TP Python : décodage d'une trame NMEA

□ Objectifs

- Découvrir et utiliser la méthode `split`
- Comprendre le protocole NMEA-0183
- Traiter des données brutes et les interpréter

□ Pré-requis

- Bases de Python
- Listes

1 Présentation

Un récepteur GPS (ou plus simplement GPS) reçoit des informations d'une collection de satellites qui sont :

- géostationnaires pour quelques un d'entre eux
- en orbite vers 20000km d'altitude pour les autres

Quatre satellites sont une quantité minimale pour que le GPS puisse être opérationnel. Et ceci parce que le calculateur du GPS doit résoudre un système d'équations mathématiques où 4 variables interviennent : latitude, longitude, altitude et heure.
Au mieux, un maximum de 12 satellites sont pris en compte simultanément même si le GPS capte des informations issues de plus de 12 satellites.
Un GPS "assemble" les informations issues des satellites et construit des **trames NMEA** qu'il met à disposition des autres appareils électroniques.
Ces trames respectent le **protocole NMEA-0183** : il s'agit, comme tout protocole, d'un ensemble de règles à respecter pour communiquer.

Le développeur d'un dispositif souhaitant interroger un appareil GPS sait qu'il pourra exploiter cette trame pour obtenir des informations de date et de position.

Dans ce TP, nous allons découvrir ce qui se cache derrière une obscure **trame NMEA** comme celle-ci :

$GPGGA,123519.00,4222.7833,N,00856.7333,E,1,08,1.25,2706.4,M,46.8,M, , *4B

2 Les bases du protocole

Dans le protocole NMEA-0183, toutes les données sont transmises sous forme trames (phrases) composées de caractères ASCII. Chaque trame commence par le caractère $, suivi par :

- un groupe de 2 lettres pour identifier l'origine du signal. Les principaux préfixes sont :
  - **BD** ou **GB** pour *BEIDOU* (Chine)
  - **GA** pour *GALILEO* (Union Européenne)
  - **GP** pour *GPS* (États-Unis)
  - **GL** pour *GLONASS* (Russie)
- un groupe de 3 lettres pour le type de trame, comme :
  - **GGA** : GPS Fix et Date
  - **GLL** : Positionnement géographique Longitude - Latitude
  - **VTG** : Direction et Vitesse de déplacement (en noeuds et km/h)
  - ...

Suivent ensuite un certain nombre de **champs** séparés par une **virgule** jouant le rôle de **séparateur de champs**, permettant ainsi une extraction des différentes données par un programme informatique.
Chaque champ indique une donnée bien précise : la date, l'heure, la longitude, la latitude, ...

Certaines trames se terminent par un champ appelé **checksum**, que l'on peut utiliser pour vérifier l'intégrité des données de la trame. Le checksum, s'il existe, est précédé du caractère *. Il est égal à la valeur héxadécimale du « OU Exclusif » de tous les caractères compris entre $ et *.

Au total, une trame possède au maximum **82 caractères**.

☞ *Dans ce TP, nous étudierons seulement les trois types de trames présentées ci-dessous : GGA, GLL et VTG*

2.1 La trame GGA

La trame GGA fourni des coordonnées géographiques ainsi que l'heure d'obtention du « Fix GPS » (de la position).

$GPGGA,172814.0,3723.46587704,N,12202.26957864,W,1,6,1.2,18.893,M,-25.669,M,2.0,0031*4C

Champ	Contenu	Commentaire	Traduction
0	**$GPGGA**	Identifiant de la trame	GP : GPS GGA : type de trame
1	**172814.0**	Heure du fix GPS	17h 28m 14,0s Temps universel
2	**3723.46587704**	Latitude	37° 23,46587704' (DDM)
3	**N**	Orientation	N : Nord S : Sud
4	**12202.26957864**	Longitude	122° 02,26957864' (DDM)
5	**W**	Orientation	E : Est W : Ouest
6	**1**	État du fix	0 : Pas de fix 1 : Fix valide
7	**6**	Nombre de satellites utilisés	6 satellites
8	**1.2**	Précision horizontale	< 2 : excellent 2-5 : Bon 5-10 : Moyen > 10 : Mauvais
9,10	**18.893,M**	Altitude	18,893 mètres (M) au dessus du niveau de la mer
11,12	**-25.669,M**	Correction de la hauteur de la géoïde	M : en mètres
13,14	**2.0,0031**	Informations DGPS	Vide si pas de fix DGPS
15	***4C**	Checksum	« XOR » de tous les caractères précédents

