PTSI - PT lycée la Prat's	TP N°1a (1 à 2H)	Découverte de l'environnement de travail et Représentation des nombres et des caractères	IPT

I. Découverte d'un Environnement de développement intégré (IDE)

- Distribution complète utilisée : **WinPython** (autre possible : ANACONDA.)
  Téléchargeable à l'adresse suivante : http://sourceforge.net/projects/winpython/files/WinPython_3.12/

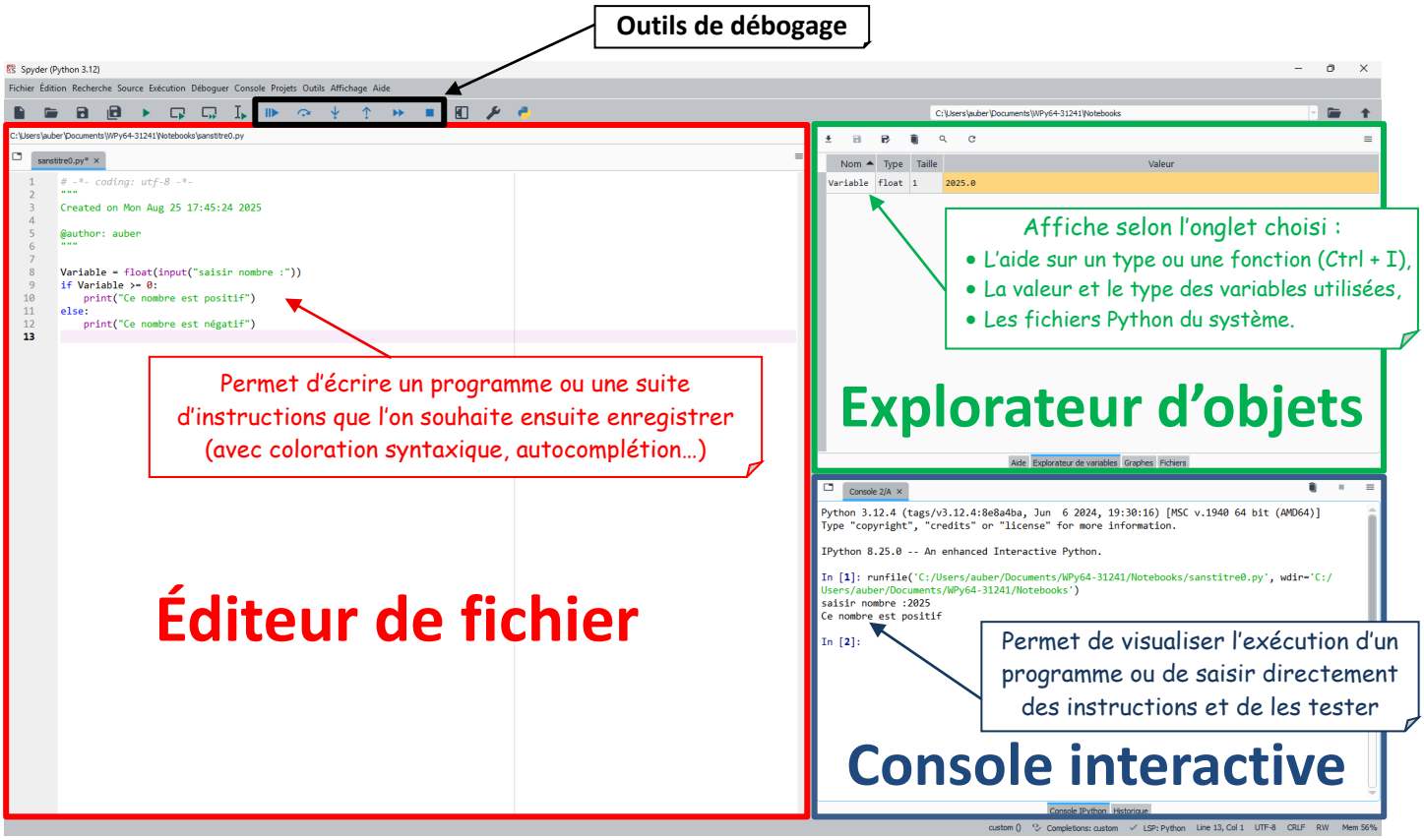
  WinPython est une distribution **Python** pour **Windows** qui fournit un environnement de développement Python portable et facile à utiliser. Cette distribution est conçue pour les développeurs travaillant sur des projets de science des données, de traitement de données ou de développement d'applications Python. Il peut être installé directement sur un **disque dur local** ou sur une **clé USB**, ce qui permet aux développeurs de travailler sur leur projet Python depuis n'importe quel ordinateur sans avoir à installer Python ou d'autres bibliothèques sur chaque machine.
  WinPython est pré-installé avec un ensemble complet de bibliothèques populaires pour la science des données (NumPy, Pandas, Matplotlib, SciPy, etc.) et **l'IDE Spyder**.

