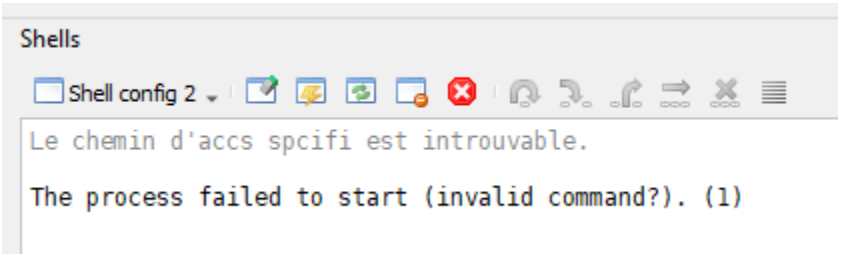
# Ds- 1Nsi Python

Se loguer avec l'identifiant : **exam02.eleve** et le mot de passe :

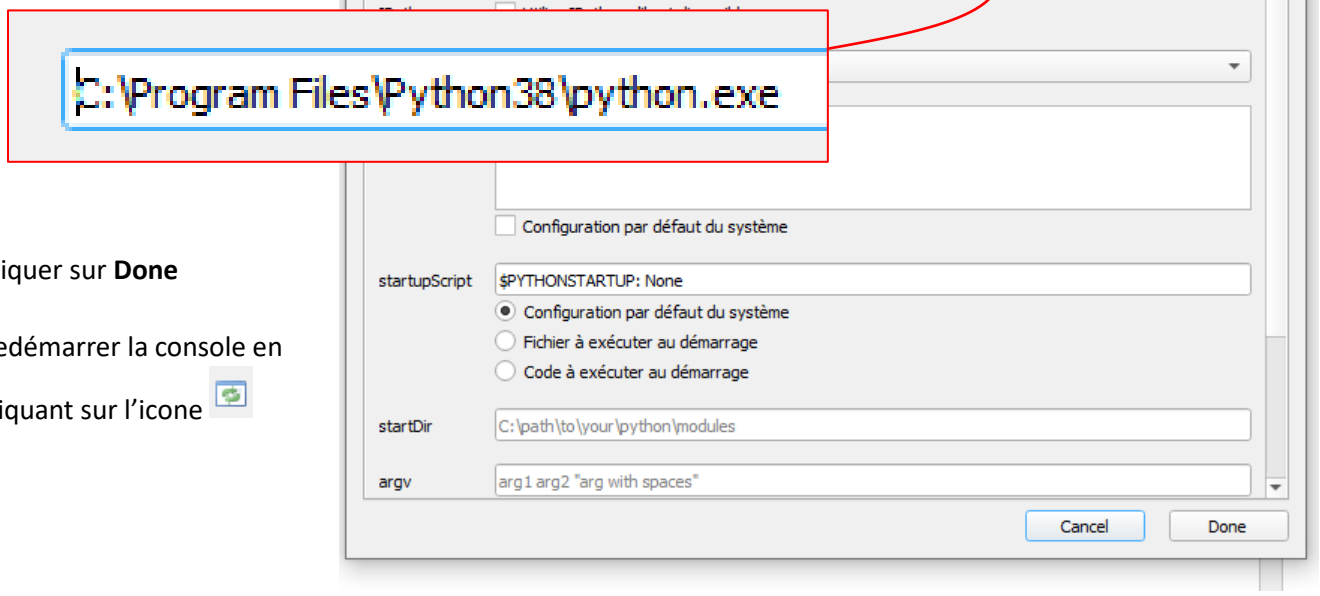
Une fois logué :

- cliquer sur l'icone 

et aller sur le serveur ***Examens(Z :)***, dans le dossier ***exam02/sujets***
- ouvrir le fichier nommé ***pyzoSallesSTI.txt***
- copier la ligne ***C:\Program Files\Python38\python.exe*** dans le presse-papier (Ctrl C ou click droit + copié)
- fermer ce fichier.
- Lancer Pyzo que vous retrouvez en cliquant sur l'icone Démarrer/3-Mathématiques/Pyzo ou par l'intermédiaire du champ de saisie Windows en bas à gauche de l'écran.
- En ouvrant Pyzo, la console n'est pas opérationnelle. Pour corriger cette erreur, il faut cliquer sur « ***Shell config*** » et choisir « ***Configuration des shells*** ».




- Il faut modifier l'adresse de la ligne exe en y collant celle que vous avez copié dans le presse-papier : faire ctrl V ou click droit + coller.




