Python - Rappels de première année

Cours de ECG-2

Lycée Gerville Réache

Chapitre 1

Sommaire I

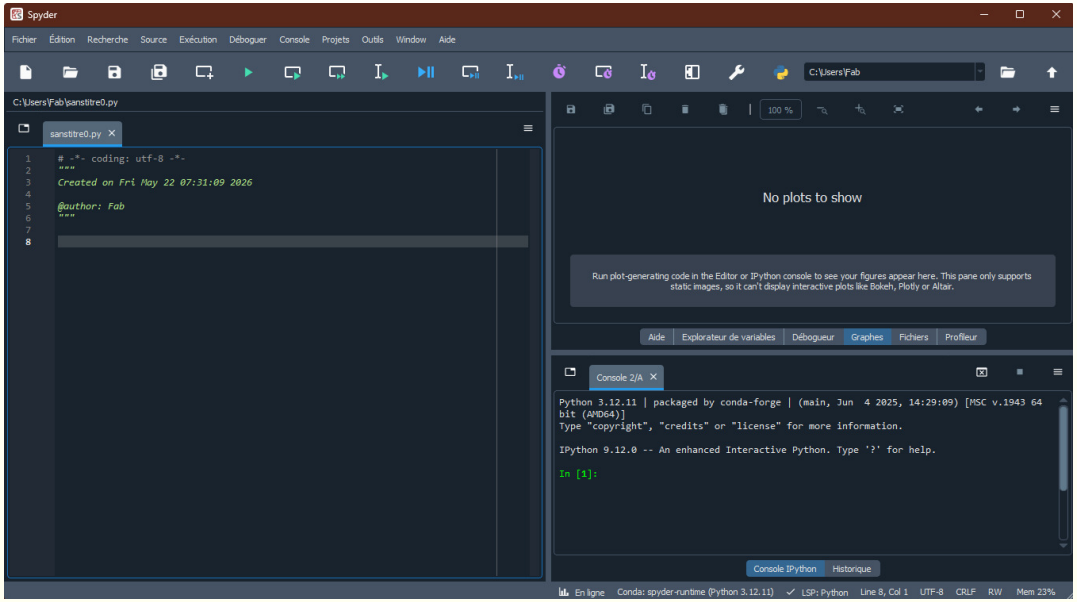
1. Prise en main
2. Rappels des principales commandes vues en première année

I. Prise en main

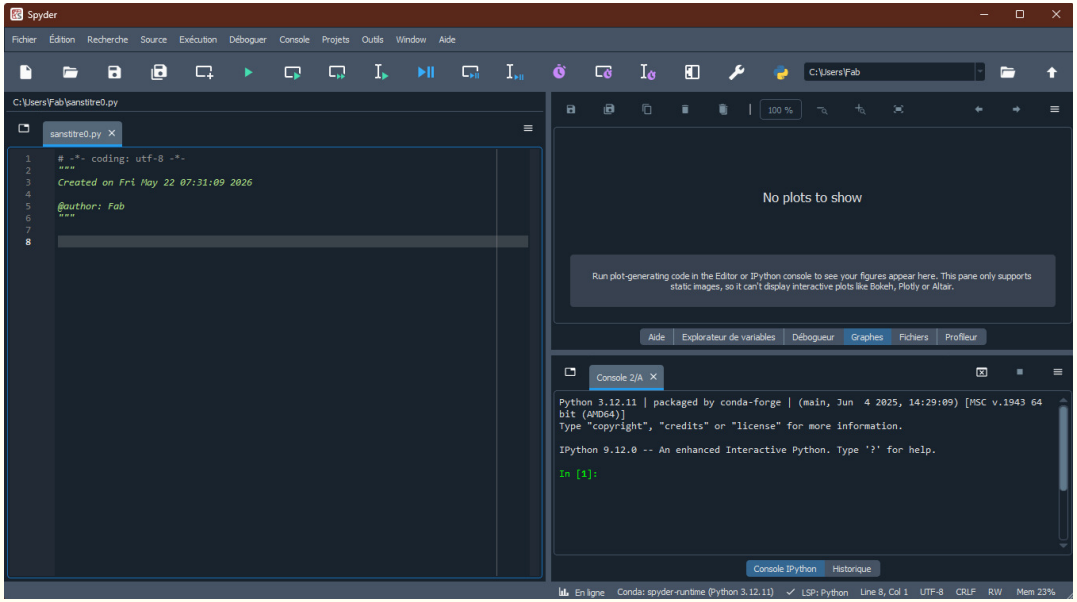
1 Prise en main

2 Rappels des principales commandes vues en première année

L'environnement de développement est constitué de plusieurs fenêtres.

