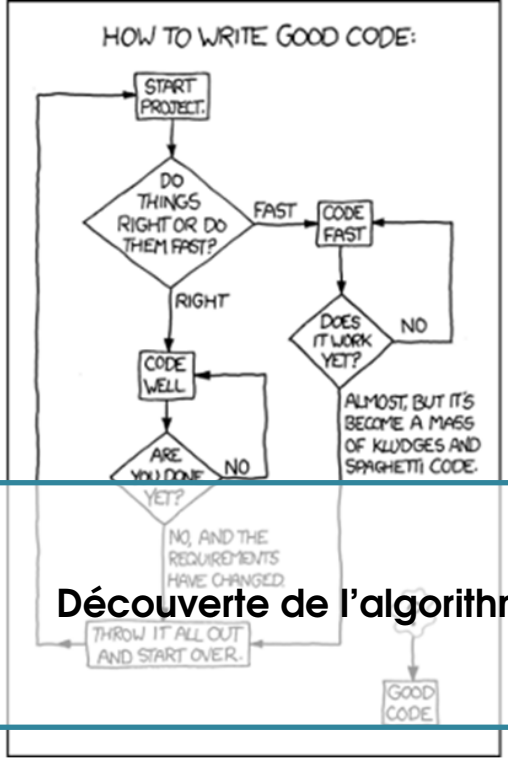
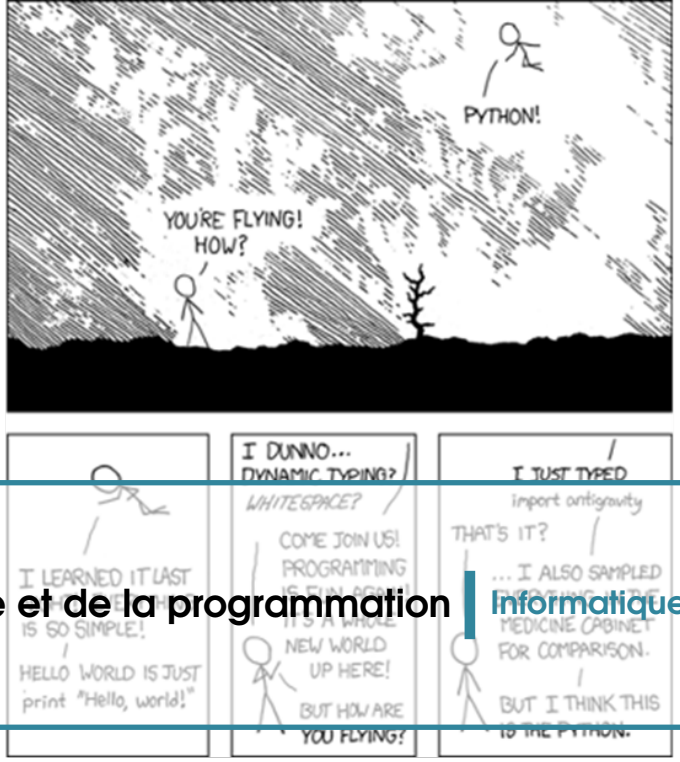




Découverte de l'algorithmique et de la programmation | Informatique

Cours

Chapitre 0

Utilisation de Python

Savoirs et compétences :

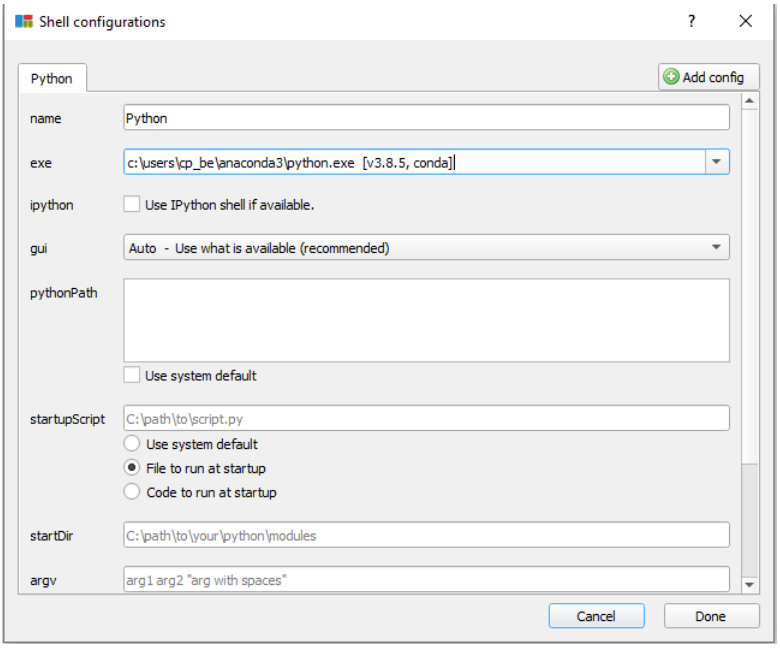
Xavier Pessoles
MPSI – PTSI

L'environnement de développement utilisé au lycée pour programmer en Pyhon est `Pyzo`.

