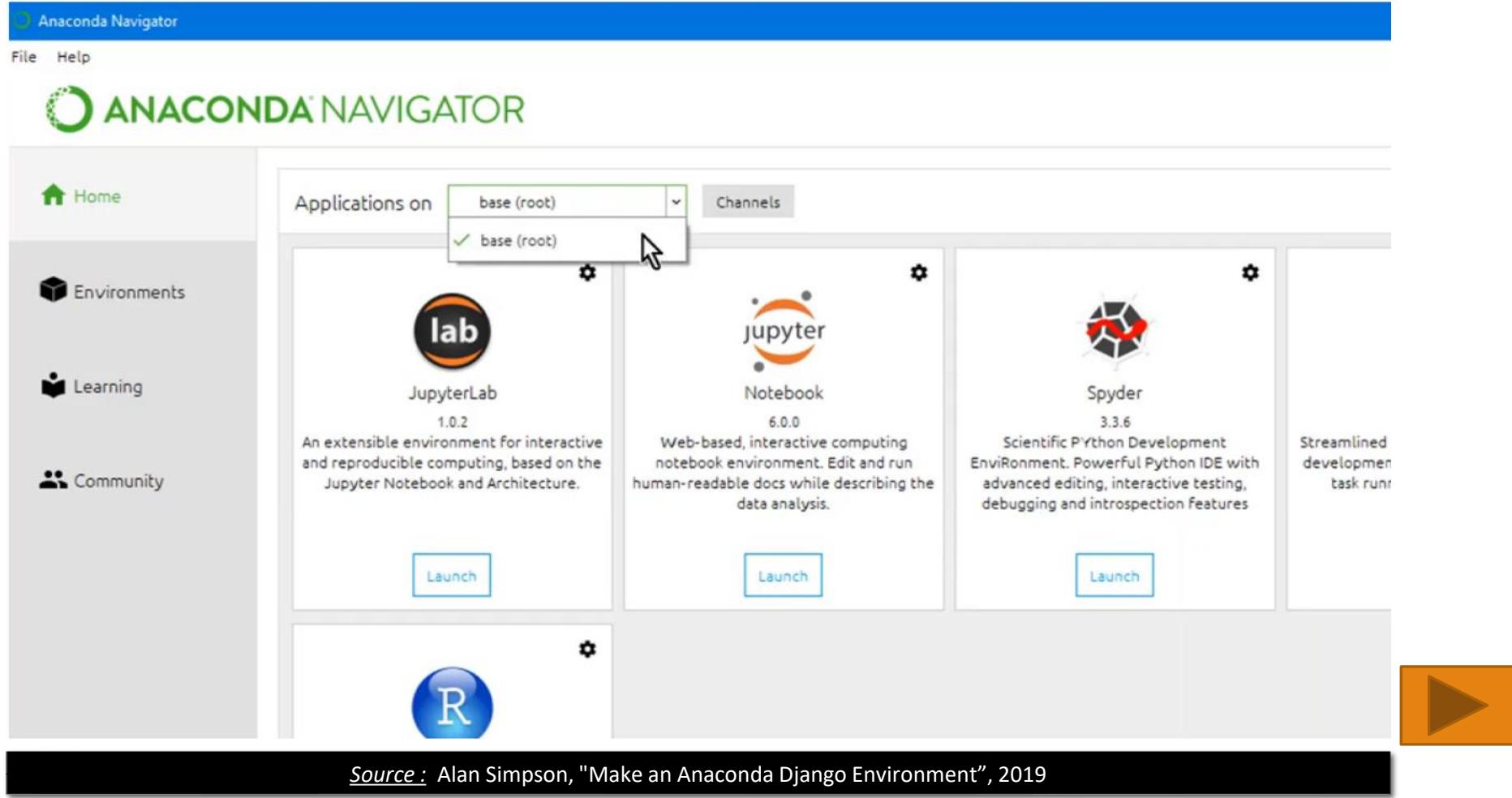
- ✓ Le Framework Django se présente sous la forme de modules Python et d'une commande *django-admin*, qui permet de faire différents opérations d'administration sur un projet.
- ✓ La portabilité de Django est un avantage, pour l'installer sous **cmd** :

apt-get install python-django

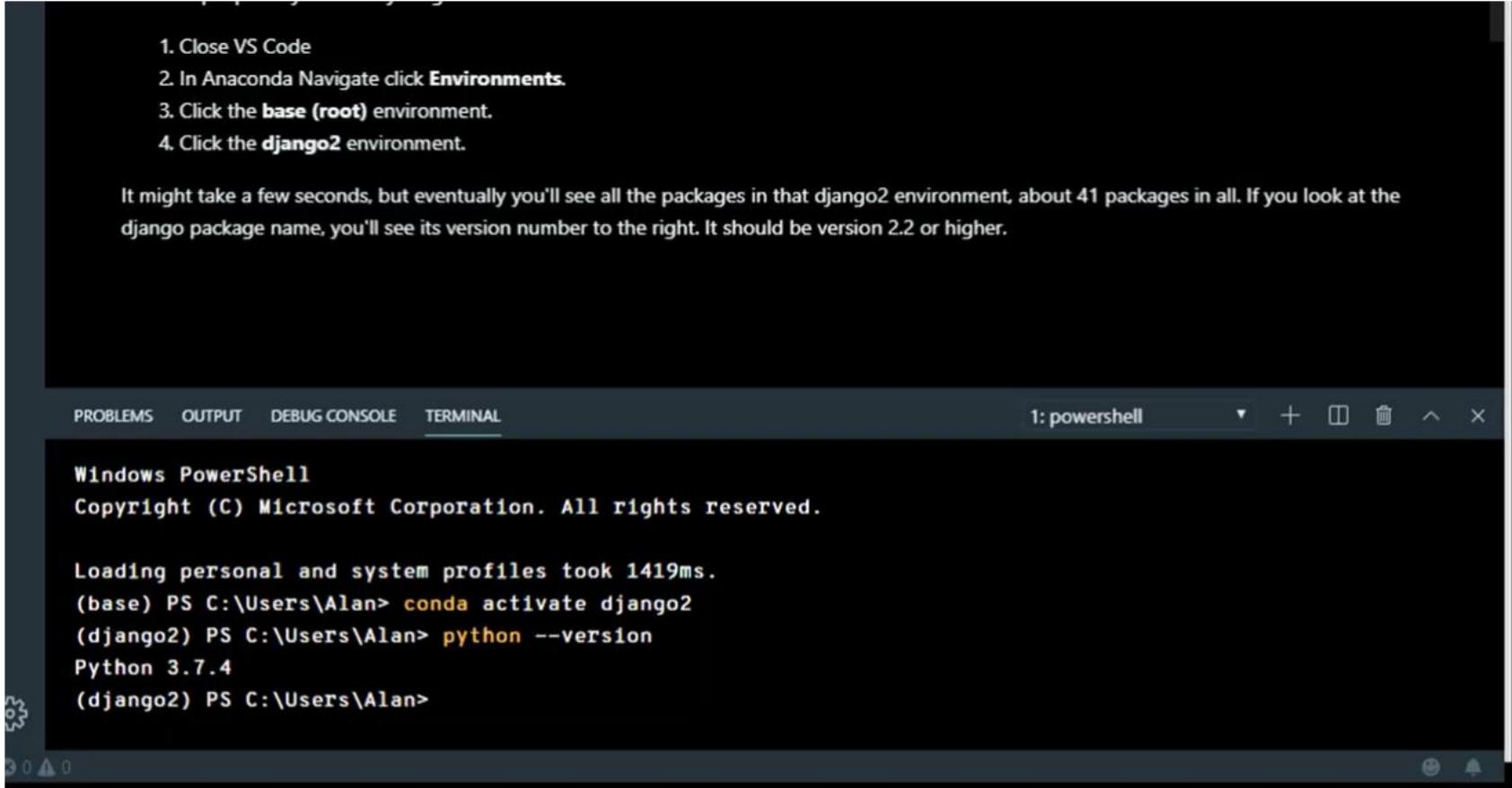
pip install django

- ✓ Soit à travers un environnement sous Anaconda

Django Framework : Installation Django



Django Framework : Installation Django



The screenshot shows a Windows PowerShell window within the VS Code interface. The terminal tab is selected at the top, showing the command line. The output pane displays the following text:

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Loading personal and system profiles took 1419ms.
(base) PS C:\Users\Alan> conda activate django2
(django2) PS C:\Users\Alan> python --version
Python 3.7.4
(django2) PS C:\Users\Alan>
```

At the top of the image, there is a list of instructions:

1. Close VS Code
2. In Anaconda Navigate click **Environments**.
3. Click the **base (root)** environment.
4. Click the **django2** environment.

A note below the instructions says: "It might take a few seconds, but eventually you'll see all the packages in that django2 environment, about 41 packages in all. If you look at the django package name, you'll see its version number to the right. It should be version 2.2 or higher."



Source : Alan Simpson, "Make an Anaconda Django Environment", 2019

Django Framework : Vérification de l'installation

Anaconda Navigator

File Help



Home Environments Learning Community Documentation Anaconda Run

Applications on django2 Channels

CMD.exe Prompt 0.1.1

Run a cmd.exe terminal with your current environment from Navigator activated

Launch

C:\WINDOWS\system32\cmd.exe - python

(c) 2017 Microsoft Corporation. Tous droits réservés.

(django2) C:\Users\User>python

Python 3.9.7 (default, Sep 16 2021, 16:59:28) [MSC v.1916 64 bit (AMD64)] :: Anaconda, Inc. on win32

Type "help", "copyright", "credits" or "license" for more information.

>>> import django

>>> print(django.get_version())

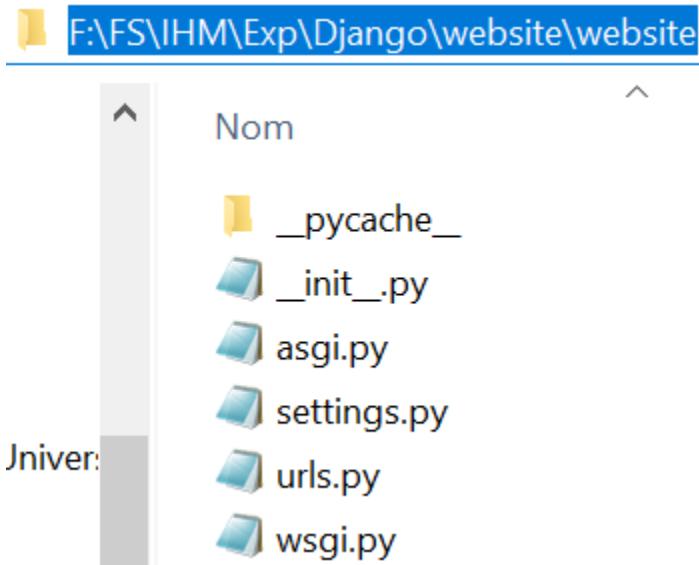
4.0

>>>

The screenshot shows the Anaconda Navigator interface. On the left, there's a sidebar with icons for Home, Environments, Learning, and Community, along with links for Documentation and Anaconda Run. Below the sidebar are social media links for Twitter, YouTube, and GitHub. The main area has a title bar with 'Applications on django2' and a 'Channels' button. A red circle highlights the 'django2' dropdown. In the center, there's a 'CMD.exe Prompt' section with a green icon. Below it, text says 'Run a cmd.exe terminal with your current environment from Navigator activated'. A red circle highlights the 'Launch' button at the bottom of this section. To the right, a terminal window is open with the command 'python' entered, showing Python version 3.9.7. A red box highlights the command 'import django' and its output '4.0'. Another red arrow points from the 'Launch' button in the CMD section to the terminal window.

Django Framework

Lancer un nouveau site



```
C:\WINDOWS\system32\cmd.exe - python manage.py runserver
(django2) F:\FS\IHM\Exp>cd Django
(django2) F:\FS\IHM\Exp\ Django>django-admin startproject website
(django2) F:\FS\IHM\Exp\ Django>ls
'ls' n'est pas reconnu en tant que commande interne
ou externe, un programme exécutable ou un fichier de commandes.

(django2) F:\FS\IHM\Exp\ Django>dir
Le volume dans le lecteur F n'a pas de nom.
Le numéro de série du volume est 6EBB-D2BE

Répertoire de F:\FS\IHM\Exp\ Django

28/12/2021 01:45 <DIR> .
28/12/2021 01:45 <DIR> ..
28/12/2021 01:45 <DIR> website
          0 fichier(s)          0 octets
          3 Rép(s) 12.571.332.608 octets libres

(django2) F:\FS\IHM\Exp\ Django>cd website
(django2) F:\FS\IHM\Exp\ website>python manage.py runserver
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).

You have 18 unapplied migration(s). Your project may not work properly until you apply the migrations
Run 'python manage.py migrate' to apply them.
December 28, 2021 - 01:45:57
Django version 4.0, using settings 'website.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
[28/Dec/2021 01:46:31] "GET / HTTP/1.1" 200 10697
[28/Dec/2021 01:46:32] "GET /static/admin/css/fonts.css HTTP/1.1" 200 423
[28/Dec/2021 01:46:32] "GET /static/admin/fonts/Roboto-Regular-webfont.woff HTTP/1.1" 200 85876
[28/Dec/2021 01:46:32] "GET /static/admin/fonts/Roboto-Bold-webfont.woff HTTP/1.1" 200 86184
[28/Dec/2021 01:46:32] "GET /static/admin/fonts/Roboto-Light-webfont.woff HTTP/1.1" 200 85692
[28/Dec/2021 01:46:32] "GET /favicon.ico HTTP/1.1" 404 2111
[28/Dec/2021 01:46:32] "GET /favicon.ico HTTP/1.1" 404 2111
```

django

[View release notes](#) for Django 4.0



The install worked successfully! Congratulations!

You are seeing this page because `DEBUG=True` is in your settings file and you have not configured any URLs.



[Django Documentation](#)
Topics, references, & how-to's



[Tutorial: A Polling App](#)
Get started with Django



[Django Community](#)
Connect, get help, or contribute

Django Framework Migration Bugs

```
(django2) F:\FS\IHM\Exp\Django\website>python manage.py runserver
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).

You have 18 unapplied migration(s). Your project may not work properly.
Run 'python manage.py migrate' to apply them.
```

```
C:\> C:\WINDOWS\system32\cmd.exe
(django2) F:\FS\IHM>cd Exp
(django2) F:\FS\IHM\Exp>cd Django
(django2) F:\FS\IHM\Exp\Django>cd website
(django2) F:\FS\IHM\Exp\Django\website>python manage.py migrate
Operations to perform:
  Apply all migrations: admin, auth, contenttypes, sessions
Running migrations:
  Applying contenttypes.0001_initial... OK
  Applying auth.0001_initial... OK
  Applying admin.0001_initial... OK
  Applying admin.0002_logentry_remove_auto_add... OK
  Applying admin.0003_logentry_add_action_flag_choices... OK
  Applying contenttypes.0002_remove_content_type_name... OK
  Applying auth.0002_alter_permission_name_max_length... OK
  Applying auth.0003_alter_user_email_max_length... OK
  Applying auth.0004_alter_user_username_opts... OK
  Applying auth.0005_alter_user_last_login_null... OK
  Applying auth.0006_require_contenttypes_0002... OK
  Applying auth.0007_alter_validators_add_error_messages... OK
  Applying auth.0008_alter_user_username_max_length... OK
  Applying auth.0009_alter_user_last_name_max_length... OK
  Applying auth.0010_alter_group_name_max_length... OK
  Applying auth.0011_update_proxy_permissions... OK
  Applying auth.0012_alter_user_first_name_max_length... OK
  Applying sessions.0001_initial... OK
(django2) F:\FS\IHM\Exp\Django\website>python manage.py migrate
```

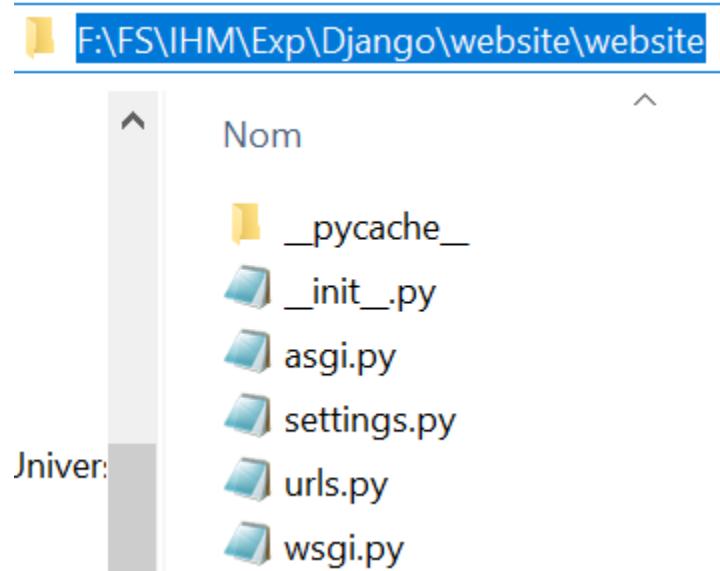
Django Framework : Création d'un projet

- ✓ Pour créer un projet :

django-admin startproject website

- ✓ Ce qui permet de créer un répertoire **website** qui contient :

- ✓ **__init__.py**, le fichier classique pour être vu comme un module Python.
- ✓ **manage.py**, un script exécutable qui permet de gérer le projet généré. C'est un *django-admin* du projet
- ✓ **settings.py**, qui contient les paramètres du projet créé
- ✓ **urls.py**, le routeur d'URLs



Django Framework : Test du projet

- ✓ Le projet est déjà fonctionnel.
- ✓ Django intègre un petit serveur Web bien pratique en phase de développement
- ✓ Pour le lancer :

python manage.py runserver

- ✓ Puis, on pointe son navigateur Web sur :

http://localhost:8000

- ✓ Django nous accueille !

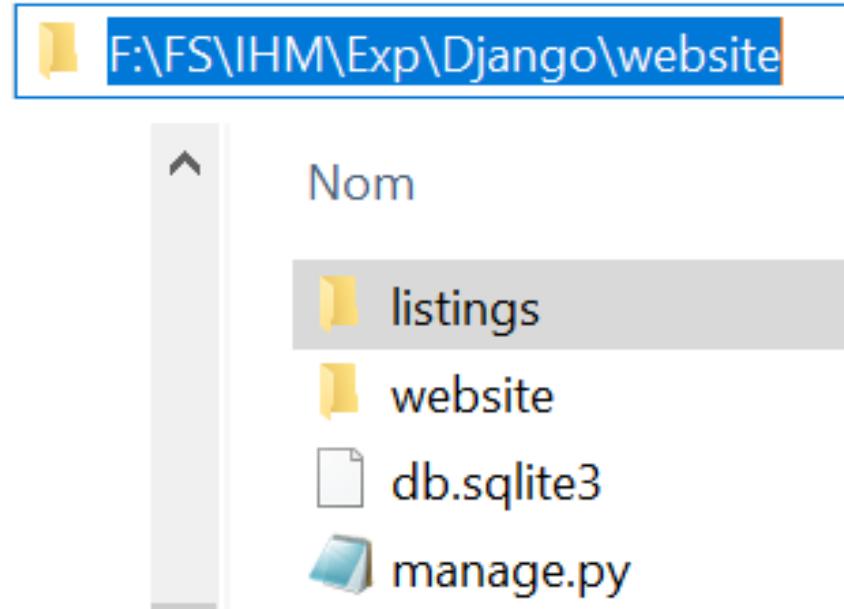
Django Framework : Générer le code d'une application Django

- ✓ Dans Django, une application est une sous-section de votre projet entier.

Django nous encourage à compartimenter notre projet entier Django en applications, pour deux raisons principales :

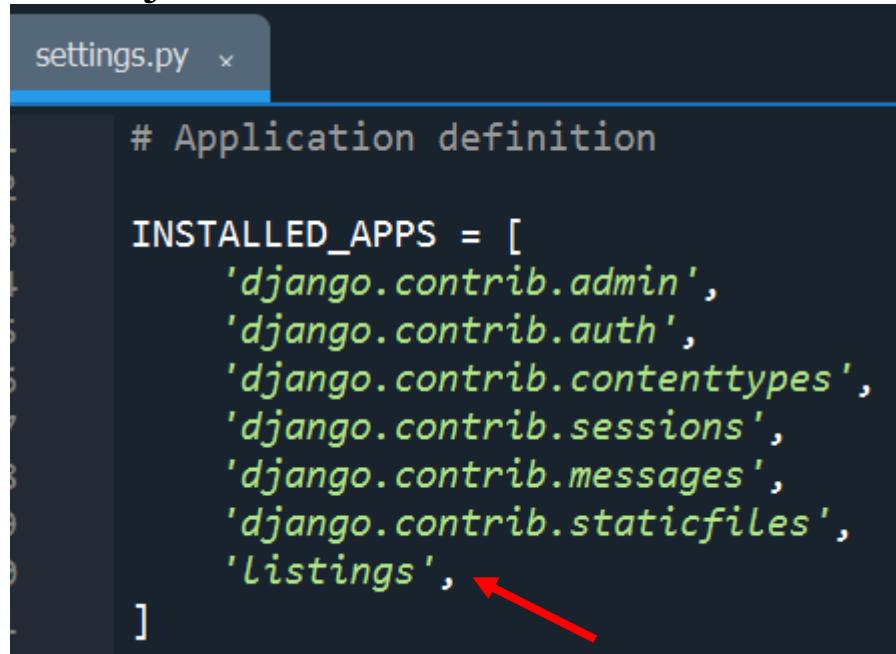
- ✓ cela permet de garder notre projet organisé et gérable au fur et à mesure qu'il se développe ;
 - ✓ cela signifie qu'une application peut éventuellement être réutilisée dans plusieurs projets.
- ✓ Pour créer cette application, en utilisant la sous-commande **startapp** :

python manage.py startapp listings



Django Framework : Générer le code d'une application Django

- ✓ La dernière étape de l'ajout de l'application « **listings** » dans le projet « **website** » consiste à « *installer* » l'application dans le projet.
- ✓ A la génération du code de base de ce projet, l'un des fichiers créés s'appelait *settings.py*. Dans une liste Python appelée **INSTALLED_APPS**, il faut ajouter la chaîne de caractères '**listings**' :



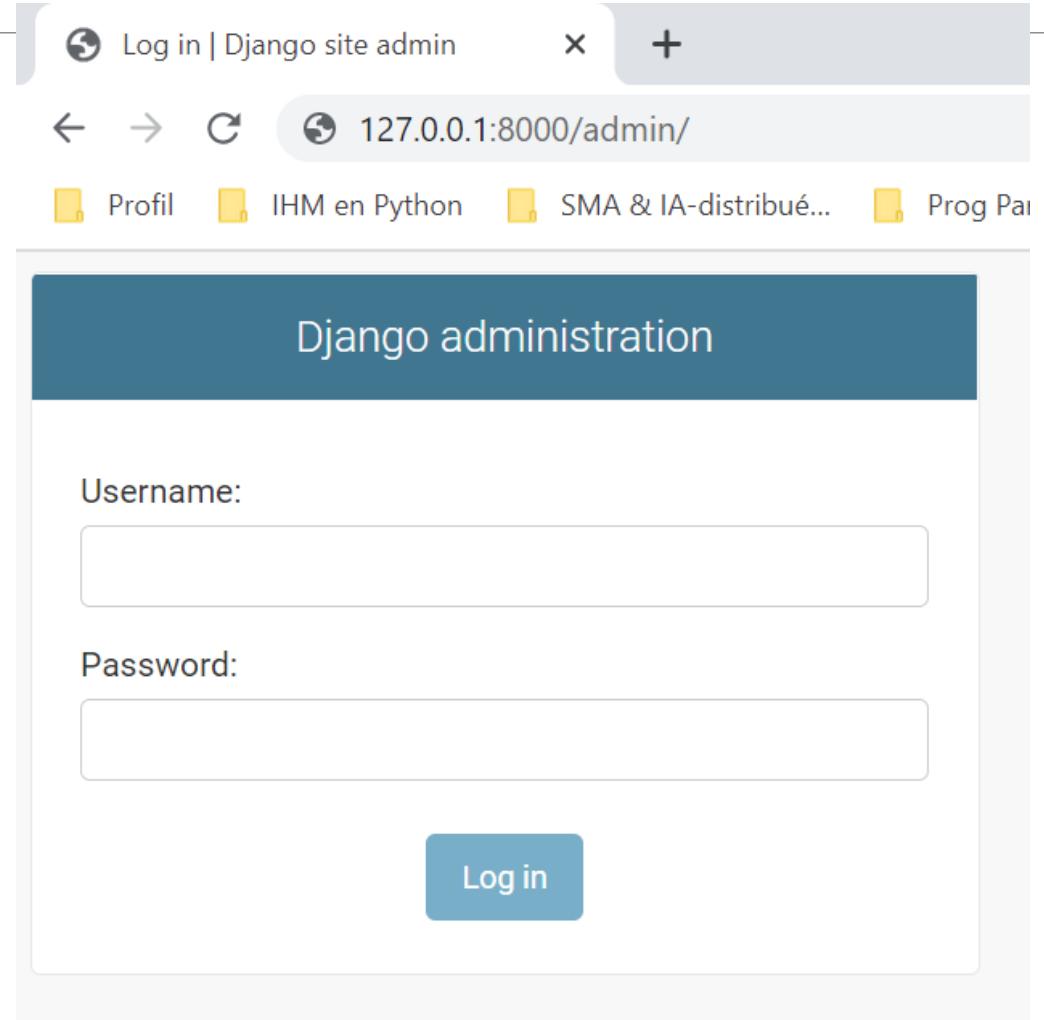
```
settings.py ×