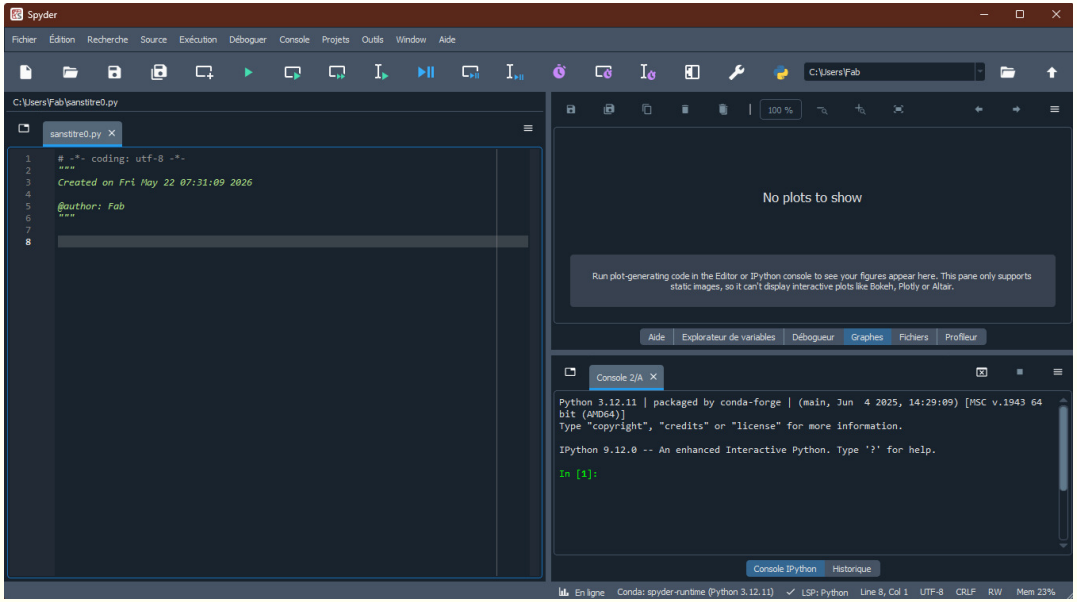


L'environnement de développement est constitué de plusieurs fenêtres.



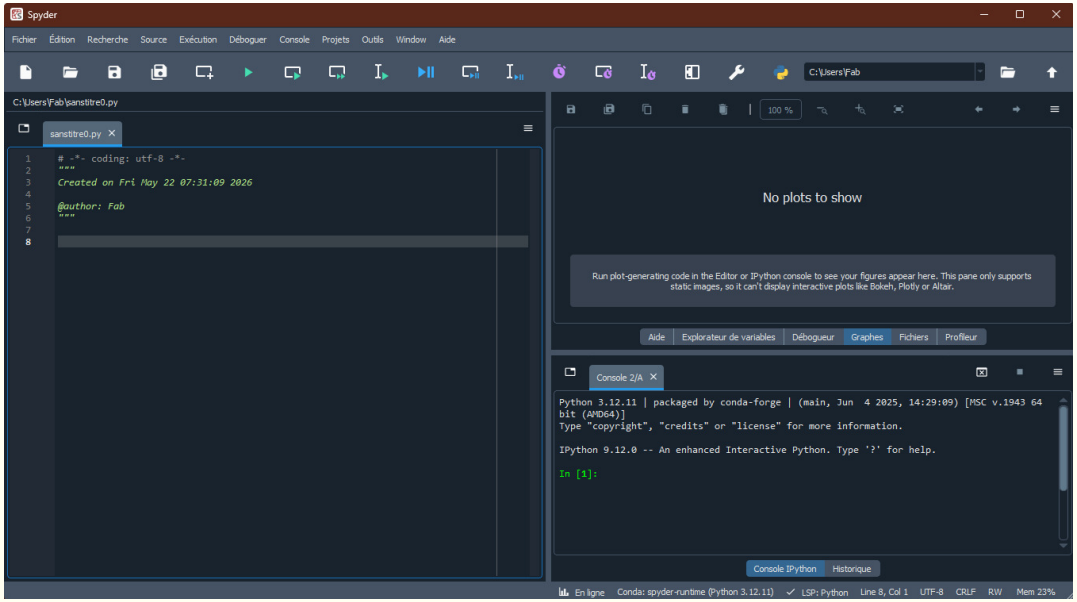
1. Un interpréteur interactif appelé **console** permet d'exécuter des instructions en ligne de commande.

L'environnement de développement est constitué de plusieurs fenêtres.



