# Timezone
# “Date & Time for dummies”

@Exirel // #Devcamp

# Définition

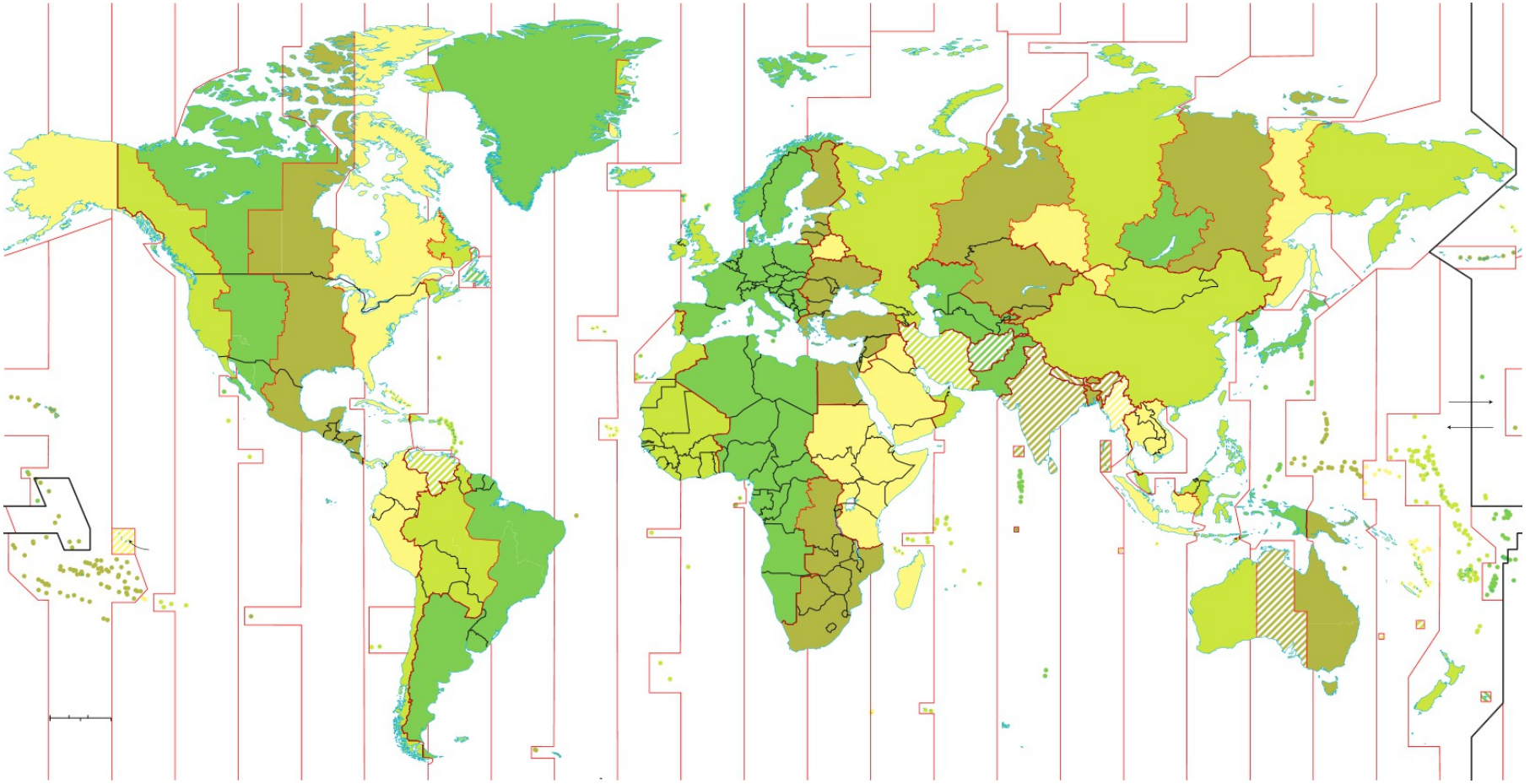
- Le **Temps universel** est une échelle de temps basée sur la rotation de la Terre.
- Un **fuseau horaire** est une zone de la surface terrestre où l'heure est identique en tout lieu.
- L'**heure d'Été** consiste, pour une zone de la surface terrestre, à changer de fuseaux horaires pour une période donnée.