# Application definition

INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'listings',
]
```

A screenshot of a code editor window titled "settings.py". The code shown is the Python settings file for a Django project. It defines the "INSTALLED_APPS" list, which contains several built-in Django apps and one custom app named "listings". A red arrow points to the word "listings" in the list, indicating it as the target of the "install" step.

Django Framework Admin UI



Microsoft Windows [version 10.0.16299.1127]
(c) 2017 Microsoft Corporation. Tous droits réservés.

(django2) C:\Users\User>f:

(django2) F:>cd FS

(django2) F:\FS>cd IHM

(django2) F:\FS\IHM>cd Exp

(django2) F:\FS\IHM\Exp>cd Django

(django2) F:\FS\IHM\Exp\ Django>cd website

(django2) F:\FS\IHM\Exp\ Django\website>python manage.py createsuperuser

Username (leave blank to use 'user'): otman

Email address: abdoun.otman@gmail.com

Password:

Password (again):

The password is too similar to the username.

Bypass password validation and create user anyway? [y/N]: y

Superuser created successfully.

(django2) F:\FS\IHM\Exp\ Django\website>

Django administration

Username:

Password:

AUTHENTICATION AND AUTHORIZATION

Groups	 Add	 Change
Users	 Add	 Change

Recent actions

My actions

None available

Site administration | Django site [x](#) [+](#)

← → C i 127.0.0.1:8000/admin/

Profil IHM en Python SMA & IA-distribué... Prog Paral Journals Projet SIMARech 3 Fatourati

Django administration

Site administration

AUTHENTICATION AND AUTHORIZATION

[Groups](#) [+ Add](#) [Change](#)

[Users](#) [+ Add](#) [Change](#)

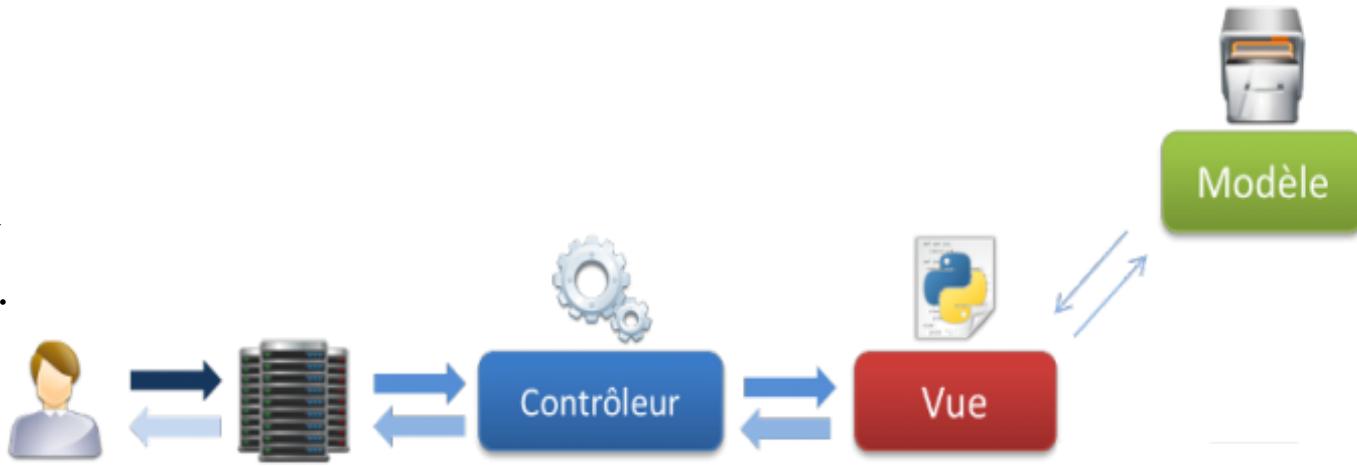
Recent actions

My actions

None available

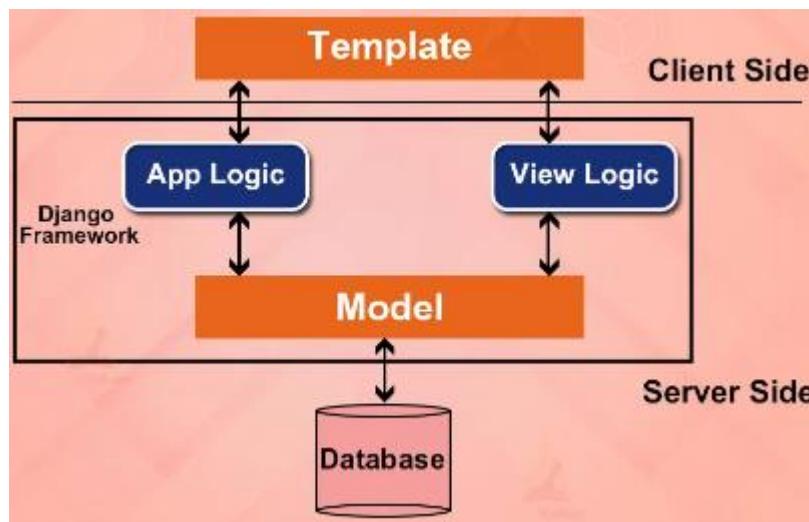
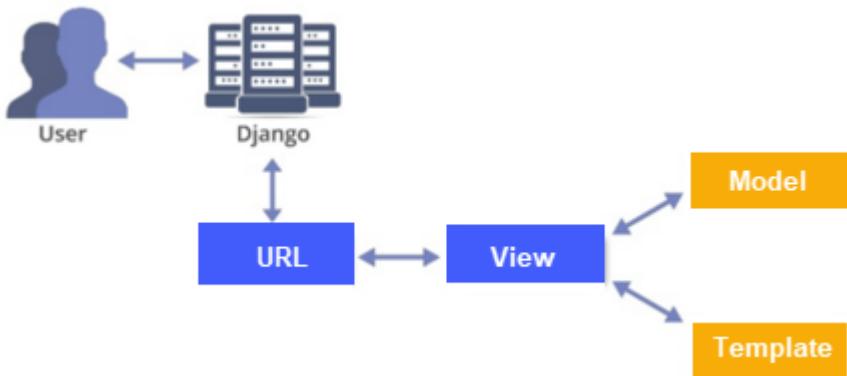
Django Framework : MVC >>> MVT

- ✓ En règle générale, toutes les applications fournissant une interface utilisateur (que ce soit sur le web ou en bureau) suivent l'architecture MVC. Django prend en charge le modèle MVC.
- ✓ Le modèle-vue-Template (MVT) est légèrement différent de MVC.
- ✓ Django s'occupe de la partie Contrôleur (Code logiciel qui contrôle les interactions entre le Modèle et la Vue).
Dont, le Template est un fichier HTML mélangé avec Django Template Language (DTL).
- ✓ Le développeur fournit la vue et le modèle, puis le mappe à une URL et Django s'occupe de le servir à l'utilisateur.



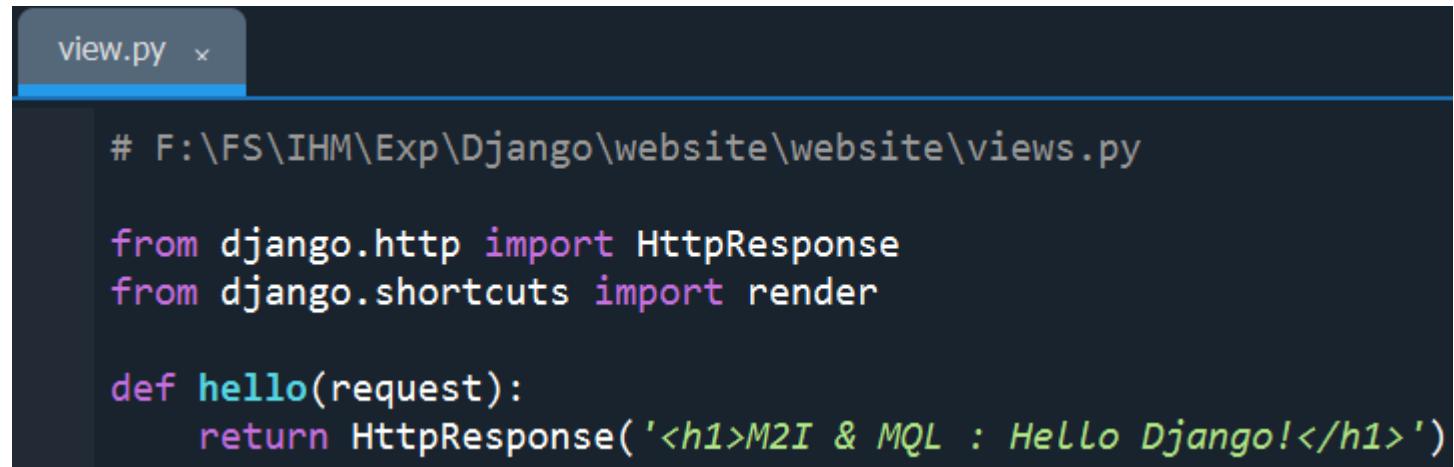
Django Framework : Architecture MVT

- ✓ En Django, l'architecture suivie est souvent appelée "MVT", qui signifie "Modèle-Vue-Template". Bien que le terme diffère légèrement de "Modèle-Vue-Contrôleur" (MVC), le concept est similaire. Dans le contexte de Django :
 - ✓ Modèle (**Model**) : Représente la structure des données et gère leur accès. Il s'occupe généralement de l'interaction avec la base de données.
 - ✓ Vue (**View**) : Gère la logique métier et la présentation. Dans le modèle MVT de Django, cette composante est plus proche du contrôleur dans une architecture MVC classique.
 - ✓ Template : Gère la présentation et la mise en forme des données. C'est l'équivalent de la vue dans une architecture MVC traditionnelle.
- ✓ Ainsi, bien que Django utilise le terme "MVT", les concepts sont similaires à ceux de l'architecture "MVC". La distinction est souvent liée à la manière dont la logique métier et la présentation sont traitées dans le cadre du développement web avec Django.



Django Framework : Vue - M&J du contenu

- ✓ La vue est l'un des blocs constitutifs de l'architecture MVT.
- ✓ Une vue a pour fonction de répondre à la visite d'un utilisateur sur le site en renvoyant une page que l'utilisateur peut voir.
- ✓ En Python : Une vue est une fonction qui accepte un objet ***HttpRequest*** comme paramètre et retourne un ***objetHttpResponse***.



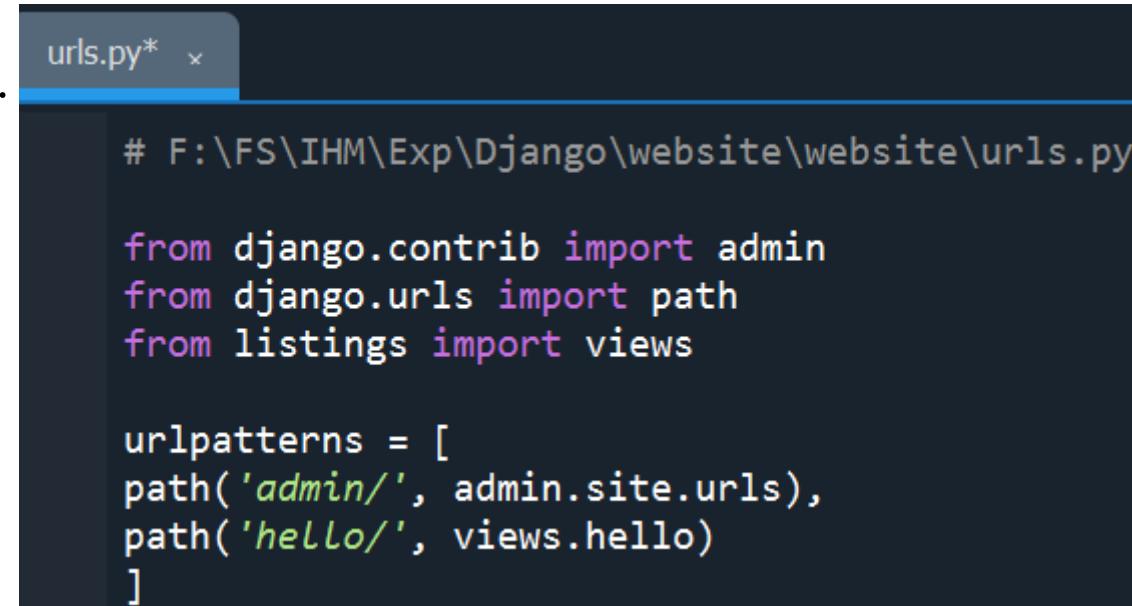
```
view.py  x
# F:\FS\IHM\Exp\Django\website\website\views.py

from django.http import HttpResponse
from django.shortcuts import render

def hello(request):
    return HttpResponse('<h1>M2I & MQL : Hello Django!</h1>')
```

Django Framework : Vue - M&J du contenu

- ✓ Un modèle d'URL, c'est pour indiquer à Django d'être à l'écoute d'une requête pour une URL donnée, puis appeler une vue spécifique pour générer une page.
- ✓ Il faut importer le module **views** crée, en ajoutant une déclaration d'import.
- ✓ Puis nous ajouterons un nouvel élément à la liste **urlpatterns**, où nous ferons référence à la fonction vue créée.
- ✓ C'est la création d'un modèle d'URL et de le lier à la vue créée.



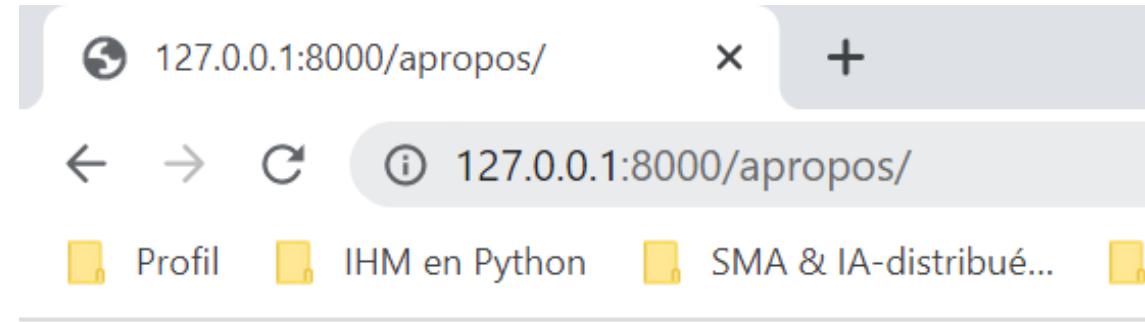
```
urls.py*  ×

