

# INF101/INF131 : devoir surveillé

Jeudi 26 Octobre 2017

- **Durée 2 heures.**
- **Lisez toutes les questions avant de commencer !** Le sujet fait quatre (4) pages.
- Respectez strictement les consignes de l'énoncé (noms de variables, format d'affichage...)
- Pensez à commenter vos programmes pour les rendre plus lisibles.
- Les exercices sont indépendants, et en ordre croissant de difficulté.
- Aucun document autorisé. L'utilisation d'une calculatrice, d'un téléphone, etc est interdite.
- Le barème est indicatif. La qualité de la rédaction et de la présentation sera prise en compte.

## 1 EXERCICE 1 : BOOLEENS (2.5 points)

On suppose que toutes les variables nécessaires sont bien initialisées. Écrire les expressions booléennes vraies si:

- a est un entier strictement positif et multiple de 3
- a est soit positif pair, soit négatif impair
- c est une lettre de l'alphabet en minuscule ou en majuscule (**remarque** : on interdit d'utiliser la fonction `isalpha` de Python; attention c pourrait être une chaîne de plusieurs caractères)
- x, y, z sont dans l'ordre croissant (non strict) et tous strictement négatifs
- x est un nombre réel non entier

## 2 EXERCICE 2 : CONVERSIONS (2.5 points)

On veut écrire un programme pour aider un coureur à calculer son allure de course. Chaque coureur connaît sa vitesse maximale en km/h (par exemple 21.2 km/h). Son entraîneur lui demande de courir à un certain pourcentage de cette vitesse maximale (par exemple 75%). Sa montre n'affiche pas la vitesse mais l'allure en mn/km (par exemple 5 mn/km correspond à 12 km/h). Écrire un programme qui réalise les opérations suivantes :

1. Demander au coureur de saisir sa vitesse maximale en km/h, qui doit être strictement positive, et inférieure ou égale à 30. Le programme filtre la valeur saisie jusqu'à ce qu'elle respecte cette contrainte (variable `vmax`).
2. Demande au coureur de saisir le pourcentage de cette vitesse auquel il doit courir (entier entre 0 et 100), et filtre cette valeur jusqu'à ce qu'elle soit bien comprise entre 0 et 100 (variable `pc`).
3. Calcule la vitesse (variable `v`) correspondant à ce pourcentage et l'affiche.
4. Calcule l'allure (variable `a`) correspondant à la vitesse `v` et l'affiche. (Attention on veut l'allure en minutes et secondes par kilomètres. Le nombre de minutes doit être un entier, le nombre de secondes un réel qu'on pourra arrondir à 2 chiffres après la virgule avec la fonction `round`).

On demande de respecter **strictement** l'affichage de l'exemple suivant (à l'espace près) :

```
Quelle est ta vitesse maximale en km/h ? 15.4
Quel pourcentage (0-100) ? 180
Quel pourcentage (0-100) ? 80
La vitesse correspondant à 80 pourcents de ton max est : 12.32 km/h
L'allure correspondant à 12.32 km/h est : 4'52''/km
```

**Rappel** : on peut spécifier le séparateur voulu dans l'affichage en précisant `sep=...`

### 3 EXERCICE 3 : PYRAMIDE (4 points)

Écrire un programme qui :

1. demande à l'utilisateur un nombre de lignes (variable `nbl`) et une lettre de départ (variable `dep`). On suppose que l'utilisateur saisit toujours une minuscule, on n'a pas besoin de le vérifier.
2. affiche une pyramide de lettres à `nbl` lignes, sous la forme suivante. Si on dépasse la fin de l'alphabet on veut recommencer à la lettre 'a' après le 'z'.

Par exemple: à gauche pour 4 lignes (`nbl=4`) et départ à 'a' (`dep='a'`); à droite pour 3 lignes (`nbl=3`) et départ à 'u' (`dep='u'`).

```
      a
     b c d
    e f g h i
   j k l m n o p
```

```
      u
     v w x
    y z a b c
```

Rappels :

- la fonction `ord(caractere)` renvoie le code ASCII du caractère reçu en paramètre. La fonction `chr(code)` renvoie le caractère ayant le code ASCII reçu en paramètre.
- `print` a 2 paramètres optionnels, `sep` qui permet de modifier le séparateur entre les chaînes affichées (espace par défaut), et `end` qui permet de modifier la fin de l'affichage (retour à la ligne par défaut).