2.2 La trame GLL

La trame GLL est plus simple que la trame GGA. Elle fournit seulement les coordonnées géographiques ainsi que l'heure du fix.

$GPGLL,4916.45,N,12311.12,W,225444,A*31

Champ	Contenu	Commentaire	Traduction
0	**$GPGLL**	Identifiant de la trame	GP : GPS GLL : type de trame
1	**4916.45**	Latitude	49° 16,45' (DDM)
2	**N**	Orientation	N : Nord S : Sud
3	**12311.12**	Longitude	123° 11,12' (DDM)
4	**W**	Orientation	E : Est W : Ouest
5	**225444**	Heure du fix GPS	22h 54m 44s Temps universel
6	**A**	Validité des données	A : valide V : non valide
7	***31**	Checksum	« XOR » de tous les caractères précédents

2.3 La trame VTG

La trame VTG est également une trame simple, elle fournit le cap (direction) et la vitesse au sol : très utile pour la navigation maritime ou aérienne.

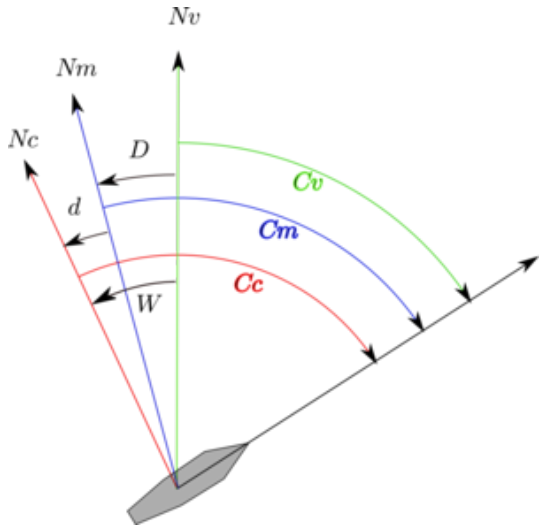
$GPVTG,054.7,T,034.4,M,005.5,N,010.2,K*48

Champ	Contenu	Commentaire	Traduction
0	**$GPVTG**	Identifiant de la trame	GP : GPS VTG : type de trame
1,2	**054.7,T**	Cap vrai en degrés	54,7° T : *True*
3,4	**034.4,M**	Cap magnétique en degrés	34,4° M : *Magnetic*
5,6	**005.5,N**	Vitesse de déplacement en Noeuds	5,5 Noeuds
7,8	**010.2,K**	Vitesse de déplacement en km/h	10,2 km/h
9	***48**	Checksum	« XOR » de tous les caractères précédents

Remarque (Wikipédia)**.** Le cap d'un mobile est la direction vers laquelle il est orienté (ou dans le cas d'un navire, la direction où pointe son étrave). C'est l'angle exprimé en degrés (de 0 à 360°), dans le sens des aiguilles d'une montre, entre la direction du nord et sa ligne de foi (son axe longitudinal). Cet angle se mesure à l'aide d'une boussole, d'un compas magnétique, gyroscopique ou satellitaire.

Il existe 3 types de cap :

- le **cap compas** (Cc) : c'est le cap indiqué par le compas.
- le **cap magnétique** (Cm) : c'est l'angle entre le nord magnétique (Nm) et la ligne de foi, une fois corrigé de la déviation (du défaut du compas magnétique inhérent à lui-même et à son environnement).
- le **cap vrai** (Cv) : c'est l'angle entre le nord géographique (ou nord vrai) (Nv) et la ligne de foi. Il peut être indiqué aujourd'hui directement par un compas satellitaire.