# F:\FS\IHM\Exp\ Django\website\website\urls.py

from django.contrib import admin
from django.urls import path
from listings import views

urlpatterns = [
    path('admin/', admin.site.urls),
    path('hello/', views.hello)
]
```

Django Framework : Vue – Ajouter une 2ème vue ABOUT



À propos

Un cours Django au profil des étudinats M2I & MQL!

Django Framework : Vue – Ajouter une 2ème vue ABOUT

views.py

```
# F:\FS\IHM\Exp\ Django\website\website\views.py

from django.http import HttpResponse
from django.shortcuts import render

def hello(request):
    return HttpResponse('<h1>M2I & MQL : Hello Django!</h1>')

def apropos(request):
    return HttpResponse('<h1>À propos </h1> <p>Un cours Django au profil des étudiants M2I & MQL!</p>')
```

urls.py

```
# F:\FS\IHM\Exp\ Django\website\website\urls.py

from django.contrib import admin
from django.urls import path
from listings import views

urlpatterns = [
    path('admin/', admin.site.urls),
    path('hello/', views.hello),
    path('apropos/', views.apropos),
]
```

Django Framework : Vue – Récap

- ✓ Lorsque Django reçoit une requête *HTTP*, il tente de trouver une correspondance pour le chemin de cette requête dans une liste de modèles d'URL, définis dans *urls.py*.
- ✓ Si le chemin correspond à un modèle d'URL, Django transmet la requête à la vue correspondante, défini dans *views.py*.
- ✓ La vue est une fonction qui accepte la demande en tant que paramètre et qui renvoie une *HttpResponse* comme valeur de retour. Cette réponse contient le *HTML* que le navigateur utilise pour afficher la page.

Django Framework : Modèle – Migration > DataBase

- ✓ Pour toute application, il faut identifier les différentes entités pour stocker des données.
- ✓ Pour chaque entité, il faut créer un modèle pour représenter cette entité.
- ✓ Un **modèle** définit les caractéristiques de stockage concernant une entité particulière.
- ✓ C'est possible d'utiliser le modèle pour créer des objets individuels, ou instances, de ce modèle, qui ont chacun leurs propres caractéristiques uniques.
- ✓ En général, dans les frameworks MVC et MVT, un modèle est également capable de stocker (ou de "persister") ses données dans une base de données pour une utilisation ultérieure.

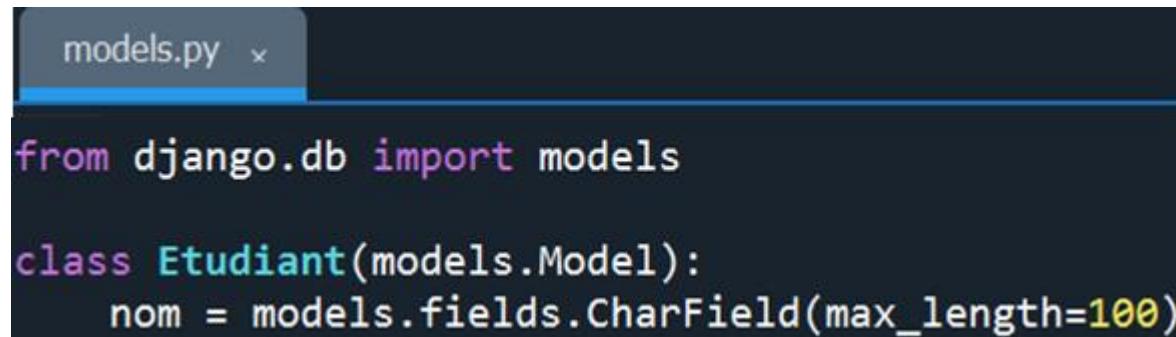
Django Framework : Modèle – Migration > DataBase

- ✓ Avec le framework **Django** : toutes les fonctionnalités de persistance des données dans une base de données sont disponibles en module.
- ✓ Il faut juste que modèle hérite de la classe ***models.Model*** de Django. Le modèle hérite ensuite de toutes les méthodes (comportements) nécessaires pour effectuer des opérations telles que la sélection et l'insertion de données dans une base de données.

N.B. : La structure d'une base de données, en termes de tables et de colonnes, est appelée **schéma**.

- ✓ Si nous construisions notre schéma de base de données manuellement, nous pourrions écrire une requête SQL ou utiliser une interface graphique de gestion de base de données, pour créer notre première table.

- ✓ La classe ***etudinat***, crée, hérite de ***models.Model***, qui est la classe de base du modèle de Django.
- ✓ Ensuite, un attribut est ajouté ***name***. Cet attribut de type ***CharField***, qui stocke des données de type ***Chaîne*** avec une longueur maximale du nom d'un Band à 100.



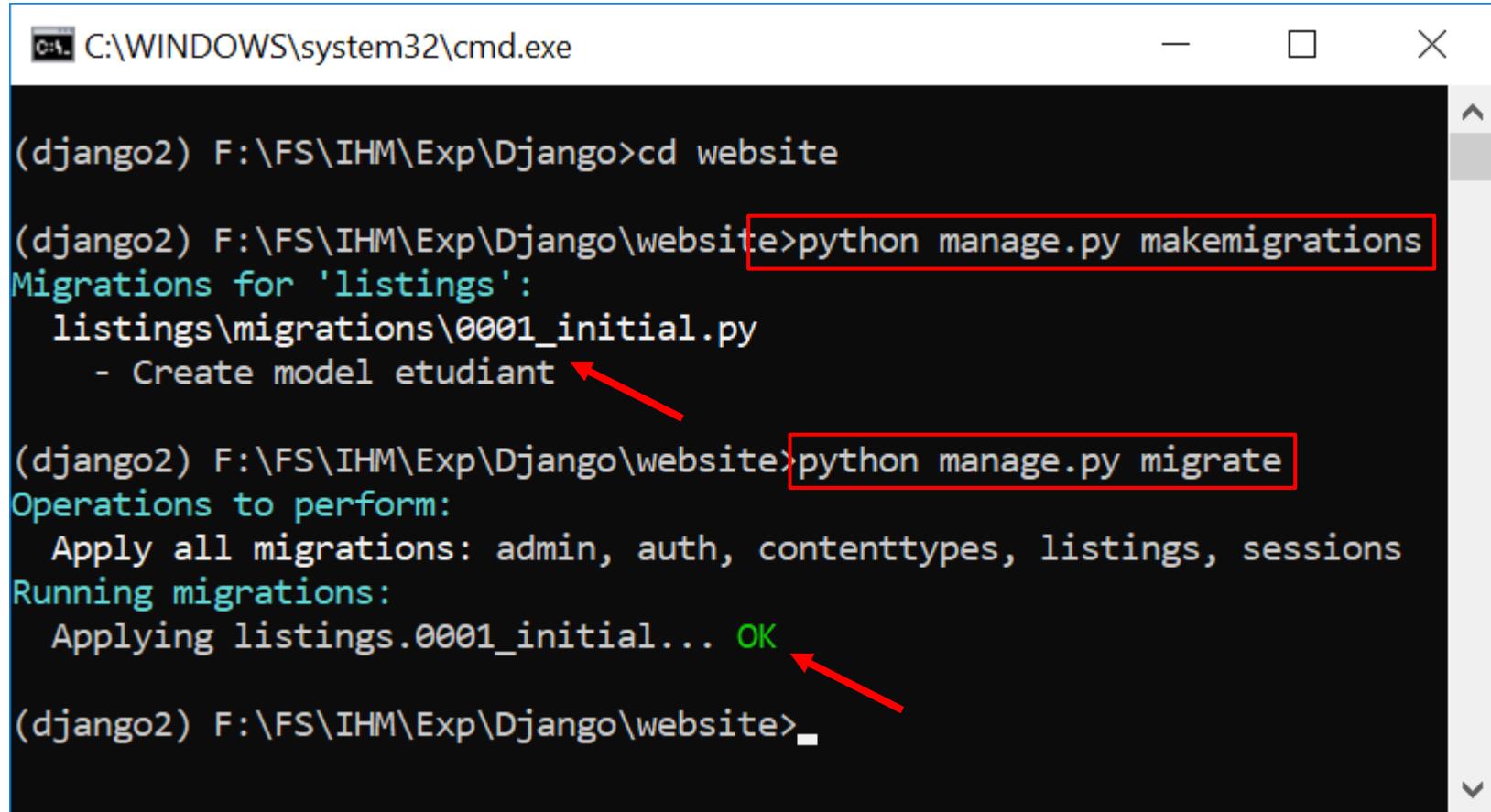
```
models.py
from django.db import models

class Etudiant(models.Model):
    nom = models.fields.CharField(max_length=100)
```

Django Framework : Modèle – Migration > DataBase

- ✓ Une **migration** est un ensemble d'instructions permettant de passer le schéma de votre base de données d'un état à un autre. Il est important de noter que ces instructions peuvent être exécutées automatiquement, comme un code.
- ✓ Sous Django, des sous-commande de l'utilitaire de ligne de commande qui va générer des instructions pour construire la table. Et ensuite, nous utilisons une autre sous-commande pour exécuter ces instructions. Ces instructions sont appelées une **migration**.
- ✓ Pour analyser le fichier ***models.py*** afin de déceler toute modification et déterminer le type de migration à générer:
python manage.py makemigrations
- ✓ Pour permettre au Django de rechercher dans chacune de ces applications installées (listings) de nouvelles migrations à exécuter :
python manage.py migrate

Django Framework : Modèle – Migration > DataBase



```
C:\WINDOWS\system32\cmd.exe
(django2) F:\FS\IHM\Exp\ Django>cd website
(django2) F:\FS\IHM\Exp\ Django\website>python manage.py makemigrations
Migrations for 'listings':
  listings\migrations\0001_initial.py
    - Create model etudiant
(django2) F:\FS\IHM\Exp\ Django\website>python manage.py migrate
Operations to perform:
  Apply all migrations: admin, auth, contenttypes, listings, sessions
Running migrations:
  Applying listings.0001_initial... OK
(django2) F:\FS\IHM\Exp\ Django\website>
```

The screenshot shows a Windows Command Prompt window titled 'cmd' with the path 'C:\WINDOWS\system32\cmd.exe'. The user is in a Django project directory 'F:\FS\IHM\Exp\ Django\website'. They run 'python manage.py makemigrations' which generates a migration file 'listings\migrations\0001_initial.py' that creates a model named 'etudiant'. Then, they run 'python manage.py migrate' which applies the migration, resulting in the message 'OK'. Red boxes highlight the command 'python manage.py makemigrations' and the output 'OK', while red arrows point to the 'etudiant' model creation and the 'OK' status message.

Django Framework : Modèle – Migration > DataBase

- ✓ Le shell de Django est simplement un shell Python (en temps réel) qui exécute application Django.
- ✓ Utiliser le shell pour créer quelques objets de groupe : **python manage.py shell**
- ✓ Puis enregistrer ces objets dans la base de données.

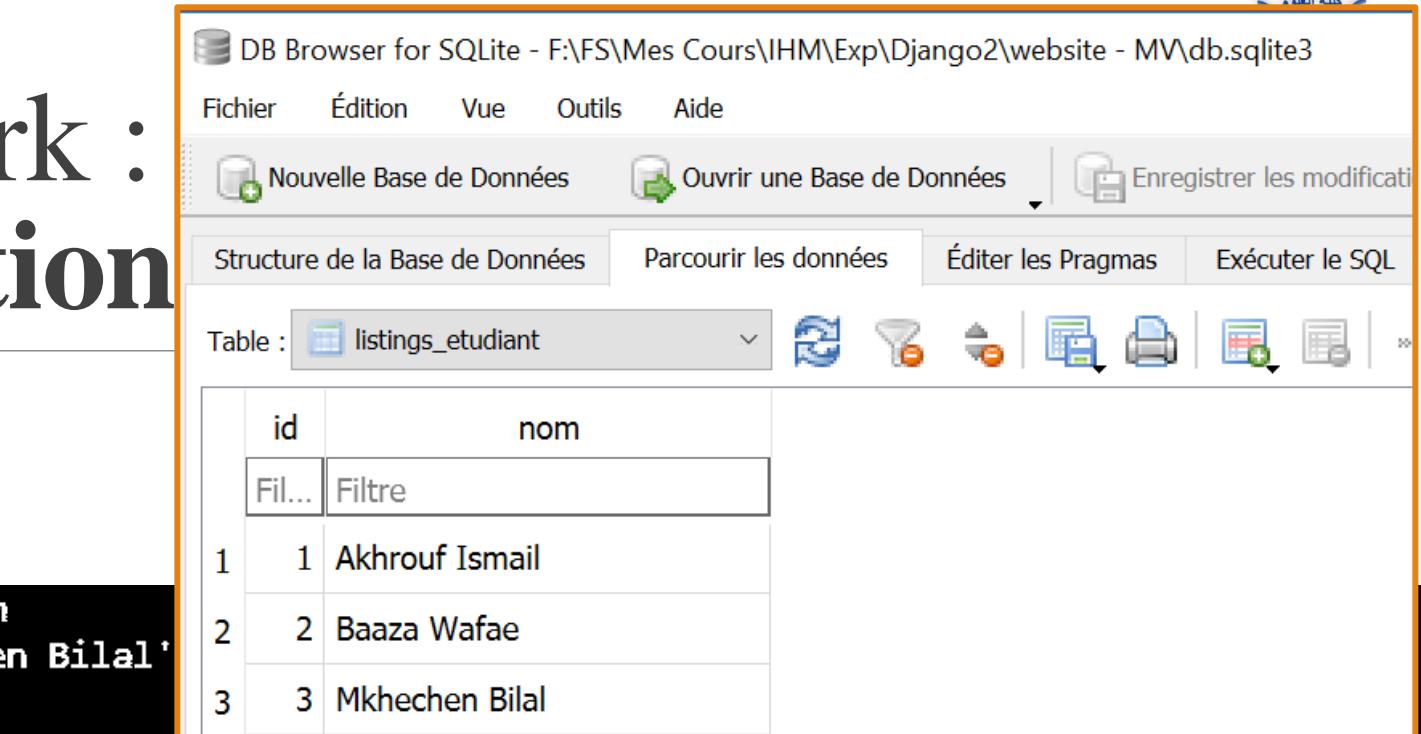
```
In [1]: from listings.models import Etudiant
In [2]: etud = Etudiant()
In [3]: etud.nom = 'Akhrouf Ismail'
In [4]: etud
Out[4]: <Etudiant: Etudiant object (None)>
In [5]: etud.save() ←
In [20]: etud = Etudiant.objects.create(nom='El Houmam Semlali Mohamed')
In [6]: etud
Out[6]: <Etudiant: Etudiant object (1)> → In [21]: etud.nom
Out[21]: 'El Houmam Semlali Mohamed'
```

```
(django2) F:\FS\IHM\Exp\ Django\website>python manage.py shell
Python 3.9.7 (default, Sep 16 2021, 16:59:28) [MSC v.1916 64 bit
Type 'copyright', 'credits' or 'license' for more information
IPython 7.29.0 -- An enhanced Interactive Python. Type '?' for help

In [1]: from listings.models import Etudiant
In [2]: etud = Etudiant()
In [3]: etud.nom = 'Akhrouf Ismail'
In [4]: etud
Out[4]: <Etudiant: Etudiant object (None)>
In [5]: etud.save()
In [6]: etud
Out[6]: <Etudiant: Etudiant object (1)> ←
In [7]: etud = Etudiant()
In [8]: etud.nom = 'Baaza Wafae'
In [9]: etud.save()
In [10]: etud
Out[10]: <Etudiant: Etudiant object (2)> ←
In [11]: etud = Etudiant()
In [12]: etud.nom = 'Mkhechen Bilal'
In [13]: etud.save()
In [14]: etud
Out[14]: <Etudiant: Etudiant object (3)> ←
```

work :

tion



DB Browser for SQLite - F:\FS\Mes Cours\IHM\Exp\ Django2\website - MV\db.sqlite3

Fichier Édition Vue Outils Aide

Nouvelle Base de Données Ouvrir une Base de Données Enregistrer les modifications

Structure de la Base de Données Parcourir les données Éditer les Pragmas Exécuter le SQL

Table : listings_etudiant

	id	nom
	Filtre	
1	1	Akhrouf Ismail
2	2	Baaza Wafae
3	3	Mkhechen Bilal

Pour l'instant, un **QuerySet** ressemble beaucoup à une **list** Python.

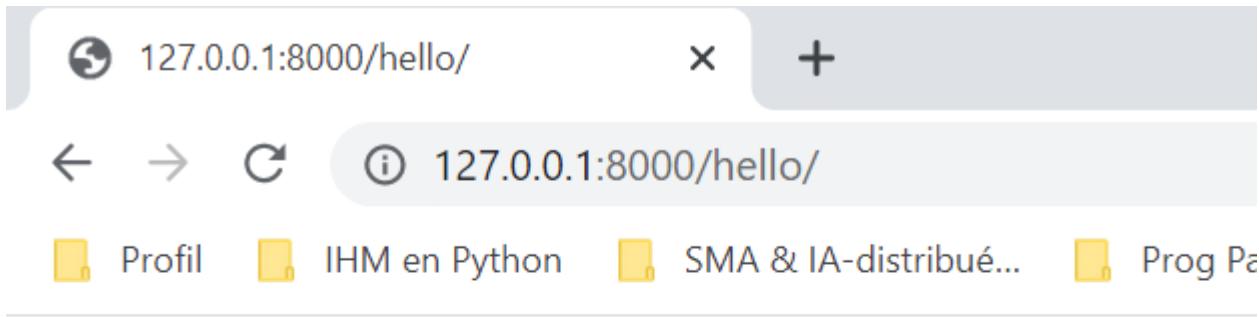
Django Framework : Modèle – M&J Vue

views.py*