1 Installation de PYZO sur votre ordinateur personnel

Rendez-vous sur le site `Pyzo.org` qui vous propose une fenêtre `Quickstart`.
L'installation se fait en quatre temps selon les éléments choisis :

1. Installation de l'IDE, choisir la version correspondant à votre système d'exploitation (Windows, MacOS, Linux) en 32 ou 64 bits ;
2. Installer la distribution `Anaconda` ;
3. Configurer votre shell au lancement de `Pyzo` la première fois dans la fenêtre en haut à gauche ;



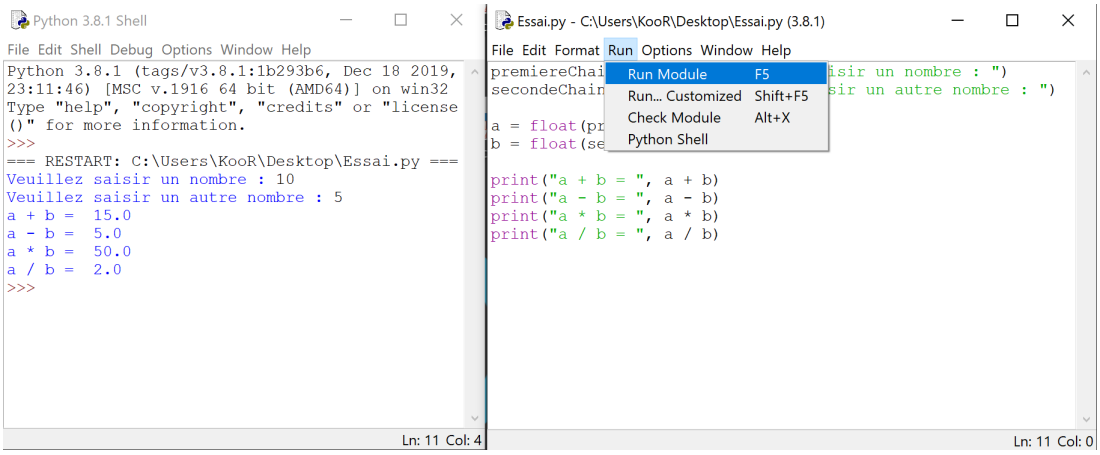
4. Les bibliothèques sont installées par défaut, l'étape 4 n'est pas utile. Sauf si vous avez dû installer une distribution plus légère.

2 Changement de thème de la fenêtre Pyzo

Si l'écran blanc de la fenêtre `Pyzo` fatigue vos yeux, vous pouvez le changer :
setting \ edit syntax styles... \
Vous pouvez choisir `solarized_dark`.

3 IDLE python

Vous serez amené en math-info à utiliser l'IDLE Python aussi installé sur les ordinateurs du lycée.
Moins convivial que `Pyzo`, il permet de réaliser les mêmes travaux en utilisant les deux types de fenêtres celle du `shell` (ou `console`) et celle d'`édition`.

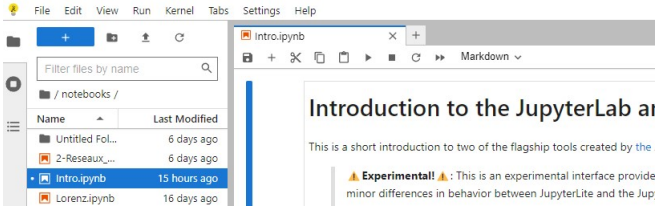


L'IDLE Python est disponible avec toute installation de Python. Avec Pyzo, aller dans dossier contenant Pyzo ou Anaconda puis dans le répertoire Lib\idlelib et lancer le programme idle.bat.
Le chemin peut être celui-ci C:\Users\...\anaconda3\Lib\idlelib par exemple.

