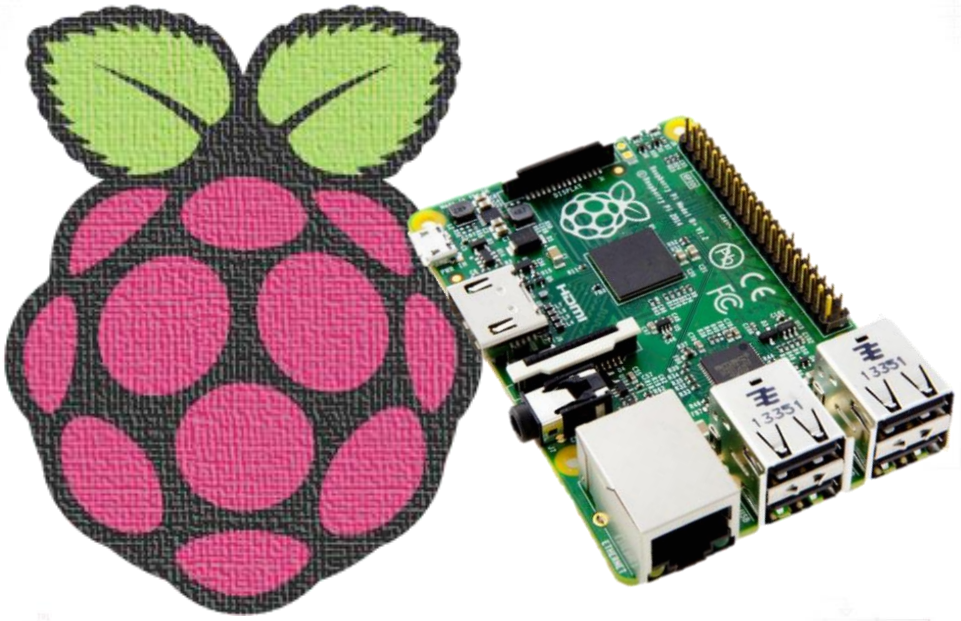
Découverte de la Raspberry Pi



Travaux pratiques
Séance n°3

Objectifs de la séance :

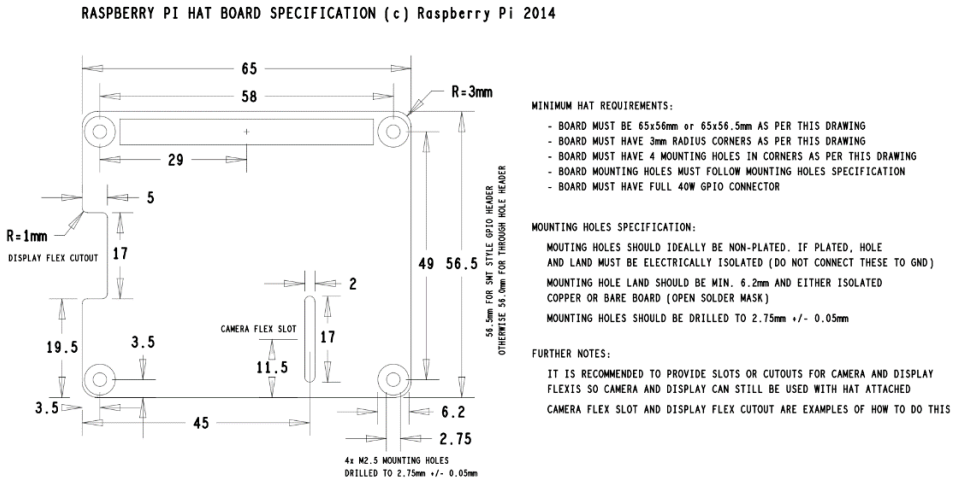
- Découvrir l'utilisation d'une carte d'extension commercialisée
- Utiliser une bibliothèque de très haut niveau disponible avec la carte
- Découvrir de nouveaux capteurs

Présentation de la carte d'extension 'Sense HAT' :

NE PAS ALIMENTER TOUT DE SUITE !!

