Séance 2: Calculer des calendriers

Université Paris-Diderot

Objectifs:

- Travailler avec des expressions arithmétiques
- Trouver des solutions efficaces pour des problèmes arithmétiques
- Nommer des calculs par des fonctions, utiliser des fonctions

Dans ce TP nous allons travailler avec deux calendriers dont un est notre calendrier moderne, et l'autre un calendrier historique. Nous nous intéressons à la traduction de dates entre des calendriers différents, ainsi que le calcul de certains dates particulières dans une année comme des jours de transition entre temps d'hiver et temps d'été.

Rédiger les réponses aux questions dans le même fichier, en indiquant (par un commentaire) de quel exercice il s'agit. Cela vous permet de faire facilement appel à des fonctions que vous avez rédigées lors des exercices précédents.

Après chaque définition d'une fonction, ajoutez des cas de tests, par exemple ceux qui sont donnés sur la feuille.

Rappel : Tous les fichiers que vous sauvegarderez ou créerez pour ce TP devront être mis dans le répertoire `TP2` sous-répertoire du répertoire `IP1-Python` que vous avez créé au TP précédent.

1 Le calendrier le plus simple du monde

Le calendrier le plus simple qu'on peut imaginer consiste simplement à compter des jours à partir de 1. Nous allons appeler ce calendrier le *calendrier compteur*, le jour numéro 1 de ce calendrier est le même que le 1 janvier année 1 du calendrier *Grégorien*, qui est notre calendrier moderne.

Par exemple, le 14 septembre 2016 du calendrier Grégorien correspond au jour 736221 du calendrier compteur (nous allons plus tard dans ce TP écrire une fonction qui permet de faire ce calcul). Ce qui montre aussi que le calendrier compteur est peu pratique pour la vie quotidienne [1].

Exercice 1 (Calculer avec le calendrier compteur, ⋆)

1. *Écrire une fonction* `diff_counter` *qui prend une date de début et une date de fin du calendrier compteur en argument, et qui envoie le nombre de jours entre ces deux dates, la date de début incluse et la date de fin exclue. Par exemple,*

```
1        diff_counter(582,584) = 2
2
```

1. Pourtant, un calendrier très similaire mais avec une définition différente du jour 1 est aujourd'hui utilisé par les astronomes.

2. *On représente les jours de la semaine par des entiers : dimanche correspond à 0, lundi à 1, ..., samedi à 6. Écrire une fonction* `weekday_of_counter` *qui prend une date compteur en argument, et qui envoie le jour de la semaine correspondant, sachant que jour 1 était un lundi (représenté par 1). Par exemple,*

```
weekday_of_counter(1) = 1
weekday_of_counter(7) = 0
weekday_of_counter(11) = 4

```

□

2 Les années bissextiles

Une différence entre le calendrier Julien et le calendrier Grégorien est la définition des années *bissextiles* (en anglais : *leap year*) :

1. Dans le calendrier Julien, les années positives[2] bissextiles sont les années divisible par 4. Ainsi, l'année 1789 n'est pas bissextile dans le calendrier Julien, tandis que les années 1900 et 2000 le sont.
2. Dans le calendrier Grégorien, les années divisibles par 4 sont les années bissextiles, avec une exception pour les années divisibles par 100 : Si l'année est divisible par 100 alors elle n'est *pas* bissextile, sauf si elle est divisible par 400. Ainsi, l'année 1789 n'est toujours pas bissextile dans le calendrier Grégorien, l'année 1900 non plus car elle est divisible par 100 et pas divisible par 400, mais 2000 est bien une année bissextile car elle est divisible par 400.

Dans les deux cas, les années bissextiles ont 366 jours et les autres 365 jours.

Exercice 2 (Les années bissextiles, $\star$)

1. *Écrire une fonction* `is_leapyear_julian` *qui prend une année (≥ 0) en argument, et qui envoie une valeur Booléenne qui indique si l'année est bissextile dans le calendrier Julien. Par exemple*

```
is_leapyear_julian(1900) = True
is_leapyear_julian(1901) = False
is_leapyear_julian(2000) = True

```

2. *Écrire une fonction* `is_leapyear_gregorian` *qui prend une année en argument, et qui envoie une valeur Booléenne qui indique si l'année est bissextile dans le calendrier Grégorien.*

```
is_leapyear_gregorian(1900) = False
is_leapyear_gregorian(1901) = False
is_leapyear_gregorian(2000) = True

```

3. *Écrire une fonction* `days_in_year_julian` *qui prend une année (≥ 0) en argument, et qui envoie le nombre de de jours dans cette année selon le calendrier Julien. Indication : Servez vous de la fonction* `is_leapyear_julian`*. Puis, écrire une fonction* `days_in_year_gregorian` *qui fait la même chose selon le calendrier Grégorien. Par exemple,*

```
days_in_year_julian(1900) = 366
days_in_year_julian(2000) = 366
days_in_year_gregorian(1900) = 365
days_in_year_gregorian(2000) = 366

```

2. la règle pour les années bissextiles négatives est différente, mais ça ne doit pas nous concerner pour ce TP

□

3 Les mois

Le calendrier Julien et le calendrier Grégorien utilisent la même règle pour le mois de février : si l'année est bissextile le mois de février a 29 jours, sinon il a seulement 28 jours.