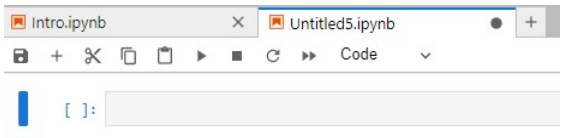
4 Python via des notebook : Jupyter, Capytale, Google Colab

4.1 Jupyter

Jupyter.org est accessible par votre moteur de recherche préféré.
Je vous propose : Try Jupyter puis JupyterLab



Ouvrir une page vierge avec le + de l'onglet et sélectionner notebook et Python.



Vous pouvez écrire des lignes de codes dans la case et les exécuter avec la flèche.

4.2 Capytale

Capytale est accessible par l'ENT du lycée via ressources numériques. L'utilisation est la même que celle de Jupyter.
Vous y trouverez des travaux à réaliser en autonomie.

Application 01

Applications – Bases

Savoirs et compétences :
❒ .

Découverte de quelques types dans l'interpréteur

Lancer PYZO.
Une fenêtre s'ouvre sur votre écran partagée en 3 zones, il est nécessaire d'ouvrir le shell en choisissant la version de python, alors quelques lignes de message introductif s'inscrivent. C'est l'**interpréteur** de Python, qu'on appelle aussi le **shell**.
Les symboles >>> invitent à taper quelque chose (invite de commande).

Fonctionnement : tapez une *instruction*, par exemple $1+1$, et tapez "Entrée" :

```
>>> 1 + 1
       2
```

Différents types de nombres

Le shell de Python peut s'utiliser comme une calculatrice pour manipuler des nombres. Selon la nature (entiers, réels, complexes) des nombres que l'on souhaite manipuler, il y aura différents *types* sous python pour les représenter.

Ci-dessus, on a manipulé 1 qui est de type `integer` (*entier* en anglais).
On manipule aussi les "nombres à virgule", dits nombres *flottants*, dont le type s'appelle `float` en python :

```
>>> 3.1 + 10.09
       13.19
>>> type(2.7)
       <type 'float'>
```

! Bien remarquer que l'on utilise la convention anglosaxonne : un point et non une virgule.

Exercice 1 –

Question 1 *Taper les deux instructions suivantes :* `3` , `3.0` *puis* `'3'` *puis demander le type de ces données. Commenter*

Question 2 *Taper les trois instructions suivantes :* `3.`, `0.1` *puis* `.1`. *Commenter.*

Exercice 2 –

Question 3 *Testez les opérateurs suivants avec des entiers et/ou des flottants :*

`-` `*` `/` `//` `%` `^` `**`

Question 4 *Que dire du résultat et du type du résultat pour les opérateurs /, // et %, par exemple comment relier les résultats obtenus de ces deux opérations : 9//2 et 9%2 ? L'opérateur ^ renvoie-t-il le résultat attendu ? Que ce passe-t-il en cas d'opération entre un entier et un flottant ?*

Les fonctions `float` et `int` permettent d'obtenir une donnée du type voulu. Taper les instructions suivantes, où l'on voit des premiers exemples d'*affectations*, ici des *variables* `x` et `y` :

```
>>> x = 42
>>> x
.................. \textit{à compléter !}
>>> float(x)
..................
>>> y = -3.67\\
>>> y \\
..................
>>> int(y)\\
..................
```

! `int` appliqué à un flottant ne fait pas un arrondi, ni ne calcule la partie entière : cela donne juste l'entier qui apparaît dans l'écriture décimale du flottant.

Les booléens

Une variable de type booléen ne peut prendre que deux valeurs : `True` ou `False`. Il s'agit souvent de propositions logiques plus ou moins longues que python peut évaluer. Quelques exemples :

```
>>> 1 > 0
       True
>>> a = (1 > 3)
>>> a
       False
>>> type(a)
       <type 'bool'>
>>> x = 3.14
```

```
>>> y = 3.1
>>> x == y
       False
```

Les opérateurs de comparaison disponibles sont :

`x == y`	x est égal à y
`x != y`	x est différent de y
`x > y`	x est strictement supérieur à y
`x < y`	x est strictement inférieur à y
`x >= y`	x est supérieur ou égal à y
`x <= y`	x est inférieur ou égal à y

Bien noter la différence entre = (pour affecter une variable se trouvant à gauche du signe) et == (pour définir une égalité qui sera un booléen).
Pour combiner des variables booléennes, on dispose de trois opérateurs logiques : `not`, `and`, `or`.

