Explorer les variables, les types et les opérateurs

Découvrir expérimentalement le rôle de l'interpréteur, suivre l'évolution des variables, distinguer les principaux types simples et construire des expressions arithmétiques et booléennes fiables.

Situation. Un système embarqué reçoit une mesure brute provenant d'un capteur. Avant de l'exploiter, il doit la convertir, vérifier sa cohérence et décider si elle signale une situation anormale. Ce TP conduit progressivement à l'écriture de ce traitement.

Méthode de travail. Pour chaque expérience :

1. prévoir le résultat sans lancer Python ;
2. exécuter le code et relever le résultat exact ;
3. expliquer les éventuels écarts et ne pas hésiter à m'appeler ;
4. conserver les notes et éventuels coded pour plus tard.

1 Découvrir la console et le script pas à pas

1.1 Repérer les zones de Spyder

Spyder est l'environnement de développement (IDE) que nous allons utiliser. On y distingue trois zones :

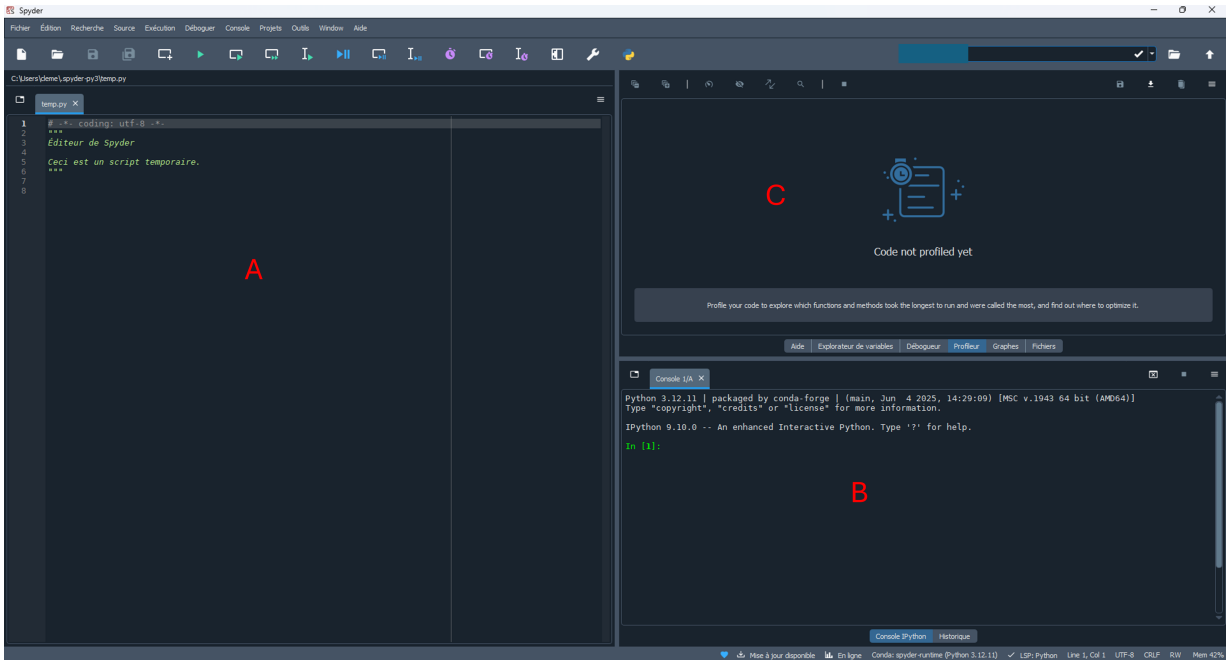

FIGURE 1 – Les principales zones de Spyder

- **Éditeur :** cette zone permet de rédiger et d'enregistrer un script dans un fichier portant l'extension `.py`.

- **Console Python :** cette zone permet d'exécuter immédiatement une instruction, d'afficher les résultats et d'échanger avec un script en cours d'exécution.
- **Panneaux complémentaires :** les onglets donnent notamment accès à l'explorateur de variables, aux graphiques, aux fichiers et à l'aide.

Question 1 : Repérer ces zones dans Spyder (figure 1) et faire la correspondance entre les zones et les lettres.