3 Coordonnées géographiques

Latitude, longitude, altitude

Pour se repérer sur Terre, on utilise un système de coordonnées géographiques : la latitude et la longitude.
Chaque point du globe est donc associé à deux valeurs :

- la **latitude** est une mesure angulaire allant 0° à l'équateur jusqu'à 90° aux pôles.
- la **longitude** est une mesure angulaire allant de 0° à 180°, relativement à un méridien donné (une « ligne droite » joignant les deux pôles). En général, le méridien d'origine choisi est le méridien de Greenwich, appelé aussi « premier méridien » .

Dans les deux cas, le nombre est suivi de la direction : Nord/Sud pour la latitude, Est/Ouest pour la longitude.

À ces deux nombres s'ajoute un troisième, l'**altitude**, désignant l'écart, en mètres, entre un point donné et un niveau de référence, généralement le niveau moyen de la mer.

Degrés, minutes et secondes

Pour situer précisément un point, il est nécessaire d'utiliser des fractions de degrés. Il est commun d'utiliser le système séxagésimal, comme pour les heures :

1 degré (1°) = 60 minutes d'arc (60') = 3600 secondes d'arc (3600')

Exemple. La ville d'Ajaccio se situe aux coordonnées (41°55'38"N, 8°44'04"E).
Sa latitude est donc de 41 degrés, 55 minutes d'arc, 38 secondes d'arc, direction Nord.
Pour localiser très précisémment un point, on peut aussi utiliser un nombre décimal de secondes, comme 38,234'.

Cette notation est appelée **DMS** (pour *Degrees, Minutes, Seconds*).
Il existe deux autres types de notation :

- la notation **DDM** pour *Degrees, Decimal Minutes* : on écrit un nombre décimal de minutes.

$$38 \text{ secondes} = \frac{38}{60} \approx 0,6338 \text{ minutes} \implies \text{la latitude d'Ajaccio est } 41°55.6338' \text{ Nord}$$

- la notation **DD** pour *Decimal Degrees* : on écrit un nombre décimal de degrés.

$$55,6338 \text{ minutes} = \frac{55,6338}{60} \approx 0,92723 \text{ degrés} \implies \text{la latitude d'Ajaccio est } 41,92723° \text{ Nord}$$

Remarque. Pour une latitude Sud ou une longitude Ouest, on peut utiliser des **nombres négatifs** dans la notation DD. Par exemple, 36,8927° Sud s'écrira -36,8927.

4 `split` : une méthode pour les séparer tous

Pour extraire des données d'une chaîne de caractères avec Python, la méthode `split` est très utile. Elle permet de former une liste à partir d'une chaîne de caractères découpées en plusieurs morceaux.
Pour savoir où couper la chaîne de caractères, on précise le **séparateur** utilisé, qui peut être n'importe quel caractère ou chaîne de caractères.

```
>>> chaine = "Ceci est un exemple"
>>> chaine.split(" ") # On découpe au niveau des espaces
['Ceci', 'est', 'un', 'exemple']
```

Un exemple d'utilisation est le suivant : considérons la chaîne de caractères `"2023/01/26 16:41:32"`, correspondant au jour et à l'heure exacte d'écriture de ces lignes.

