Informatique 2: Travaux Pratiques — Exercices de Base

Manipulation de dates et temps et expressions régulières

Le but de cette séance est de se familiariser avec la notion de date et temps, et de manipuler les expressions rationnelles en Python en Python. **Procéder par étape et tester votre code pour chaque étape.**

> **Attention**
>
> De manière générale, éviter de copier/coller du code ou du texte depuis un fichier pdf, cela peut causer des problèmes d'encodage.

Dates et temps

Question 1: (*10 minutes*) **Operations de bases**
Dans cet exercice, on veut tester le résultat de la différence pour des dates *naïves* d'une part et des dates *awares* d'autre part.

1. Définir une fonction prenant en paramètre un objet de type `datetime` et retournant une chaîne de caractères représentant cette date. La chaine de caractères en question doit être au format suivant: `'dd/mm/YY HH:MM'`.
2. Définir une fonction qui prend en paramètre un objet de type `timedelta` et qui retourne une chaîne de caractères représentant sa durée sous la forme suivante: 'D jours, MM minutes'.

   > **Attention**
   >
   > Penser à gérer le cas où le `timedelta` à une valeur négative.

3. Créer deux dates à partir des chaînes de caractères suivantes :
   `d1='27/03/2023 07h50', d2='27/03/2023 14h50'`.

   > **Attention**
   >
   > Le format est *case sensitive*, c'est-à-dire sensible à la casse (majuscules/minuscules).

4. Afficher ces deux dates et la durée qui les sépare.
5. Installer le module `pytz` avec **pip** (PyCharm → Preferences → Project → Project interpreter → +). À l'aide de ce module, créer deux `timezones` différentes: pour `'Europe/Paris'` et pour `'Asia/Macao'`
6. À l'aide de la méthode `localize` du module `pytz`, modifier les deux dates précédentes pour mettre d1 à l'heure de *Paris* et d2 à l'heure de *Macao*.
7. Comparer de nouveau ces deux dates. Que constatez-vous?

Question 2: (*20 minutes*) **Durée reparation**
On dispose d'une chaîne de caractères `s1` contenant la date d'achat d'un produit (dans un format donné décrit ci-dessous) et la durée de la garantie (exprimée en jours) séparées par une virgule et une espace d'une part et une chaîne de caractères `s2` contenant la date de départ en réparation (dans un format donné décrit ci-dessous) et la date du retour de la réparation (dans un format donné décrit ci-dessous), séparées par un point-virgule, d'autre part. On veut calculer et afficher (dans un format donné décrit ci-dessous) la nouvelle date d'expiration de la garantie en la prolongeant de la durée de la réparation. Les informations se trouvent dans le fichier *contract_dates.txt* disponible sur Moodle dans le Matériel pour les exercices.

> **Exemple**
>
> Les dates sont:
>
> ```
> Purchase: 05/01/23 10h45, 500 jours
> ```

```
Repair: 2023-08-10 1:05pm;2023-08-18 4:10pm
```

Pour ces dates on affichera :
`'Votre garantie a été prolongée, elle expirera le Jeudi 27 mai 2021 à 13h50'`

1. Lire le fichier et stocker la date d'achat dans `s1` et les informations à propos de la réparation dans `s2`.
2. Découper `s1` pour séparer la date/heure d'achat et la durée de la garantie.
3. Créer un objet de type `datetime` pour représenter la date/heure d'achat (utiliser `datetime.strptime`).
4. Créer un objet de type `timedelta` pour représenter la durée de la garantie.
5. Afficher le nom du jour de la semaine du jour de l'achat.
6. Découper `s2` pour séparer les dates/heures de début et de fin de réparation.
7. Créer des objets de type `datetime` pour manipuler ces dates/heures.

Attention

Pour pouvoir *parser* des heures au format AM/PM avec `strptime` il faut que la langue locale de l'ordinateur le supporte. Pour ce faire, il faut saisir : `locale.setlocale(locale.LC_TIME, 'en_US')` après avoir importé le module `locale`. Sur Windows, utiliser la notation correspondante (`'usa'` au lieu de `'en_US'`)[a].

[a] https://docs.microsoft.com/en-us/cpp/c-runtime-library/language-strings?view=vs-2017

8. Calculer la durée de la réparation (objet de type `timedelta`).
9. Calculer la (nouvelle) date/heure d'expiration de la garantie.
10. Afficher le message annonçant cette date/heure au format susmentionné si cette date est antérieure au 1er janvier 2024 à midi, date/heure (fictive) de changement de la réglementation sur les garanties.
11. Déterminer si la garantie est encore valide au moment de l'exécution du programme.