# Problèmes

# Théorie

- Un **fuseau horaire** est une zone où l'heure est identique en tout lieu.
- **24 fuseaux horaires** de même taille.
- Le premier fuseau est centré sur **le méridien de Greenwich**.
- La zone fait 15° et est centrée sur un méridien dont la longitude est multiple de 15°

# Pratique



Carte des fuseaux horaires

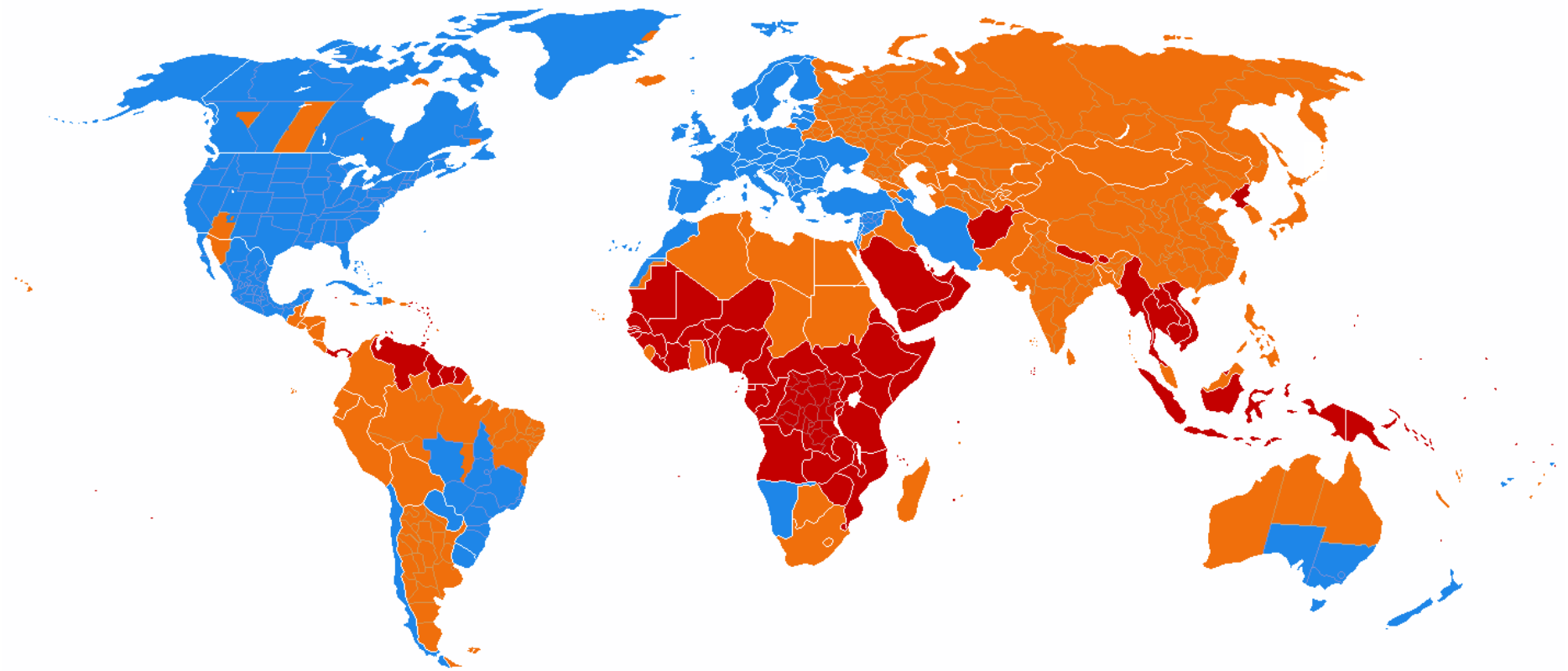
# Répartition géographique

- **43 fuseaux ou zones horaires** (42 officiels)
- Dont 37 toute l'année
- Mais pas forcément par les mêmes pays
  - deux seulement à l'heure d'hiver boréale (UTC-3:30 et UTC+3:30)
  - un uniquement à l'heure d'hiver australe (UTC+12:45)
  - un exclusivement à l'heure d'été boréale (UTC-2:30)
  - un dernier uniquement à l'heure d'été australe **(UTC+13:45)**.

