
Introduction

I. Installation de Pyzo et de Python 3

Il est **indispensable** d'installer Pyzo sur un ordinateur personnel : au cours de l'année, vous aurez des devoirs d'informatique à faire à la maison. En salle de TP, des ordinateurs seront disponibles mais vous pouvez, si vous le souhaitez, apporter votre ordinateur portable.

Pour installer Pyzo et Python, suivez les instructions suivantes :

1) Rendez-vous sur la page d'installation de Pyzo : <https://pyzo.org/start.html>

2) Installation de Pyzo

- ★ Dans la partie **Step 1 : install the Pyzo IDE**, téléchargez le fichier d'installation correspondant à votre système d'exploitation (OS) :
 - Windows : Pyzo installer (64 bit)
 - MacOS : Pyzo dmg
 - Linux : Pyzo tarball

Si la version de votre OS n'est pas à jour, cliquez sur le lien [all releases](#) et choisissez la version de Pyzo qui vous correspond.

- ★ Une fois le fichier téléchargé, installez le programme Pyzo en suivant les instructions. Sous Windows, il est conseillé de cliquer sur la ligne *Associate ".py" extension (need admin privileges)*.

3) Installation de Python 3

- ★ Toujours sur la même page <https://pyzo.org/start.html>, dans la partie **Step 2 : install Python environment**, cliquez sur le lien [regular Python](#) qui vous mènera sur la page <https://www.python.org/>.
- ★ Téléchargez le fichier d'installation de Python (version 3.13.5 à ce jour) qui correspond à votre système d'exploitation.

4) Configurer l'IDE Pyzo

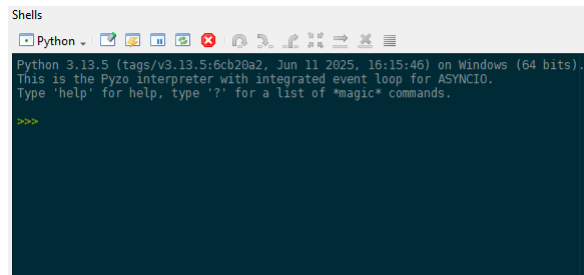
- ★ Ouvrez le logiciel Pyzo.
- ★ Dans le menu **Tools**, décochez tous les éléments sauf **Interactive help** et **Workspace**.
- ★ Dans le menu **Settings/Edit syntax styles...** vous pouvez modifier le style de l'éditeur. Par exemple, pour un meilleur confort visuel, il est conseillé d'utiliser le style *solarized_dark*.
- ★ Si vous le souhaitez, vous pouvez changer la langue de Pyzo en Français dans le menu **Settings/Select language** mais cela changera seulement la langue des menus : l'aide de Python restera en anglais.

II. Découverte de l'IDE Pyzo

Un **IDE** (*Integrated Development Environment* ou *Environnement de Développement Intégré*) est un logiciel qui regroupe plusieurs outils destinés à faciliter l'écriture, l'exécution et le débogage du code informatique.

L'IDE Pyzo est un environnement simple et adapté aux débutants. Il repose sur le langage Python et offre deux fenêtres principales (le shell et l'éditeur) et d'autres fenêtres secondaires que l'on peut ajouter ou supprimer :

★ [Le Shell ou la console interactive](#)



Le shell correspond à l'interpréteur Python proprement dit. Voici ses principaux modes de fonctionnement :

- les instructions sont exécutées une à une c'est-à-dire qu'on peut écrire une instruction, taper sur la touche **Entrée** pour l'exécuter, puis écrire une autre instruction
- les instructions tapées ici ne sont pas enregistrées, une fois Pyzo fermé, elles sont perdues définitivement
- si on s'est trompé dans l'instruction que l'on vient de taper, on ne peut la modifier que si on n'a pas encore validé, sinon il faut retaper l'instruction (on pensera à utiliser la touche ↑ pour gagner du temps – voir la partie IV.)

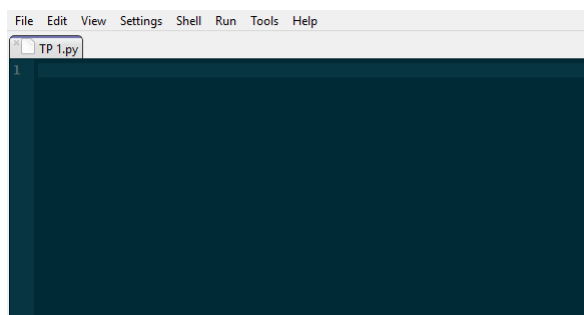
Le shell est donc pratique pour tester des instructions rapidement et de façon isolée du reste du programme ou pour lancer l'exécution d'une fonction mais ce n'est pas là qu'il faut écrire nos programmes.