Exercice 3 –

Question 5 *Créer une variable* `x` *égale à l'entier 3, et tester les trois opérateurs logiques ci-dessus sur les expressions* `x < 4.1` *et* `x % 2 == 0`.

Question 6 *Mathématiquement, à quoi correspond le booléen* `x % 2 == 0` *?*

Les chaînes de caractères

Python permet aussi de manipuler du texte. Le type utilisé s'appelle une chaîne de caractère : *string* en anglais, abrégé en `str` en Python. Il suffit de délimiter le texte par des apostrophes (*quote*) ou par des guillemets (*double quote*) :

```
>>> c = 'ptsi'
>>> d = "bonjour àtous"
```

On peut avoir accès aux caractères qui composent une chaîne de caractères ; attention, en Python, ils sont numérotés à partir de 0, c'est-à-dire que le premier caractère est d'indice 0.

Question 7 *Noter les résultats et commenter.*

```
>>> c[0]
>>> c[2]
>>> c[-1]
>>> c[0] = 'm'
```

Trois autres choses à retenir sur le type `string` :

- la longueur d'une chaîne de caractères, c'est-à-dire le nombre de caractères qui la composent, s'obtient avec la fonction `len`.
  Tester avec `c` et `d`.
- on peut concaténer des chaînes de caractères (c'est-à-dire les "accoler") à l'aide de `+`.
- comme pour les fonctions `int` et `float`, il existe une fonction `str` qui permet d'obtenir un objet de type string.

Question 8 *Créer une chaîne de caractère* `e` *obtenue en concaténant* `c` *et* `d`.

Question 9 *Tester les instructions suivantes ; comment modifier* ***la dernière*** *pour qu'il n'y ait pas de message d'erreur ?*

```
>>> numero = 2
>>> f = c + numero
```

Remarque : Le fait qu'un même opérateur (ici +) puisse fonctionner différemment en fonction du contexte illustre le mécanisme appelé *surcharge des opérateurs*.

Remarque : Pour ne pas avoir à tout recopier à chaque ligne du shell en cas d'erreur...
Sous PYZO (windows), Avec la flèche ↑ (éventuellement plusieurs fois successivement), on peut remonter dans l'historique de commandes.
Sous IDLE, Se placer sur la ligne à répéter et "entrée".

Application 02

Applications – Bases

Savoirs et compétences :
❒ .

Prise en main en mode « exécution d'un fichier »

Lancer Pyzo

Pour écrire de véritables programmes et les conserver, il faut créer un fichier lstlisting exécutable, ce qu'on appelle aussi un "script".
Pour cela, à partir de l'interpréteur :

aller dans "File" et cliquer sur "New"

(plus rapide : CTRL + N).

Avant toute chose, enregistrer ce fichier dans votre répertoire "TP0" sous le nom TP0.py; il ne faut pas oublier de préciser l'extension .py ("File" puis "Save" ou CTRL + S).

Taper dans le fichier les lignes suivantes :

```
2 * 3
print(3 * 3)
a = 4 * 3
print(a + 2)
```

Sauvegarder, puis **exécuter** (*run* en anglais)

en tapant CTRL + E

ou en cliquant sur "Execute file" dans le menu "Run"

Si votre script fait appel à des fichiers stockés dans le même dossier que votre script, la première exécution doit repérer le chemin d'accès aux données, dans ce cas là, exécuter :

en cliquant droit sur l'onglet de votre script et en sélectionnant "Run file as script".

Question 10 *Commenter*

Exercice 4 –

Question 11 *Créez dans votre script deux variables* `x` *et* `y` *contenant des entiers de votre choix.*

1. Complétez votre script pour qu'à l'exécution, il s'affiche dans le shell le booléen disant si `x` est pair ou non.
2. Complétez votre script pour qu'à l'exécution, il s'affiche dans le shell le booléen disant si `x` et `y` sont de même parité ou non.

Exercice 5 –

Question 12 *Créez dans votre script deux variables* `prenom` *et* `nom` *de type chaîne de caractère.*

Question 13 *Complétez votre script pour qu'à l'exécution, il s'affiche vos initiales (par exemple* `LM` *pour Léa Martin).*

Après ce premier test et ces deux exercices, votre script ressemble à peu près à cela :

```
2 * 3
print(3 * 3)
a = 4 * 3
print(a + 2)
x = 2
y = 5
###
# ce que vous avez répondu pour l'exo précé✓
    dent
###
prenom = 'Léa'\\
nom = 'Martin'\\

###
# ce que vous avez répondu pour cet exercice.
###
```