Exercice 3 (Les mois, ⋆)
Écrire une fonction `days_in_month` *qui prend en argument un mois et un Booléen qui indique s'il s'agit d'une année bissextile, et qui renvoie le nombre de jours dans ce mois. Les mois sont numérotés à partir de 1 : janvier=1, février=2,. . .,décembre=12. Par exemple*

```
days_in_month(1,False)=31
days_in_month(2,False)=28
days_in_month(2,True)=29
days_in_month(11,True)=30

```

□

4 Convertir des dates vers des dates compteurs

Maintenant on va combiner les fonctions écrites dans les exercices précédentes pour convertir de dates Juliennes et Grégoriennes en des dates compteur.

Il y a une deuxième différence, en plus de la règle des années bissextiles, entre les deux calendriers : le 1 janvier de l'an 1 dans le calendrier Grégorien est le même jour que le 3 janvier de l'an 1 dans le calendrier Julien ! En d'autres mot, tandis que le 1 janvier an 1 Grégorien est le jour 1 du calendrier compteur, le 1 janvier an 1 Julien est le jour -1 dans le calendrier compteur.

Une solution naïve pour convertir une date Grégorienne $(year, month, day)$ est d'additionner les durées (en nombre de jours) de toute les années strictement plus petites que $year$, puis d'y ajouter les durées des mois strictement plus petits que $mois$, et finalement d'y ajouter day. On fait pareil pour le calendrier Julien sauf qu'il faut ajuster le résultat selon le décalage expliqué en haut.

Exercice 4 (Conversion naïve de dates en compteur, ⋆⋆)

1. *Écrire une fonction* `gregorian_to_counter` *qui prend une année, un mois et un jour en argument, et renvoie comme résultat la conversion de cette date du calendrier Julien vers une date compteur. Suivez l'algorithme naïve expliqué en haut, en vous servant des fonctions écrites aux exercices précédentes, et des boucles* `for`*. Par exemple,*

```
gregorian_to_counter(1,1,1)=1
gregorian_to_counter(2,2,2)=398
gregorian_to_counter(101,1,1)=36525
gregorian_to_counter(2016,9,14)=736221

```

2. *Écrire, de la même façon, une fonction* `julian_to_counter` *qui convertit une date du calendrier Julien en calendrier compteur. Attention au décalage entre calendrier Grégorien et calendrier Julien. Par exemple,*

```
julian_to_counter(1,1,1)=-1
julian_to_counter(2,2,2)=396
julian_to_counter(101,1,1)=36524
```

```
julian_to_counter(2016,9,14)=736234
```

□

Exercice 5 (Applications de la conversion, ⋆)

1. *Écrivez, en vous servant des fonctions précédentes, une fonction* `weekday_of_gregorian` *qui prend une date* $(year, month, day)$ *du calendrier Grégorien en argument, et qui envoie le jour de la semaine (représenté comme en entier entre* 0 *et* 6*, comme expliqué à l'exercice 1). Par exemple,*

```
weekday_of_gregorian(2016,9,14)=3
```

2. *Dans la plupart des pays Européens, le passage du calendrier Julien au calendrier Grégorien était dans la nuit du 4 octobre 1582 (dans l'ancien calendrier Julien). Il fut décrété que le jour suivant cette date sera le 15 octobre 1582 du nouveau calendrier Grégorien. Vérifier, à l'aide des fonctions que vous déjà écrites, que le 15 octobre 1582 Grégorien était effectivement le lendemain du 4 octobre 1582 Julien.*
3. *En Europe, le passage de l'heure d'hiver à l'heure d'été se fait toujours le dernier dimanche du mois de mars, selon le calendrier Grégorien. Écrivez une fonction qui prend une année en entrée, et qui calcule le jour dans le mois de mars qui est le dernier dimanche du mars dans cette année. Indication : le jour le plus tôt possible est le 25. Calculez d'abord le jour de la semaine qui est le 25 mars de l'année en question, cela vous permet de trouver facilement le résultat. Par exemple,*

```
day_of_summertime(2017)=26
```

□

Exercice 6 (Une meilleure fonction de conversion, ⋆⋆)
Les fonctions de conversions vers le calendrier compteurs étaient un peu naïves, et pas très efficaces, à cause des boucles sur toutes les années qui précèdent l'année de la date qu'on cherche à convertir. Par exemple, pour une date de l'année 2016, la première boucle `for` *fera 2015 itérations seulement pour calculer combien de jours se sont écoulés jusqu'au 31 décembre 2015.*

Écrivez une fonction `julian_to_counter_clever` *qui calcule le même résultat que la fonction de conversion* `julian_to_counter` *de l'exercice 4, mais qui calcule plus efficacement le nombre de jours qui se sont écoulé dans les années précédentes. Pour simplifier le problème on considère que le calendrier Julien.*

Indication : Combien de jours y-a-t'il dans un bloc de 4 années successives., dans le calendrier Julien ?

Tester que votre nouvelle implémentation donne les mêmes résultats que l'implémentation naïve. □