Structure conditionnelle if — TP2

Dans tout langage de programmation, il est très utile de pouvoir utiliser des *structures conditionnelles* : cela permet de faire une opération uniquement si une certaine condition est satisfaite. Il s'agit donc d'un test, qui en Python prend la forme d'un "**`if`**" : **si** une condition est satisfaite **alors** on fait quelque chose.
La syntaxe d'un test est la suivante :

```
1 if ma_condition :
2     # des instructions à suivre si ma_condition est satisfaite
3 # des instructions à suivre en dehors du test
```

Si la condition `ma_condition` est vérifiée, Python "rentre dans le `if`" et exécute les instructions de la ligne 2, puis enchaîne sur les instructions de la ligne 3. Si elle ne l'est pas, Python, ne fait rien et passe directement aux instructions en dehors du test, ligne 3.
Faisons quelques remarques :

- La condition `ma_condition` est une variable booléenne.
- Ne pas oublier les **deux points** à la fin de la condition.
- **L'indentation** est capitale : c'est elle qui délimite les instructions dépendant du `if` de celles en dehors du test.

Exercice 1 — Un premier essai

Q1 Dans le script, définissez une variable `x` valant une valeur réelle positive de votre choix. Essayez alors le premier exemple suivant :

```
1 if x>0:
2     x = x + 2
```

Combien vaut `x` après exécution du code ? Réessayez avec une valeur initiale de `x` négative.

Q2 Dans le script, définissez une variable `y` valant une valeur réelle de votre choix. Essayez alors le deuxième exemple suivant :

```
1 if y<2:
2     y = y + 2
3 y = y - 2
```

Combien vaut `y` après exécution du code ? Réessayez avec une autre valeur initiale de `y`. Précisez quand les instructions des lignes 2 et 3 sont prises en compte par Python.

Q3 Dans le script, définissez la fonction suivante :

```
1 def fun(z):
2     res = -1
3     if z>0:
4         res = 1
5     return res
```

Que renvoie `fun(3)` ? `fun(-2)` ?

Exercice 2 **Guess and check**

Dans chacun des exemples suivants, prédisez la valeur de la variable `a` après exécution du code, puis vérifiez votre prédiction sur l'ordinateur. Attention à l'indentation...

Q1

```
a = 1
if a > 1 :
    a = a + 2
a = a + 3
```

```
a = 1
if a > 1 :
    a = a + 2
    a = a + 3
```

Q2

```
a = -1
if a > 1 :
    a = a - 1
if a < 1 :
    a = a + 2
```

```
a = 1.5
if a > 1 :
    a = a - 1
if a < 1 :
    a = 0
```

Q3 On peut aussi imbriquer des tests `if` ! Voici une petite variation de la question 2, la consigne reste la même : guess and check !

```
a = -1
if a > 1 :
    a = a - 1
    if a < 1 :
        a = a + 2
```

Si/sinon

Bien souvent, on veut traiter une alternative de deux cas : "si `ma_condition` est vérifiée alors faire ceci, et sinon faire cela".
Cela peut se faire avec deux tests `if` à la suite : l'un testant la condtion `ma_condition` et l'autre testant **`not`**`(ma_condition)`. Complétez par exemple la fonction suivante pour qu'elle renvoie x^2 si $x \geq 1$ et 0 sinon :

```
def fun2(x):
    if x >= 1:
        return x**2
    if      :
        return 0
```

Pour alléger le code[1] dans ce cas de figure, il existe la commande `else` ("sinon") qui suit la syntaxe suivante :

```
if ma_condition :
    # des instructions si ma_condition est satisfaite
else :
    # des instructions si ma_condition n'est pas satisfaite
# des instructions en dehors du test
```

1. et surtout pour ne pas demander à Python d'effectuer un calcul dont on connaît déjà le résultat : si on vient d'évaluer la condition `ma_condition`, il est inutile d'évaluer juste ensuite la condition **`not`**`(ma_condition)` !

Si la condition est satisfaite, Python exécute les instructions du bloc `if` puis ignore celles du bloc `else`. Inversement si la condition n'est pas satisfaite, Python ignore les instructions du bloc `if` et exécute celle du bloc `else`. Faisons quelques remarques :

- La condition `ma_condition` ne s'écrit **qu'une seule fois** : après le `if` et pas après le `else`.
- Le bloc `else` est au même niveau d'indentation que le bloc `if`.
- Les deux points et l'indentation sont toujours indispensables.

Par exemple, la fonction précédente `fun2` peut aussi s'écrire comme ceci :

```
1 def fun2(x):
2     if x>=1 :
3         return x**2
4     else :
5         return 0
```

Exercice 3 Absolument !

Q1 Écrire une fonction `valabs` prenant en argument un réel `x` et renvoyant $|x|$.

Q2 Écrire une fonction `un_ou_deux` prenant en argument un réel `x` et renvoyant la chaîne de caractères `"oui"` si `x` vaut 1 ou 2, et la chaîne de caractères `"non"` sinon.