```
# F:\FS\IHM\Exp\ Django\website\website\views.py

from django.http import HttpResponseRedirect
from django.shortcuts import render
from listings.models import Etudiant

def hello(request):
    etudiants = Etudiant.objects.all()
    return HttpResponseRedirect(f"""
        <h1>M2I & MQL : Hello Django !</h1>
        <p>La Liste des étudinats :<p>
        <ul>
            <li>{etudiants[0].nom}</li>
            <li>{etudiants[1].nom}</li>
            <li>{etudiants[2].nom}</li>
            <li>{etudiants[3].nom}</li>
        </ul>
    """)
```



M2I & MQL : Hello Django !

La liste des étudinats :

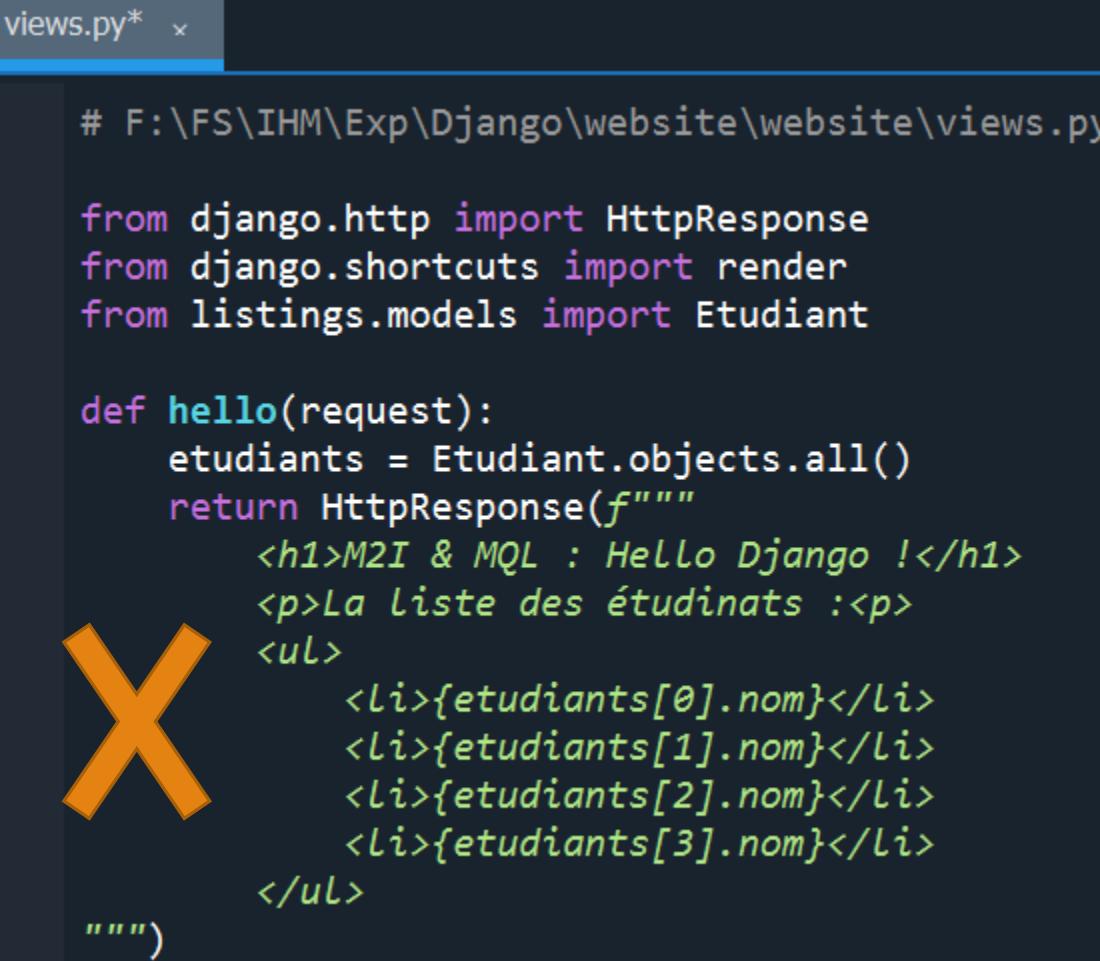
- Akhrouf Ismail
- Baaza Wafae
- Mkhechen Bilal
- El Houmam Semlali Mohamed

Django Framework : Modèle/Migration – Récap

- ✓ Un modèle définit les caractéristiques et les comportements d'un objet appartenant à l'application.
Ça ressemble à une classe standard, un modèle est capable d'enregistrer (« persister ») ses données dans une base de données.
- ✓ Une migration est un ensemble d'instructions qui font passer notre base de données d'un état à un autre, par exemple en créant une nouvelle table. Nous pouvons utiliser le shell de Django pour générer et exécuter les migrations.
- ✓ Dans une vue, nous pouvons récupérer des objets dans la base de données et afficher leurs données dans nos pages.
- ✓ Grâce aux modèles, le « M » de l'architecture MVT, nous avons enregistré des données dans notre base de données et nous les affichons dans nos pages.

Django Framework : Gabarits – Anti-pattern

- ✓ Notre **vue** commence déjà à avoir l'air un **peu chargée** et la quantité de HTML.
- ✓ Le problème est que notre vue a maintenant deux responsabilités :
 - ✓ **Sélectionner tous les objets** Etudiant de la base de données : la logique;
 - ✓ **Afficher les noms** de ces étudiants parmi d'autres contenus comme les titres et les paragraphes : la présentation.
- ✓ **Anti-pattern** : cette vue est une partie de notre application a trop de responsabilités, ça devient ingérable !!!!
- ✓ **Design pattern**, il faut déplacer la responsabilité de la présentation hors de la vue et en la plaçant à sa place légitime : un **gabarit**.



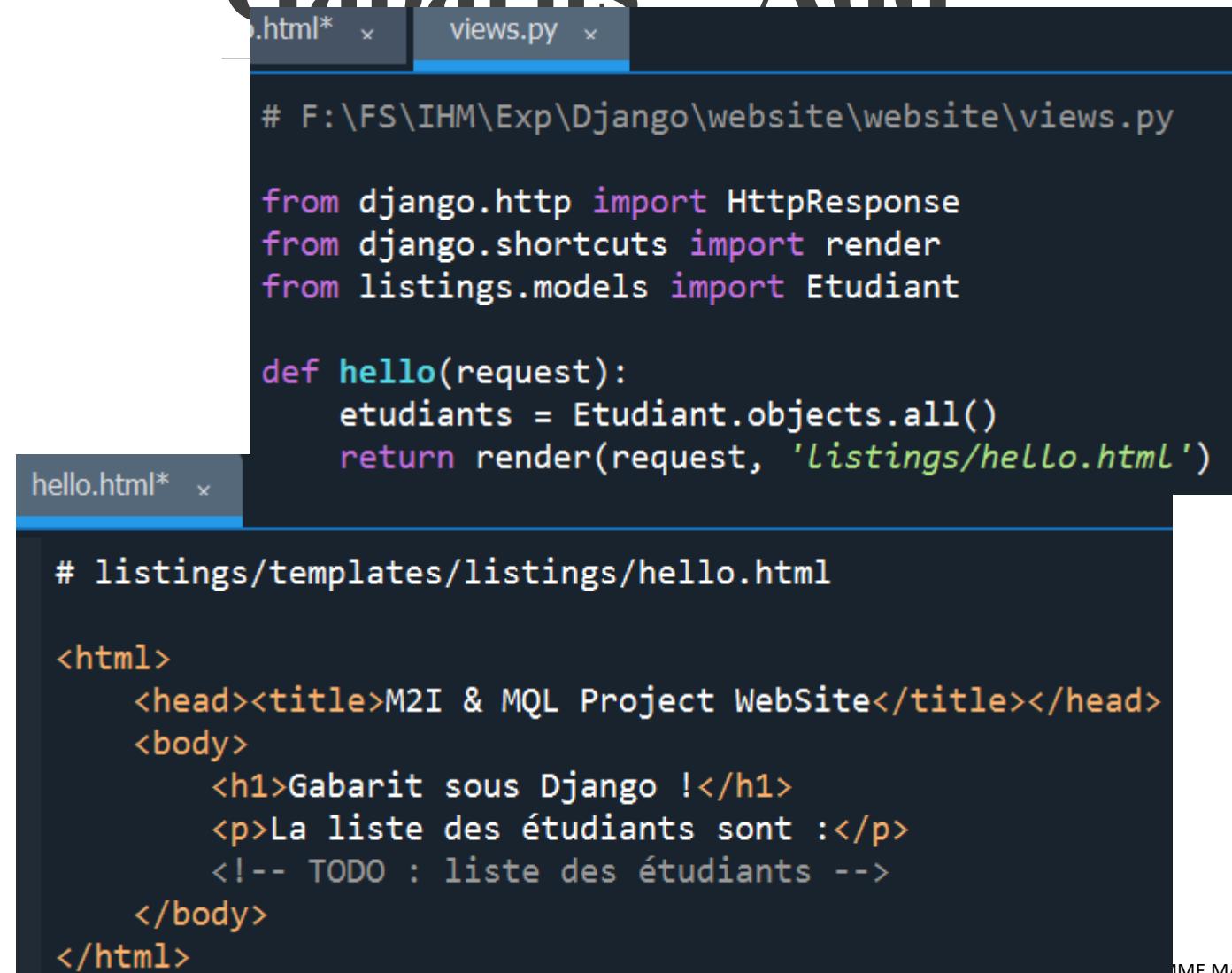
The screenshot shows a code editor window with a dark theme. The file is named 'views.py*' and contains Python code. A large orange 'X' is overlaid on the left side of the code area, indicating that the code is problematic. The code itself is as follows:

```
# F:\FS\IHM\Exp\ Django\website\website\views.py

from django.http import HttpResponse
from django.shortcuts import render
from listings.models import Etudiant

def hello(request):
    etudiants = Etudiant.objects.all()
    return HttpResponse(f"""
        <h1>M2I & MQL : Hello Django !</h1>
        <p>La liste des étudiants :</p>
        <ul>
            <li>{etudiants[0].nom}</li>
            <li>{etudiants[1].nom}</li>
            <li>{etudiants[2].nom}</li>
            <li>{etudiants[3].nom}</li>
        </ul>
    """)
```

Django Framework : Gabarits – Add



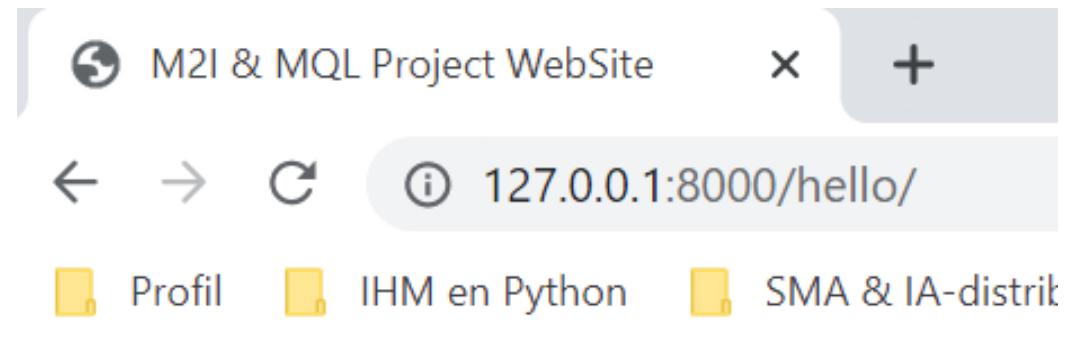
```
# F:\FS\IHM\Exp\ Django\website\website\views.py

from django.http import HttpResponseRedirect
from django.shortcuts import render
from listings.models import Etudiant

def hello(request):
    etudiants = Etudiant.objects.all()
    return render(request, 'listings/hello.html')

# listings/templates/listings/hello.html

<html>
    <head><title>M2I & MQL Project WebSite</title></head>
    <body>
        <h1>Gabarit sous Django !</h1>
        <p>La liste des étudiants sont :</p>
        <!-- TODO : liste des étudiants -->
    </body>
</html>
```



listings/templates/listings/hello.html

Gabarit sous Django !

La liste des étudiants sont :

Django Framework :

```

html  x  views.py  x

# F:\FS\IHM\Exp\ Django\website\website\views.py

from django.http import HttpResponse
from django.shortcuts import render
from listings.models import Etudiant

def hello(request):
    etudiants = Etudiant.objects.all()
    return render(request, 'listings/hello.html', {'etudiants': etudiants})

```

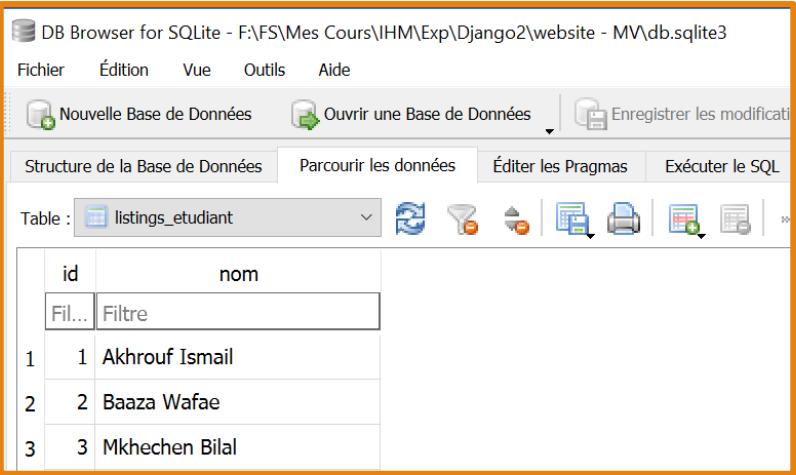
```

hello.html  x  views.py  x

# listings/templates/listings/hello.html

<html>
  <head><title>M2I & MQL Project WebSite</title></head>
  <body>
    <h1>Gabarit sous Django !</h1>
    <p>La liste des étudiants sont :</p>
    <ul>
      <li>{{ etudiants.0.nom }}</li>
      <li>{{ etudiants.1.nom }}</li>
      <li>{{ etudiants.2.nom }}</li>
    </ul>
  </body>
</html>

```



M2I & MQL Project WebSite

← → ⌂ ⓘ 127.0.0.1:8000/hello/

Profil IHM en Python SMA & IA-distril

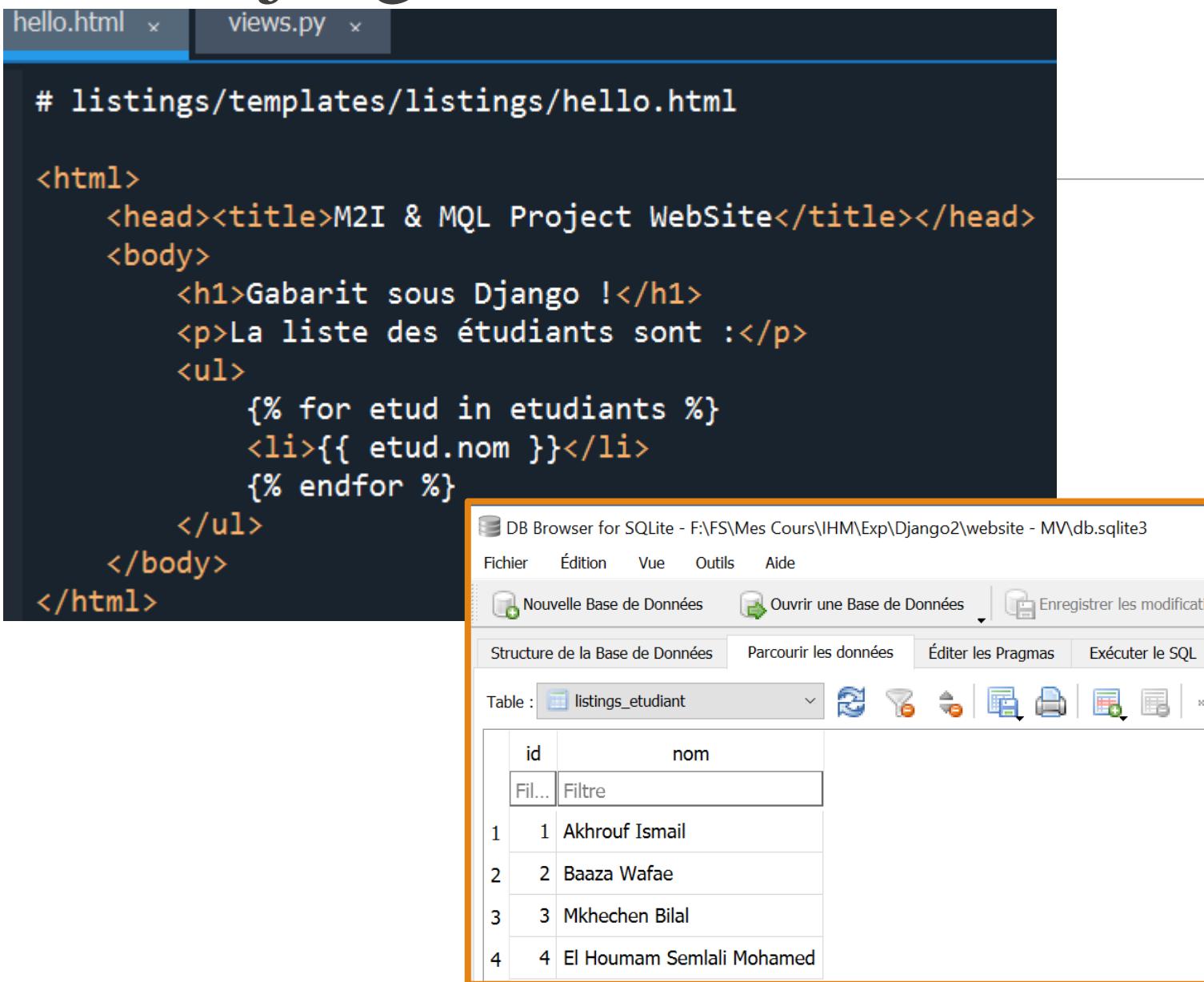
listings/templates/listings/hello.html

Gabarit sous Django !

La liste des étudiants sont :

- Akhrouf Ismail
- Baaza Wafae
- Mkhechen Bilal

Django Framework : Gabarits – Add



The screenshot shows a code editor with two tabs: 'hello.html' and 'views.py'. The 'hello.html' tab contains the following code:

```
# listings/templates/listings/hello.html

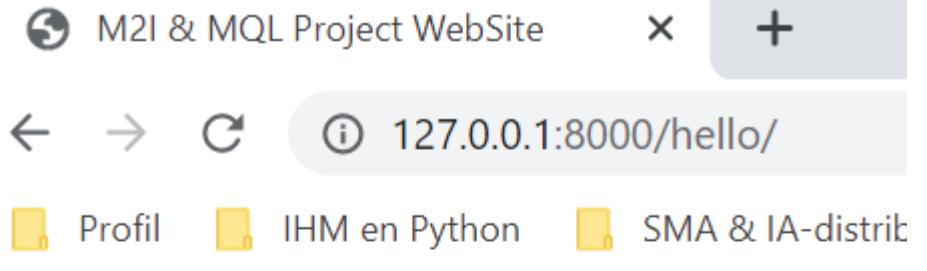
<html>
  <head><title>M2I & MQL Project WebSite</title></head>
  <body>
    <h1>Gabarit sous Django !</h1>
    <p>La liste des étudiants sont :</p>
    <ul>
      {% for etud in etudiants %}
      <li>{{ etud.nom }}</li>
      {% endfor %}
    </ul>
  </body>
</html>
```

The 'views.py' tab contains the following code:

```
# listings/templates/listings/hello.html
```

Below the code editor is a screenshot of DB Browser for SQLite showing a table named 'listings_etudiant' with four rows of data:

	id	nom
Filtre	1	Akhrouf Ismail
	2	Baaza Wafae
	3	Mkhechen Bilal
	4	El Houmam Semlali Mohamed



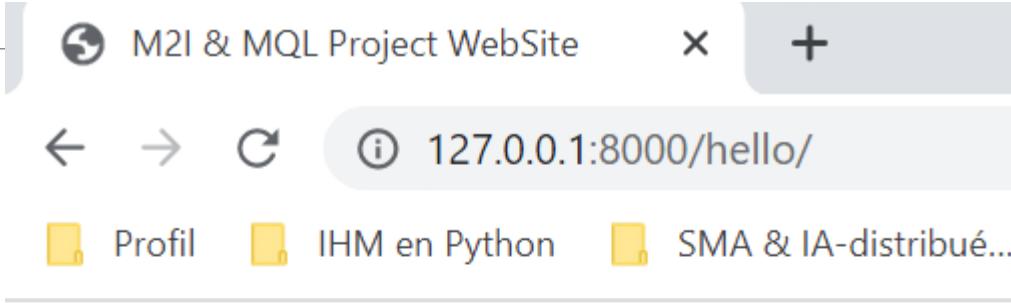
```
# listings/templates/listings/hello.html
```

Gabarit sous Django !

La liste des étudiants sont :

- Akhrouf Ismail
- Baaza Wafae
- Mkhechen Bilal
- El Houmam Semlali Mohamed

Django Framework : Gabarits – Add



The screenshot shows a browser window titled "M2I & MQL Project WebSite". The address bar shows the URL "127.0.0.1:8000/hello/". Below the address bar are three menu items: "Profil", "IHM en Python", and "SMA & IA-distribué...". The main content area displays the rendered Django template "hello.html".