- Icône de lancement du logiciel Spyder :

  Spyder signifie "**S**cientific **Py**thon **D**evelopment **Envi**ronment", qui est un IDE spécialement conçu pour faciliter le développement en Python, en particulier pour les scientifiques et ingénieurs. Il est reconnu pour son interface intuitive, ses fonctionnalités avancées et sa compatibilité avec de nombreux outils scientifiques.
  Il propose une interface utilisateur unique qui combine plusieurs fonctionnalités :
  - **Éditeur de texte avancé :** facilite l'écriture et la compréhension du code Python (coloration syntaxique, auto-complétion et pliage de code),
  - **Explorateur de variables graphiques :** permet de visualiser et d'interagir avec les données (analyse des données plus intuitive),
  - **Console Python interactive :** permet d'exécuter un programme ou de tester des instructions isolées,
  - **Outils de débogage :** permet d'identifier l'origine d'une erreur de code.

A. Description de la fenêtre d'accueil de Spyder

Trois zones de travail :



1. Console interactive :

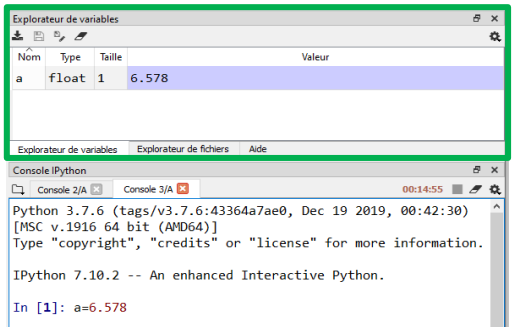
La console interactive de Python ou encore IPython (zone bleue) peut être utilisée comme une simple « machine à calculer », respectant les règles mathématiques sur les priorités des calculs :

Exposant (**) ; multiplication (*) ; division (/ et //) ; addition (+) ; soustraction (-)

Taper les scripts suivants dans **l'interpréteur** actif de la console : (essayer de prévoir le résultat obtenu)

```
3*2
5**2
4-2**3*2+1
4-2**3/2+1
(Noter la différence de « type » de résultats)
5//3
5./3
1j*3j
print ('bonjour')
```

2. Explorateur de variables :



- Cliquer sur l'onglet « explorateur de variables » de l'explorateur d'objets (zone verte).
  Un tableau affiche pour chaque variable du programme en cours d'exécution : son nom, son type, sa taille et sa valeur.

- Taper dans la console interactive : **(essayer de prévoir le résultat obtenu et regarder l'évolution de « a » et « b » au cours des frappes)**

```
a=3
a+a
b+1
```

  Pourquoi ce message d'erreur ?

```
b=4*a/2
c=[52,3,7,128]
d=5+23j
```

  Noter la différence de type et de taille entre les variables a, b, c et d.

```
b=b+1 (ou b+=1)
b=b-1 (ou b-=1)
b=b*2 (ou b*=2)
print(a,b)
```

  Remarque : *au cours de la frappe de : « **print(** » l'aide sur la fonction s'affiche automatiquement*

```
b="Informatique pour tous -"
a=" PTSI Cluny"
```

  Remarque : *les variables a et b ont changé de type au cours de ces quelques lignes.*
  En Python, le typage des variables est dynamique – implicite *(lors de l'affectation d'une variable, Python détecte automatiquement son type et en déduit les fonctions qu'il peut lui appliquer).*

```
c=b+a   # concaténation de deux chaînes de caractères
print(c)
print(b,a)
type(a)
```

- Ouvrir une nouvelle console IPython (ou interpréteur) (clic droit **sur** l'onglet Console), et taper

```
print(a)
```

  Que se passe-t-il ? Pourquoi ?

Conclusion :

Une variable est définie dans l'univers d'un programme, ici un interpréteur donné. Un autre interpréteur sera totalement indépendant du précédent, sans aucun lien : ils ne se « parlent » pas. Les variables de l'un sont inconnues de l'autre, ce sont deux bulles étanches.

En particulier, quand vous exécutez un programme (depuis l'éditeur) il vaut mieux le faire dans un nouvel interpréteur, pour qu'il n'y ait pas de variable déjà affectée qui traîne et le perturbe.