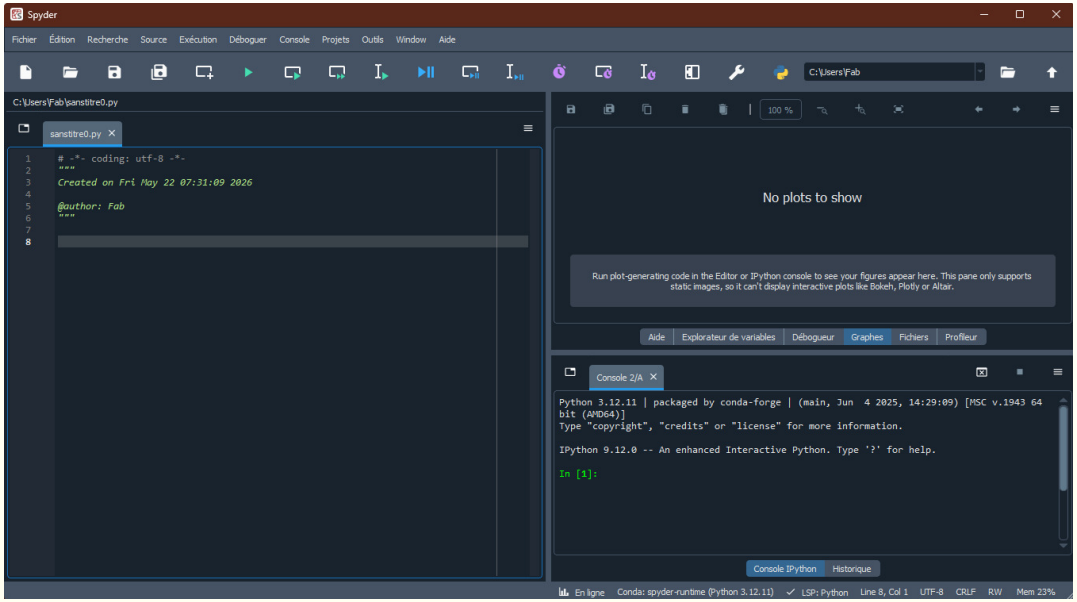
1. Un interpréteur interactif appelé **console** permet d'exécuter des instructions en ligne de commande.
2. Un **éditeur** de texte destiné à la saisie des programmes qui offre par défaut un certain nombre de fonctionnalités qui améliore l'expérience utilisateur. Après enregistrement et compilation du code, l'exécution se déroule dans la console.

L'environnement de développement est constitué de plusieurs fenêtres.



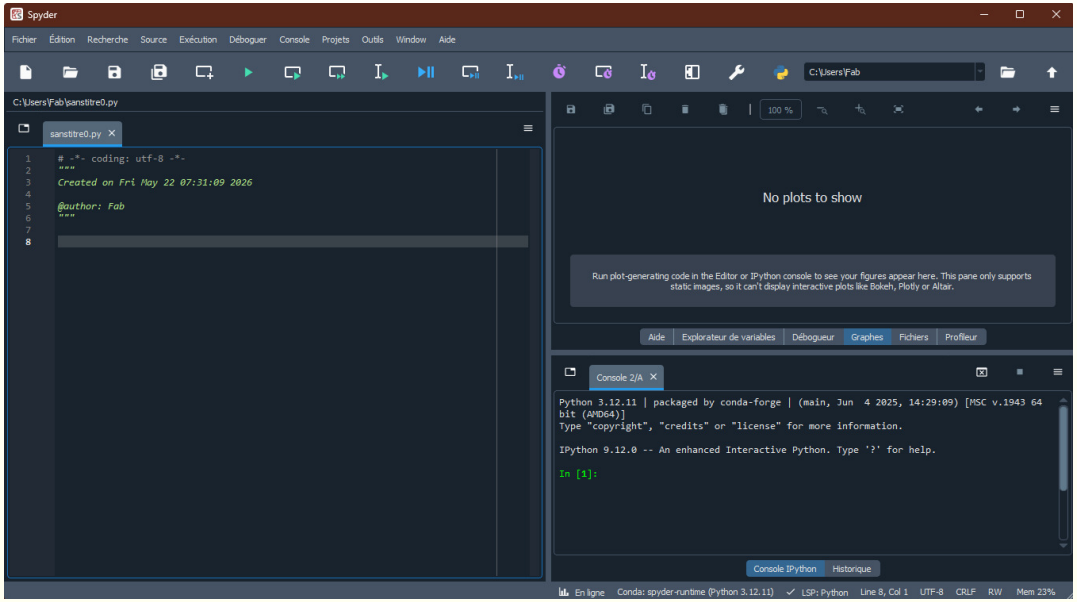
1. Un interpréteur interactif appelé **console** permet d'exécuter des instructions en ligne de commande.
2. Un **éditeur** de texte destiné à la saisie des programmes qui offre par défaut un certain nombre de fonctionnalités qui améliore l'expérience utilisateur. Après enregistrement et compilation du code, l'exécution se déroule dans la console.
3. Un **affichage** pour les graphiques et histogrammes créés.