### 4 Exercice 4 : mélange de liste (3 points)

1. Écrire une fonction `melange_liste` qui reçoit en paramètre une liste, et renvoie une liste contenant les mêmes éléments dans le désordre.  
**Attention** : on ne veut PAS modifier la liste reçue en paramètre mais renvoyer une nouvelle liste.  
**Indice** : on peut faire une copie de la liste reçue, dans une nouvelle variable qui pourra elle être modifiée.
2. Écrire ensuite un programme principal qui :
  - (a) crée une liste vide
  - (b) demande à l'utilisateur de saisir des entiers, et les ajoute à cette liste. L'utilisateur doit taper -1 pour terminer la saisie (-1 n'est pas ajouté à la liste)
  - (c) affiche la liste ainsi constituée de tous les éléments saisis (dans l'ordre de saisie)
  - (d) mélange cette liste en appelant la fonction écrite ci-dessus
  - (e) affiche la liste mélangée

**Indice** : on utilisera la fonction `randint` du module `random`. **Remarque** : le module `random` contient aussi déjà une fonction `shuffle` qui permet de mélanger une liste, mais on demande de ne **PAS** l'utiliser ici, le but de l'exercice est de la réécrire. (Par ailleurs cette fonction modifie la liste manipulée, alors qu'on demande ici de renvoyer une copie mélangée de la liste de départ sans modifier celle-ci.)

**Rappel de quelques fonctions utiles sur les listes** : `len` (longueur d'une liste) ; `append` (ajouter un élément à la fin d'une liste) ; `pop` (récupérer l'élément à un certain indice et le supprimer de la liste).

## 5 Exercice 5 : message secret (8 points)

### 5.1 Fonctions

Définir les fonctions suivantes permettant de coder un mot de différentes manières selon le jour de la semaine :

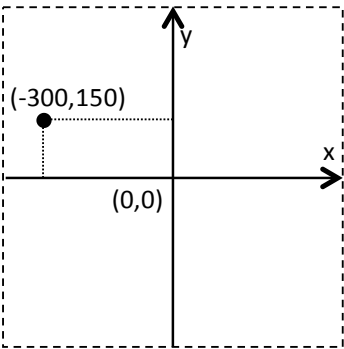
1. La fonction `lundi` reçoit un mot, choisit un entier `n` au hasard entre 2 et 7, et renvoie le mot répété `n` fois, séparés par des `+`. Par exemple si le mot est "salut" et que l'entier tiré est 3, le résultat est "salut+salut+salut". On remarquera qu'il n'y a **pas** de `+` à la fin de la chaîne.
2. La fonction `mardi` reçoit un mot, et renvoie ce mot écrit à l'envers. Par exemple si le mot est "bonjour", le résultat est "ruojnob".  
**Rappel** : si `c` est une chaîne de caractères alors `list(c)` permet d'obtenir la liste de toutes ses lettres.
3. La fonction `mercredi` reçoit un mot, choisit un entier `x` au hasard entre 1 et 10, et opère un décalage de `x` positions sur chaque lettre du mot (en rebouclant sur 'a' après le 'z' ou sur 'A' après le 'Z'). Par exemple le mot "Salut" avec un décalage de 7 sera codé en "Zhsba" car 'Z' est situé 7 lettres après 'S' dans l'alphabet en majuscules, 'h' est situé 7 lettres après 'a' dans l'alphabet en minuscules, etc. Pour 'u' et 't' on voit qu'on dépasse le 'z' et on revient donc au début de l'alphabet. On pourra écrire une fonction auxiliaire qui opère le décalage sur une lettre (en considérant bien les majuscules et les minuscules).
4. La fonction `jeudi` reçoit un mot, supprime toutes ses voyelles (majuscules comme minuscules), et renvoie le mot constitué des lettres (consonnes) restantes. Par exemple "salut" devient "slt", "Alice" devient "lc". On pourra écrire une fonction auxiliaire qui teste si une lettre est une voyelle.
5. La fonction `vendredi` reçoit un mot, change les minuscules en majuscules, et vice-versa. Par exemple "Salut" devient "sALUT". On pourra écrire une fonction auxiliaire qui transforme une minuscule en majuscule et vice versa.  
**Remarque** : on **interdit** ici d'utiliser les fonctions `upper`, `lower`, `isupper`, `islower`, déjà définies dans Python; on veut les reprogrammer.
6. La fonction `samedi` reçoit un mot, mélange ses lettres, et renvoie le mot mélangé.  
**Remarque**: on pourra cette fois utiliser la fonction `shuffle` du module `random`. **Rappel** : la fonction `shuffle` reçoit une liste, la mélange, et ne renvoie rien; c'est la liste reçue qui est modifiée pour être mélangée.
7. La fonction `dimanche` reçoit un mot mais ne fait aucun codage et le renvoie tel quel.