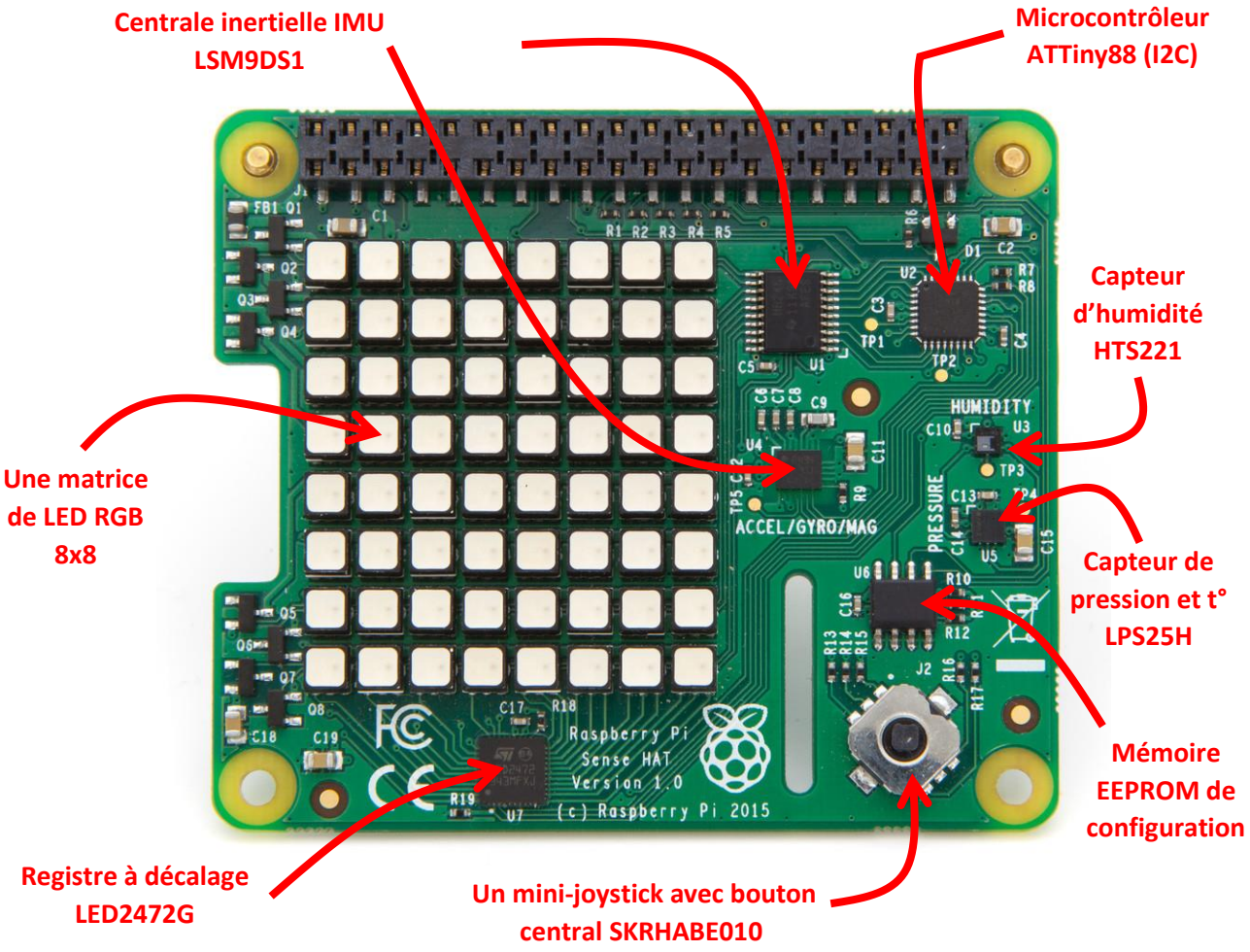
Nous allons utiliser au cours de ce TP une carte d'extension pour la Raspberry pi sortie en septembre 2015. C'est actuellement la seule carte d'extension proposée par la fondation 'Raspberry pi', celle qui est à l'origine du développement de la Raspberry Pi.
Remarque : pour information, cette carte est vendue à environ 40€, ce qui est le prix de la Raspberry !! Bon nombre de cartes d'extension, même parfois très simples, sont vendues assez cher. Si celle-ci concentre beaucoup de technologie dans un espace restreint, pas mal d'autres cartes se résument à des composants standards. Il devient alors beaucoup plus intéressant de la réaliser soi-même avec les compétences acquises à l'IUT… 😉