L'environnement de développement est constitué de plusieurs fenêtres.



1. Un interpréteur interactif appelé **console** permet d'exécuter des instructions en ligne de commande.
2. Un **éditeur** de texte destiné à la saisie des programmes qui offre par défaut un certain nombre de fonctionnalités qui améliore l'expérience utilisateur. Après enregistrement et compilation du code, l'exécution se déroule dans la console.
3. Un **affichage** pour les graphiques et histogrammes créés.
4. Un explorateur de variables, qui permet de connaitre les valeurs contenues dans les variables en cours d'utilisation.

L'environnement de développement est constitué de plusieurs fenêtres.



1. Un interpréteur interactif appelé **console** permet d'exécuter des instructions en ligne de commande.
2. Un **éditeur** de texte destiné à la saisie des programmes qui offre par défaut un certain nombre de fonctionnalités qui améliore l'expérience utilisateur. Après enregistrement et compilation du code, l'exécution se déroule dans la console.
3. Un **affichage** pour les graphiques et histogrammes créés.
4. Un explorateur de variables, qui permet de connaitre les valeurs contenues dans les variables en cours d'utilisation.
5. Un explorateur de fichiers qui permet de se déplacer dans l'arborescence de fichier et d'en charger dans l'éditeur de texte.

II. Rappels des principales commandes vues en première année

1 Prise en main

2 **Rappels des principales commandes vues en première année**
- Ce que dit le programme
- Les affectations
- Les `Types`
- Les fonctions d'entrée et de sortie
- Instructions conditionnelles
- Les fonctions

Toute la richesse du langage Python ne peut pas être entièrement maîtrisée par un étudiant, aussi le paragraphe ci-dessous liste limitativement les éléments du langage Python (version 3 ou supérieure) dont la connaissance est exigible des étudiants.

Il s'agit de la liste des commandes utiles pour les travaux pratiques des deux années de formation.

- Connaître et savoir manipuler les différents objets de base du langage `Python` (les entiers, les nombres flottants, les booléens, les chaînes de caractères, les variables).

- Connaître et savoir manipuler les différents objets de base du langage `Python` (les entiers, les nombres flottants, les booléens, les chaînes de caractères, les variables).
- Savoir programmer en `Python` : fonctions d'entrée et de sortie, boucles `if`, ...

- Connaître et savoir manipuler les différents objets de base du langage `Python` (les entiers, les nombres flottants, les booléens, les chaînes de caractères, les variables).
- Savoir programmer en `Python` : fonctions d'entrée et de sortie, boucles `if`, ...
- Savoir définir une `fonction` en `Python`.

- Opérations sur les entiers et les flottants : `+` , `-`, `*`, `/`, `**`

- Opérations sur les entiers et les flottants : `+` , `-`, `*`, `/`, `**`
- Comparaison et connecteurs logiques pour les booléens : `==`, `>`, `<`, `>=`, `<=`, `!=`, `True`, `False`, `and`, `or`, `not`

- Opérations sur les entiers et les flottants : `+` , `-`, `*`, `/`, `**`
- Comparaison et connecteurs logiques pour les booléens : `==`, `>`, `<`, `>=`, `<=`, `!=`, `True`, `False`, `and`, `or`, `not`
- Programmation : `=`, `input`, `print`, `if`, `elif`, `else`, `def`, `return`, ...

La notion de `variable` est essentielle en informatique. Elle s'articule autour de 3 principes :

- stockage d'une information,

La notion de `variable` est essentielle en informatique. Elle s'articule autour de 3 principes :

- stockage d'une information,
- accès au contenu du stockage,

La notion de `variable` est essentielle en informatique. Elle s'articule autour de 3 principes :

- stockage d'une information,
- accès au contenu du stockage,
- modification de l'information stockée.

La notion de `variable` est essentielle en informatique. Elle s'articule autour de 3 principes :

- stockage d'une information,
- accès au contenu du stockage,
- modification de l'information stockée.

La notion de `variable` est essentielle en informatique. Elle s'articule autour de 3 principes :

- stockage d'une information,
- accès au contenu du stockage,
- modification de l'information stockée.

Dans la pratique, une variable est repérée par son nom auquel on associe une valeur `val`. Ce mécanisme d'association `nom = val` est appelé affectation.