```
# listings/templates/listings/hello.html

<html>
  <head><title>M2I & MQL Project WebSite</title></head>
  <body>
    <h1>Gabarit sous Django !</h1>
    <p>La liste des étudiants sont :</p>
    <ul>
      {% for etud in etudiants %}
        <li>{{ etud.nom|upper }}</li>
      {% endfor %}
    </ul>
    <p>La liste contient {{ etudiants|length }} étudiants.</p>
  </body>
</html>
```

listings/templates/listings/hello.html

Gabarit sous Django !

La liste des étudiants sont :

- AKHROUF ISMAIL
- BAAZA WAFAE
- MKHECHEN BILAL
- EL HOUMAM SEMLALI MOHAMED

La liste contient 4 étudiants.



Flask
web development,
one drop at a time

Application Web sur Python

FLASK

Application Web sur Python FLASK : About

- ✓ Flask est un micro-framework python pour réaliser des applications web évolutives. Flask dépend de la boîte à outils **WSGI** de **Werkzeug** et du moteur de templates **Jinja**.
- ✓ Un micro framework est un framework qui tente de fournir uniquement les composants absolument nécessaires à un développeur pour créer une application.
- ✓ Flask, un micro framework est conçu pour la construction d'API pour un autre service ou une autre application.
- ✓ Le micro dans le micro framework signifie que Flask vise à garder le code de base simple mais extensible.
- ✓ En définissant seulement le moteur de templates et un système de routes, Flask vous laisse le choix de personnaliser (en ajoutant des packages ou en développant les vôtres) pour la gestion des formulaires par exemple, vous avez donc la main sur votre code.



Application Web sur Python

FLASK : KeyWords / Install

- ✓ Pour comprendre ce qu'est Flask, vous devez comprendre quelques termes généraux.
- ✓ **WSGI** : Web Server Gateway Interface (WSGI) a été adopté comme norme pour le développement d'applications Web Python. WSGI est une spécification pour une interface universelle entre le serveur Web et les applications Web.
- ✓ **Werkzeug** : Il s'agit d'une boîte à outils WSGI, qui implémente des requêtes, des objets de réponse et d'autres fonctions utilitaires. Cela permet de créer un framework Web par-dessus. Le framework Flask utilise Werkzeug comme l'une de ses bases.
- ✓ **jinja2** : jinja2 est un moteur de création de modèles populaire pour Python. Un système de création de modèles Web combine un modèle avec une certaine source de données pour rendre des pages Web dynamiques.
- ✓ Pour installer FLASK : **pip install Flask**



Application Web sur Python

FLASK : Lancer une application

- ✓ Lancer l'application :

set FLASK_APP=Hello.py

flask run

- ✓ Taper sur le navigateur:

http://127.0.0.1:5000/

```
Sélection C:\WINDOWS\system32\cmd.exe - flask run
(base) C:\Users\User>cd f:
F:\FS\IHM\Exp\Flask

(base) C:\Users\User>f:

(base) F:\FS\IHM\Exp\Flask>cd fs
Le chemin d'accès spécifié est introuvable.

(base) F:\FS\IHM\Exp\Flask>set FLASK_APP=Hello.py

(base) F:\FS\IHM\Exp\Flask>flask run
 * Serving Flask app "Hello.py"
 * Environment: production

     Use a production WSGI server instead.
 * Debug mode: off
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

Application Web sur Python FLASK : Hello Application

- ✓ Après l'importation de la classe Flask, il faut instancier la classe Flask.
- ✓ Le décorateur **route()** pour définir l'url à laquelle la méthode **hello()** sera accessible.
- ✓ Dans la méthode **hello()**, défini la réponse à envoyer à l'utilisateur.

The image shows a code editor window and a browser window side-by-side. The code editor on the left displays the file F:\FS\IHM\Exp\Flask\Hello.py with the following content:

```
F:\FS\IHM\Exp\Flask\Hello.py
Hello.py* ×

1 from flask import Flask
2
3 app = Flask(__name__)
4
5 @app.route('/')
6 def hello():
7     return 'M2I & MQL : Test du Framework Flask ! '
```

The browser window on the right shows the URL 127.0.0.1:5000 in the address bar, and the page content "M2I & MQL : Test du Framework Flask !" below it. The browser interface includes back, forward, and refresh buttons, as well as links to other files like 'Profil', 'IHM en Python', and 'SMA & IA-distribué'.

Application Web sur Python

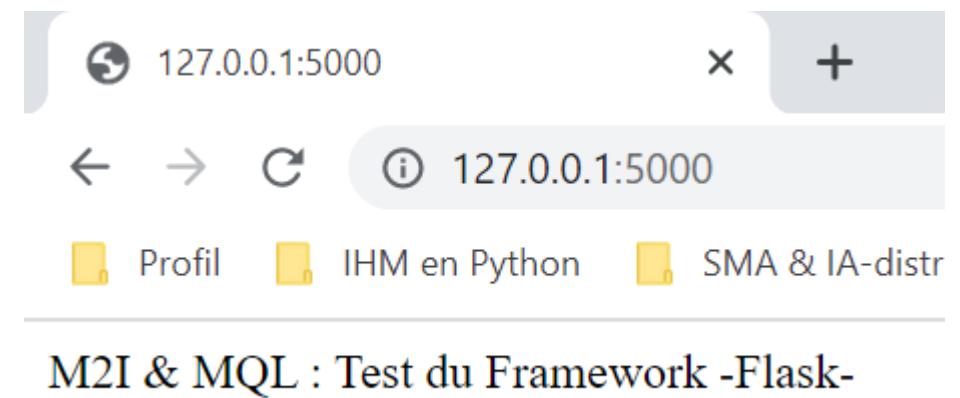
FLASK : Run Application in code

```
from flask import Flask

app = Flask(__name__)

@app.route('/')
def hello():
    return 'M2I & MQL : Test du Framework -Flask-'

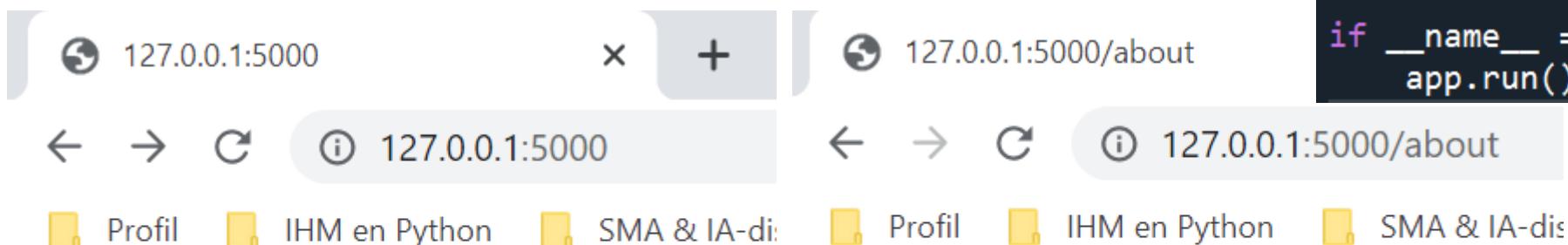
if __name__ == '__main__':
    app.run()
```



Application Web sur Python

FLASK : Les routes

- ✓ Dans tout framework web, il y a un système de gestion de routes.
- ✓ Les routes vont nous permettre de faire le lien entre la requête envoyée par l'utilisateur et la réponse que nous devons renvoyer à l'utilisateur.
- ✓ Pour déclarer une route, il faut utiliser le décorateur **route()** de la classe **Flask**, puis nous lui donner en paramètre la fonction à exécuter quand l'URL correspond.



M2I & MQL -Flask- : Home Page

M2I & MQL -Flask- : About Page

```
from flask import Flask

app = Flask(__name__)

@app.route('/')
def homepage():
    return 'M2I & MQL -Flask- : Home Page'

@app.route('/about')
def about():
    return 'M2I & MQL -Flask- : About Page'

if __name__ == '__main__':
    app.run()
```

Application Web sur Python

FLASK : Les routes avec paramètres

- ✓ GETTER : Pour faire passer une variable à travers l'url et l'intégrer comme paramètre de la route:

```
from flask import Flask

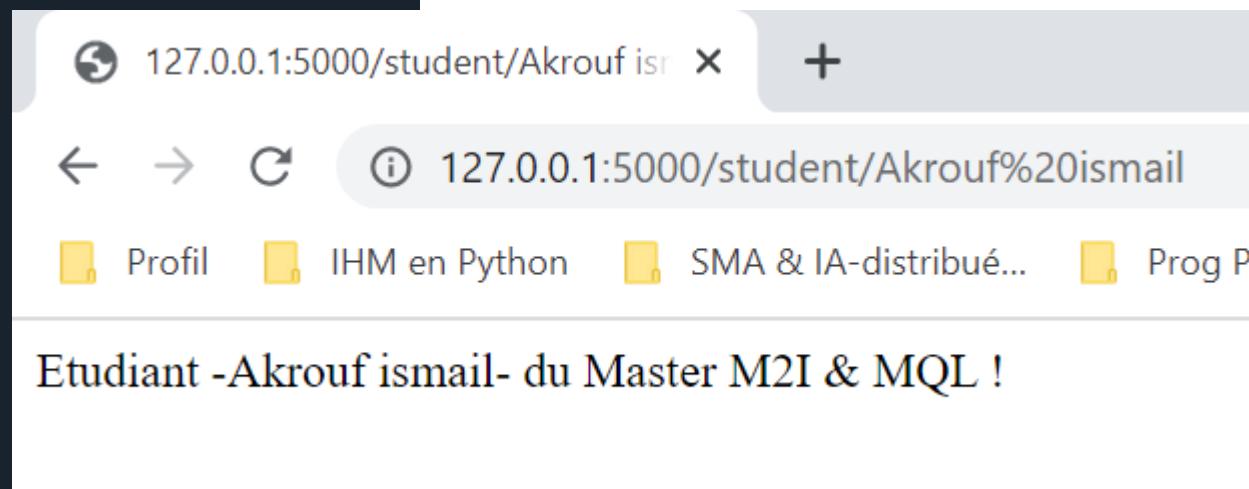
app = Flask(__name__)

@app.route('/')
def homepage():
    return 'M2I & MQL -Flask- : Home Page'

@app.route('/about')
def about():
    return 'M2I & MQL -Flask- : About Page'

@app.route('/student/<name>')
def hello(name):
    return 'Etudiant -{}- du Master M2I & MQL !'.format(name.capitalize())

if __name__ == '__main__':
    app.run()
```



Application Web sur Python FLASK : Les Templates

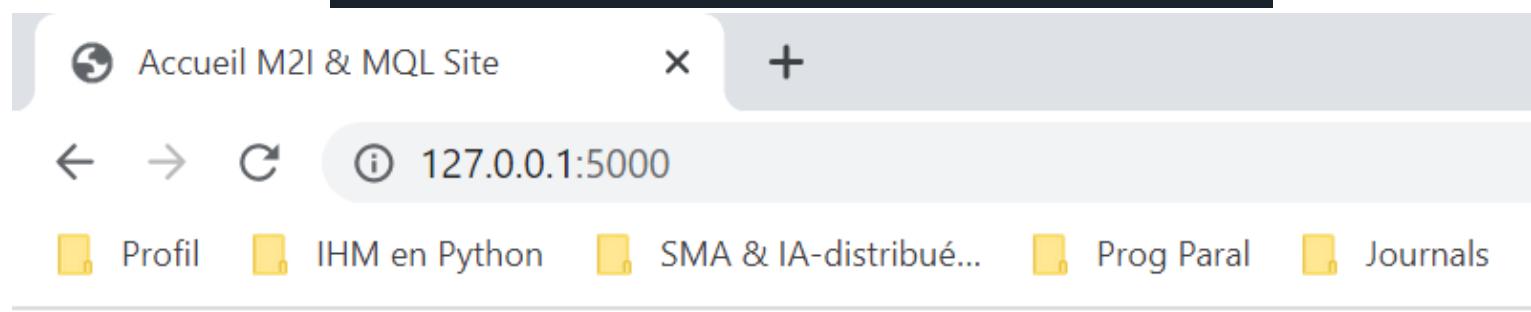
- ✓ Flask utilise **Jinja2** comme moteur de templates, cela permettre d'avoir des fichiers lisibles (retour des pages en HTML), qui ne contiennent pas du Python. Pour retourner un Template, nous allons utiliser la méthode ***render_template()*** du module Flask.

```
accueil.html ×
1 <!-- templates/homepage.html -->
2
3 <!DOCTYPE html>
4 <html>
5   <head>
6     <title>Accueil M2I & MQL Site</title>
7     <meta charset="utf-8">
8   </head>
9   <body>
10    <h1>Page d'Accueil du site fondé en Flask.</h1>
11  </body>
12</html>
```

```
from flask import Flask, render_template

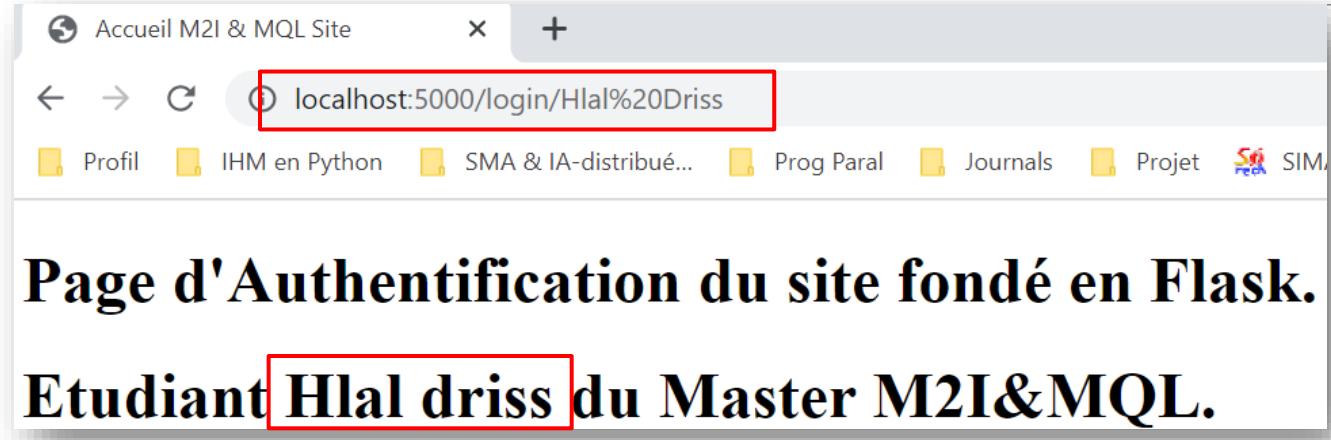
app = Flask(__name__)

@app.route('/')
def accueil():
    return render_template('accueil.html')
```



Page d'Accueil du site fondé en Flask.

Application Web sur Python FLASK : Les Templates paramétrés



The screenshot shows a browser window titled "Accueil M2I & MQL Site". The address bar contains "localhost:5000/login/Hhal%20Driss". The page content is a login form with the placeholder text "Etudiant Hhal driss du Master M2I&MQL.".

```
<!-- templates/login.html -->

<!DOCTYPE html>
<html>
<head>
  <title>Accueil M2I & MQL Site</title>
  <meta charset="utf-8">
</head>
<body>
  <h1>Page d'Authentification du site fondé en Flask.</h1>
  <h1>Etudiant {{ name|capitalize }} du Master M2I&MQL.</h1>
</body>
</html>
```

```
templateparama.py*  x

from flask import Flask, render_template

app = Flask(__name__)

@app.route('/')
def homepage():
    return render_template('homepage.html')

@app.route('/about/')
def about():
    return render_template('about.html')

@app.route('/login/')
@app.route('/login/<name>')
def hello(name='Abdoun'):
    return render_template('Login.html', name=name)

if __name__ == '__main__':
    app.run()
```

Application Web sur Python **FLASK : Using HTTP methods**

- ✓ Possible d'utiliser des méthodes HTTP dans Flask.
- ✓ Le protocole HTTP est la base de la communication de données sur le World Wide Web. Différentes méthodes de récupération de données à partir de l'URL spécifiée sont définies dans ce protocole. Les méthodes sont décrites ci-dessous.
 - ✓ GET: envoie des données sous forme simple ou non chiffrée au serveur.
 - ✓ HEAD: envoie des données sous forme simple ou non chiffrée au serveur sans corps.
 - ✓ HEAD: envoie les données du formulaire au serveur. Les données ne sont pas mises en cache.
 - ✓ PUT: remplace la ressource cible par le contenu mis à jour.
 - ✓ DELETE: supprime la ressource cible fournie en tant qu'URL.
- ✓ Par défaut, la route Flask répond aux requêtes GET. Cependant, cette préférence peut être modifiée en fournissant un argument de méthodes au décorateur route().

Application Web sur Python FLASK : Using HTTP methods

- ✓ *templateparam.py* : Utiliser la méthode POST dans le routage d'URL : un formulaire HTML (*login.html*) use la méthode *POST* pour envoyer des données de formulaire à une URL.



M2I & MQL - FLASK: Login Page

Fichier | F:/FS/IHM/Exp/Flask/TemplateParam/login.html

Profil IHM en Python SMA & IA-distribué... Prog Paral Journ

Page d'Authentification en Flask.

Saisir votre Nom :

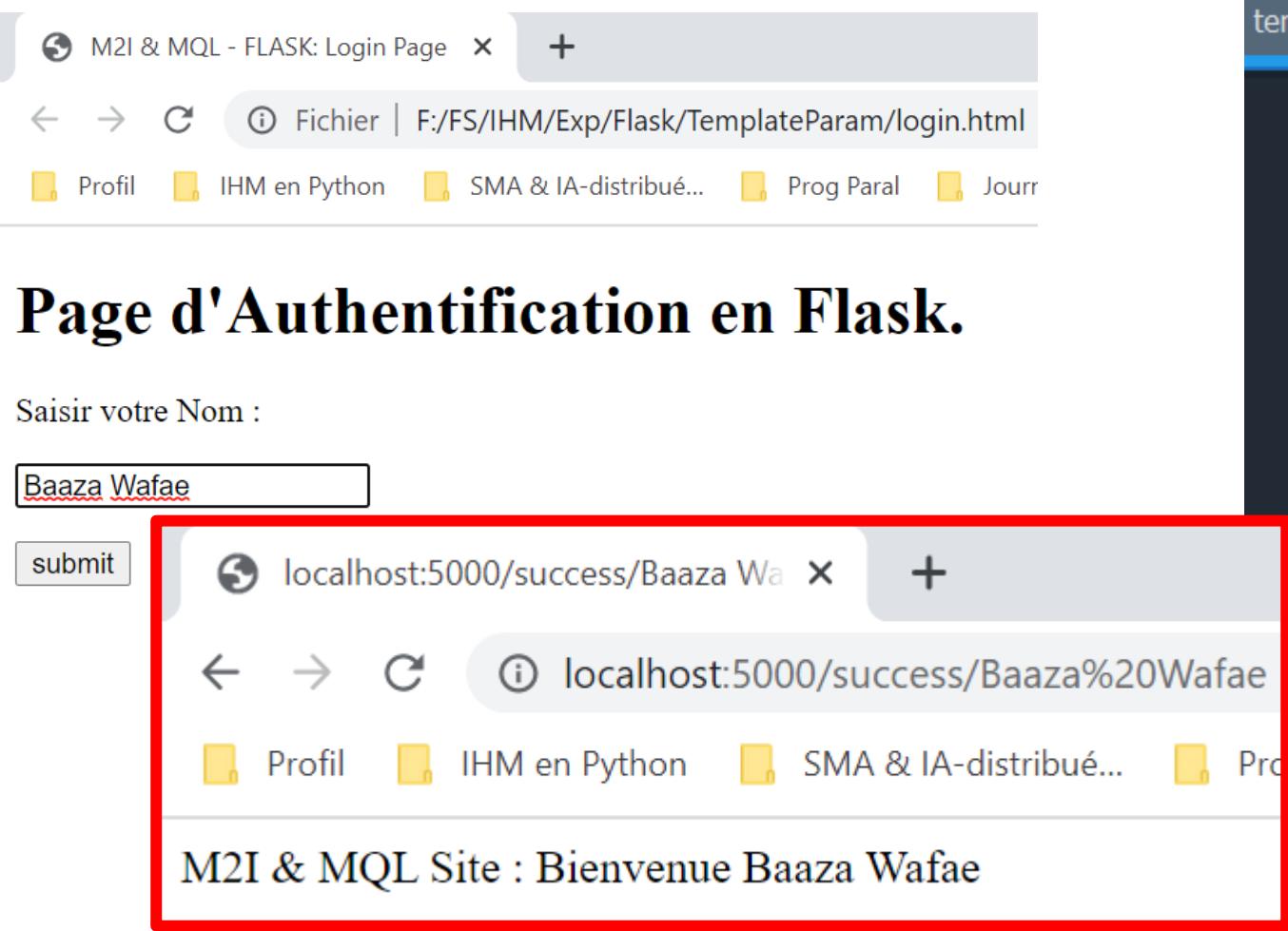
Baaza Wafae

submit

```
<!-- |TemplateParam/login.html -->

<!DOCTYPE html>
<html>
<head>
    <title>M2I & MQL - FLASK: Login Page</title>
    <meta charset="utf-8">
</head>
<body>
    <h1>Page d'Authentification en Flask.</h1>
    <form action = "http://localhost:5000/login" method = "post">
        <p>Saisir votre Nom :</p>
        <p><input type = "text" name = "nm" /></p>
        <p><input type = "submit" value = "submit" /></p>
    </form>
</body>
</html>
```

Application Web sur Python FLASK : Using HTTP methods



M2I & MQL - FLASK: Login Page

Fichier | F:/FS/IHM/Exp/Flask/TemplateParam/login.html

Profil IHM en Python SMA & IA-distribué... Prog Paral Journ

Page d'Authentification en Flask.