# Daylight Saving Time

- **L’heure d’Été** est un système consistant à ajuster l’heure officielle
- Ajoute une heure par rapport à l'heure locale
- Du printemps jusqu'à la fin de l'été ou le début de l’automne
- **En Europe, dates normées et fixées au moins 5 ans à l'avance.**

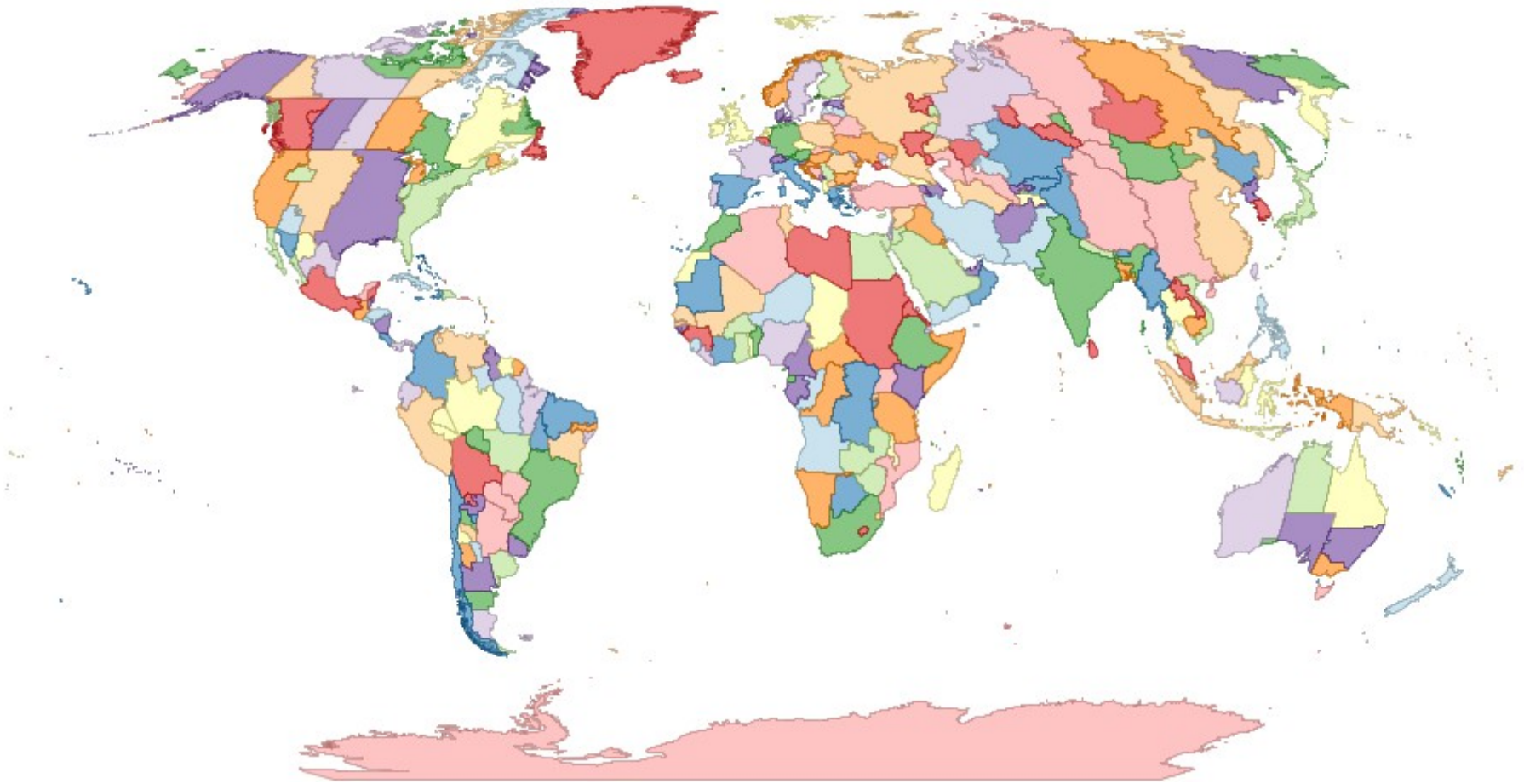
# Carte de l'heure d'Été



**Bleu** : l'utilise toujours. **Orange** : ne l'utilise plus. **Rouge** : ne l'a jamais utilisé.

# Résultat

# La carte des zones horaires



Timezone map (TZ Database)

# IANA Time Zone Database

- Créée par Arthur David Olson
- Convention de nommage spécifique :
  - America/Chicago, America/New_York
  - Europe/Paris, Europe/London
  - Asia/Seoul, Asia/Tel_Aviv
- Gère le DST
- Gère les secondes intercalaires

# Solution (du code)

# Use python!

```
>>> import pytz
>>> from datetime import datetime
```

# UTC

```
>>> datetime.utcnow()
datetime.datetime(2013, 11, 13, 10, 53,
52, 978000)
>>> pytz.utc.localize(datetime.utcnow())
datetime.datetime(2013, 11, 13, 10, 53,
59, 463000, tzinfo=<UTC>)
>>>
```

# Local time

```
>>> naive = datetime.utcnow()
>>> aware = pytz.utc.localize(naive)
>>> tz = pytz.timezone('Europe/Paris')
>>> local = aware.astimezone(tz)
>>> local
datetime.datetime(2013, 11, 13, 12, 4, 30,
931000, tzinfo=<DstTzInfo 'Europe/Paris'
CET+1:00:00 STD>)
```

# Explications

# Naive vs Aware

- Naive datetime :
  - **Sans information** de Timezone
  - `naive = datetime.now()`
  - `aware_local = local_tz.localize(naive)`
- Aware datetime :
  - Avec une Timezone
  - `aware_utc = aware.astimezone(pytz.utc)`