### 5.2 Programme principal

8. Définir ensuite un programme principal qui :
  - (a) Opère en boucle en demandant un mot à l'utilisateur, jusqu'à lire le mot "stop".
  - (b) Pour chaque mot saisi par l'utilisateur (sauf "stop"), lui demande un jour de la semaine et affiche le résultat du codage de ce mot selon la fonction correspondant au jour choisi.
  - (c) Recommence ensuite avec le mot suivant.
9. **Question bonus** : Écrire un nouveau programme principal qui, au lieu de lire un seul mot, lit un texte entier. Ensuite le programme demande un jour de la semaine, code chaque mot du texte selon la fonction correspondante, et affiche le texte résultant. De la même manière qu'à la question 8, ce programme s'arrête si le texte saisi est "stop".

Par exemple pour le texte "Bonjour a tous" codé avec la fonction du mardi, on affichera "ruojnoB a suot". (On notera que ce n'est pas le texte qui est à l'envers, mais bien chaque mot qui est à l'envers.)

**Rappel** : la fonction `split()` permet de découper une chaîne de caractères selon un séparateur (ici le séparateur est l'espace entre les mots), et renvoie la liste des mots ainsi obtenue. Par exemple `"Bonne chance a tous".split(" ")` renvoie la liste `["bonne", "chance", "a", "tous"]`.

<p align="center"><b>Opérations sur les types</b></p> <p><code>type()</code> : pour connaître le type d'une variable  <code>int()</code> : permet la transformation en entier  <code>float()</code> : permet la transformation en flottant  <code>str()</code> : permet transformation en chaîne de caractères</p>	<p align="center"><b>Définition d'une fonction</b></p> <pre>def nom_fonction(arg) :     instructions     return valeurs_ou_variables</pre>
<p align="center"><b>Instructions de lecture et écriture console</b></p> <pre>print(a1,a2,...,an, sep=xx, end=yy)</pre> <ul style="list-style-type: none"> <li>Pour imprimer une suite d'arguments de a1 à an</li> <li>sep : permet de définir le séparateur affiché entre chaque argument (par défaut " ")</li> <li>end : permet de définir le ce qui sera affiché à la fin (optionnel, par défaut : saut de ligne)</li> </ul> <pre>res = input(message)</pre> <ul style="list-style-type: none"> <li>Pour lire une suite de caractères au clavier terminée par &lt;Enter&gt;</li> <li>La chaîne de caractères résultante doit être affectée (ici à la variable res).</li> <li>l'argument est optionnel : c'est un message explicatif destiné à l'utilisateur</li> </ul>	<p align="center"><b>Instructions conditionnelles</b></p> <pre>if condition :     instructions  if condition :     instructions else :     instructions  if condition1 :     instructions elif condition2 :     instructions else :     instructions</pre>
<p align="center"><b>Opérateurs arithmétiques</b></p> <p>+ : addition,        - : soustraction  * : multiplication, ** : puissance,  / : division,       // : division entière (quotient),  % : reste de la division entière (modulo)</p>	<p align="center"><b>Turtle : instructions et fenêtre</b></p> <p>Curseur initialement en (0,0), dirigé vers la droite.</p> <pre>reset() : efface tout goto(x, y) : place le crayon au point x, y forward(d) : avance de d backward(d) : recule de d up() : relève le crayon down() : abaisse le crayon left(a) : tourne à gauche de a° right(a) : tourne à droite de a°</pre> 
<p align="center"><b>Opérateurs booléens</b></p> <p><b>and</b> : et logique   <b>or</b> : ou logique   <b>not</b> : négation</p>	
<p align="center"><b>Opérateur de comparaison</b></p> <p>== égalité            != différence  &lt; inférieur,           &lt;= inférieur ou égal  &gt; supérieur,           &gt;= supérieur ou égal</p>	
<p align="center"><b>Opérateurs sur les chaînes de caractères</b></p> <p><code>len(s)</code> : renvoie la longueur de la chaîne s  <code>s1+s2</code> : concatène les chaînes s1 et s2  <code>s*n</code> : construit la répétition de n fois la chaîne s  exemple : "ta"* 3 donne "tatata"</p>	
<p align="center"><b>Itération tant que</b></p> <pre>while condition :     instructions</pre>	
<p align="center"><b>Itération for, et range</b></p> <pre>for e in list_chaine_ou_dico :     instructions</pre> <p>Itère les instructions avec e prenant chaque valeur dans la liste, chaîne ou dictionnaire.</p>	<pre>for var in range (deb, fin, pas) :     instructions</pre> <p><b>range(a)</b> : séquence des valeurs [0, a[  <b>range(b,c)</b> : séquence des valeurs [b, c[ (pas de 1)  <b>range(b, c, g)</b> : idem avec un pas de g  <b>range(b,c,-1)</b> : valeurs de b(incl.) à c (excl.) , pas -1</p>