Expressions régulières

Question 3: (*30 minutes*) **Plaques d'immatriculation**

1. Ecrire une expression régulière Python pour reconnaître les chaînes de caractères composées d'exactement deux lettres majuscules.

Exemple

`"A"`	✗
`"AB "`	✓
`"ABC"`	✗
`"ab"`	✗

2. Modifier l'expression régulière Python précédente pour reconnaître les chaînes de caractères composées d'exactement deux lettres majuscules ou deux lettres minuscules (mais *pas de mélange de majuscules et de minuscules*).

Exemple

```
"Aa"  ✗
"AA"  ✓
"bb"  ✓
```

3. Ecrire une expression régulière Python pour reconnaître les chaînes de caractères composées de 1 à 6 chiffres.

Exemple

```
""          ✗
"7"         ✓
"076726"    ✓
"17354576"  ✗
```

4. Modifier l'expression régulière Python précédente pour reconnaître les chaînes de caractères composées de 1 à 6 chiffres *ne commençant pas par un zéro*.

Exemple

```
"0"         ✗
"076726"    ✗
"1"         ✓
"36"        ✓
"5787368"   ✗
```

5. Ecrire une expression régulière Python pour reconnaître les plaques d'immatriculation suisses définies comme suit :
   (a) Un code à deux lettres (tout en minuscule ou tout en majuscule) pour identifier le canton (code canton)
   (b) Possiblement des espaces
   (c) Un code numérique comprenant entre 1 et 6 chiffres ne commençant pas par un zéro (numéro d'identification)

Attention

Ne pas oublier d'importer le module `re`.

Utiliser les raccourcis pour les symboles numériques, alphabétiques et les caractères d'espacement.

6. Implémenter cette expression en Python et tester sur quelques chaînes de caractères (valides et invalides). Tester dans le code Python et directement avec PyCharm. Le programme affichera un message d'erreur si la chaîne de caractères saisie par l'utilisateur n'est pas reconnue comme valide. Tester avec `['AB123456', 'VD011111', 'VD 1234567', 'VD123456', 'vd123456', 'Vd123456', 'VD 123456', 'VD  123456', 'VD \t 123456']`

7. On suppose que le nombre maximum de chiffres dans le numéro d'identification n'est pas connu au moment où on écrit l'expression régulière mais qu'il sera connu au moment de l'exécution et stocké dans une variable `n`. Modifier le programme pour prendre ce cas en compte en générant à l'exécution la chaîne de caractères décrivant l'expression régulière.

8. On suppose qu'on dispose d'un dictionnaire contenant les codes des cantons avec leur noms complets. Modifier le programme de sorte à utiliser ce dictionnaire pour générer la chaîne de caractères décrivant l'expression régulière. Le dictionnaire se trouve dans le fichier *cantons.py* disponible sur Moodle dans le Matériel pour les exercices

```
d = {'AG': 'Argovie', 'AI': 'Appenzell Rhodes-Intérieures',
'AR': 'Appenzell Rhodes-Extérieures', 'BE': 'Berne',
'BL': 'Bâle-Campagne', 'BS': 'Bâle-Ville', 'FR': 'Fribourg',
'GE': 'Genève', 'GL': 'Glaris', 'GR': 'Grisons', 'JU': 'Jura',
'LU': 'Lucerne', 'NE': 'Neuchâtel', 'NW': 'Nidwald',
'OW': 'Obwald', 'SG': 'Saint-Gall', 'SH': 'Schaffhouse',
'SO': 'Soleure', 'SZ': 'Schwytz', 'TG': 'Thurgovie',
'TI': 'Tessin', 'UR': 'Uri', 'VD': 'Vaud', 'VS': 'Valais',
'ZG': 'Zoug', 'ZH': 'Zurich'}
```

9. Extraire les informations importantes de la chaîne de caractères représentant une plaque d'immatriculation, à savoir le code du canton et le numéro d'identification, en utilisant la méthode `groups()`, et les stocker dans deux variables.

10. Modifier le programme pour effectuer l'extraction à l'aide de groupes nommés `canton` et `identification`.

11. Afficher la plaque d'immatriculation ainsi traitée sous forme normalisée, c'est à dire avec un code de canton en majuscule et le numéro d'identification séparés par un unique espace.

12. Pour augmenter la visibilité des plaques d'immatriculation, il a été décidé que le code du canton devrait être répété après le numéro d'identification. Par exemple `"VD 12345 VD"`. Modifier le programme pour gérer ce nouveau format (le programme devra continuer de gérer les anciennes plaques).

13. Que se passe-t-il quand le programme traite la plaque `"vd 12345 VD"` ?

14. Modifier le programme pour utiliser des expressions régulières *compilées*.