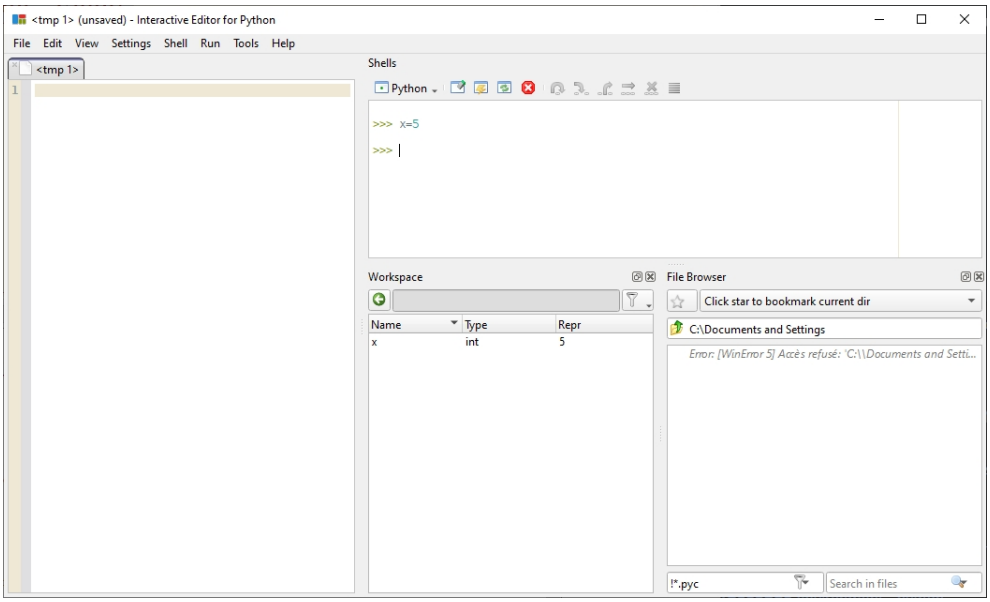
INTRODUCTION À PYTHON : IF, FOR, WHILE

Pour les TP de cette année, nous utiliserons le logiciel **Pyzo**. Lancer Pyzo depuis le bureau. La fenêtre qui s'ouvre est divisée en plusieurs parties*.



* Il se peut que la fenêtre soit agencée différemment ou qu'à la place du Workspace, vous voyez une autre fenêtre, telle que « source structure ». Pas de panique. On reviendra sur le Workspace plus bas.

1 Console et opérations simples

La *console* (ou *Shell*) est comme une calculatrice, on tape quelque chose, on valide avec Entrée et le résultat apparait. Essayer avec les instructions ci-dessous.

a) L'addition +
```
2+3
7+1.4
```

b) La soustraction -
```
5.5 - 1.1
-2 - 3
```
(les espaces aident à relire)

c) La multiplication *
```
2*3
1+0.5*2
(1+0.5)*2
```
(prioritaire sur + et -)*

d) La division /
```
3/4
1+2/3
1/(2*2)
1/2*2
```
(idem pour /)
*(pas de priorité entre * et /)*

e) La puissance (ou exponentiation) **
```
5**2
4**(1/2)
3**1/2
```
(rappel : $\sqrt{x} = x^{1/2}$)
*(** prioritaire sur * et /)*

f) La division entière (quotient de la div. euclidienne) //
```
10//3
8//2
```

g) Le reste de la division euclidenne %
```
10%3
8%2
```

Exercice 1. Quel est le reste de la division euclidienne de 123456789^{10} par 9 ? Que peut-on en déduire ?

2 Éditeur et variables

L'éditeur (partie gauche de l'image ci-dessus) permet d'écrire des *scripts* : ce sont des fichiers qui contiennent des lignes de code python. Chaque ligne sera exécutée l'une après l'autre. Taper le code suivant **dans l'éditeur**, puis l'exécuter avec Ctrl + E.

```
## Mon premier script
print("Salut le monde")

a = 3
b = 1/2
c = a**b
print(c)    # doit afficher 1.7302(...) dans la console
```

Quelques remarques :

- La fonction `print(...)` permet d'afficher le résultat **dans la console**.
- `a`, `b` et `c` sont des *variables* : ce sont des « boîtes » qui stockent une *valeur*. Attention à l'ordre : la syntaxe est toujours `<variable> = <expression>`. Par exemple, `annee = 2023` est correct mais pas ~~`2023 = annee`~~.
- Tout ce qui est écrit après un # est un commentaire et est ignoré par Python. Les commentaires permettent d'apporter des précisions pour rendre le code plus compréhensible.
- Le script peut être divisé en cellules. Une cellule est délimitée par deux lignes de commentaires qui commencent par ##. **N'effacez pas le code que vous venez d'écrire dans l'éditeur**. Dans la suite de ce TP, vous devrez toujours rajouter le code à la suite du précédent en commençant par ## pour créer une nouvelle cellule.
- Sauvegardez le script que vous venez de taper dans votre dossier élève personnel (U :// puis cherchez votre prénom et nom). Appelez ce fichier `TP1.py`.