Listes	Dictionnaires
<p><b>list_exple = []</b> Création d'une liste vide</p> <p><b>list_exple = [e1, e2, e3]</b> Création d'une liste, ici à 3 éléments e1, e2, et e3</p> <p><b>list_exple[i]</b> obtenir l'élément à l'index <i>i</i> (<i>i</i> ≥ 0) Les éléments sont indexés à partir de 0 Si <i>i</i> &lt; 0, les éléments sont accédés à partir de la fin de la liste. Ex : <b>list_exple[-1]</b> permet d'accéder au dernier élément de la liste</p> <p><b>list_exple.append(element)</b> Ajout d'un seul élément à la fin de la liste</p> <p><b>list_exple.extend(liste2)</b> Ajout de tous les élem. de la liste <b>liste2</b> à la fin de la liste</p> <p><b>list_exple.insert(i, element)</b> Ajout d'un seul élément à l'index <i>i</i></p> <p><b>res = list_exple.pop(index)</b> retire l'élément présent à la position <i>index</i> et le renvoie, ici dans la variable <i>res</i>.</p> <p><b>list_exple.remove(element)</b> retire l'élément donné (le premier trouvé)</p> <p><b>len(list_exple)</b> Pour avoir le nombre d'éléments d'une liste</p> <p><b>l2_exple = list_exple</b> créé un synonyme (donne un 2<sup>ème</sup> nom à la liste)</p> <p><b>l3_exple = list(list_exemple)</b> Pour créer une copie (= un clone)</p> <p><b>elem in list_exple</b> Pour tester si un élément est dans une liste. Vaut <b>True</b> ou <b>False</b>.</p> <p><b>list_exple.shuffle()</b> (module random) Mélange la liste (effet de bord).</p>	<p><b>dico_exple = {}</b> Création d'un dictionnaire vide</p> <p><b>dico_exple = { c1:v1, c2:v2, c3:v3 }</b> Création d'un dictionnaire, ici à 3 entrées</p> <p><b>e = dico_exple[c1]</b> Les valeurs du dictionnaire sont accessibles par leurs clés. Ici, <b>e</b> prendra la valeur <b>v1</b>. Provoque une erreur si la clé n'existe pas.</p> <p><b>dico_exple[c3] = v3</b> Pour ajouter une nouvelle valeur au dictionnaire (<b>ici v3</b>) avec une clé (<b>ici c3</b>). Si la clé existe déjà, la valeur associée est modifiée.</p> <p><b>del dico_exple[C3]</b> Pour supprimer une association dans le dictionnaire. La clé doit exister.</p> <p><b>c in dico_exemple</b> Pour vérifier l'existence d'une clé dans le dictionnaire. Vaut <b>True</b> ou <b>False</b>.</p> <p><b>dic2 = dico_exple</b> créé un synonyme (donne un 2<sup>ème</sup> nom au dictionnaire)</p> <p><b>dic3 = dict(dico_exple)</b> Pour créer une copie (= un clone)</p>
<h3>Gestion des fichiers</h3>	
<p><b>f=open('data.txt')</b> Pour ouvrir un fichier en lecture seule</p> <p><b>f=open('data.txt', 'w')</b> Pour ouvrir un fichier en écriture</p> <p><b>texte = f.read()</b> pour lire tout le fichier en une seule fois</p> <p><b>lignes = f.readlines()</b> Lire en une fois toutes les lignes du fichier et les stocker dans une liste (un élément = une ligne)</p> <p><b>for ligne in f:</b> <b>instructions</b> Lire le fichier ligne par ligne dans une boucle for</p> <p><b>f.write(texte)</b> Pour écrire dans un fichier (<b>texte</b> doit oblig. être un <b>str</b>). Ne saute pas de ligne automatiquement à la fin du texte. <b>'\n'</b> code un saut de ligne.</p> <p><b>f.close()</b> Pour fermer un fichier</p>	
<h3>Chaîne de caractères (complément)</h3>	
<p><b>ch.split(arg)</b> où <b>ch</b> est une chaîne de carac. Retourne la liste des mots ici de <b>ch</b>, en coupant à chaque occurrence de <b>arg</b> (par défaut <b>arg = " "</b> (un espace) )</p>	