Exemple :

```
>>> 1+1
2

>>> print("Hello world!")
Hello world!
```

★ [L'éditeur](#)



C'est un simple éditeur de texte où l'on peut écrire du code Python qui pourra être ensuite exécuté. Principes de fonctionnement de l'éditeur :

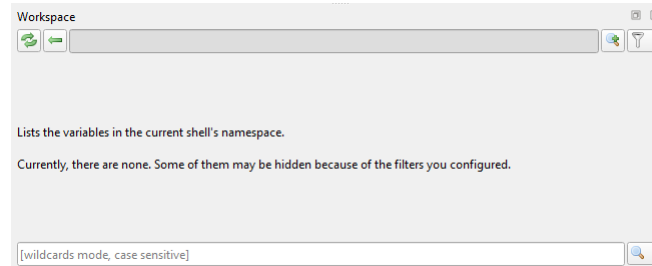
- on peut écrire plusieurs instructions avant de les exécuter toutes d'un coup (dans l'ordre logique du script)
- on peut enregistrer dans un fichier .py tout ce que l'on écrit dans l'éditeur
- on peut ajouter des commentaires pour expliquer le programme
- si on s'est trompé dans une instruction, on peut la modifier à tout moment

L'éditeur est donc la partie où l'on doit écrire nos programmes.

Exemple :

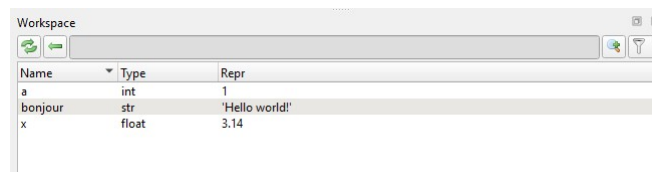
```
1 a = 1
2 b = 2
3 c = 3
4 delta = b**2 - 4*a*c
5 print(delta)
```

★ L'environnement (Workspace)

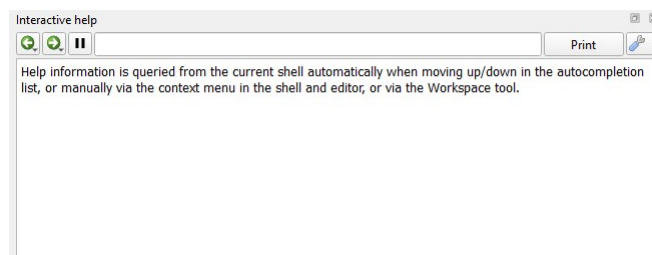


Cette partie est facultative. Elle permet de voir la liste des variables que l'on a créée ainsi que leur type et leur valeur.

Exemple :

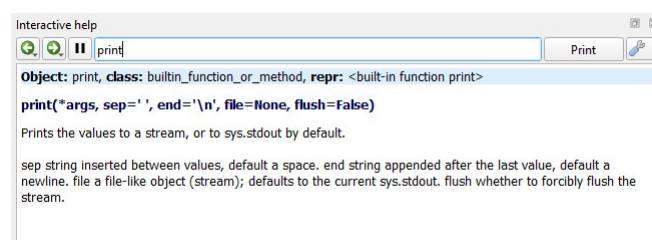


★ L'aide interactive (Interactive help)



Cette partie est également facultative. Elle permet de consulter l'aide de Python pour comprendre le comportement d'une fonction.

Exemple :



III. Bonnes pratiques d'organisation et de programmation

Pour bien s'y retrouver et travailler efficacement, il est important d'adopter de bonnes habitudes :

- ★ **Créer un dossier par TP** : tous les fichiers liés à un même TP (scripts, données, images) doivent être regroupés.
- ★ **Créer un fichier python principal par TP** : créer un nouveau fichier et l'enregistrer sous le nom TP n.py où n est le numéro du TP.
- ★ **Organiser son code** : séparer les différentes parties d'un TP (exercices ou groupes d'exercices) en cellules. Une cellule commence par un double-hashtag suivi d'un titre. Par exemple :

```
## Exercice 1

print("Hello world!")

## Exercice 2

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

- ★ **Donner des noms explicites** aux variables et aux fonctions. Par exemple :

```
## Code obscur

def f(x):
    y = 3.14 * x * x
    return y

a = 5
b = f(a)
print(b)