Le Workspace. En plus de la console et de l'éditeur, Pyzo permet de rajouter différentes « fenêtres-outils ». Pour afficher le Workspace, cliquer sur Tools > Workspace. Vous pouvez enlever des outils ou les repositionner comme avec une fenêtre normale. Le *Workspace* affiche toutes les variables qui sont « enregistrées » ainsi que leur type et leur valeur. Lorsqu'une variable est enregistrée, elle est utilisable dans n'importe quel script ou dans la console.

Exercice 2. Après avoir exécuté votre script, taper successivement dans la console les commandes

`a+b` `x=c**4` `x` `y=X`

Lisez le message d'erreur de la dernière commande et expliquer ce qui cause cette erreur.

Question 1. Essayez de *prédire* ce que va afficher le script suivant, puis le taper et l'exécuter pour vérifier.

```
## Exercice de prédiction
compteur = 5
compteur = compteur + compteur
compteur = compteur + compteur
print(compteur)              # qu'est-ce qui va s'afficher ?
```

Question 2. Est-ce qu'il y a une différence entre les instructions `a = b` et `b = a` ?

3 Bloc conditionnel : if / else (si / sinon)

Si (`if`) une condition est vraie, alors on exécute une instruction, sinon (`else`) on en exécute une autre. La syntaxe générale est la suivante (ne pas recopier !) :

```
if <condition> :
    ...     # Ce bloc est exécuté si <condition> est vraie
    ...
else :
    ...     # Ce bloc est exécuté si <condition> est fausse (optionnel)
    ...
...         # fin du bloc : cette partie sera toujours
...         # exécutée, indépendamment de la condition
```

Exemple. Recopier le script suivant et l'exécuter (avec Ctrl + E). Comparer pour différentes valeurs de a, b et c :

```
## Bloc if
a = 11
b = 3
c = 17
if a + b == c :
    print("L'égalité est juste")
else :
    print("L'égalité est fausse")
    print(c-b-a)
print(c)                            # Ceci s'affiche dans les deux cas
```

Syntaxe d'un bloc `if`. Les deux points « : » marque le début du bloc. L'espace en début de ligne *l'indentation*. Cela indique qu'on est dans le bloc. Pour indiquer qu'on sort du bloc, il faut enlever l'indentation en supprimant cet espace. Pyzo gère automatiquement l'indentation, mais s'il faut indenter manuellement, on utilisera la touche Tab →.

Différence entre = et == . Le signe = sert à affecter une valeur à une variable.

Le signe == teste une égalité et renvoie un booléen, à savoir `True` (Vrai) ou `False` (Faux). Pour écrire une condition d'un bloc `if`, il faut donc utiliser == et non pas = !

Remarque. Si on souhaite uniquement exécuter une cellule (et pas tout le script), cliquer n'importe où dans la cellule puis utiliser Ctrl + Enter.

Question 3. Est-ce qu'il y a une différence entre les instructions `a == b` et `b == a` ?

Exercice 3. Recopier et compléter le script suivant. Vérifier avec différentes valeurs de a et b.

```
## Exercice de divisibilité
a = 437
b = 19
if ... :      # vérifier si a est divisible par b
    print("a est divisible par b. Le résultat de la division est :")
    print(...)
else:
    print("a n'est pas divisible par b")
print("fini")
```

Remarque. Pour écrire une condition, on peut utiliser :

- Des opérateurs de comparaison booléenne : == (égalité), != (qui signifie « n'est pas égal à ») ainsi que >, >=, <, <=.
- Des connecteurs logiques : and (et), ou (ou), `not(...)` (non)

```
a = 3
b = 3.5
if ... and ... :
    print("b est dans l'intervalle [a,a+1[")
if ... or ... :
    print("b n'est pas dans l'intervalle [a-1,a+1]")
```

Remarque. Enfin, après un bloc `if`, on peut aussi utiliser `elif <condition>`, qui combine `else` et `if`. C'est l'équivalent de « sinon si ». Les scripts A et B ci-dessous affichent le même résultat si `maNote` est dans $[0,20]$.

```
## Script A : if avec elif
maNote = ... # mettez votre note

if maNote == 20:
    print("Excellent !")
elif maNote >= 15:
    print("Très bien !")
elif maNote >= 10:
    print("Vous avez la moyenne")
else:
    print("ARGH")
```

```
## Script B : if SANS elif
maNote = ... # mettez votre note

if maNote == 20:
    print("Excellent !")
if maNote >= 15 and maNote < 20:
    print("Très bien !")
if maNote >= 10 and maNote < 15:
    print("Vous avez la moyenne")
else:
    print("ARGH")
```

Question 4. Qu'est-ce qui est affiché par chaque script lorsque `maNote` est dans $\mathbb{R}_-^*$? lorsque `maNote` est dans $]20,+\infty[$?