Ainsi, on ne voit pas bien la délimitation des différents exercices; de plus, lorsqu'on exécute ce script, on a les affichages "parasites" du test et de tous les exercices, alors qu'on aimerait passer à autre chose...Il ne faut pas effacer votre travail pour autant!
Il convient donc d'utiliser des **commentaires** à l'aide du symbole #; tout ce qui suit sur la même ligne ne sera pas "lu". On s'en sert pour améliorer la présentation, et pour supprimer les affichages intempestifs lors de l'exécution. Par exemple, il convient de transformer notre fichier ainsi :

```
##### TP 0 Prise en main de Python #####
## Premier test

2 * 5
#print(3 * 5)
a = 4 * 5 \\
#print(a + 2)
```

```
## Exercice 4
x = 2
y = 5
# ...ce que vous avez répondu pour l'exo 4, ↙
    que vous pouvez mettre en commentaire !

## Exercice 5\\
prenom = 'Léa'\\
nom = 'Martin'\\
# ...ce que vous avez répondu pour l'exo 5, ↙
    que vous pouvez mettre en commentaire !
```

On peut mettre tout un paragraphe en commentaire en le sélectionnant, puis en allant dans le menu "Format" et en cliquant sur "Comment Out Region". L'opération inverse se fait avec "Uncomment Region".

Lorsqu'on ferme PYZO et qu'on le relance, votre fichier TP0.py se retrouvera facilement en cliquant sur "Open" dans le menu "File", ou bien en faisant CTRL + O.

Exercice 6 –

Question 14 *Toujours en se servant des variables* `prenom` *et* `nom`*, écrire les instructions permettant d'afficher par exemple le message :* `'Bonjour, Léa Martin !'` *lorsque* `prenom` *contient* `'Léa'` *et* `nom` *contient* `'Martin'`.

Application 03

Applications – Bases

Savoirs et compétences :
❒ .

Les fonctions

Nous avons déjà rencontré des fonctions python prédéfinies : `type`, `int`, `len`, `print`...

Lorsqu'on écrit le nom d'une fonction `fct` dans l'interpréteur et sa parenthèse ouvrante, des informations s'affichent (les **arguments** attendus et optionnels, parfois le type du résultat...).
Pour obtenir plus de détails, on peut aussi taper `help(fct)`.

Exercice 7 –

Question 15 *À l'aide de ces deux techniques : vérifier que la fonction* `abs` *renvoie bien la valeur absolue, comprendre le fonctionnement de la fonction* `round`*, et tester avec des exemples pour vérifier.*

Fonctions issues d'une bibliothèque

Souvent, nous aurons besoin de fonctions non disponibles par défaut, qu'il faut alors importer à partir d'une bibliothèque [1]. C'est le cas pour de nombreuses fonctions mathématiques ; par exemple :

```
>>> cos(1)
       Traceback (most recent call last):
       File "<pyshell#27>", line 1, in <module>
           cos(1)
NameError: name 'cos' is not defined
```

```
>>> import math as m # ceci signifie "importer la bibliothèque math sous le nom m"
>>> m.cos(1)
       0.5403023058681398
```

`floor` (partie entière habituelle : `floor(x)` est l'entier inférieur le plus proche de `x`),
`ceil` (partie entière supérieure : `ceil(x)` est l'entier supérieur le plus proche de `x`).

```
>>> from math import *
```

Définir vos propres fonctions

Vous pouvez définir vos propres fonctions. Pour cela il faut se placer dans le script TP0.py et plus dans le shell. Une fonction se définit de la façon suivante :

```
def f(x): # Def appelle une fonction et f indique son nom
    return(x**2)
```

On dit que la variable `x` est l'*argument* de la fonction `f`, et que cette fonction *renvoie* (ou *retourne*) x^2.

C'était la même chose pour la structure `for`.

```
def g(n):
    return( (n \% 2) == 0 )
```

Une fonction peut avoir plusieurs arguments, que l'on sépare alors par des virgules ; le bloc d'instruction peut être plus long :

```
def h(x,y):
    z = 2 * y
    return(x + z)
```

La syntaxe générale d'une fonction est la suivante :

```
def nom_de_la_fonction(arguments):
```

Exercice 8 –

Question 16 *Écrire une fonction* `suite_cos` *qui prend pour argument un entier naturel* `n` *et qui affiche toutes les valeurs de* $\cos(i)$ *pour* i *variant de* 0 *à* `n-1`.

1. Remarque : en anglais, bibliothèque se dit *library*.