One Can Not Simply Mix **Naive** and **Aware** **datetime** in the same operation.

# Naive

```
>>> d1_naive.strftime(date_format)
“27 October 2013, 02:00”
>>> d2_naive.strftime(date_format)
“27 October 2013, 02:00”
>>> d2_naive – d1_naive
0
```

# Aware

```
>>> d1_aware.strftime(date_format)
“27 October 2013, 02:00”
>>> d2_aware.strftime(date_format)
“27 October 2013, 02:00”
>>> d2_aware – d1_aware
```

# Aware

```
>>> d1_aware.strftime(date_format)
“27 October 2013, 02:00”
>>> d2_aware.strftime(date_format)
“27 October 2013, 02:00”
>>> d2_aware – d1_aware
1
```

# Aware

```
>>> d1_aware.strftime(date_format)
“27 October 2013, 02:00 UTC+02:00”
>>> d2_aware.strftime(date_format)
“27 October 2013, 02:00 UTC+01:00”
>>> d2_aware – d1_aware
1
```

# Timezone avec TZ Data

- Zone horaire :

```
tz_local = pytz.timezone('Europe/Paris')
```

- Une zone horaire change de fuseau horaire pour l'heure d'Été :
  - Avant : **CEST** (Central European **Summer** Time)
  - Après : **CET** (Central European Time)

# Bonnes pratiques

# UTC in, Local out

- Lire vers UTC

```
pytz.utc.localize
```

- Ecrire depuis UTC

```
local_tz = pytz.timezone('Europe/Paris')
utc_aware.astimezone(local_tz)
```

# Utiliser les timezones

- Récupérer une timezone :

```
pytz.timezone('Europe/Paris')
```

- Localiser une date naïve :

```
tz.localize
```

- Transformer d'une TZ à une autre :

```
aware_local.astimezone(tz_far_away)
```

# Ne réinventez pas la roue

- TZ Database existe déjà, utilisez la !
- Ne manipulez pas les dates comme des nombres entiers
- Si votre langage ne peut pas le faire pour vous...
- … changez de langage !

Merci !