Saisir votre Nom :

submit

localhost:5000/success/Baaza Wa

localhost:5000/success/Baaza%20Wafae

Profil IHM en Python SMA & IA-distribué... Prog Paral

M2I & MQL Site : Bienvenue Baaza Wafae

```
templateparam.py
```

```
from flask import Flask, redirect, url_for, request

app = Flask(__name__)

@app.route('/success/<name>')
def success(name):
    return 'M2I & MQL Site : Bienvenue %s' % name

@app.route('/Login',methods = [ 'POST', 'GET'])
def login():
    if request.method == 'POST':
        user = request.form['nm']
        return redirect(url_for('success',name = user))
    else:
        user = request.args.get('nm')
        return redirect(url_for('success',name = user))

if __name__ == '__main__':
    app.run()
```

Application Web sur Python

DJANGO # FLASK



Flask est un micro-Framework de développement web, très léger et facile à prendre en main. Flask est aujourd’hui très utilisé pour le développement d’API dans une architecture en micro-services.

- ✓ Année création : 2010
- ✓ Langage : Python
- ✓ Open source : Oui, sur GitHub
- ✓ Nb. contributeurs : + de 600

Officiel : <https://flask.palletsprojects.com/>



Django est un Framework de développement web full stack codé en Python. Django permet d'uniformiser le développement de son application en suivant une architecture bien précise.

- ✓ Année création : 2005
- ✓ Langage : Python
- ✓ Open source : Oui, sur GitHub
- ✓ Nb. contributeurs : + de 2000

Officiel : <https://docs.djangoproject.com/>

Application Web sur Python **DJANGO : PYRAMID # FLASK**

- ✓ **Django** est un **framework Python open-source complet et gratuit**.
- ✓ Les fonctionnalités de Django le rend très évolutif, vraiment rapide et très polyvalent. :
 - ✓ Authentification,
 - ✓ Routage d'URL,
 - ✓ Moteur de modèles,
 - ✓ **ORM** (Object Relational Mapper)
 - ✓ Migration de schémas de bases de données (Django v.1.7+),
- ✓ Django utilise son **ORM** pour mapper des objets sur des tables de base de données. Le même code fonctionne avec différentes bases de données et n'est pas difficile à transférer d'une base de données à l'autre. Les principales bases de données avec lesquelles Django travaille sont PostgreSQL, MySQL, SQLite et Oracle.
- ✓ Avec Django, c'est possible de créer n'importe quelle application web, du projet à petite échelle au site web complexe.



Application Web sur Python

FLASK : PYRAMID # DJANGO

- ✓ Flask est un framework Python (**licence BSD**). Flask dépend de la boîte à outils **WSGI** de Werkzeug et du template **Jinja2**.
- ✓ Flask permet d'aider à construire une base solide d'applications web.
- ✓ La légèreté et la modularité de Flask le rendent facilement adaptable aux besoins des développeurs. Il comprend un certain nombre de fonctionnalités utiles prêtes à l'emploi :
 - ✓ Un serveur de développement intégré et un débogueur rapide
 - ✓ Support intégré aux essais unitaires
 - ✓ Envoi de la demande RESTful
 - ✓ Prise en charge des cookies sécurisés (sessions côté client)
 - ✓ Conformité à la norme WSGI 1.0
 - ✓ Basé sur l'Unicode
 - ✓ Possibilité de brancher n'importe quel ORM
 - ✓ Traitement des requêtes HTTP



Application Web sur Python

PYRAMID : DJANGO # FLASK

- ✓ **Pyramid** est une **application web** basée sur Python, **en open-source**, et le deuxième framework le plus populaire. Son but est de faire le plus possible avec un minimum de complexité.
- ✓ Fonctionnant sous Python 3, **Pyramid** suit les progrès technologiques.
- ✓ La caractéristique la plus frappante de Pyramid est sa capacité à bien fonctionner avec de petites et grandes applications, et aussi :
 - ✓ Applications à fichier unique
 - ✓ Génération d'URL
 - ✓ Configuration extensible
 - ✓ Modèles globaux et spécifications des actifs
 - ✓ Authentification et autorisation souples
 - ✓ Test, soutien et documentation complète sur les données
 - ✓ Prédictions de vue et nombreuses vues par itinéraire,
- ✓ Avec Pyramid, un développeur peut décider du langage de Template, des bibliothèques de génération et de la couche de base de données.



Pyramid™

IHM : Others use cases !?

1. SMART IHM : LA CONCEPTION D'IHM BASÉE SUR L'INTELLIGENCE ARTIFICIELLE
2. SOYA : IHM EN 3D
3. INDUSTRIE 4.0 : SYSTÈME DE CONTRÔLE INDUSTRIEL - SCADA

IHM : Other Use Cases !?

Smart IHM : Déf

- ✓ **Smart IHM** permet d'avoir des interfaces homme-machine (IHM) assistées par l'Intelligence Artificielle (IA).

- ✓ **Smart IHM** sont des systèmes qui impliquent des interactions entre le système technique d'une machine et un opérateur dans un contexte et à travers divers canaux :
 - ✓ la machine est elle-même une hiérarchie de systèmes interconnectés capables d'acquérir des données en temps réel, de les traiter, d'utiliser l'intelligence artificielle et de créer des modèles ;
 - ✓ l'opérateur présente des spécificités et des performances intra-individuelles et inter-individuelles, notamment la mobilisation de capacités cognitives et d'entraînement pour une tâche donnée.



IHM : Other Use Cases !?

Smart IHM : Search !

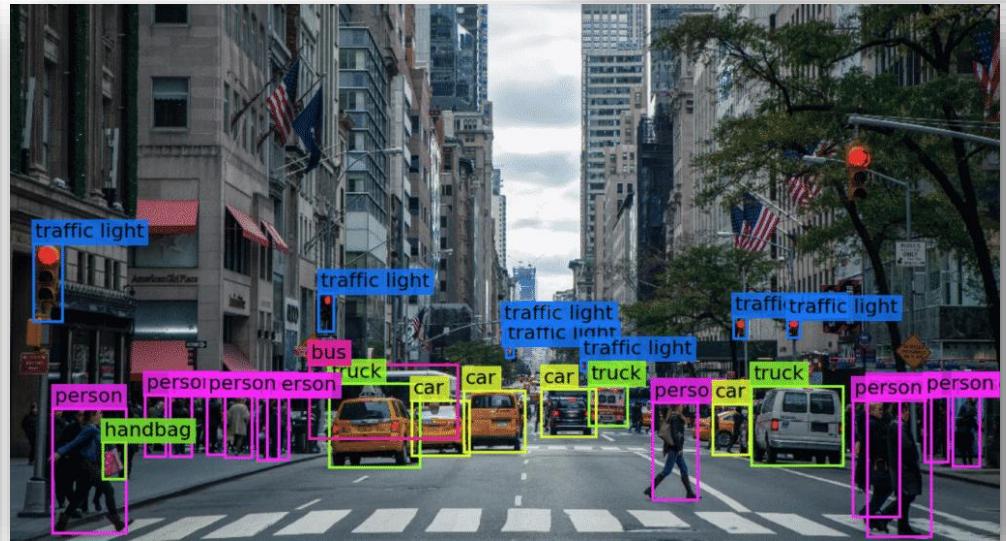
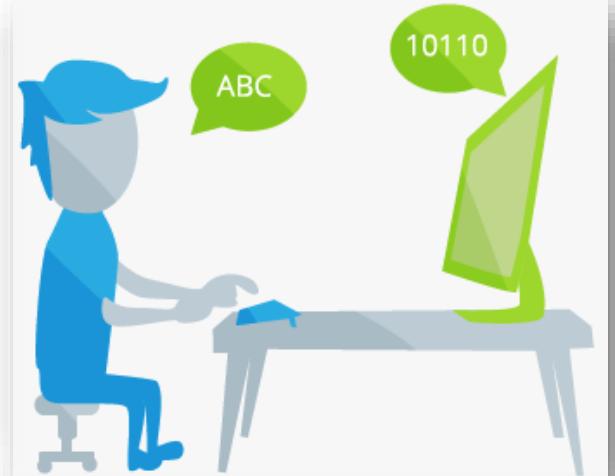
- ✓ Smart IHM impliquent à la fois des échanges directionnels de machine à homme et d'homme à machine.
- ✓ Smart IHM visent à fournir des informations, à mettre en place des décisions, à agir et à vérifier les conséquences de l'action pour la meilleure récupération.
- ✓ L'IHM basée sur l'IA est actuellement une **tendance** de fond pour les **consommateurs** et **l'industrie**, mais c'est aussi un domaine universitaire actif. Les activités de recherche fournissent des informations prospectives sur les tendances industrielles et de consommation.



IHM : Other Use Cases !?

Smart IHM : Apps

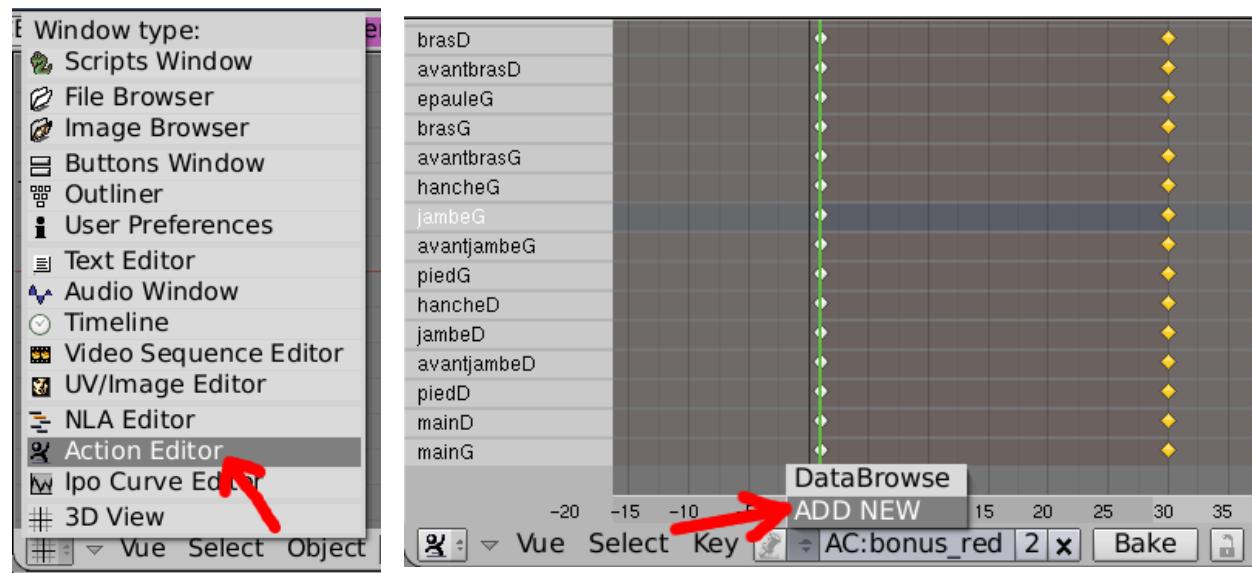
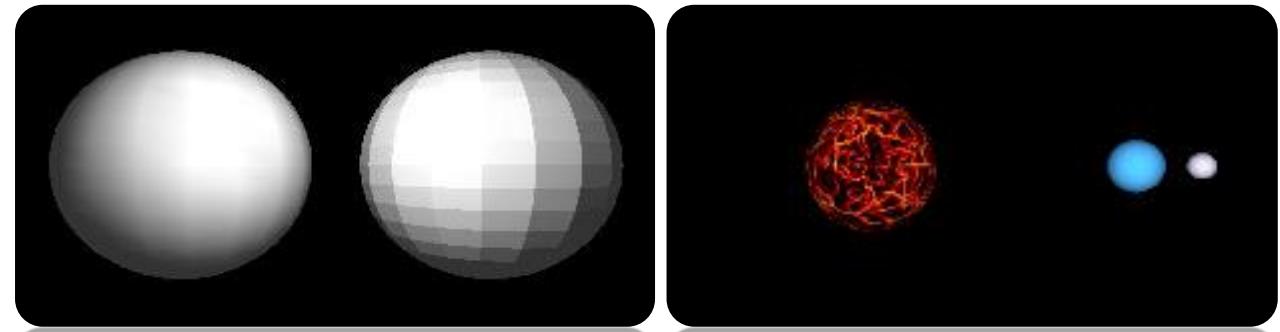
- ✓ Traitement automatique des langues
- ✓ Représentation des connaissances
- ✓ Vision par ordinateur
- ✓ Intelligence artificielle distribuée
- ✓ Planification / Optimisation
- ✓ Philosophie de l'intelligence artificielle



IHM : Other Use Cases !?

Soya : IHM en 3D

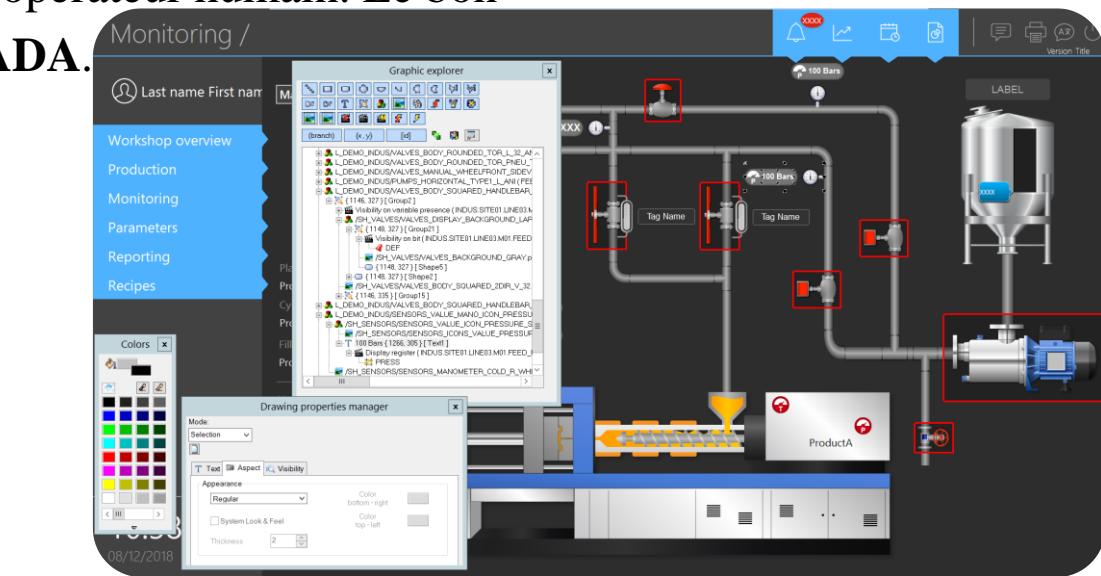
- ✓ **Soya 3D** est un moteur 3D orienté objet écrit en Python, développé par Jean-Baptiste "Jiba" Lamy.
- ✓ **Soya 3D** a pour principale qualité de permettre un développement rapide de jeux ou d'interface 3D, le tout en Python.
- ✓ **Soya 3D** offre les principales capacités que l'on peut attendre d'un moteur 3D comme la gestion basique de scène, système des ombres portées, etc. Ainsi que certaines caractéristiques uniques visant à rendre le développement de jeux plus facile et plus rapide.
- ✓ **Soya** est un logiciel libre multi-plateformes, sous licence GNU GPL.
- ✓ **Soya** est édité en Pyrex (un mix de C et de Python) et Python.



IHM : Other Use Cases !?

SCADA - Industrie 4.0 : About

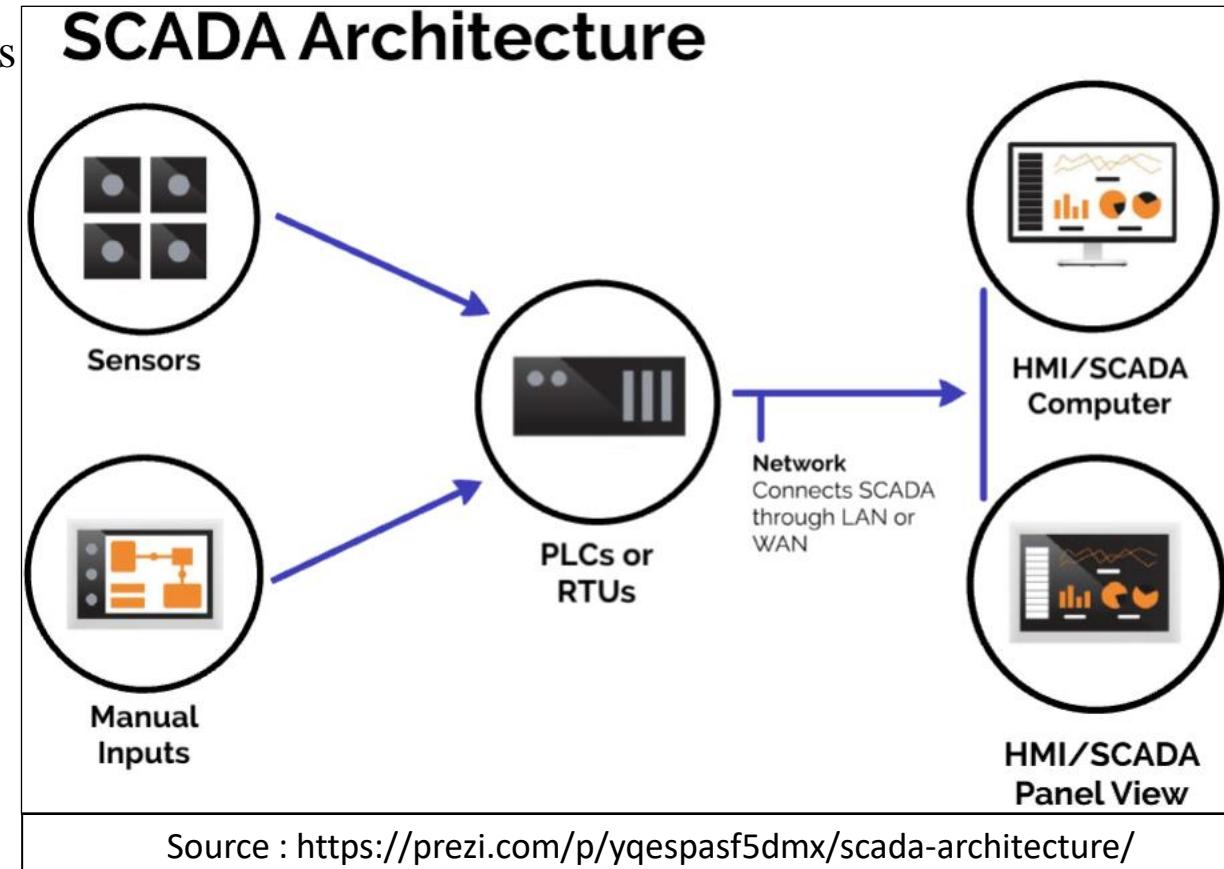
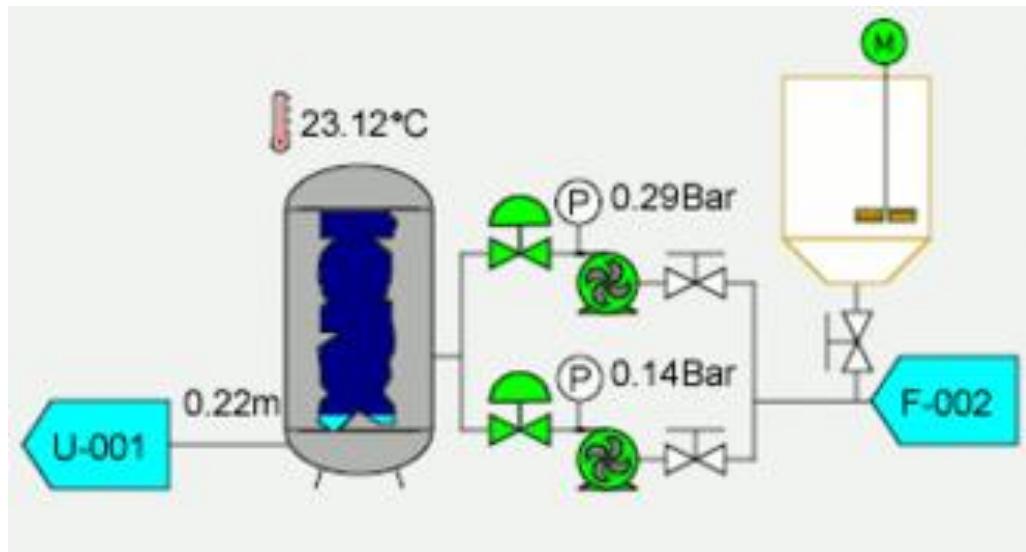
- ✓ **SCADA** est un système de contrôle, de supervision et d'acquisition de données utilisé en industrie. Les processus industriels et d'infrastructure sont normalement surveillés par des *Dashboard systems*.
- ✓ **SCADA** fait référence à la surveillance et au contrôle de supervision dans les organisations industrielles et IHM est un sous-ensemble de SCADA.
- ✓ IHM se connecte à tous les processus, puis présente ces données à un opérateur humain. Le bon fonctionnement de l'IHM est essentiel au bon fonctionnement du **SCADA**.