Éditeur

```
1  a=4          # crée la variable a et lui affecte la valeur 4
2  a,b=1,2          # a vaut 1 et b vaut 2
```

La notion de `variable` est essentielle en informatique. Elle s'articule autour de 3 principes :

- stockage d'une information,
- accès au contenu du stockage,
- modification de l'information stockée.

Dans la pratique, une variable est repérée par son nom auquel on associe une valeur `val`. Ce mécanisme d'association `nom = val` est appelé affectation.

Éditeur

```
1  a=4          # crée la variable a et lui affecte la valeur 4
2  a,b=1,2          # a vaut 1 et b vaut 2
```

Définition 1 – Portée d'une variable :

La portée d'une variable est la zone de code dans laquelle la variable vit (on peut récupérer sa valeur, la mettre à jour).

Illustrons cette notion, plus subtile qu'il n'y parait, sur des exemples.

Exercice 1 :

1. On considère tout d'abord le programme suivant.

Éditeur

```
1  a = 1
2  b = 1
3  for i in range (3):
4      print(a)
```

Quel est le résultat produit par ce programme ? Commenter.

Illustrons cette notion, plus subtile qu'il n'y parait, sur des exemples.

Exercice 1 :

1. On considère tout d'abord le programme suivant.

Éditeur

```
1  a = 1
2  b = 1
3  for i in range (3):
4      print(a)
```

Quel est le résultat produit par ce programme ? Commenter.

1. Les variables `a` et `b` sont des variables définies dans le bloc de code principal du programme. Elles sont dites globales. Elles sont accessibles partout dans le module.

Exercice 1 :

2. On ajoute alors les instructions suivantes.

Éditeur

```
1  def f():
2      b = 2
3      c = 3
4      print(a, b, c)
5
6  print(b)
```

Quel est la valeur de `b` affichée par ce programme ? Commenter.

Exercice 1 :

2. On ajoute alors les instructions suivantes.

Éditeur

```
1  def f():
2      b = 2
3      c = 3
4      print(a, b, c)
5
6  print(b)
```

Quel est la valeur de `b` affichée par ce programme ? Commenter.

2. La valeur affichée est 1.
En effet, la définition `b=2` est une définition locale à la fonction `f`. Cette définition n'est accessible qu'au sein du bloc de code de la fonction.
Lors de l'appel à la fonction `f`, une case mémoire est créée pour stocker la valeur de `b`. Cette case est supprimée à la fin de l'exécution de `f`.

entiers	nombres à virgule	booléens	chaînes de caractère	objets matriciels
int	float	bool	str	np.array

entiers	nombres à virgule	booléens	chaînes de caractère	objets matriciels
int	float	bool	str	np.array

Éditeur

```
1  type(a)          # affiche le type de la variable a
```

Exemples 1 :

Console

```
>>> type(45)
<class 'int'>
>>> type('Hello there')
<class 'str'>
>>> type(3.14159)
<class 'str'>
>>> type(['world',24,15.78,[57]])
<class 'list'>
>>> type(True)
<class 'bool'>
```

addition	soustraction	multiplication	division	puissance
a + b	a − b	a ∗ b	a/b	a ∗ ∗ b

Table : Opérations arithmétiques de base

addition	soustraction	multiplication	division	puissance
a + b	a − b	a ∗ b	a/b	a ∗ ∗ b

Table : Opérations arithmétiques de base

vrai	faux	et	ou	non	=	<	>	⩽	⩾	≠
true	false	and	or	not	==	<	>	<=	>=	! =

Table : Opérateurs logiques

Définition 2 :

- **Instructions d'affichage :**
  L'instruction `print('Mon message')` écrit le message `Mon message` dans la console.
  L'instruction `print(x)` affiche le contenu de la variable `x`.

Définition 2 :

- **Instructions d'affichage :**
  L'instruction `print('Mon message')` écrit le message `Mon message` dans la console.
  L'instruction `print(x)` affiche le contenu de la variable `x`.
- **Instructions de saisie :**
  L'instruction `x=input('Mon message')` écrit le message `Mon message` dans la console puis donne la main à l'utilisateur pour qu'il rentre la valeur à affecter à la variable `x`.

Remarques :

- L'instruction `print(x,y,z)` provoque l'affichage des contenus des variables `x`, `y`, `z` dans cet ordre sur une même ligne. Il est aussi possible de combiner affichage de messages et affichage de contenus de variables.

Remarques :

- L'instruction `print(x,y,z)` provoque l'affichage des contenus des variables `x`, `y`, `z` dans cet ordre sur une même ligne. Il est aussi possible de combiner affichage de messages et affichage de contenus de variables.