☞ *La notation de la date est « à l'envers » : il s'agit en réalité de la notation anglo-saxonne de la date, très pratique en informatique lorsqu'il est question de comparer deux dates données...*

```
>>> dateEtHeure = "2023/01/26 16:41:32"
```

Pour extraire chacune des 6 informations (jour, mois, année, heure, minutes, secondes) on peut utiliser `split`.
On commence par séparer la date de l'heure, puis on récupère la date et l'heure dans 2 variables séparées :

```
>>> informations = dateEtHeure.split(" ")
>>> informations
['2023/01/26', '16:41:32']
>>> date = informations[0]
>>> heure = informations[1]
>>> date
'2023/01/26'
>>> heure
'16:41:32'
```

Comme toujours, il y a une astuce. On peut aller plus vite en écrivant simplement :

```
>>> date, heure = dateEtHeure.split(" ")
>>> date
'2023/01/26'
>>> heure
'16:41:32'
```

Enfin, il se reste plus qu'à « spliter » la date et l'heure, en précisant à chaque fois le séparateur :

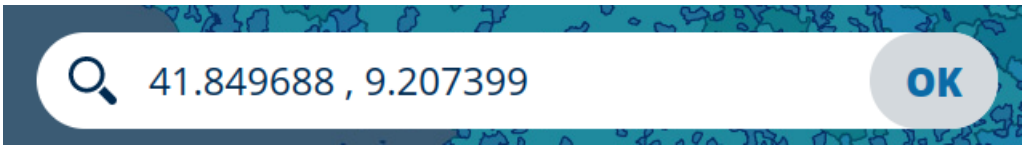
```
>>> annee, mois, jour = date.split("/")
>>> heures, minutes, secondes = heure.split(":")
>>> annee
'2023' # /!\ On récupère une chaîne de caractères, et non un entier /!\
```

5 Afficher une carte avec Géoportail

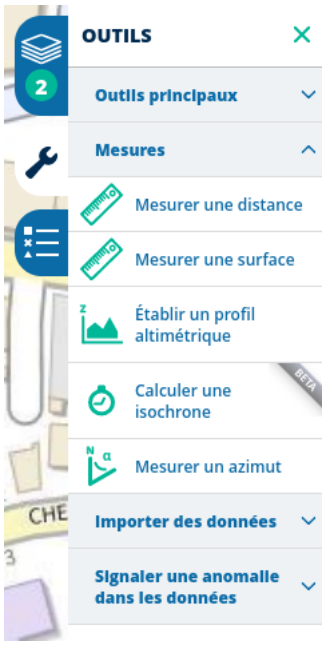
L'IGN (Institut Géographique National) a mis en oeuvre Géoportail, une application WEB donnant accès à des cartes et des photos aériennes de toute la France.

`https://www.geoportail.gouv.fr/`

En plus de rechercher des lieux, on peut afficher l'emplacement exact d'un point du globe dont on connaît les coordonnées géographiques. Pour faire cela, il faut calculer les coordonnées décimales (DD) du point, et les rentrer dans la barre de recherche, séparées par un espace ou une virgule :



L'inverse est également possible : en faisant un clic droit sur un point de la carte, on peut afficher ses coordonnées géographiques ainsi que l'altitude du point.
On peut également utiliser les outils cartographiques accessibles en cliquant sur la clé à molette sur la droite de l'écran, afin de mesurer une distance, une surface...



Exercices

Compréhension

Exercice 01 On considère la trame NMEA suivante :

$GPGGA,123519,4807.038,N,01131.324,E,1,08,0.9,545.4,M,46.9,M,,*42

1. Avec une trame de ce type (GPGGA), de quelles informations peut-on disposer ?
2. Déterminer les coordonnées géographiques du lieu, en DMS.
3. Déterminer l'altitude du lieu.
4. Combien de satellites ont été utilisés par l'appareil qui a généré cette trame ?

Exercice 02 On considère la trame NMEA incomplète suivante :

$GPGLL,,,,,194826.00,V,N*4A

1. Quelles informations peut-on extraire de cette trame NMEA ?
2. Que signifie le caractère `V` à la fin de la trame ?
3. Quelles données sont manquantes dans cette trame ?

Exercice 03 On considère la trame NMEA suivante :

$GPGGA,145822.384,4920.960,S,07013.173,E,1,12,1.0,0.0,M,0.0,M,,*74

1. Donner, au format DDM, les coordonnées contenues dans cette trame NMEA.
2. Convertir ces coordonnées en DD.
3. Convertir ces coordonnées en DMS.