IHM : Other Use Cases !?

SCADA - Industrie 4.0 : Architecture

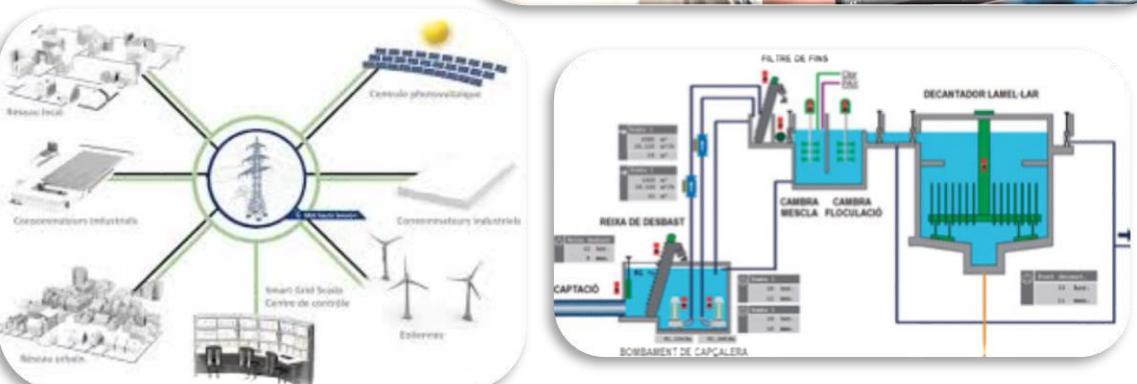
- ✓ L'IHM est essentielle au succès de tout SCADA car ce composant fournit toutes les données à un opérateur humain. Cette entrée est analysée par l'opérateur pour prendre les décisions en conséquence.
- ✓ IHM est associé aux bases de données de SCADA et fournit des informations cruciales pour la maintenance et le dépannage.
- ✓ L'opérateur reçoit généralement des informations de l'IHM sous forme de graphiques ou de synoptiques.



IHM : Other Use Cases !?

SCADA - Industrie 4.0 : Apps

- ✓ SCADA a des applications dans la *fabrication*, la production, la *production d'énergie*,
- ✓ Même les processus d'installation dans telles que les *aéroports*, les *gares de chemin de fer*, les *navires* et les stations spatiales. utilisent le SCADA pour surveiller et contrôler divers processus.
- ✓ L'IHM utilisé pour le SCADA varie considérablement d'un téléphone cellulaire à un *réacteur nucléaire*.





Ergonomie des IHMs

30 CONCEPTS-CLÉS DE L'UTILISABILITÉ
D'INTERFACE HOMME-MACHINE (IHM)

Ergonomie des IHMs : 30 concepts-clés de l'utilisabilité d'IHM

-
- ✓ Le principe des 7±2 éléments
 - ✓ La règle des 2 secondes
 - ✓ La règle des 3 clics
 - ✓ La loi de Pareto (20/80)
 - ✓ Les 8 règles d'or de la conception
 - ✓ La loi de Fitts
 - ✓ La pyramide inversée
 - ✓ Satisfaction
 - ✓ Le syndrome de l'oisillon
 - ✓ Banner blindness
 - ✓ Les effets Cliché et Zeigarnik
 - ✓ Les lois de la Gestalt
 - ✓ L'effet d'autoréférence
 - ✓ L'eye-tracking (ou oculométrie)
 - ✓ Le pli (fold)
 - ✓ La zone fovéale
 - ✓ Les annotations
 - ✓ Dégradation élégante
 - ✓ La granularité
 - ✓ Les zones sensibles
 - ✓ La lisibilité perceptive
 - ✓ La navigation en démineur
 - ✓ La navigation mystère
 - ✓ La cohérence visuelle
 - ✓ L'enrichissement progressif
 - ✓ La lisibilité cognitive
 - ✓ La conception centrée utilisateur
 - ✓ La vigilance
 - ✓ Le design intuitif (Walk-Up-And-Use)
 - ✓ Les Wireframes

Ergonomie des IHMs : 30 concepts-clés de l'utilisabilité d'IHM

1. Le principe des 7±2 éléments : « Limiter le nombre d'options de navigation d'un menu à 7 »

Le **cerveau humain** étant limité dans sa **capacité à traiter l'information**, celui-ci aborde la complexité en traitant l'information par blocs. Les études menées par *George A. Miller* tendent à montrer que notre mémoire à court-terme ne peut *retenir que 5 à 9 éléments* à la fois.

2. La règle des 2 secondes : « Minimiser le temps d'utilisateur d'interface en 2 Seconds d'attente de réponse, Passage, Lancement, ... »

En vertu de ce principe, un utilisateur **ne** devrait **pas** avoir à **attendre plus de 2 secondes** certaines réponses du système, comme le passage d'une application à une autre, ou le lancement d'un programme. Le choix des 2 secondes est arbitraire, mais il s'agit d'un ordre de grandeur raisonnable. Ce qui est sûr, c'est que **moins l'utilisateur attend, meilleure est son expérience**.

3. La règle des 3 clics : « 3 clics pour arriver à l'objectif »

Selon cette règle, les utilisateurs tendent à **abandonner un site** lorsqu'ils ne sont pas capables d'accéder à l'information ou au service en l'espace de **3 clics**. Cette règle met donc l'accent sur la nécessité de fournir une **navigation claire**, d'une **structuration logique**, et d'une **arborescence simple** à appréhender afin d'arriver à l'**objectif en 3 clics**.

4. La loi de Pareto ou la règle des 20/80 : « Identifier les actions dont les impacts seront les plus significatifs, en ciblant les 20% d'utilisateurs »

La loi de Pareto postule que **80% des effets** viennent de **20% des causes**. Bien connue du marketing, cette loi peut aussi s'appliquer en matière d'utilisabilité. Par exemple, des progrès très importants peuvent souvent être accomplis en identifiant les **20% d'utilisateurs**, des activités ou des processus qui représentent 80% du profit, et en ciblant les efforts sur cette population.

Ergonomie des IHMs : 30 concepts-clés de l'utilisabilité d'IHM

5. Les 8 règles d'or de la conception d'interface : « à respecter »

Ben Shneiderman a proposé un ensemble de règles empiriques pour la **conception d'interfaces**, applicables dans la plupart des **systèmes interactifs** :

- ✓ Recherchez la **cohérence** avant tout.
- ✓ Offrez des **raccourcis** aux utilisateurs avancés.
- ✓ Offrez un retour d'information (**feedback**).
- ✓ Concevez des **dialogues** avec une fin explicite.
- ✓ Offrez des moyens simples de **gestion des erreurs**.
- ✓ Permettez des **retours** en arrière **simples**.
- ✓ Faites en sorte que l'**utilisateur** se sente le **maître**.
- ✓ Ne sollicitez pas trop la mémoire à court-terme de l'utilisateur.

6. La loi de Fitts : « Adopter la taille et la distance de l'interface pour plus d' efficacité et d'accessibilité des interfaces »

Enoncé par *Paul Fitts* en 1954, ce modèle du mouvement humain prévoit que le temps nécessaires pour atteindre rapidement une **zone cible** est en fonction de la **distance** et de la **taille de la cible**. C'est le cas par exemple d'un mouvement de souris. Jouer sur la **taille** et la **distance** de la cible peut alors influer sur l'**accessibilité** et l'**efficacité** de l'interface.

7. La pyramide inversée : « Appliquer le principe de la pyramide inversée, car les utilisateurs recherchant une gratification immédiate »

La **pyramide inversée** est un **mode d'écriture** où le **résumé d'un article** est présenté dès le **début** de celui-ci. Cette approche est bien connue des **journalistes** : l'**article débute** donc par une **conclusion**, suivie d'**éléments-clés** et finalement d'**élément de moindre importance**. Les internautes recherchant une gratification immédiate, la pyramide inversée est importante aussi bien en matière d'écriture qu'en matière d'expérience utilisateur.

Ergonomie des IHMs : 30 concepts-clés de l'utilisabilité d'IHM

8. Satisfaction : « Résoudre efficacement la requête Web user »

Les **utilisateurs** du web ne recherchent pas la meilleure façon de résoudre leur problèmes, mais bien la **résolution effective** de ceux-ci. Ils ne sont pas intéressés par la solution la plus **intelligente**, mais aux contraire favorisent les solutions qu'ils jugent « *satisfaisantes* », même s'il existe des solutions plus performantes sur le long terme.

9. Le syndrome de l'oisillon : « Attention au re-design »

Le **syndrome de l'oisillon** décrit la tendance naturelle des **utilisateurs** à **adhérer au premier type d'interface expérimenté**, et à **juger** les **autres** en fonction de leur degré de **similarité avec le premier**. La conséquence est que les **utilisateurs préfèrent** en général les **systèmes** auxquels ils sont **habitues et rejettent** ceux qui ne leur sont **pas familiers**. Ce qui posent des **problèmes** en matière de **re-design** : les utilisateurs, habitués à un design, se sentent souvent mal à l'aise lorsque une nouvelle interface est mise en place.

10. Banner blindness : « Mettre plus des liens et des zones de texte, les utilisateurs d'interface cherchent l'information et ignorent toute publicité »

Les **internautes** tendent à **ignorer** tout ce qui ressemble à de la **publicité**, et ils sont même assez bons à ce jeu. La publicité est bien perçue, mais elle est ignorée la plupart du temps. La raison en est que les **utilisateurs cherchant de l'information** se concentrent uniquement sur les zones où ils pensent pouvoir trouver celle-ci, c-à-d les liens et les zones de texte.

11. Les effets Cliffhanger et Zeigarnik : « Les internautes mémorisent mieux les publicités, l'exploité pour fidéliser les utilisateurs d'un site : RSS »

L'**être humain** ne supporte **pas l'incertitude** : qu'une **question** lui vienne à l'esprit, il lui faut absolument une **réponse**. L'effet *Cliffhanger*, qui est fréquemment utilisé pour pimenter les intrigues, exploite cette tendance en introduisant une interruption soudaine qui laisse l'utilisateur face à révélation choc, ou à une situation non-résolue. Cet effet est aussi très souvent utilisé dans la publicité, où **poser des questions ouvertes ou provocantes pousse l'utilisateur à lire une publicité ou à cliquer sur une bannière**. Découvert en 1927, l'effet *Zeigarnik* désigne le fait qu'une **tâche a tendance à être mieux mémorisée** lorsque elle est **interrompue**. Appliqué au web, cet effet établit une connexion émotionnelle avec les lecteurs, et est extrêmement efficace en terme de marketing. Les internautes mémorisent alors mieux les publicités. Cet effet est aussi exploité pour fidéliser les utilisateurs d'un site (ex : « **Abonnez-vous à notre fil RSS pour ne pas manquer la suite de l'article** »).

Ergonomie des IHMs : 30 concepts-clés de l'utilisabilité d'IHM

12. Les lois de la Gestalt : « Respecter la psychologie de la forme pour une conception d'interfaces assez cohérente »

Aussi appelé « *psychologie de la forme* », cet ensemble de principes constitue un des fondamentaux de la conception d'interfaces.

- ✓ ***Loi de proximité*** : lorsque nous percevons une collection d'objets, nous considérons les objets proches les uns des autres comme formant un groupe.
- ✓ ***Loi de similitude*** : lorsque nous voyons des objets jugés similaires, nous les voyons comme formant un groupe.
- ✓ ***Loi de Prägnanz*** : au sein de notre champ visuel, certains objets sont perçus comme étant de premier plan (les figures), tandis que les autres sont perçus comme d'arrière plan (le fond).
- ✓ ***Loi de symétrie*** : nous concevons souvent les objets comme des formes symétriques ayant un centre.
- ✓ ***Loi de bonne continuité*** : nous tendons naturellement à clôre ou compléter des objets qui ne sont en réalité pas clos (ex : le logo IBM)



13. L'effet d'autoréférence : « Exploiter l'autoréférence des utilisateurs (ses concepts, ses expériences) pour une communication plus efficace »

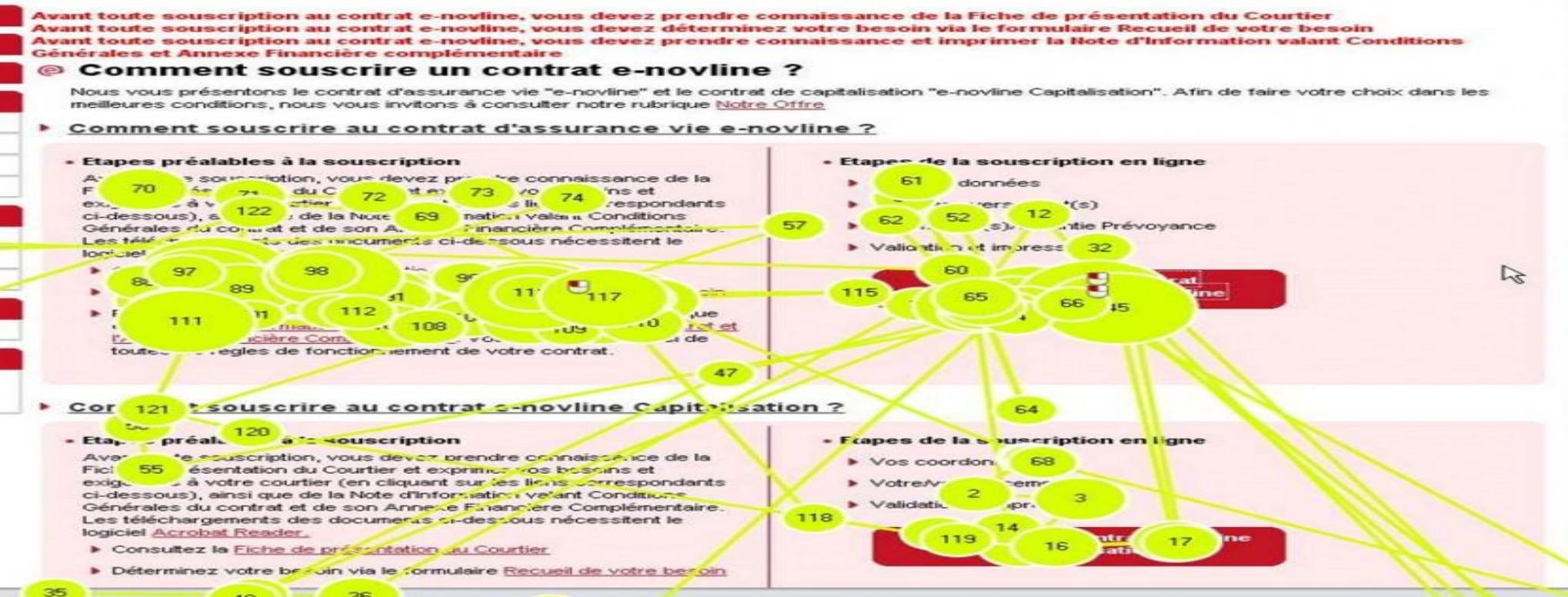
L'autoréférence est particulièrement importante en matière d'écriture web, et peut améliorer significativement la communication entre l'auteur et le lecteur. Nous nous souvenons en effet davantage des choses lorsque elles ont un lien avec nos concepts, nos expériences. Il est donc souhaitable d'exploiter l'autoréférence quand on souhaite communiquer plus efficacement.

14. L'eye-tracking (ou oculométrie) : « Tracer le parcours du regard sur une interface, pour une post-évaluation du mode de navigation d'un site »

L'eye-tracking est un procédé qui consiste à **enregistrer le parcours du regard** sur une **interface**, pour ensuite **dresser une carte des zones les plus vues**. Cette technique permet notamment d'**évaluer la facilité de navigation d'un site**.

15. Le pli (fold) : « Gérer rentablement les flods, pour une exploitation améliorée de la zone d'interface »

Le « **pli** » se définit comme étant la **limite inférieure** en dessous de laquelle le **contenu du site n'est plus visible**. Cette limite est dépendante de la résolution de l'écran. La région au dessus du pli, visible sans scrolling, est appelée « **page écran** ».



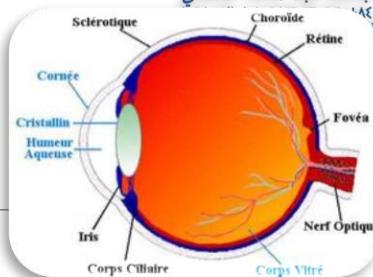
14. L'eye-tracking (ou oculométrie) : « Tracer le parcours du regard sur une interface, pour une post-évaluation du mode de navigation d'un site »

L'**eye-tracking** est un procédé qui consiste à **enregistrer le parcours du regard** sur une **interface**, pour ensuite **dresser** une carte des **zones les plus vues**. Cette technique permet notamment d'**évaluer la facilité de navigation d'un site**.

15. Le pli (fold) : « Gérer rentablement les flops, pour une exploitation améliorée de la zone d'interface »

Le « pli » se définit comme étant la **limite inférieure** en dessous de laquelle **le contenu du site n'est plus visible**. Cette limite est dépendante de la résolution de l'écran. La région au dessus du pli, visible sans scrolling, est appelée « page écran ».

Ergonomie des IHMs : 30 concepts-clés de l'utilisabilité d'IHM



16. La zone fovéale : « Percevoir tout le détail du contenu dans la zone fovéale, pour plus du regard »

La **fovée** est la **partie** de l'**œil humain** en charge de la ***vision centrale***. Cette ***vision centrale*** nous **sert à lire, conduire, regarder la télévision, et à accomplir toutes les activités où le besoin de détail est très important**. La **zone fovéale** désigne la **zone du regard** où le **maximum de détail** peut être perçu. cette zone occupe un angle de vision d'environ 2 degrés.

17. Les annotations : « Ajouter plus d'annotation pour une meilleure usabilité des interfaces du site »

L'**annotation** est une action automatique **fournissant des informations** sur la destination d'un lien ou la fonction d'un contrôle. Concernant les liens, une annotation peut être délivrée grâce à l'attribut « *title* ». Les **annotations** participent à l'**usabilité d'un site**, en permettant aux utilisateurs de savoir précisément où ils vont.

18. Dégradation élégante : « Fournir une information ou un service et rendre le site utilisable quelle que soit la configuration de l'utilisateur »

La **dégradation élégante** désigne la **capacité d'un site ou d'une page à fournir une information ou un service** même si certains de ses **composants ne peuvent être utilisés**. En pratique, cela revient à s'assurer que le **site reste utilisable** quelle que soit la **configuration de l'utilisateur**.

19. La granularité : « Adapter la granularité de l'information en fonction de l'utilisateur visé »

La **granularité** mesure à quel degré un **ensemble complexe d'informations** a été réduit en un nombre **plus petit d'éléments**. La granularité de l'information peut être adapté en fonction du public visé.