Cette carte '**Sense Hat**' utilise le format standardisé des cartes : HAT (Hardware Attached on Top). Ce standard formalise les dimensions, les caractéristiques mécaniques et électriques des cartes d'extension depuis juillet 2014. On peut le comparer aux shields pour l'Arduino.



Une caractéristique importante des cartes 'HAT' est la possibilité d'embarquer une mémoire EEPROM I2C qui communiquera avec la Raspberry dès la mise sous tension. Cette mémoire permettra notamment de fournir la configuration des différentes broches GPIO de la Raspberry pour qu'elles soient utilisables par la carte d'extension.

Qu'est-ce que l'on retrouve sur cette carte d'extension :



Chaque LED dispose de trois broches indépendantes R, G et B permettant d'obtenir par combinaison toutes les couleurs souhaitées.
Les LEDs sont câblées en matrice et pilotées par un registre à décalage LED2472G, lui-même commandé par un microcontrôleur Atmel ATTiny88. Un autre composant, le registre 74AHCT245 participe également au pilotage.
L'ensemble est commandé via le bus I2C du microcontrôleur (un bus de communication au même titre que la liaison série mais un peu plus compliqué à piloter). L'adresse I2C est **0x46**.

Le mini-joystick SKRHABE010 est également associé au microcontrôleur et possède donc la même adresse I2C.

Le capteur de pression LPS25H est directement connecté au bus I2C, son adresse est **0x5C**. Sa plage de mesure s'étend de 60 à 1260 hPa.
Le capteur d'humidité et de température HTS221 est lui aussi directement connecté au bus I2C. Son adresse est **0x5F**. Sa plage pour l'humidité est de 0 à 100% avec une précision de 4,5% et pour la température de -40° à +120° avec une précision maximale de 0,5°C.

La centrale inertielle IMU de référence LSM9DS1 contient un accéléromètre 3 axes, un gyroscope 3 axes et un magnétomètre 3 axes. Ce composant est connecté sur le bus I2C. L'adresse de l'accéléromètre/gyroscope est **0x6A** tandis que le magnétomètre est sur **0x1C**.
L'accéléromètre dispose d'une plage ±2/4/8/16 g
Le gyroscope dispose d'une plage angulaire ± 245/500/2000 dps (degrees per second)
Le magnétomètre dispose d'une plage ± 4/8/12/16 gauss

J'ai évoqué que le bus I2C qui permet d'accéder aux différents composants de la carte est relativement complexe en compréhension et en mise en œuvre. Toutefois ici, on va utiliser une librairie d'un niveau si élevée que le bus I2C deviendra totalement transparent, tout comme le pilotage de la matrice de LED via le registre à décalage.

Vous pouvez dès à présent installer votre carte Raspberry avec ses périphériques (clavier, souris, câble d'écran, câble Ethernet), sa carte µSD. Vérifier que vous êtes bien hors tension, et installer alors la carte d'extension Sense Hat sur la Raspberry.

Vous pouvez alors alimenter la Raspberry avec son adaptateur secteur. Vous devez voir le voyant d'alimentation de la carte d'extension s'allumer.