- Cliquer sur **Done**
- Redémarrer la console en cliquant sur l'icone 

Ce DS est composé de 3 exercices indépendants pour lesquels il faudra à chaque fois créer une fonction python. Les scripts de ces fonctions et les lignes qui les appellent seront écrits dans un fichier que vous enregistrerez avec le nom *mon_nom.py* dans le répertoire ***Travail(U :) /exam02.eleve../***

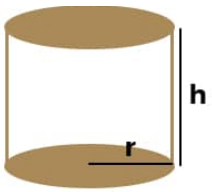
Ce même fichier sera à copier-coller en fin d'épreuves dans le répertoire : ***Examens(Z :) /exam02/copies/***

***On commence à présent le travail à faire :***

## 1- FONCTION QUI CALCULE UN VOLUME : *(6,5 points)*

Le volume d'un cylindre est donné par la relation ci-contre avec $\pi \approx 3.14$ .

**volume = π x r² x h**




Une fonction nommée *volumeCylindre()* renvoie le volume d'un cylindre, en fonction de son rayon et de sa hauteur mis en arguments.

En exécutant le code ci-contre, on obtient dans la console :

```
>>> (executing file "dsSynthesePython.py")
6.28 m3
les valeurs doivent être positives
les valeurs doivent être positives
```

```
# Programme principal
vol = volumeCylindre(1,2)
print(vol)

sortie = volumeCylindre(-1,2)
print(sortie)

message =  volumeCylindre(1,-2)
print(message)
```

⇨ Créer le script de cette fonction *volumeCylindre ()* et tester l'ensemble.

*Corrigé*

```
def volumeCylindre(r,h) :
    if r < 0 or h < 0 :
        return "les valeurs doivent être positives"
    else :
        volume = 3.14 * r**2 * h
        message = str(volume) + " m3"
        return message
```

## 2- FONCTION QUI CALCULE UNE MOYENNE PONDEREE : *(7 points)*

Une fonction nommée *moyennePonderee()* prend en argument une liste de listes contenant des notes avec leur coefficient. Cette fonction retourne la moyenne pondérée de ces notes. Dans le cas ou la somme des coefficients est nulle, elle retourne le mot clé ***None***. On suppose que les notes et les coefficients sont des nombres positifs.

En exécutant le code ci-contre, on obtient dans la console :

```
# Programme principal
moy = moyennePonderee([[8,2] , [12,0] , [13.5,1] , [5,0.5]])
print(moy)
print(moyennePonderee([[3,0] , [5,0]]))
```

Valeur de la note : 3/20

Valeur du coefficient de cette note : coef 0

```
>>> (executing file "dsSynthesePython.py")
9.142857142857142
None
```

Dans le premier exemple, la moyenne a été calculée de la manière suivante :

$$\frac{8 \times 2 + 12 \times 0 + 13{,}5 \times 1 + 5 \times 0{,}5}{2 + 0 + 1 + 0{,}5}$$

⇨ Créer le script de cette fonction *moyennePonderee()* et tester l'ensemble.

*Corrigé*

```
def moyennePonderee(liste) :
    somme = 0
    sommeCoefficient = 0
    for note in liste :
        somme = somme + note[0] * note[1]
        sommeCoefficient = sommeCoefficient + note[1]
    if sommeCoefficient == 0 :
        return None
    else :
        moyenne = somme / sommeCoefficient
        return moyenne
```

## 3- FONCTION QUI REALISE DES MULTIPLICATIONS : *(6,5 points)*

Une fonction nommée *multiplication()* prend en argument 2 nombres entiers relatifs $n$ et $m$. Elle renvoie le produit $n \times m$ de ces 2 nombres. Sa particularité, c'est que son script est écrit en utilisant uniquement les opérateurs `+` et `-` (addition et soustraction). **Il n'utilise pas l'opérateur `*` de multiplication .**

```
# Programme principal
print(multiplication(3,5))
print(multiplication(-4,-8))
print(multiplication(-2,6))
print(multiplication(-2,0))
```

En exécutant le code ci-dessus, on obtient dans la console :

```
>>> (executing file "dsSynthesePython.py")
15
32
-12
0
```

Remarque : pour changer le signe d'une variable, on prend son opposée en plaçant un `-` devant la variable. Par exemple :

```
>>> a = 10

>>> -a
-10
```

⇨ Créer le script de cette fonction *multiplication()* **sans utiliser l'opérateur `*`** et tester l'ensemble.

*Corrigé*

```
def multiplication(n,m) :
    # On gère les signes :
    if n < 0  :
        n = -n
        if m < 0 :
            m = -m
            signe = "positif"
        else :
            signe = "negatif"
    else :
        if m < 0 :
            m = -m
            signe = "negatif"
        else  :
            signe = "positif"
    # On réalise la multiplication avec n et m qui sont positifs :
    somme = 0
    for i in range(n) :
        somme = somme + m
    # On gère le signe du résultat
    if signe == "negatif" :
        somme = - somme
    return somme
```