20. Les zones sensibles: « Profiter de la propriété CSS (:focus) pour bien gérer les zones sensibles des interfaces »

Les **zones sensibles** sont des **zones cliquables** qui changent de **forme** ou d'**apparence** une fois survolées ou **cliquées**. Ces zones utilisent souvent la propriété CSS **:focus**.

Ergonomie des IHMs : 30 concepts-clés de l'utilisabilité d'IHM

21. La lisibilité perceptive: « Assurer une Legibility du contenu des interface pour une perception visuelle assez apparente »

La lisibilité perceptive (legibility) mesure la facilité d'un texte à être lu. Elle ne s'intéresse qu'à la perception visuelle, et non à la compréhension du texte.

22. La navigation en démineur : « Le mode de navigation Minesweeping est très populaire, pour cela il faut éviter tout défaut de conception de l'interface »

Ce terme (*minesweeping*) désigne un mode de navigation à l'aveugle où l'utilisateur promène sa souris sur la page, en utilisant le changement d'apparence du curseur pour détecter les liens. Un tel comportement indique la plupart du temps un défaut de conception de l'interface.

23. La navigation Mystère : « MMN complique l'utilisation des interfaces de navigation car user est incapable d'identifier la destination des liens »

La navigation mystère (MMN : Mystery Meat Navigation) est un terme décrivant des interfaces de navigation difficilement utilisables du fait que les destinations des liens ne sont pas clairement identifiables. C'est souvent le cas des éléments de navigation à base d'icônes non-standards.

24. La cohérence visuelle : « Insister sur la cohérence visuelle, pour un repérage et une navigation efficaces »

La cohérence visuelle décrit la cohérence de l'agencement physique du site (places du logo, des éléments de navigation, typographie...). La cohérence est essentielle pour un repérage et une navigation efficaces.

25. L'enrichissement progressif : « Suivre une stratégie de conception d'interfaces à base des couches, pour une accessibilité des interfaces plus attractive »

L'enrichissement progressif est une stratégie de conception d'interfaces où les fonctionnalités sont ajoutées par couches. À la couche de base, accessible à tous les navigateurs, sont ajoutées des fonctionnalités supplémentaires pour les utilisateurs de navigateurs modernes. L'avantage principal de ce mode de conception est l'accessibilité des interfaces ainsi produites.

Ergonomie des IHMs : 30 concepts-clés de l'utilisabilité d'IHM

26. La lisibilité cognitive : « Utiliser des phrases et des vocabulaire plus simple pour améliorer la lisibilité cognitive de vos interfaces »

La **lisibilité cognitive** (readability) mesure *la facilité d'un texte à être compris*, en se basant sur la complexité des phrases et sur le vocabulaire employé. Le plus souvent, ce type de **lisibilité** se mesure en terme d'**âge** ou d'**années d'étude** nécessaires pour **comprendre le texte**. La lisibilité cognitive est à distinguer de la lisibilité perceptive.

27. La conception centrée utilisateur : « Établir une conception centrée sur les utilisateurs de l'interface en terme : Besoins, intérêts, comportements, »

La **conception centrée utilisateur** (CCU) est un mode de conception où les **utilisateurs**, leurs **besoins**, leurs **intérêts**, et leurs **comportements** définissent les **fondations d'un site** en termes de **structure**, de **navigation**, et d'**information**. La CCU est un standard, en particulier grâce à l'essor du web 2.0 où le contenu est généré pour une large part par les utilisateurs.

28. La vigilance : « Limiter la vigilance des utilisateurs, en mettant en Background toutes tâches longues, monotones ou répétitives »

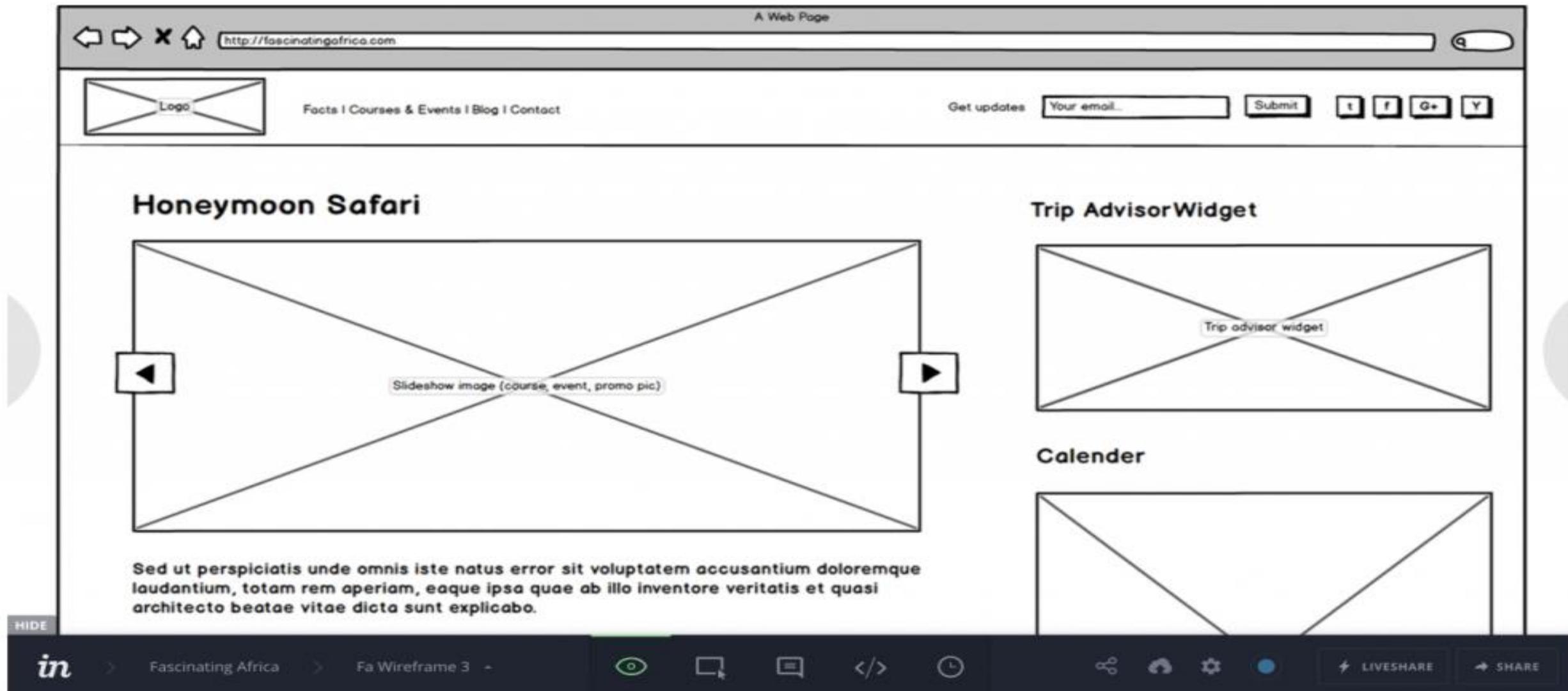
La **vigilance**, ou état de veille, est la **capacité à rester attentif** lors de l'**accomplissement de tâches longues, monotones ou répétitives**, comme la **relecture de documents**, la gestion d'emplois du temps, la sauvegarde régulière de fichiers etc. Dans les applications web modernes, les tâches de cet ordre sont automatisées et gérées en arrière-plan, ce qui améliore l'**usabilité** puisque la **vigilance** des utilisateurs est limitée.

29. Le design intuitif : « Utiliser la conception W2U pour une utilisation claire dès la première fois, sans aucune introduction ou formation »

Le **design intuitif** (Walk-Up-and-Use Design) : quand un **utilisateur** est capable d'**utiliser l'interface dès la première fois, sans aucune introduction ou formation**.

30. Les Wireframes : « Élaborer une maquette fonctionnelle du projet, afin de bien décrire les idées, les concepts et la structure de vos IHM »

Le **wireframe**, ou « **maquette fonctionnelle** », est un **squelette** qui décrit les **idées**, les **concepts** et la **structure d'une interface** ou d'un site (exemples ici et là). Les wireframes peuvent être utilisés pour présenter un projet de site aux parties prenantes. Le design d'un wireframe est en général très sommaire, et il s'agit même fréquemment de simples croquis (JPG) réalisés sur papier.



30. Les Wireframes : « Élaborer une maquette fonctionnelle du projet, afin de bien décrire les idées, les concepts et la structure de vos IHM »

Le wireframe, ou « maquette fonctionnelle », est un squelette qui décrit les **idées**, les **concepts** et la **structure d'une interface** ou d'un site (exemples ici et là). Les wireframes peuvent être utilisés pour présenter un projet de site aux parties prenantes. Le design d'un wireframe est en général très sommaire, et il s'agit même fréquemment de simples croquis (JPG) réalisés sur papier.

Ergonomie des IHMs : Récap 1

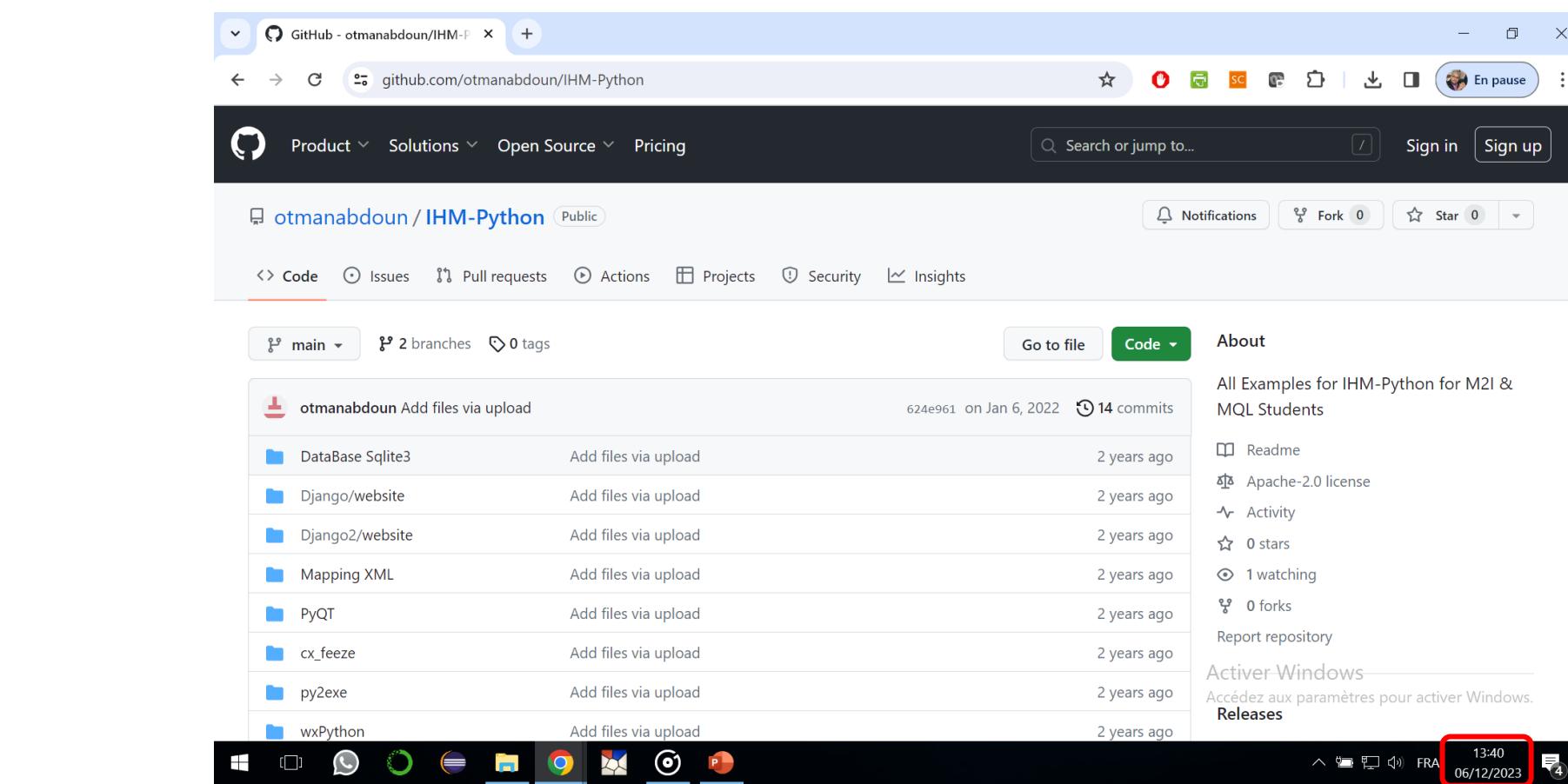
1. Limiter le nombre d'options de navigation d'un menu à 7. Cela dit ce principe fait débat.
2. Minimiser le temps d'utilisateur d'interface (2 Seconds d'attente de réponse, Passage, Lancement, ...).
3. Fournir une navigation claire, d'une structuration logique, et d'une arborescence simple à appréhender (3 clics pour arriver à l'objectif).
4. Identifier les actions dont les impacts seront les plus significatifs, en ciblant les 20% d'utilisateurs.
5. Respecter les 8 règles d'or de la conception d'interface proposé par *Ben Shneiderman*.
6. Adopter la taille et la distance de l'interface pour plus d'efficacité et d'accessibilité, selon *Paul Fitts*.
7. Appliquer le principe de la pyramide inversée, car les utilisateurs recherchant une gratification immédiate.
8. Résoudre efficacement la requête Web user, car il ne recherchent pas la meilleure façon de résoudre leur problèmes.
9. Attention au redesign : les utilisateurs, habitués à un design, se sentent souvent mal à l'aise lorsque une nouvelle interface est mise en place.
10. Mettre plus des liens et des zones de texte, les utilisateurs d'interface cherchent l'information et ignorent toute publicité.

Ergonomie des IHMs : Récap 2

11. Les internautes mémorisent alors mieux les publicités. Cet effet est aussi exploité pour fidéliser les utilisateurs d'un site : RSS.
12. Respecter la psychologie de la forme pour une conception d'interfaces assez cohérente.
13. Exploiter l'autoréférence des utilisateurs (ses concepts, ses expériences) pour une communication plus efficace.
14. Tracer le parcours du regard sur une interface, pour une post-évaluation du mode de navigation d'un site.
15. Gérer rentablement les flods, pour une exploitation améliorée de la zone d'interface.
16. Percevoir tout le détail du contenu dans la zone fovéale, pour plus du regard.
17. Ajouter plus d'annotation pour une meilleure usabilité des interfaces du site.
18. Fournir une information ou un service et rendre le site utilisable quelle que soit la configuration de l'utilisateur.
19. Adapter la granularité de l'information en fonction de l'utilisateur visé.
20. Profiter de la propriété CSS « :focus » pour bien gérer les zones sensibles des interfaces.

Ergonomie des IHMs : Récap 3

21. Assurer une Legibility du contenu des interface pour une perception visuelle assez apparente.
22. Le mode de navigation Minesweeping est très populaire, pour cela il faut éviter tout défaut de conception de l'interface.
23. Mystery Meat Navigation complique l'utilisation des interfaces de navigation car user est incapable d'identifier la destination des liens.
24. Insister sur la cohérence visuelle, pour un repérage et une navigation efficaces.
25. Suivre une stratégie de conception d'interfaces à base des couches, pour une accessibilité des interfaces plus attractive.
26. Utiliser des phrases et des vocabulaire plus simple pour améliorer la lisibilité cognitive de vos interfaces.
27. Établir une conception centrée sur les utilisateurs de l'interface en terme : Besoins, intérêts, comportements.
28. Limiter la vigilance des utilisateurs, en mettant en arrière plan toutes tâches longues, monotones ou répétitives.
29. Instaurer un design d'interface plus intuitif, pour une utilisation claire dès la première fois, sans aucune introduction ou formation.
30. Élaborer une maquette fonctionnelle du projet, afin de bien décrire les idées, les concepts et la structure de vos IHM.



All ressources in GITHUB

[HTTPS://GITHUB.COM/OTMANABDOUN/IHM-PYTHON.GIT](https://github.com/otmanabdoun/IHM-Python.git)

Module IHM :

Références

- ✓ Kim, Jin Woo, "Human Computer Interaction", Ahn graphics, 2012.
- ✓ Stéphanie Jean-Daubias, "Introduction à l'Interface Homme Machine", SJD LIRIS UCBL, 2020.
- ✓ Michel Beaudouin-Lafon, "40 ans d'interaction homme-machine : points de repère et perspectives", Interstices, 2007.
- ✓ Vitaly Friedman, "30 Usability Issues to be Aware of", 2007.
- ✓ Karim Bouzoubaa, "Interface Homme-Machine", EMI - Université Mohamed 5, 2013.
- ✓ Magnaudet, M. "Qu'est-ce que programmer une interface adaptative?", 2014.
- ✓ Gérard Swinnen, "Apprendre à programmer avec Python 3", Eyrolles, 2012.
- ✓ Bob Cordeau, Laurent Pointal, " Une introduction à Python 3", 2020.
- ✓ Mark Lutz, Learning, "Python : Powerful Object-Oriented Programming", O'Reilly Media, 6 octobre 2009.
- ✓ Patrick Fuchs et Pierre Poulain, "Introduction à la programmation Python", Université de Paris, 2021.
- ✓ Benoît Petitpas, "Bases de données en Python", 2015.
- ✓ Marvie, "Python et les interfaces graphiques", Université de Lille 1, 2020.
- ✓ Pierre Denis, Thibaut Cuvelier, "Créer des applications graphiques en Python avec PyQt5", 2017.
- ✓ Yassine Ben Salah, "Interface graphique : Qt Designer et PyQt5", 2019.
- ✓ Développez votre site web avec le framework Django, Rest de Savoir 2019.
- ✓ Gérard Rozsavolgyi, "Tuto-Django", 2021.



**End of course
-IHM-**



جامعة عبد المالك السعدي
Université Abdelmalek Essaadi

Interface Homme Machine

Pr. Abdoun Otman
Département Informatique,
Faculté des Sciences,
Université Abdelmalek Essaâdi, Tétouan

Module IHM :

Plan 1

1. Introduction aux IHM
2. Principes ergonomiques des IHM : Règles et standards
3. Sciences cognitives, ergonomie et guides de style
4. Présentation des technologies du Web : HTML/XHTML
5. Développement en PHP et interaction en AJAX
6. Manipuler les interfaces graphiques avec XML/XSLT
7. Programmation d'interface homme machine en langage Java : AWT
8. Les composants Java avancés : Swing
9. Développement Web en J2EE : Servlet & JSP

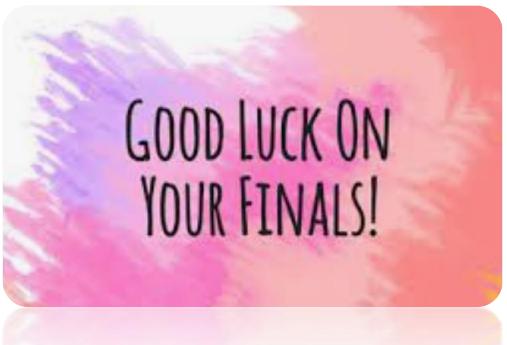


Module IHM :

Plan 2 : Proposé & Achevé

1. Introduction aux IHM : Définition et Historique
2. Principes ergonomiques des IHM : Règles et standards
3. Sciences cognitives, ergonomie et guides de style
4. **Programmation sur Python**
5. **Data Management en Python:**
 - a. Sérialisation avec **PICKLE** et **JSON**
 - b. Stockage avec **SQLITE3**
 - c. Mapping en **XML/XSLT**
6. Interface graphique pour Python :
 - a. **TKINTER** pour Tk
 - b. **PYQT** pour Qt
 - c. **WXPYTHON** pour wxWidgets
7. Application Web sur Python : **DJANGO**, **FLASK**,...
8. IHM challenge : Smart IHM, Soya, SCADA





Date de l'examen écrit :
Mercredi 03 Janvier
2024 à 13h - Amphi F



جامعة عبد المالك السعدي
تولوز | تولوز | Université Abdelmalek Essaadi

IHM

Interface Homme Machine

Pr. Abdoun Otman

Département Informatique,
Faculté des Sciences
Université Abdelmalek Essaâdi, Tétouan