Exemple 2 :

1. Taper dans la console les instructions suivantes :

Console

```
>>> print('Hello there')
```

Remarques :

- L'instruction `print(x,y,z)` provoque l'affichage des contenus des variables `x`, `y`, `z` dans cet ordre sur une même ligne. Il est aussi possible de combiner affichage de messages et affichage de contenus de variables.

Exemple 2 :

1. Taper dans la console les instructions suivantes :

Console

```
>>> print('Hello there')
```

2. Taper dans l'éditeur de texte les instructions suivantes puis exécuter :

Éditeur

```
1 n = int(input('Entrer n :'))
2 print('Le nombre qui suit ',n,' est ',n+1)
```

- Attention : en faisant `x=input ('Mon message')`, la valeur entrée par l'utilisateur et affectée à la variable `x` est systématiquement de type chaîne de caractères `str`. Si on veut un entier ou un nombre flottant, il faudra utiliser les commandes `int` et `float`.

Exercice 2 :

1. 1. Exécuter l'instruction ci-dessous :

Console

```
>>> a=input('Entrer un nombre: ')
>>> type(a)
```

Exercice 2 :

1. 1. Exécuter l'instruction ci-dessous :

Console

```
>>> a=input('Entrer un nombre: ')
>>> type(a)
```

2. Que remarquez-vous ?

Exercice 2 :

1. 1. Exécuter l'instruction ci-dessous :

Console

```
>>> a=input('Entrer un nombre: ')
>>> type(a)
```

   2. Que remarquez-vous ?

2. 1. Exécuter les instructions ci-dessous :

Console

```
>>> a=int(input('Entrer un entier: '))
>>> type(a)
>>> a=float(input('Entrer un nombre: '))
>>> type(a)
>>> a=eval(input('Entrer un contenu de type quelconque: '))
>>> type(a)
```

Exercice 2 :

1. 1. Exécuter l'instruction ci-dessous :

Console

```
>>> a=input('Entrer un nombre: ')
>>> type(a)
```

   2. Que remarquez-vous ?

2. 1. Exécuter les instructions ci-dessous :

Console

```
>>> a=int(input('Entrer un entier: '))
>>> type(a)
>>> a=float(input('Entrer un nombre: '))
>>> type(a)
>>> a=eval(input('Entrer un contenu de type quelconque: '))
>>> type(a)
```

   2. Commenter chaque résultat. (Pour la dernière instruction, on fera varier le type du contenu).

Définition 3 :

Pour effectuer un bloc d'instructions sous certaines conditions, on utilise une boucle `if`.

Définition 3 :

Pour effectuer un bloc d'instructions sous certaines conditions, on utilise une boucle `if`.

Voici la syntaxe :

Éditeur

```
1  if condition_1 :
2      instructions_1
3  elif condition_2
4      instructions_2
5  ...
6  elif condition_{k-1}
7      instructions_{k-1}
8  else :
9      instructions_k
```

Si la condition_1 est vraie, le premier bloc d'instructions (instructions_1) est effectué.

Définition 3 :

Pour effectuer un bloc d'instructions sous certaines conditions, on utilise une boucle `if`.

Voici la syntaxe :

Éditeur

```
1  if condition_1 :
2      instructions_1
3  elif condition_2
4      instructions_2
5  ...
6  elif condition_{k-1}
7      instructions_{k-1}
8  else :
9      instructions_k
```

Si la $condition_1$ est vraie, le premier bloc d'instructions ($instructions_1$) est effectué.

Si la $condition_1$ est fausse, on fait un deuxième test.

Définition 3 :

Pour effectuer un bloc d'instructions sous certaines conditions, on utilise une boucle `if`.

Voici la syntaxe :

Éditeur

```
1  if condition_1 :
2      instructions_1
3  elif condition_2
4      instructions_2
5  ...
6  elif condition_{k-1}
7      instructions_{k-1}
8  else :
9      instructions_k
```

Si la condition_1 est vraie, le premier bloc d'instructions (instructions_1) est effectué.

Si la $\textit{condition}_1$ est fausse, on fait un deuxième test.

Si la condition_2 est vraie, le deuxième bloc d'instructions (instruction_2) est effectué.

Définition 3 :

Pour effectuer un bloc d'instructions sous certaines conditions, on utilise une boucle `if`.

Voici la syntaxe :

Éditeur

```
1  if condition_1 :
2      instructions_1
3  elif condition_2
4      instructions_2
5  ...
6  elif condition_{k-1}
7      instructions_{k-1}
8  else :
9      instructions_k
```

Si la condition_1 est vraie, le premier bloc d'instructions (instructions_1) est effectué.

Si la $\textit{condition}_1$ est fausse, on fait un deuxième test.

Si la condition_2 est vraie, le deuxième bloc d'instructions (instruction_2) est effectué.