1.2 Premiers calculs dans la console

Dans la console Python, cliquer après l'invite `In [...]:` et saisir les instructions suivantes une par une. Valider chaque ligne avec la touche `Entrée`.

```
largeur = 8
hauteur = 5
largeur * hauteur
```

Question 2 : Déterminer le moment où chaque instruction est exécutée, puis distinguer l'effet d'une affectation de celui d'une expression saisie seule dans la console.

1.3 Créer, enregistrer et exécuter un script

Effectuer les opérations suivantes dans l'ordre indiqué.

1. Créer un dossier `TP1` dans votre répertoire d'informatique (ITC).
2. Dans Spyder, choisir *Fichier* > *Nouveau fichier*.
3. Enregistrer immédiatement le fichier dans `TP1` sous le nom `TP1_C1.py`.
4. Saisir le programme ci-dessous dans l'éditeur.
5. Enregistrer avec `Ctrl`+`S`, puis exécuter avec `F5` ou le bouton *Exécuter le fichier* (flèche verte 'start').

```
largeur = 8
hauteur = 5
aire = largeur * hauteur
aire
```

Question 3 : Prévoir puis relever les lignes affichées ou pas dans la console. Expliquer l'absence d'affichage produite par la ligne contenant seulement `aire` dans le script.

Question 4 : A partir de l'explorateur de variable (zone C), déterminer le nom et la valeur des variables `largeur`, `hauteur` et `aire`.

1.4 Échanger avec le script grâce à `print` et `input`

La fonction `print(...)` affiche dans la console les objets placés entre parenthèses. A l'inverse, la fonction `input(...)` affiche un message, suspend l'exécution du script et attend une saisie au clavier dans la console.

> **Remarque :** Elle renvoie toujours le texte saisi, donc une variable de type `str`. Nous en parlerons plus tard.

Remplacer le script précédent par les lignes suivantes, enregistrer, puis exécuter de nouveau le fichier.

```
print("Debut du script")
prenom = input("Entrer le prenom : ")
age = input("Entrer l'age : ")
print("Bonjour", prenom)
print("Votre age est :", age)
```

1.5 Retrouver dans la console les objets créés par le script

Après l'exécution du script, revenir à l'invite de la console sans relancer le fichier et saisir ligne à ligne :

```
prenom
age
aire
aire / 2
```

Observer également *l'explorateur de variables* de Spyder.

Question 5 : Relever les objets encore accessibles après la fin du script et montrer que la console permet de poursuivre leur manipulation.

À retenir. La console exécute immédiatement une instruction saisie. Un script est un fichier enregistré qui permet de conserver et de réexécuter plusieurs instructions dans l'ordre. `print` envoie des informations vers la console ; `input` récupère une saisie effectuée dans la console. Après l'exécution, les objets créés par le script restent accessibles dans le contexte de l'interpréteur.

2 Suivre les variables comme un programme

Question 6 : Sans exécuter le code, compléter mentalement le tableau d'évolution associé.

```
a = 7
b = a
a = a + 3
c = (a == b)
b += 1
```

Après l'instruction	Valeur de a	Valeur de b	Valeur de c
`a = 7`			
`b = a`			
`a = a + 3`			
`c = (a == b)`			
`b += 1`			

Question 7 : Exécuter ensuite le code ligne par ligne en consultant l'explorateur de variables. Corriger le tableau et donner les valeurs finales de a, b et c.

Question 8 : Expliquer la validité de l'instruction `a = a + 3` en Python malgré l'impossibilité de l'égalité mathématique $a = a + 3$, puis justifier que la modification de a ne modifie pas automatiquement b.

À retenir. Une variable est caractérisée par un nom, une valeur et un type. Une affectation modifie uniquement la variable placée à gauche du symbole =.

3 Les types

Question 9 : Pour le script suivant, associer chaque valeur à son type Python, puis identifier les objets qui représentent un nombre.

```
n = 42
x = 42.0
```

```
z = 1 + 2j
test = n > 0
texte = "monTexte"

print(type(n), type(x), type(z))
print(type(test), type(texte))
```

Question 10 : Exécuter le script (à la suite ou dans un nouveau .py) puis déterminer ce que fait la fonction `type(...)`.

