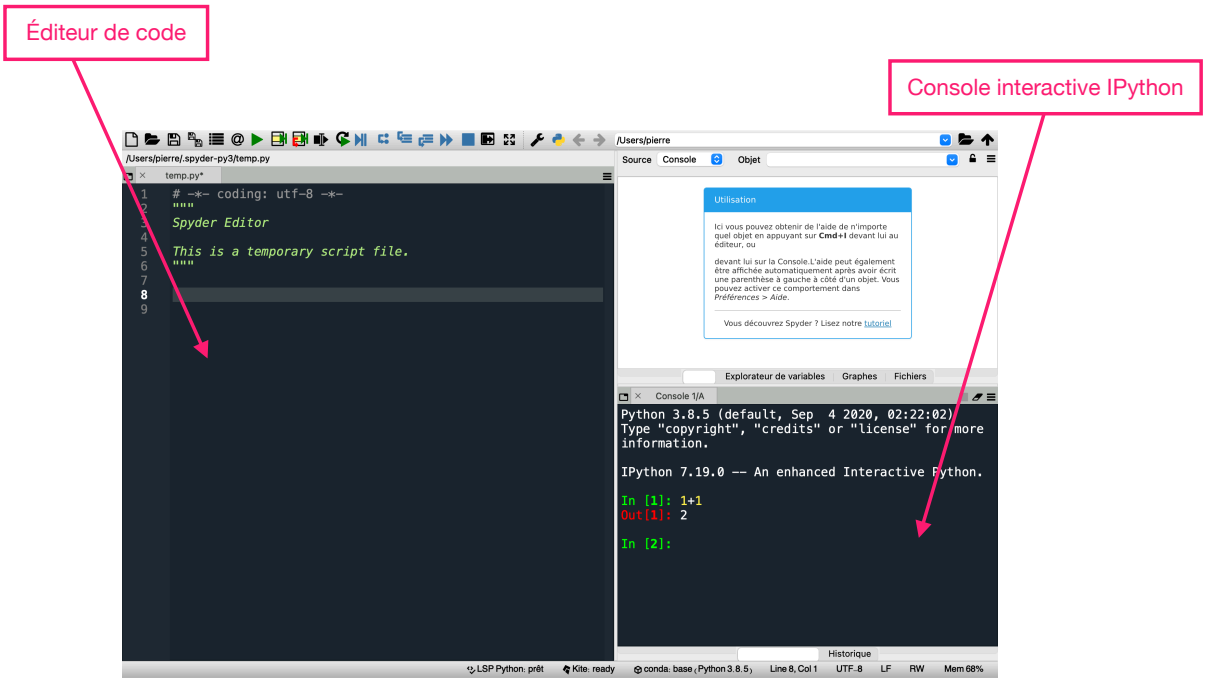
▶ Exercice 0 – Préliminaires

Il existe de multiples façons de travailler avec Python. Pour nos TP en classe préparatoire ECG, nous utiliserons l'environnement de développement intégré (IDE en anglais) nommé Spyder. Il est inclus dans la distribution Python nommée Anaconda qui est extrêmement populaire. Un IDE contient de nombreux outils pour programmer dans un langage donné. Il contient notamment un éditeur de code dans lequel les différents éléments du code sont colorés en fonction de leur nature : on parle de « coloration syntaxique ».

1. Mise en route

- Allumer le PC devant vous. Attendre le temps nécessaire. ;-)
- Depuis le menu « Démarrer » chercher « Anaconda » puis « Spyder ».



2. Utilisation interactive avec IPython
Dans la console interactive IPython située en bas à droite taper un calcul très simple (du genre de $1+1$) et valider. Le résultat s'affiche immédiatement en dessous.

3. Utilisation classique avec l'éditeur de code
Dans l'éditeur de code taper un script basique tel que (par exemple) :

```
x = 1 + 3
print(x)
```

Pour lancer l'exécution de votre script, fouillez dans les menus ou bien taper simplement la touche F5 (cette manière de lancer l'exécution est propre à Spyder).

▶ Exercice 1 – Fonctions

1. Saisir la fonction suivante puis tester là avec quelques valeurs particulières de n (supposé entier, même si cela n'a pas d'importance) :

```
def carre(n):
    """
    Prend en argument un entier n et renvoie son carré.
    """
    return n*n
```

Il faut pour cela rajouter quelques commandes : à vous de jouer!