4 Boucle `for` et boucle `while`

La boucle `for` permet de répéter un bloc d'instructions en changeant à chaque fois la valeur d'une variable. La syntaxe est la suivante (ne pas recopier !) :

```
for <variable> in <iterable>:
    ...     # ce bloc est exécuté une fois pour chaque valeur dans <iterable>
    ...     # à chaque itération, la valeur de <variable> est modifiée
...
...         # ce bloc est exécuté après la boucle for
```

Un *itérable* est un objet que l'on peut « parcourir » : une liste, un *string*, etc. Recopier, compléter et exécuter le script suivant :

```
## Une boucle for
monPrenom = "..."           # mettez votre prénom (ou autre...)
for lettre in monPrenom :
    print(lettre)
print("stop")
```

Question 5. Que se passe-t-il si on indente `print("stop")` une fois vers la droite ? Prédire le résultat, puis vérifier.

La fonction `range` permet de générer facilement une liste d'entiers. Étant donnés deux *entiers* `m` et `n` :

- `range(n)` permet d'itérer sur les valeurs `[0,1,...,n-1]` : on commence à `0` inclus et on s'arrête à `n` exclu.
- `range(m,n)` permet d'itérer sur les valeurs `[m,m+1,...,n-1]` : on commence à `m` inclus et on s'arrête à `n` exclu.

Exercice 4. Recopier et compléter le script ci-dessous pour afficher « et 1, et 2, et 3, zéro ! ».

```
## Un "for" avec un "range"
for i in range( ... , ... ) :
    print("et",i)
print("zéro !")
```

Les boucles `for` sont très utiles pour calculer des sommes ou des produits, grâce à un accumulateur : il s'agit d'une variable qu'on initialise en dehors de la boucle et qui change de valeur à chaque itération. Recopier et exécuter :

```
## La variable s est un accumulateur
s = 0
for i in range(101) :
    s = s + i
    print(s)
```

Question 6. À quoi correspond (mathématiquement) la dernière valeur de `s` affichée ?

Exercice 5. Écrire un script qui affiche le résultat de $3+4+5+\cdots+22$. Réponse : 250.

Exercice 6. Écrire un script qui affiche le résultat de $3^2+4^2+5^2+\cdots+22^2$. Réponse : 3790.

La boucle `while` permet de répéter un bloc d'instructions *tant que* une condition est vraie.

```
## Ma première boucle while
N = 3
while N != 0 :
    print(N)        # tant que N≠0, ce bloc s'exécute
    N = N-1
print("partez !")
```

Remarque. Contrairement à la boucle `for`, une boucle `while` peut ne jamais s'arrêter ! C'est par exemple ce qui se passe si on remplace la ligne `N=N-1` par `N=N+1` (tester !). **Astuce** : pour forcer l'arrêt d'un script, utiliser Ctrl + I.

Question 7. Que se passe-t-il si on remplace la ligne `N=3` par `N=0` ? par `N=-3` ? Prédire, puis tester.

5 Exercices pour s'entrainer

Exercice 7. Écrire un script qui commence par `a = ...` (choisissez une valeur), puis qui affiche un message qui donne le signe de `a` : positif, négatif, ou nul (0 est considéré comme ayant un signe « nul »). Tester pour différentes valeurs de `a`.

Exercice 8. Écrire un script qui commence par `a = ...` et `b = ...`, puis qui affiche 1 si `a` et `b` ont même signe, 0 sinon.

Exercice 9. Écrire un script qui affiche la somme des entiers *pairs* de 10 à 30. Réponse : 220.

Exercice 10. Écrire un script qui affiche le produit des entiers de 1 à 8 avec une boucle. Réponse : 40320.

Exercice 11. Écrire un script qui réalise l'algorithme suivant : on assigne la valeur 1 à la variable `n`, puis, tant que `n` est inférieur à 1000, on affiche `n` puis on le multiplie par 2.

Exercice 12 (⋆)**.** Écrire un script qui commence par `a = ...` (entier positif) et affiche successivement tous les chiffres de `a` (unités, dizaines, etc.). Vous ne devez utiliser qu'une boucle `while` (sans `if`). Interdiction de convertir en « string ».