Récupération des fichiers utiles pour cette séance :

Pour cela, tapez la commande suivante dans le terminal :
$ sudo git clone https://github.com/MbottinIUT/IUT_IPE_TP3.git
[rem : il n'y a pas d'espace dans ce lien, juste des '_']

Une fois la tâche effectuée, vous devriez retrouver, en ouvrant un explorateur de fichiers, un dossier '**IUT_IPE_TP3**' sous '**/home/pi**'.

Recommandation importante concernant les programmes à écrire :

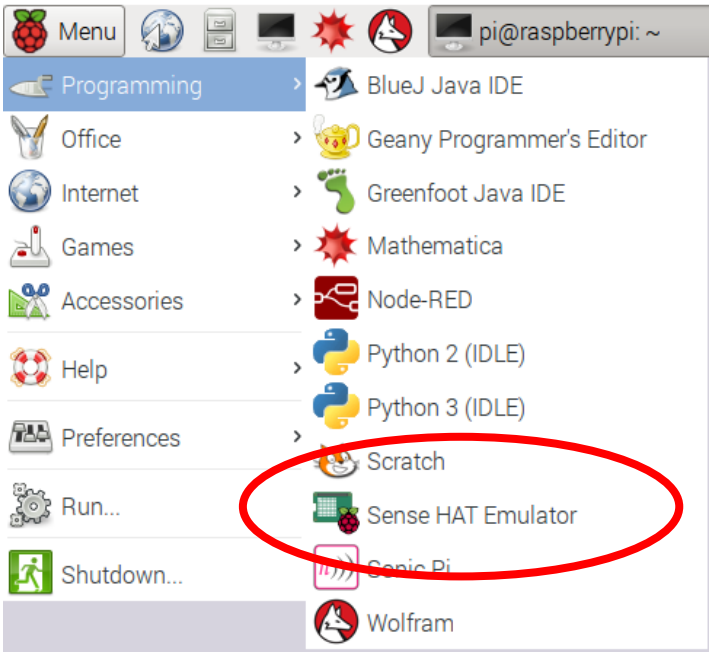
On utilisera comme d'habitude l'environnement de développement '**Geany**' et vous baserez vos programmes sur le template '**main_IPE.py**' comme vous l'avez fait pour les TPs précédents.

On va utiliser pour ces exercices une librairie python de très haut niveau dédiée à l'utilisation de cette carte. Vous trouverez la documentation (in english !!) de cette librairie dans le 'Memento Rpi' à partir de la page 37 ou encore avec la documentation PDF dans le dossier 'IUT_IPE_TP3/Documentation'.
Chaque méthode de la classe 'Sense hat' est bien détaillée et accompagnée d'un exemple.
Ceci devrait vous permettre d'écrire vos programmes avec une certaine autonomie !!