2. Écrire une fonction `moyenne(a, b, c)` prenant en entrée trois nombres (entiers ou flottants, peu importe) et qui renvoie leur moyenne.
Tester cette fonction avec différentes valeurs de (a,b,c).

3. Écrire une fonction `somme_produit(x, y)` prenant en entrée deux valeurs numériques et renvoyant le **couple** formé par la somme et le produit des deux nombres.
Précision – *Un **couple** en Python est exactement l'analogue de ce que l'on rencontre en mathématiques : deux éléments encadrés par des parenthèses et séparés par une virgule. Pour un nombre quelconque de termes on parle de **tuples**. À ne pas confondre avec la notion de **liste** sur laquelle nous reviendrons.*
4. On rappelle que la température t en degrés Fahrenheit correspond à la température $(t-32)\times\frac{5}{9}$ en degrés Celsius. Écrire deux fonctions `F_vers_C` et `C_vers_F` effectuant les conversions d'un système dans un autre.
À combien de degré Celsius correspond la température de 451 degrés Fahrenheit ?
À quel écrivain cette température fait-elle référence ?

▶ Exercice 2 – Structure conditionnelle

1. On considère le script suivant :

```
x = -3
if x >= 0:
    y = x
else:
    y = -x
print(y)
```

Quel affichage ce script va-t-il provoquer ?
2. Écrire en fonction `valeur_absolue(x)` prenant en argument une valeur numérique x (entier ou flottant, peu importe) et qui renvoie la valeur absolue de x.
3. Écrire une fonction `trinome(a, b , c)` qui prend comme arguments trois nombres a, b et c et qui renvoie le nombre de solutions réelles de l'équation $ax^2+bx+c=0$.
4. Écrire une fonction `voyelle` qui prend en argument une chaîne de caractères (composée uniquement de lettres minuscules) et qui indique par un booléen (`True` ou `False`) suivant que cette chaîne commence ou pas par une voyelle (`a, e, i, o, u, y`).

▶ Exercice 3 – Structure itératives : for et while

1. Saisir, exécuter et comprendre le script suivant :

```
for n in range(1,10):
    print(n*n)
```

Quelle est la dernière valeur affichée ? Que remarque-t-on ?
2. Saisir, exécuter et comprendre le script suivant :

```
k = 1
while k < 11:
    print(k)
    k = k + 2
```

Quelles sont les valeurs affichées ? Que remarque-t-on ?

▶ Exercice 4 – Quelques boucles simples

1. Écrire une fonction `somme(n)` prenant en argument un entier naturel non nul et renvoyant la somme des entiers entre 1 et n, c'est-à-dire $1+2+3+\cdots+n$.
2. Écrire une fonction `factorielle(n)` prenant en argument un entier naturel et renvoyant la factorielle de n c'est-à-dire $n!=1\times2\times3\times\cdots\times n$ (avec la convention $0!=1$).
3. Écrire une fonction `somme_impairs(n)` prenant en argument un entier naturel n non nul et renvoyant la somme de tous les entiers impairs inférieurs ou égaux à n.
4. Écrire une fonction `puissance(n)` prenant en argument un entier naturel n non nul et qui renvoie la plus grande puissance de 2 inférieure ou égale à n.

▶ Exercice 5 – Conjecture de Syracuse

Pour tout entier naturel a, on peut considérer la suite $(u_n)_{n\in\mathbb{N}}$ définie par $u_0=a$ et pour tout $n\in\mathbb{N}$:

$$u_{n+1}=\begin{cases}\frac{1}{2}u_n & \text{si } u_n \text{ est pair}\\ 3u_n+1 & \text{si } u_n \text{ est impair}\end{cases}$$

1. Écrire une fonction `suite(a, n)` prenant comme arguments a et n et qui renvoie le terme u_n de la suite correspondante.
2. Écrire une fonction `attente(a)` prenant un entier naturel a et qui renvoie le plus petit entier n tel que $u_n=1$ (pour la suite démarrant à a) ou qui renvoie -1 si l'on n'a toujours pas rencontré la valeur 1 au bout de 1000 termes de la suite.