Exercice 04 Un bateau est perdu au large de la Corse. À l'horizon, aucune côte n'est visible. Le GPS est endommagé, mais affiche encore les dernières trames NMEA reçues :

$GPGGA,115823.384,4218.587,N,00726.923,E,1,12,1.0,0.0,M,0.0,M,,*66
$GPVTG,197.28,T,192.4,M,005.5,N,016.1,K*73

1. Déterminer les coordonnées géographiques du bateau.
2. Localiser l'emplacement du bateau à l'aide de Géoportail.
3. Déterminer le cap vrai suivi par le bateau ainsi que sa vitesse, en km/h.
4. À l'aide de Géoportail, dire si le cap suivi par le bateau est correct pour rentrer au port d'Ajaccio.
5. À l'aide de Géoportail, calculer la durée approximative que mettra le bateau à rentrer au port s'il modifie son cap mais conserve sa vitesse actuelle.
6. À quelle heure arrivera-t-il au port, sachant que nous sommes le 26 Janvier 2023 ?

Programmation

Un peu de split

Exercice 05 Écrire une fonction `dateFR(chaine)` qui prend en paramètre une chaîne de caractères décrivant une date et une heure sous la forme `AAAA/MM/JJ` et qui retourne la date comme dans l'exemple suivant :

```
>>> dateFR("2023/01/26")
'26 Janvier 2023'
```

Exercice 06 On considère une chaîne composée uniquement des caractères numériques et de points, formant une série de nombres séparés par des points. Écrire une fonction `somme(chaine)` retournant la somme des nombres composant cette chaîne.

```
>>> somme("123.456.789")
1368
>>> somme("12.34.56.78.9")
189
```

Trames NMEA

Exercice 07 Écrire une fonction `heureFR(heureNMEA)` qui convertit une heure au format trame NMEA en heure lisible.

```
>>> heureFR("124722")
'12:47:22'
```

Exercice 08 Écrire une fonction `DDCoord(coordNMEA, direction)` qui convertit une coordonnée au format NMEA en notation DD (*Decimal Degree*).

```
>>> DDCoord("2759.291", "N")
27.988183
>>> DDCoord("11232.33", "W")
-112.538833
```

☞ ***La valeur de retour doit être un flottant arrondi à 6 décimales***

Exercice 09 Écrire une fonction `decodeGGA(trame)` qui retourne un tuple `(heure, latitude, longitude, altitude)` correspondant aux informations d'une trame GGA donnée. L'heure doit être donnée sous la forme *hhmmss*. La latitude et la longitude doivent être données au format DD, avec 6 décimales maximum.

```
>>> decodeGGA("$GPGGA,215354.523,4851.494,N,00217.668,E,1,12,1.0,33.87,M,0.0,M,,*6C")
('21:53:54', 48.858233, 2.294467, 33.87)
>>> decodeGGA("$GPGGA,122156.134,4549.962,N,00651.912,E,1,12,1.0,4810.35,M,0.0,M,,*69")
('12:21:56', 45.8327, 6.8652, 4810.35)
```

Extraction des données d'un fichier

Exercice 10 (Récupération des données)

1. Se rendre à l'adresse `https://nsi.dellasantina.corsica/files/premiere/tp-gps/trames_nmea.txt` et récupérer le fichier contenant des trames NMEA.
2. Récupérer le contenu de ce fichier dans une variable `contenu`, avec la syntaxe suivante :

```
with open("fichier.nmea") as fichier:
    contenu = fichier.read()
```

3. Récupérer dans une variable `trames` chaque ligne du fichier (y compris d'éventuelles lignes vides) à l'aide de la méthode `split`.