3.1 Influence du nom sur le type

Exécuter successivement les instructions suivantes dans la console.

```
mesure = 12
print(mesure, type(mesure))
mesure = 12.0
print(mesure, type(mesure))
mesure = "12"
print(mesure, type(mesure))
```

Question 11 : Déterminer si le nom `mesure` impose un type, puis indiquer ce qui fixe le type de la variable à chaque étape.

3.2 Conversions explicites

Pour chacune des expressions suivantes, prévoir la valeur et le type du résultat, puis vérifier dans la console.

Expression	Valeur prévue	Type prévu
`float(7)`		
`int(7.9)`		
`complex(3)`		
`bool(0)`		
`bool(-2)`		
`int("12") + 3`		
`"12" + "3"`		

Question 12 : Expliquer en particulier la différence entre les deux dernières expressions et l'effet de `int(7.9)`.

À retenir. En Python, le type appartient à l'objet manipulé. Les fonctions `int`, `float`, `complex`, `bool` et `str` permettent de demander explicitement certaines conversions.

4 Choisir le bon opérateur

4.1 Retrouver une division euclidienne

Un enregistrement audio dure 7 384 secondes.

Question 13 : Sans utiliser de fonction de conversion, calculer avec `//` et le modulo le nombre d'heures, de minutes et de secondes orrespondant. Vérifier que le programme affiche "2 h 3 min 4 s".

Question 14 : Écrire une unique expression booléenne qui vérifie la recomposition de la durée à partir de `heures`, `minutes` et `secondes`.

4.2 Précision des nombres flottants

Question 15 : Exécuter, une à une, les instructions suivantes. Prévoir les résultats, exécuter, comparer puis conclure.

```
somme = 0.1 + 0.2
print(somme)
print(somme == 0.3)
print(abs(somme - 0.3))
```

À retenir. Un `float` est une représentation approchée. Pour comparer deux résultats calculés, on teste généralement si leur écart est inférieur à une tolérance choisie.

5 Construire et vérifier des expressions booléennes

5.1 Table de vérité expérimentale

Question 16 : Pour chaque couple de valeurs de `a` et `b`, exécuter les expressions `not a`, `a and b` et `a or b`, puis compléter le tableau. Déduire de l'expérience la règle qui rend `a and b` vraie, puis celle qui rend `a or b` vraie.

a	b	not a	a and b	a or b
False	False			
False	True			
True	False			
True	True			

5.2 Traduire une spécification

Une température `T` est considérée comme plausible si elle appartient à l'intervalle $[-20;80]$. Une alerte est déclenchée lorsqu'une température plausible atteint ou dépasse 30 °C.

Question 17 : Écrire les expressions booléennes `plausible` et `alerte`, puis les tester successivement pour `T = -30`, `T = 22.5`, `T = 30` et `T = 95`.

Question 18 : Écrire une expression `fonctionnement_normal` vraie uniquement pour une mesure plausible sans alerte active.

6 Défi de synthèse : qualifier une mesure de capteur

Créer un nouveau fichier .py (un nouveau script) pour cette partie.
Un convertisseur fournit un entier `brut` compris entre 0 et 1023. La tension associée est donnée par

$$U = 5\,\frac{\text{brut}}{1023},$$

et le capteur délivre une température en degrés Celsius selon

$$T = 100\,(U - 0{,}5).$$

Question 19 : Écrire un programme qui, à partir de `brut = 164`, calcule `U` et `T`, puis affiche leur valeur et leur type.

Question 20 : Ajouter les booléens `code_valide`, `temperature_plausible`, `alerte` et `fonctionnement_normal`. Ils doivent traduire respectivement : $0 \leq$ `brut` ≤ 1023, $-20 \leq T \leq 80$, une température plausible supérieure ou égale à 30 °C, puis une situation plausible sans alerte.

Question 21 : Tester le programme avec `brut = -1`, `brut = 100`, `brut = 164` et `brut = 1100`. Présenter les résultats dans un tableau et justifier chaque booléen obtenu.

brut	T	**Code valide**	T **plausible**	**Alerte**	**Normal**	
−1						
100						
164						
1100						

Question 22 : Rédiger en trois phrases un bilan distinguant clairement affectation, comparaison et opération booléenne.