## Code très clair

def aire_cercle(rayon):
    pi = 3.14
    aire = pi * rayon * rayon
    return aire

rayon_cercle = 5
aire = aire_cercle(rayon_cercle)
print(aire)
```

- ★ **Commenter son code** : tout ce qu'on écrit après un hashtag est un commentaire qui n'est pas lu par Python et permet d'expliquer le code. Un bon commentaire explique ce que fait un bloc d'instructions sans paraphraser ligne par ligne. Par exemple :

```
# Liste de nombres entiers
nombres = [3, 6, 9, 10, 12, 15, 18]

# Initialisation du compteur de nombres pairs
nb_paires = 0

# Parcours de la liste pour compter les nombres pairs
for n in nombres:
    if n % 2 == 0:
        nb_paires += 1

# Affichage du nombre de valeurs paires trouvées
print(f"Nombre de valeurs paires : {nb_paires}")
```

- ★ **Espacer son code** : utiliser les sauts de lignes et les espaces pour améliorer la lisibilité du code. Voici quelques conseils :

- Dans un script, les lignes vides sont utiles pour séparer visuellement les différentes parties du code.

Il est recommandé de laisser deux lignes vides avant la définition d'une fonction.

On peut aussi laisser une ligne vide dans le corps d'une fonction pour séparer les sections logiques de la fonction, mais cela est à utiliser avec parcimonie.

- Mettre un espace **avant** et **après** les opérateurs : + - / * == != >= <= > < **not and or**
- Mettre un espace **seulement après** les symboles : , :
- **Ne pas mettre d'espace** autour des parenthèses délimitant les arguments d'une fonction :
`ma_fonction(argument1, argument2)`

- ★ **Tester son code** : il est très important de tester régulièrement son code pour voir si tout fonctionne comme prévu. En particulier, si Python affiche un message d'erreur, on ne continue pas à écrire son programme tant qu'on n'a pas compris et corrigé l'erreur ! En effet, on risque d'ajouter d'autres erreurs et rendre très difficile le débogage du code.

IV. Les raccourcis-clavier

Afin d'être rapide et efficace en programmation, il est nécessaire de maîtriser certains raccourcis-clavier de Pyzo. En voici une liste non exhaustive :

Commande	Effet
CTRL + C	copier
CTRL + V	coller
CTRL + X	couper
CTRL + R	commenter la sélection
CTRL + T	décommenter la sélection
TAB	indenter la sélection
SHIFT + TAB	désindenter la sélection
CTRL + I	interrompre l'exécution en cours
CTRL + K	redémarrer le shell
CTRL + E	exécuter tout le fichier
CTRL + ENTREE	exécuter une cellule
ALT + ENTREE	exécuter la sélection ou la ligne

Autres astuces très utiles :

- ★ Pour modifier ou ré-exécuter une commande déjà tapée dans le shell, on peut la ré-afficher en tapant sur la flèche du haut  autant de fois que nécessaire.
- ★ Lorsqu'on commence à taper le nom d'une variable, fonction ou commande quelconque dans Pyzo, un menu déroulant apparaît pour nous proposer des noms de variables ou de fonctions déjà définies ou de commandes préexistantes. On peut alors sélectionner la commande voulue à l'aide des touches haut  et bas , puis valider cette sélection à l'aide de la touche tabulation  (et non de la touche .
- ★ Vous pouvez éditer vos propres raccourcis-clavier dans la partie **Settings/Edit key mappings....**

V. Informatique aux concours

1) Concours Agro-Véto

★ Épreuve écrite « Modélisation mathématique et informatique » (3h)

Extrait de la notice du concours :

L'usage de la calculatrice est autorisé pour cette épreuve.

Cette épreuve porte sur le traitement mathématique d'un problème concret au sens où il est issu d'une situation liée à la biologie, la géologie, la physique, la chimie ou à la géographie. Le problème étudié ne doit pas nécessiter de connaissances supplémentaires en dehors de celles qui figurent au programme de mathématiques. Le candidat peut ainsi être amené à proposer ou à analyser un ou des modèles, à en étudier les propriétés, à les valider ou à les critiquer, sur des questions théoriques ou pratiques (l'énoncé pouvant s'accompagner par exemple de documents à étudier). La démarche mathématique reste le cadre opérationnel, et le calcul numérique a toute sa place dans ce contexte.

La partie informatique est évaluée dans cette épreuve par le moyen de questions d'algorithmique liées à la simulation du modèle. Afin de garantir une bonne articulation avec la partie mathématique de l'épreuve, l'accent sera mis sur les chapitres 1, 2.4 et 3.1 du programme d'informatique. Comme mentionné à l'annexe A de ce programme, aucun élément de syntaxe qui relève des modules, y compris numpy ou matplotlib, n'est exigible.

Pour information, les chapitres 1 et 2.4 du programme d'informatique de BCPST mentionnés portent sur :