Définition : **Type d'une variable :**

Un type est une étiquette sur la variable, qui va décrire comment la manipuler (et parfois aussi les valeurs possibles de cette variable : IEEE 754 pour les flottants). Le type peut être déclaré a priori par l'utilisateur (typage explicite) ou détecté par la machine (typage implicite).
En Python, la fonction type() permet de connaitre le type d'une variable.

3. Éditeur :

- Taper le code suivant dans l'éditeur :

```
print('Bonjour')     #message de bienvenue
X=8                  # on affecte 8 à la variable X
print(x**2)          # calcul et affichage de x^2
```

  - La casse est importante pour les variables. Modifier le programme pour que celui-ci fonctionne.

- Modifier le programme pour que l'utilisateur rentre au clavier la valeur de X à afficher en utilisant la fonction : input(),

```
X=input("X=")
```

  - ***Que se passe-t-il ?***

La fonction input() permet à l'utilisateur de rentrer des informations au clavier, mais Python considère ce qui est saisi comme du texte (type : « str »). Il faut alors modifier le type de l'entrée de façon adaptée :

Exple : X= float(input("X=")) pour transformer l'entrée en réel « flottant » et l'affecter à X
X= int(input("X=")) pour transformer l'entrée en entier « int » et l'affecter à X

Rq : le type de variable peut être aussi imposé lors du calcul : print(int(X)**2) ou print(float(X)**2)

4. Débogage de programme :

- Taper le programme suivant dans l'éditeur :

```
X=10
Y=15
X=X+Y
Y=X
Z=8*X/(X-Y)
print(Z)
```

  - Enregistrer ce fichier : « débogage.py »
  - Utiliser le mode « débogueur » en cliquant sur l'icône « déboguer le script » :  ou (Ctrl+ F5)
  - Dans ce mode chaque ligne du programme est copiée dans « la console interactive » puis est testée à chaque appui sur l'icône « exécuter la ligne en cours » :  ou (Ctrl + F10)
  - On peut suivre l'avancement des variables dans « l'explorateur des variables » jusqu'à la fin du programme.
  - Si le programme a un problème, on peut connaitre facilement la ligne qui génère une erreur.

  - ***Quel est le problème de ce programme ?***

- Taper le programme suivant dans l'éditeur : (La structure ci-dessous sera détaillée dans les cours suivants.)

```
i=15
while i!=0              # tant que i est différent de 0
i=2-i
print(i)                # afficher i
```

  - **Remarque** : Spyder corrige automatiquement les « grosses » erreurs de frappes et possède une coloration syntaxique :
    - Les mots clés (while, for, …) sont en bleu et à la fin de la ligne il y a un « : »,
    - Les nombres en marron,
    - Les objets intégrés (print(), input() et toute autre fonction connue du langage) sont en violet.

Vous devez avoir à l'écran :

```
i=15
while i !=0 :
        i=2-i              # Spyder a indenté l'instruction du While
        print(i)
```

- Enregistrer ce fichier : « débogage_1.py ».
- Lancer le programme normalement en cliquant sur l'icône « exécuter le fichier » :  ou (F5).
- Lorsqu'un programme ne se termine pas (boucle sans fin), il est possible de le fermer en utilisant l'icône « interrompre la commande en cours » :  en haut à droite de la console ou bien en fermant la console (onglet Python1).
- Utiliser le mode « débogueur » :  puis exécuter le programme pas à pas  .

Quel est le problème de ce programme ?

On pourra placer un point d'arrêt (icône  , double clic droit ou F12) dans la marge de gauche de l'éditeur en face d'une ligne bien choisie du programme puis cliquer sur  ou (Ctrl+ F12) pour continuer l'exécution du programme jusqu'au prochain point d'arrêt.

II. Représentation des nombres :

Pour tous les exercices, utiliser l'annexe du cours sur les différents types de nombres.

Exercice 1. Affectation de variables

- Fermer toutes les consoles IPython dans la zone de la console interactive.
- Ouvrir un nouvel interpréteur dans l'éditeur.
- Taper les commandes suivantes dans la console, puis prédire et vérifier la valeur des variables (à l'aide de l'explorateur de variables) :

1.
```
>>> x=42
>>> y=10
>>> x=y
>>> y=x
```

2.
```
>>> x=42
>>> y=10
>>> z=x
>>> x=y
>>> y=z
```

3.
```
>>> (x,y)=(42,10)
>>> (x,y)=(y,x)
```

Remarque : La deuxième ligne du cas N°3 permet une affectation simultanée

Exercice 2. Les Booléens (type : Bool)