Ce procédé est répété jusqu'à la condition_{k-1}.

Si la condition_{k-1} est vraie, le bloc d'instructions ($\text{instructions}_{k-1}$) est effectué.

Remarques :

- Il faut penser aux deux points après chaque condition et après le `else`.

Remarques :

- Il faut penser aux deux points après chaque condition et après le `else`.
- Bien respecter l'indentation : c'est elle qui permet à `Python` de reconnaître le bloc d'instructions à exécuter et où se termine la structure conditionnelle. Une fois les instructions écrites, si l'on doit poursuivre le script, on revient au même niveau d'indentation que le `if`.

Remarques :

- Il faut penser aux deux points après chaque condition et après le `else`.
- Bien respecter l'indentation : c'est elle qui permet à `Python` de reconnaître le bloc d'instructions à exécuter et où se termine la structure conditionnelle. Une fois les instructions écrites, si l'on doit poursuivre le script, on revient au même niveau d'indentation que le `if`.

Exemple 3 :

Taper dans l'éditeur de texte le script suivant puis l'exécuter. Que calcule-t-il ?

Éditeur

```
1  a = float(input('Entrer un réel a : '))
2  if a >= 0 :
3      print(a)
4  else :
5      print(-a)
```

Exercice 3 :

1. Décrire le déroulement du programme ci-dessous.

Éditeur

```
1  k=int(input('Quel est votre age ?'))
2  if k<=12 :
3          print('vous avez droit au menu enfant !')
4  elif k>12 and k<= 18 :
5      print('vous avez droit au menu spécial jeune !')
6  else :
7      print("vous n'avez droit à rien !")
```

Exercice 3 :

1. Décrire le déroulement du programme ci-dessous.

Éditeur

```
1  k=int(input('Quel est votre age ?'))
2  if k<=12 :
3          print('vous avez droit au menu enfant !')
4  elif k>12 and k<= 18 :
5      print('vous avez droit au menu spécial jeune !')
6  else :
7      print("vous n'avez droit à rien !")
```

2. Que se passe-t-il si l'utilisateur entre la valeur 10 ? la valeur 18 ? la valeur 25 ?

Exercice 4 :

Compléter le programme ci-dessous qui demande trois réels a, b et c et qui affiche le nombre de racines réelles du polynôme du second degré $P : x \longmapsto ax^2 + bx + c$. Selon les cas, le programme devra afficher le message « P n'admet pas de racine », « P admet une racine » ou « P admet deux racines ».

Éditeur

```
1   a = float(input('Donner une valeur non nulle pour a : '))
2   b = ... ... 'Donner une valeur pour b : '
3   c = ... ... 'Donner une valeur pour c : '
4   d = b^2-4*a*c
5   if ... :
6       print('P admet deux racines racines réelles.')
7   elif ... :
8       print('P admet une racine réelle.')
9   else :
10      print('P admet zero racine réelle.')
```

Définition 4 :

- La commande

Éditeur
```
1  def nom(x) :
2      instructions
3      return ...
```

permet de créer une fonction, nommée ici `nom`, qui à chaque variable `x` associe la quantité définie après le **`return`**.

Définition 4 :

- La commande

Éditeur
```
1  def nom(x) :
2      instructions
3      return ...
```

  permet de créer une fonction, nommée ici `nom`, qui à chaque variable `x` associe la quantité définie après le **`return`**.
- Une fonction peut posséder plusieurs variables. L'en-tête est alors `def nom(x1,x2,...,xn) :`

Exemple 4 :

Pour définir la fonction $f : x \in \mathbb{R} \longmapsto x^2 + 2x + 1$, recopier le script suivant dans l'éditeur de texte :

Éditeur

```
1  def f(x) :
2      return x**2 + 2*x + 1
```

Après avoir exécuté, on peut alors utiliser cette fonction dans la console. Taper par exemple :

```
>>> f(1), f(0), f(-1)
```

Exemple 4 :

Pour définir la fonction $f : x \in \mathbb{R} \longmapsto x^2 + 2x + 1$, recopier le script suivant dans l'éditeur de texte :

Éditeur

```
1  def f(x) :
2      return x**2 + 2*x + 1
```

Après avoir exécuté, on peut alors utiliser cette fonction dans la console. Taper par exemple :
`>>> f(1), f(0), f(-1)`

Console

```
>>> f(1), f(0), f(-1)
(4, 1, 0)
```

Remarques :

- Toujours la même remarque : attention aux deux points et à l'indentation !

Remarques :

- Toujours la même remarque : attention aux deux points et à l'indentation !
- Les fonctions seront systématiquement à définir dans l'éditeur de texte.