- les algorithmes élémentaires opérant par boucles simples (chapitres 2 et 4)
- les algorithmes opérant par boucles dans un tableau unidimensionnel (chapitre 5)
- la lecture et l'écriture dans un fichier texte (compléments)
- l'utilisation de modules et de bibliothèques (chapitre 3 et compléments)
- les algorithmes opérant par boucles imbriquées (chapitre 7)
- la recherche dichotomique (chapitre 8)
- les fonctions récursives (chapitre 9)
- les matrices de pixels et d'images (compléments)
- les algorithmes de tris (chapitre 8)
- les méthodes numériques (chapitre 10)
- la simulation de variables aléatoires et l'estimations de probabilités, d'espérances et de variances (chapitre 10)

Le chapitre 3.1 du programme d'informatique sera vu en BCPST 2.

★ Épreuve orale « Mathématiques pratiques et informatique » (40 min de préparation + 40 min d'interrogation)

Extrait de la notice du concours :

L'oral devant le jury dure 40 min. Il est précédé de 40 min de préparation, formalités d'accueil comprises (vérification de la pièce d'identité, rappel succinct des règles, installation près de la console informatique et mise en route de celle-ci, remise du sujet à préparer).

Le jury de l'épreuve de Mathématiques pratiques et informatique est composé d'un seul examinateur.

Le candidat prépare pendant 40 min un exercice de mathématiques ainsi qu'une courte question portant sur le cours de mathématiques (énoncé ou application immédiate d'une propriété, illustration d'une définition par un exemple ou un dessin). **L'exercice peut comporter au moins une question d'informatique.**

Le premier temps de l'interrogation permet la restitution devant le jury de l'exercice et de la question de cours. Cette restitution dure au maximum 30 min. **Le jury pose ensuite un bref exercice non préparé. Cet exercice porte essentiellement sur des questions d'informatique.**

Au moins l'un des deux exercices (l'exercice préparé et celui sans préparation) commence par une question d'informatique. En informatique, l'accent est mis sur les chapitres 1, 2.4 et 3.1 du programme.

Sur un barème de 20 points, la part accordée au contrôle du cours n'excède pas 2 points et celle accordée à l'informatique n'excède pas 5 points. L'exercice avec préparation, la question de cours, et l'exercice sans préparation abordent, au moins deux des trois grands domaines du programme de mathématiques (probabilités-statistiques, analyse, algèbre linéaire).

2) Concours G2E

- ★ Épreuve orale optionnelle « Épreuve oral d'informatique » (20 min de préparation + 20 min d'interrogation)

Extrait de la notice du concours :

L'épreuve orale d'informatique, au choix avec la chimie, dure 40 minutes : 20 minutes de préparation suivies de 20 minutes d'exposé devant l'examinateur.

Pendant la 1ère partie de l'exposé (10 minutes) le candidat est amené à présenter la résolution d'un exercice tiré au sort et préparé pendant les 20 minutes de préparation. Pendant la seconde moitié de l'exposé, il travaille sur un exercice non préparé proposé par l'examinateur.

L'objectif des exercices proposés est de vérifier la capacité du candidat à pouvoir transformer un problème élémentaire en un algorithme, à déterminer les étapes permettant de mettre en uvre cet algorithme et à identifier les fonctions et types de structures nécessaires à sa programmation. Le programme qui en résulte est écrit dans le langage de programmation *Python*.

Les exercices se présentent sous forme de problèmes généraux ne faisant pas appel nécessairement à des notions mathématiques, physiques ou biologiques...

Pendant l'exposé, le candidat est convié à présenter une solution pour résoudre le problème posé et à répondre à des questions qui peuvent être liées à la solution exposée, prolongements, variantes, efficacité de l'algorithme proposé. Les interventions de l'examinateur sont destinées à obtenir des précisions, corriger des erreurs ou de mauvaises démarches, elles ne sont jamais faites pour perturber le candidat.

L'évaluation tient compte d'aspects strictement « algorithmiques » et de « programmation » :

- exactitude de l'algorithme présenté

- maîtrise des concepts de programmation manipulés
- efficacité du programme, prise en compte des cas particuliers

Plus généralement d'autres qualités ont aussi été appréciées :

- vivacité et rapidité suite aux remarques de l'examineur
- aptitude à défendre les solutions proposées
- capacité à relier le problème à des problèmes plus généraux
- maîtrise du langage et « élégance » des solutions présentées

Dans la 2ème partie de l'exposé, le candidat est interrogé sur un deuxième exercice. Les qualités estimées du candidat sont les mêmes que dans la 1ère partie.