Exercice 4 Missa brevis et spiritus maxima [2]

Q1 Écrire une fonction `maxi` prenant en arguments deux réels a et b et renvoyant le plus grand des deux. Testez votre fonction sur plusieurs exemples.

Q2 Écrire une fonction `maxi3` prenant en arguments trois réels a, b et c et renvoyant le plus grand des trois. Dans un premier temps, on demande d'écrire une fonction utilisant plusieurs tests `if`. Testez votre fonction sur plusieurs exemples, en particulier sur des cas où deux des arguments sont égaux.

Q3 Écrire une deuxième version de la fonction `maxi3` mais n'utilisant pas de `if`. On réutilisera judicieusement la fonction `maxi` de la première question (qui, elle, utilise un `if`). Testez votre fonction sur plusieurs exemples.

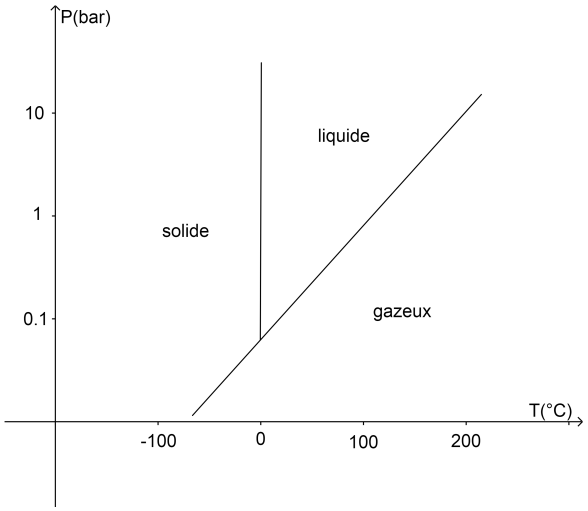
Exercice 5 Diagramme de l'eau simplifié

> **Remarque**
>
> Pour la deuxième question de cet exercice, vous utiliserez la fonction logarithme. Celle-ci n'est pas directement disponible dans Python : il faut la charger en "important un paquet". Par exemple le paquet `numpy` qui contient plusieurs fonctions mathématiques. Voici comment procéder :
>
> ```
> 1 import numpy as np # À positionner au début du programme
> 2 x = np.log(7) # cette commande affecte à x la valeur ln(7)
> ```
>
> Signalons que Python utilise la notation anglo-saxonne "log" pour désigner la fonction logarithme népérien "ln".

Q1 Écrire une fonction `etat1` prenant en argument un réel `T` correspondant à la température en degrés Celsius d'un volume d'eau dans des conditions habituelles de pression, et renvoyant son état (solide, liquide ou gazeux).

2. (ça veut rien dire mais...)

De manière plus générale, l'eau pure se présente sous 3 phases (solide, liquide ou gazeuse) en fonction des conditions de température et de pression. La phase (ou le mélange de plusieurs phases) correspondant à une température `T` et une pression `P` est donnée par le diagramme de phase de l'eau pure dont une version simplifiée est donnée ci-contre.



Sur ce diagramme, où l'axe de la pression est en échelle logarithmique, la courbe de fusion (passage de l'état solide à l'état liquide) est modélisée par la droite d'équation `T` $= 0$, et les courbes de vaporisation et de sublimation (passage de l'état liquide ou solide à l'état gazeux) sont modélisées par la droite d'équation $\log_{10}(\texttt{P}) = \frac{1}{50}\texttt{T} - 2$.

On rappelle que pour $x > 0$ on définit : $\log_{10}(x) = \dfrac{\ln(x)}{\ln(10)}$.

Q2 Écrire une fonction `etat2` prenant en argument deux réels `T` et `P` correspondant à la température en degrés Celsius et à la pression en bar d'un volume d'eau et renvoyant son état. Vérifiez en particulier que `etat2(T,1) = etat1(T)`. Comment cette relation se lit-elle sur le diagramme de l'eau ?

Exercice 6 Résolution d'une équation polynomiale du second degré

Q1 Écrire une fonction `racines` prenant en argument trois réels `a`, `b` et `c` tels que `a` $\neq 0$, et qui renvoie les solutions réelles de l'équation $\texttt{a}x^2 + \texttt{b}x + \texttt{c} = 0$. Pour indiquer qu'il n'y a pas de solution réelle, la fonction renverra la chaîne de caractères `"pas de solution réelle"`.

Q2 Définissez une fonction `racines2` prenant aussi en compte le cas `a` $= 0$ en renvoyant alors les solutions réelles de l'équation $\texttt{b}x + c = 0$. Votre fonction `racines2` devra utiliser la fonction `racines`.

Q3 Testez votre fonction en appelant `racines(a,b,c)` pour `(a,b,c)` prenant les valeurs suivantes : $(1, -4, 3); (1, 4, 4); (1, 0, 1); (0, 2, 1); (0, 0, 1); (0, 0, 0)$.