Remarques :

- Toujours la même remarque : attention aux deux points et à l'indentation !
- Les fonctions seront systématiquement à définir dans l'éditeur de texte.
- Les variables utilisées dans le corps d'une fonction sont des variables locales : elles n'existent pas à l'extérieur de la fonction.

Remarques :

- Toujours la même remarque : attention aux deux points et à l'indentation !
- Les fonctions seront systématiquement à définir dans l'éditeur de texte.
- Les variables utilisées dans le corps d'une fonction sont des variables locales : elles n'existent pas à l'extérieur de la fonction.
- Lorsqu'on définit une fonction, inutile d'ajouter les fonctions d'entrée et de sortie `input` et `print` à celle-ci. L'utilisateur est censé connaitre l'argument à rentrer pour la fonction. `Python` renverra le contenu des variables placées après le return.

Remarques :

- Toujours la même remarque : attention aux deux points et à l'indentation !
- Les fonctions seront systématiquement à définir dans l'éditeur de texte.
- Les variables utilisées dans le corps d'une fonction sont des variables locales : elles n'existent pas à l'extérieur de la fonction.
- Lorsqu'on définit une fonction, inutile d'ajouter les fonctions d'entrée et de sortie `input` et `print` à celle-ci. L'utilisateur est censé connaitre l'argument à rentrer pour la fonction. `Python` renverra le contenu des variables placées après le return.
- L'instruction `return` arrête le déroulement de la fonction. Si cette instruction est rencontrée, le programme s'arrête donc (en affichant le contenu de la variable à retourner) et le code situé après le `return` ne s'exécutera pas.

Exemple 5 :

Recopier la fonction suivante dans l'éditeur de texte :

Éditeur

```
1  def minimum(x,y) :
2      if x<=y :
3          return x
4      else :
5          return y
```

Enregistrer et exécuter cette fonction puis tester la sur quelques exemples.

Exercice 5 – La valeur absolue :

Définir une fonction `valabs`, qui prend en entrée un flottant et qui renvoie sa valeur absolue.

Éditeur

```
1  def valabs(x) :
2      if ... :
3          return ...
4      else :
5          return ...
```

Exercice 6 – Fonctions et `fonction` :

1. Compléter la fonction f suivante sur `Python` qui définit la fonction :

$$f : x \longmapsto \begin{cases} \dfrac{1}{x} & \text{si } x \neq 0, \\ 0 & \text{sinon.} \end{cases}$$

Éditeur

```
1  def f(x) :
2      if ... :
3          return ...
4      else :
5          return ...
```

Exercice 6 – Fonctions et `fonction` :

1. Compléter la fonction f suivante sur `Python` qui définit la fonction :

$$f : x \longmapsto \begin{cases} \dfrac{1}{x} & \text{si } x \neq 0, \\ 0 & \text{sinon.} \end{cases}$$

Éditeur

```
1  def f(x) :
2      if ... :
3          return ...
4      else :
5          return ...
```

2. On donne le script suivant :

Éditeur

```
1  def g(x) :
2      if x<-1 :
3          y=-2
4      if x>=-1 and x<=1 :
5          y=2 * x
6      if x>1 :
7          y=2
8      return y
```

Exercice 6 – Fonctions et `fonction` :

1. Compléter la fonction f suivante sur `Python` qui définit la fonction :

$$f : x \longmapsto \begin{cases} \dfrac{1}{x} & \text{si } x \neq 0, \\ 0 & \text{sinon.} \end{cases}$$

Éditeur
```
1  def f(x) :
2      if ... :
3          return ...
4      else :
5          return ...
```

2. On donne le script suivant :

Éditeur
```
1  def g(x) :
2      if x<-1 :
3          y=-2
4      if x>=-1 and x<=1 :
5          y=2 * x
6      if x>1 :
7          y=2
8      return y
```

1. Déterminer l'expression mathématique de la fonction g ainsi définie.

Exercice 6 – Fonctions et `fonction` :

1. Compléter la fonction f suivante sur `Python` qui définit la fonction :

$$f : x \longmapsto \begin{cases} \dfrac{1}{x} & \text{si } x \neq 0, \\ 0 & \text{sinon.} \end{cases}$$

Éditeur

```
1  def f(x) :
2      if ... :
3          return ...
4      else :
5          return ...
```

2. On donne le script suivant :

Éditeur

```
1  def g(x) :
2      if x<-1 :
3          y=-2
4      if x>=-1 and x<=1 :
5          y=2 * x
6      if x>1 :
7          y=2
8      return y
```

1. Déterminer l'expression mathématique de la fonction g ainsi définie.
2. Proposer un script analogue au précédent qui utilise `elif`.