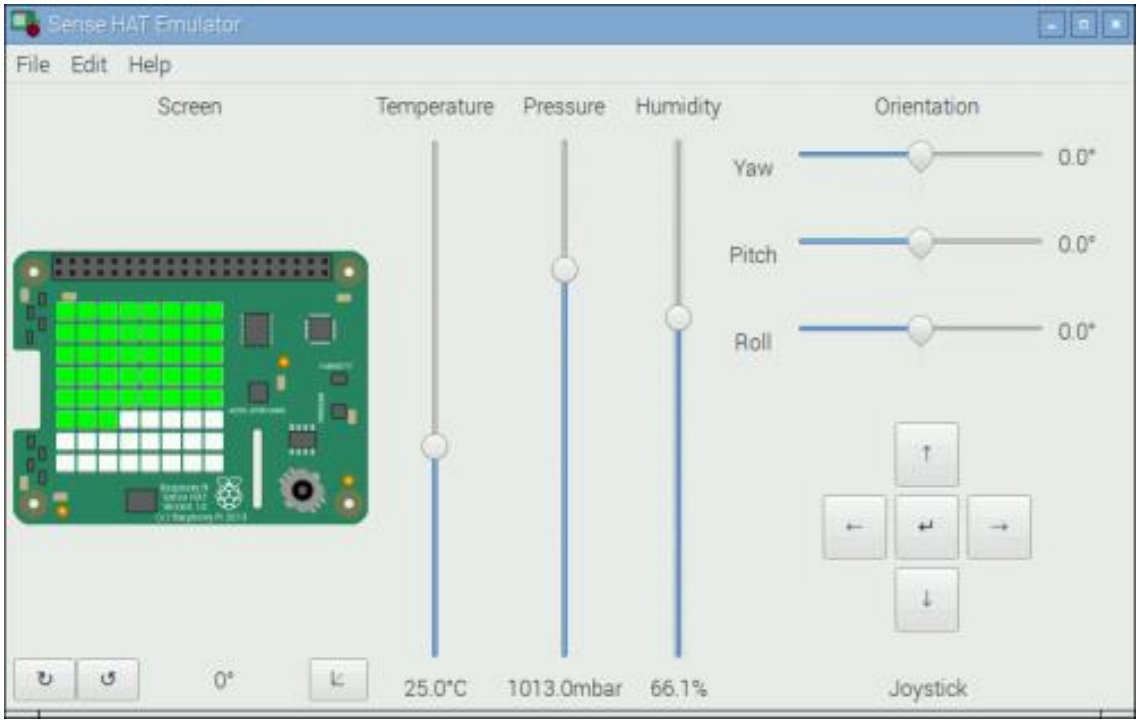
Cette librairie est installée par défaut dans l'OS. Toutefois, il est toujours intéressant de la maintenir à jour (quelques bugs peuvent avoir été corrigés récemment).

Veuillez donc taper la commande suivante dans un terminal :
$ sudo apt-get install sense-hat

Enfin, sachez que si vous possédez une carte Raspberry mais que vous trouvez la carte 'Sense Hat' un peu chère, il y a sous Stretch un simulateur de la carte :

