Il existe une multitude de langages utilisés pour communiquer avec l'ordinateur : fortran, C++, java, python, IDL, perl… pour n'en citer que certains. En NSI, nous choisissons de travailler sous Windows et de programmer en langage python en utilisant le logiciel *Spyder* et le site *Capytale*.

Spyder et Capytale

La fenêtre qui s'ouvre est divisée en plusieurs parties, dont les deux plus importantes sont nommées « Console » et « Éditeur ».

La console (à droite en général) est la fenêtre où on exécute les commandes directement, une à une.
L'éditeur (à gauche en général) est la fenêtre où l'on peut entrer les commandes sous forme d'un fichier texte exécutable.

Partie I : Calculer et afficher

Première leçon à absolument retenir avant de commencer :

Un ordinateur est très efficace (il peut résoudre les calculs les plus complexes), mais aussi très TRÈS bête (il ne fait QUE ce qu'on lui dit de faire, ni plus, ni moins) et psycho-rigide (il faut lui dire quoi faire de la bonne façon, sinon, il ne comprend pas).

La console permet de donner des instructions à l'ordinateur une à une. Elle peut par exemple servir de calculatrice.

I.1. Faire les calculs suivants dans la console et compléter le tableau ci-dessous.

*5**2 5//2 5%2 5 > 5 5 >= 5 5 == 2 5 != 2*

Instr.	***Utilité***	***Instr.***	***Utilité***
+	Addition	**	
*	Multiplication	//	
/	Division simple	%	
>		>=	
==		!=	

I.2. Écrire en python les expressions booléennes traduisant les conditions suivantes. Les nombres mentionnés sont tous des entiers.

- **a.** L'entier *n* est divisible par 5.
- **b.** Les entiers *m* et *n* sont de même signe (sans faire de multiplication).
- **c.** Au moins deux des trois entiers *m, n, p* sont différents.
- **d.** Les trois entiers *m, n, p* sont tous différents.

Pour aller un peu plus loin…

- **e.** Les entiers *m* et *n* sont tels que l'un est multiple de l'autre.
- **f.** *n* est le plus grand multiple de 7 inférieur à 100

A vos ordinateurs !

Pour afficher on utilise la commande print().
En particulier, les chaines de caractères s'affiche avec des apostrophes (') simples, doubles ou triples.
Exemple : print('Hello !')

I.3. Afficher les messages suivants, en utilisant un nombre d'apostrophe approprié :

- **a.** Bonjour !
- **b.** C'est bien l'informatique
- **c.** Mon voisin me dit "il est cool ton ordi"

I.4. Quel est l'intérêt d'utiliser des guillemets simples ? Doubles ? Triples ?

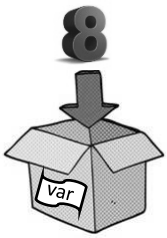
L'utilisation exclusive de la console pour entrer des instructions est, on le voit rapidement, peu pratique, puisqu'on ne peut pas modifier une ligne de code déjà écrite. Pour cela, on utilise l'éditeur : on entre les lignes de code comme dans un fichier texte (on l'appelle alors « script »), puis on exécute ce code (on envoie les instructions à l'ordinateur).

Partie II : Les variables

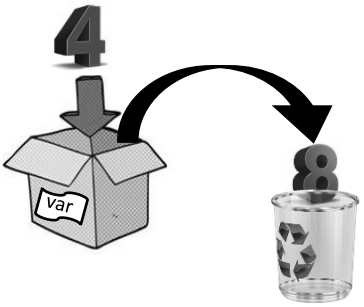
Une variable est comme une « boite » dans laquelle on stock une valeur. Pour créer une variable, on tape :

`nomdevariable = valeurdevariable`

Par exemple, taper `var = 8` revient à mettre la valeur 8 dans la boite var.



Attention, si on tape ensuite `var = 4`, la valeur qui était dans la boite var disparait et est remplacé par la valeur 4.



Attention : il faut à tout prix éviter les caractères spéciaux (accents par exemple) dans les noms de variable. Elles ne peuvent pas non plus commencer par un chiffre et sont ***sensibles au majuscules/minuscules*** *(ex : NOM ≠ Nom ≠ nOm)*

Pour afficher la valeur d'une variable, on tape : `nomdevariable` ou `print(nomdevariable)`.

II.1. Dans la console, taper les commandes suivantes :

```
1 | x = 8
2 | y = 5
3 | x + y
4 | print(x+y)
```

Faire la même chose dans un script et l'exécuter.
Quelle est la différence entre les deux façons d'afficher des variables ?

Il existe différents types de variables : les entiers « integer », les nombres à virgule « float », les chaines de caractère « string », etc… Nous les découvrirons au fur et à mesure du cours.
Attention : Une fois qu'une variable a été créée, le type de valeur que l'on a initialement mis dedans défini le type de la variable.

On peut aussi inclure des commentaires (lignes non-exécutées) dans le script en plaçant un # au début de la ligne. Ces commentaires sont extrêmement utiles, en particulier pour lire le code d'une autre personne (petite pensée pour votre prof là !!! ☺)

II.2. Taper puis exécuter le code ci-dessous

```
1 | print("Jouons ensemble !")
2 | prenom = input("Quel est ton nom ?\n")
3 | age = input("Quel est ton age ?\n")
4 | age = int(age)
5 | print("Salut",prenom,"qui a",age,"ans.")
```

II.3. Quelles sont les variables utilisées dans ce code ? Quel est leur type respectif ?