Exercice 11 (Filtrage des données)

1. Récupérer dans une variable `tramesGGA` la liste des trames GGA du fichier précédent.
2. Écrire une fonction `getTrames(fichier, type)` qui retourne la liste des trames de type donné dans le fichier donné.

```
>>> getTrames("trames_NMEA.nmea", "GGA")
['$GPGGA,,,,,,0,,,,,,,,*66',
 ↪  '$GPGGA,072709.00,4155.633115,N,00844.026411,E,1,00,1.3,65.8,M,48.4,M,,*58', ... ]
```

Exercice 12

1. Écrire une fonction `trameGGAValide(trame)` qui retourne `True` si, pour une trame GGA donnée, le fix GPS est valide.
2. Écrire une fonction `getTramesGGAValides(fichier)` qui retourne la liste des trames GGA valides contenues dans le fichier donné.s

Exercice 13 (Exploitation des données)

Écrire une fonction `readGPSData(fichier)` qui retourne la liste des tuples `(heure, latitude, longitude, altitude)` extraites à partir des trames GGA du fichier donné.

```
>>> readGPSData("trames_NMEA.nmea")
[('072709', 41.927219, 8.733774, 65.8), ('072710', 41.92722, 8.733786, 67.7), ...]
```

Exercice 14 Vérifier à l'aide de Géoportail le lieu où ont été enregistrées les trames NMEA du fichier précédent.

Calculs de checksum

Exercice 15 Écrire une fonction `getChecksum(trame)` qui retourne le checksum de la trame NMEA donnée.

```
>>> getChecksum("$GPGGA,174748.079,2759.291,N,08655.504,E,1,12,1.0,8848.7,M,0.0,M,,*65")
'0x65'
```

La valeur retournée doit être un nombre héxadécimal : on utilisera la fonction `hex`.

☞ ***Il ne faut pas calculer le checksum, mais juste lire sa valeur telle qu'elle apparaît dans la trame.***

Exercice 16 Écrire une fonction `XOR(chaine)` qui retourne le *OU Exclusif* de tous les caractères d'une chaîne.

```
>>> XOR("abc")
0x17
```

Quelques précisions sur le calcul effectué par la fonction XOR :

- Pour calculer le XOR de deux caractères, il faut au préalable transformer ces caractères en nombres entiers, c'est à dire déterminer leur valeur décimale dans la table ASCII. Ce que fait pour nous la fonction `ord`.
- Le XOR, ou « OU Exclusif », est une opération bit-à-bit définie dans Python par l'opérateur `^`

Exercice 17 Écrire une fonction `computeChecksum(trame)` qui **calcule** le checksum de la trame NMEA donnée.

```
>>> computeChecksum("$GPGGA,174748.079,2759.291,N,08655.504,E,1,12,1.0,8848.7,M,0.0,M,,*65")
'0x65'
```

Quelques informations :

- pour calculer le checksum, on utilise tous les caractères entre les symboles $ et * ($ et * exclus)
- la valeur retournée doit être un nombre héxadécimal

Exercice 18 Écrire une fonction `verifyChecksum(trame)` qui retourne `True` si le checksum d'une trame NMEA est correct, c'est à dire si le checksum calculé est égal au checksum lu dans la trame.

```
>>> verifyChecksum("$GPGGA,174748.079,2759.291,N,08655.504,E,1,12,1.0,8848.7,M,0.0,M,,*5E")
True
>>> verifyChecksum("$GPGGA,064036.289,4836.5375,N,00740.9373,E,1,04,3.2,200.2,M,,,,0000*2A")
False
```

Exercice 19

1. Que signifie en pratique un checksum erroné dans une trame NMEA ?
2. Parmi toutes les trames NMEA présentes dans ce TP, une seule présente un checksum erroné. Laquelle ?

Sources

- *Les Trames NMEA*

  `http://www.cedricaoun.net/eie/trames%20NMEA183.pdf`

- *NMEA-0183 messages : Overview*

  `https://receiverhelp.trimble.com/alloy-gnss/en-us/NMEA-0183messages_MessageOverview.html`

- `http://www.eauxturquoises.fr/a_tutopencpn/3_tutopencpn/36/367_NMEA/367_NMEA.htm`