- Tester les commandes suivantes :

```
>>> 3==5
>>> 841>0
>>> 2**10==1024
```

- Saisir : `>>> x, y = 10, -5` tester :

```
>>> x>0 and y<0
>>> x<0 and y<0
>>> not (x>0)
>>> not (x==-15)
>>> x>0 or y>0
>>> x<0 or y<0
```

puis tester :

```
>>> x>0 or blabla
>>> blabla or x>0
```

Exercice 3. Les entiers naturels (type : int)

Numération de position et bases

a. Tester les commandes : 173//11 et 173%11 et indiquer leur lien avec la division euclidienne.

b. Trouver la représentation du nombre 14_{10} en base 2, ainsi que de 1010111_2 en base 10.

c. Trouver la représentation du nombre 183_{10} en base 16, ainsi que de $7E4_{16}$ en base 10.

- Exécuter les instructions : `>>> bin(25)` `>>> int('11001',2)` `>>> 0b1010101`
- Quels résultats obtient-on ? Indiquer leur type. Valider vos résultats du 3.b.

- Même question avec : `>>> hex(27)` `>>> int('1b',16)` `>>> 0x5ad9`
- Quels résultats obtient-on ? Indiquer leur type. Valider vos résultats du 3.c.

- Ecrire chaque digit du nombre $5b3f_{16}$ en binaire sur 4 bits et le comparer à sa conversion en binaire. Proposer une méthode de correspondance entre les bases 2 et 16.

Exercice 4. Les nombres réels – où à virgule flottante (type : float)

- Observer l'effet des commandes suivantes :

1. `>>> 1` **2.** `>>> float(1)` **3.** `>>> 1.` **4.** `>>> int(1.278)`

- A quel type appartiennent ces nombres ?

- Observer l'effet des commandes suivantes :

`>>> 1 + 2**-53 - 1` `>>> 1 - 1 + 2**-53` `>>>1.0 + 2**53 == 2**53` `>>> (1+ 2**53) - 2**53`

- Expliquer le phénomène « d'absorption ».

Exercice 5. Dépassement de capacité

- Observer l'effet des commandes suivantes :

`>>> 10.**100**100` `>>> 2.**1023` `>>> 2.**1024` `>>>2**1024`

- Expliquer ces différents résultats.

💻 Observer l'effet des commandes suivantes :

>>> 1 + 10**-15 > 1 >>> 1 + 10**-16 > 1 >>> 1 + 10**-16 >= 1

Rq : Un nombre flottant N se code : $N = (-1)^S \times 1,M \times 2^E$ (S : signe, E : exposant biaisé, M : mantisse)
Python code les flottants en double précision, sur 64 bits : S, E et M sont définis sur 1, 11 et 52 bits.
L'exposant biaisé E codé sur 11 bits, peut donc prendre des valeurs comprises entre : -1024 et 1023
La plus petite valeur de M est donc : $2^{-52} \approx 2,22.10^{-16}$
La plus grande valeur de N est d'environ 2^{1024}

Exercice 6. Quelques bizarreries

💻 Observer les résultats de :
1.1+2.7, 1.1*2.7, (1./3**2)*3**2 et (1./3**6)*3**6

✍ Expliquer le phénomène.

Pour ceux qui savent programmer :

Exercice 7. Codage d'un entier naturel en base k

Représenter en base k un entier naturel exprimé en base 10 (avec k<9).
Revoir si nécessaire la méthode vue en cours.
L'objectif est d'obtenir l'expression de l'entier n en base k, sous la forme : $a_i \ldots a_1 a_0$.

Algorithme proposé en pseudo code :

Données : n, k
Tant que n≠0 **faire**
  a ← reste de la division euclidienne de n par k
  n ← quotient de la division euclidienne de n par k
Résultat : la représentation $a_i \ldots a_1 a_0$

Exercice 8. Valeur absolue

Ecrire un programme qui permet de donner la valeur absolue d'un nombre rentré au clavier, sans utiliser le module math : abs(x), mais en traduisant en Python l'algorithme suivant :

Si valeur <0 **alors**
  Retourne – valeur
Sinon
  Retourne valeur
Fin Si

Exercice 9. Décomposition d'une somme d'argent

On souhaite programmer une machine qui rend la monnaie. Elle dispose de pièces de 2 euros, 1 euro, 50cts d'euros, 20cts d'euros, 10cts d'euros.
Connaissant la somme à payer et la somme versée, calculer le nombre de pièces de chaque sorte à rendre, de façon à minimiser le nombre total de pièces rendues.

- On conseille de saisir les sommes sous forme de centimes (valeur entière).
- On suppose que la machine n'est jamais à cours de monnaie.

💻 Ecrire le programme permettant de réaliser ce travail.