II.4. Que fait la commande `input()` ?

II.5. Que fait la commande `int()` ?

II.6. A quoi servent les virgules dans la commande `print()` ?

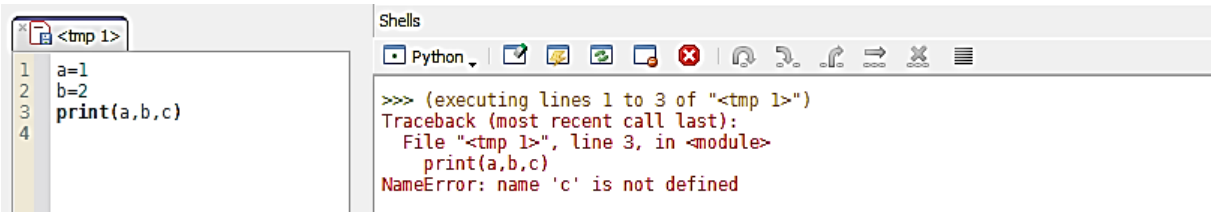
Vérifions que nous avons compris : Exercice flash !
(Disponible dans la correction de l'activité sur l'ENT)

Débogage

Lorsqu'on écrit un code, il est exceptionnel de ne pas faire d'erreur ! Il existe de multiples façons de corriger un code (en langage informatique, on appelle cela le débogage).

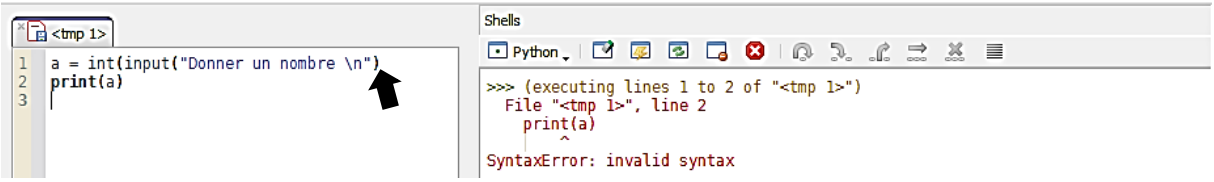
La première étape du débogage est de compiler le code. A ce moment, le compilateur python peut **indiquer dans la console les problèmes rencontrés et à quelles lignes**.

Exemple : La variable c présente dans le print n'existe pas !



Parfois, le compilateur indiquera la ligne directement en dessous de la ligne contenant l'erreur : c'est le cas lorsqu'on a oublié une parenthèse.

Exemple : Il manque une parenthèse ligne 1 !



Partie III : Exercices pour s'entrainer

Dans chacun des cas suivants, proposer un pseudo-code (ce que le code doit accomplir, en langage humain) puis écrire le code qui réalise la tâche demandée.

III.1. Demander à l'utilisateur deux nombres, les stocker dans deux variables *a* et *b* puis échanger les contenus de ces variables (la valeur de *a* dans *b* et vice-versa) et les afficher.

III.2. Demander à l'utilisateur trois nombres, les stocker dans trois variables *a*, *b* et *c*, puis échanger les contenus des variables pour que *a* contienne l'ancienne valeur de *b*, *b* celle de *c* et *c* celle de *a*, et les afficher.

Pour aller un peu plus loin…

Proposer un pseudo-code puis écrire le code qui demande à l'utilisateur trois nombres **entiers positifs non-nuls**, les stocke, dans l'ordre, dans trois variables *a*, *b* et *c*, puis calcul le résultat de la division euclidienne de $\left(\frac{a - b}{c} + c\right)$ par *b* dans une variable nommée *res* qui est ensuite affichée.

Test : Avec a = 2, b = 3 et c = 4, le résultat est 1.

Ce qu'il faut retenir

Un ordinateur est très efficace (il peut résoudre les calculs les plus complexes), mais aussi très TRÈS bête (il ne fait QUE ce qu'on lui dit de faire, ni plus, ni moins) et psycho-rigide (il faut lui dire quoi faire de la bonne façon, sinon, il ne comprend pas).

❖ Instructions mathématiques

Instruction	***Utilité***
**	Puissance
/	Division simple
//	Résultat de la division euclidienne
%	Reste de la division euclidienne
>=	Supérieur ou égal (booléen)
==	Egal (booléen)
!=	Différent (booléen)

❖ Variables

Une variable est comme une « boite » dans laquelle on stock une valeur.
Pour créer une variable, on tape : `nomdevariable = valeurdevariable`

Il existe différents types de variables. Une fois qu'une variable a été créée, le type de valeur que l'on a initialement mis dedans défini le type de la variable.

❖ Commandes

Commande.	***Utilité***
`var = valeur`	Assigner une valeur à une variable
`print()`	Affichage
`var = input()`	Demander une valeur à l'utilisateur et l'enregistrer dans la variable *var*
`int()`	Convertir une valeur en format « integer » (entier)