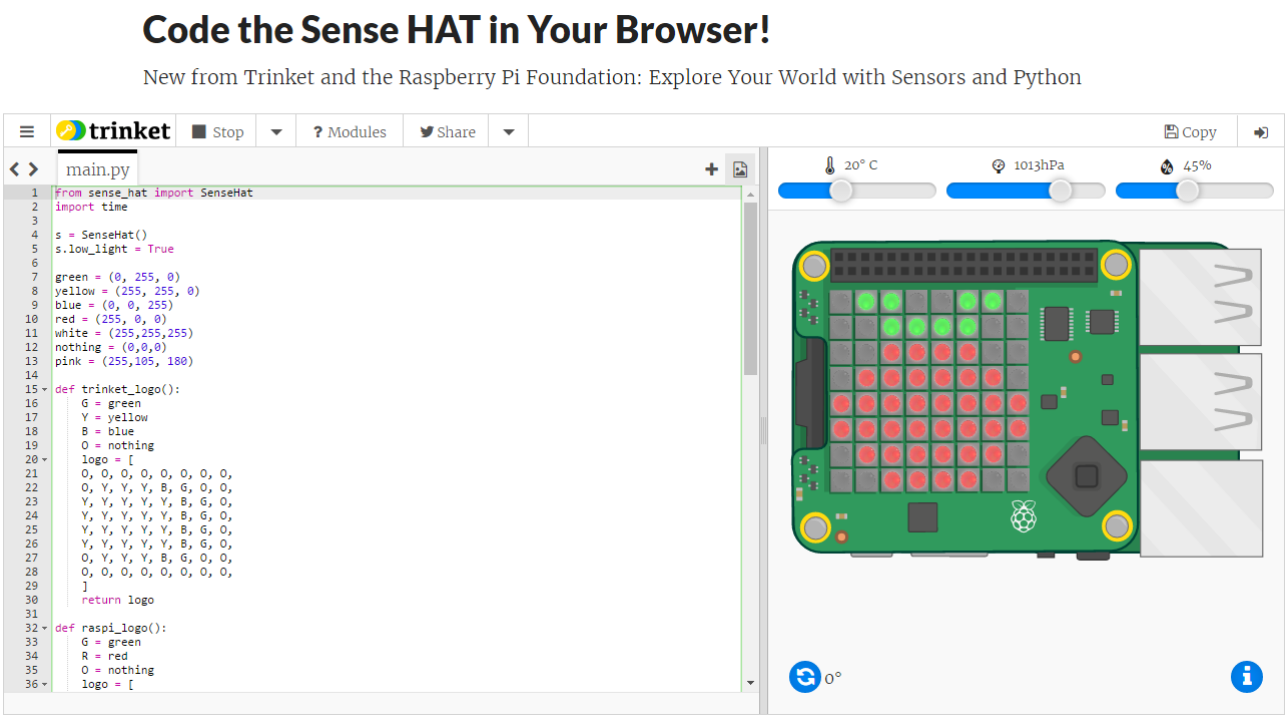


Ce dernier vous permet de tester votre code Python à travers une interface graphique simulant tous les composants de la carte 'Sense Hat' :



Et pour ceux et celles qui n'ont pas encore de Raspberry (il est encore temps pour envoyer une lettre au père Noël !!), sachez qu'il existe un simulateur de carte 'Sense Hat' en ligne. Vous avez juste besoin d'un navigateur et d'une connexion internet.
L'adresse est la suivante : https://trinket.io/sense-hat

Et voilà à quoi ressemble l'interface :



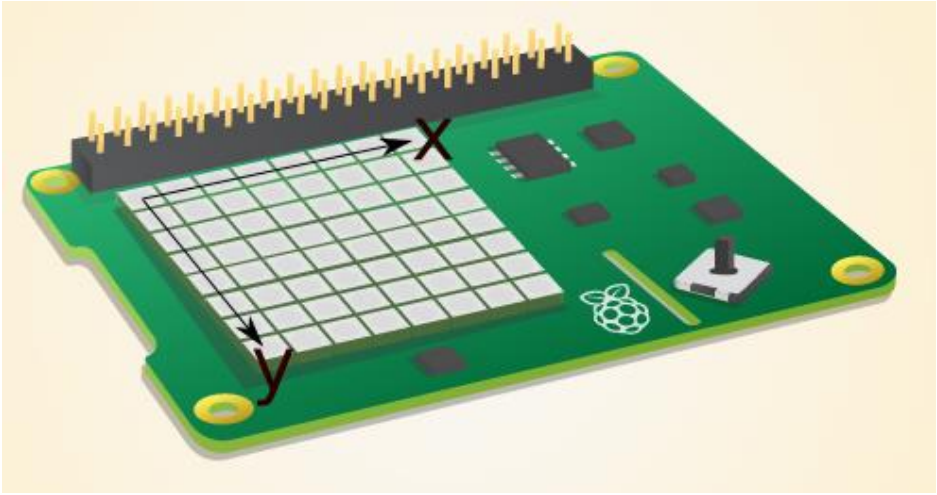
EXERCICE 1 : 'Ex01.py'

Objectif de l'exercice :

Pour commencer ce TP, on utiliser la matrice de LED RGB pour la remplir de pixels, chacun à une position aléatoire et d'une couleur aléatoire.

Informations utiles :

- Pour gérer la carte, il faut commencer par importer la librairie python correspondante. C'est l'instruction '**from sensehat import SenseHat**' qui permet cela. On aura donc à l'importer dans tous les exercices de ce TP.
- De la même façon, il faut créer (instancier) un objet basé sur la classe d'objet définie dans cette librairie. L'instruction qui s'en charge se retrouvera donc dans l'ensemble de nos exercices également : '**sense = SenseHat()**'
- **IMPORTANT :** dès lors, dès que vous voulez accéder à une méthode relative à cet objet, vous utiliserez la syntaxe '**sense.methode()**' avec ou sans paramètre dans les parenthèses. Et si vous voulez accéder à un attribut de cet objet, vous utiliserez la syntaxe '**sense.attribut**'.
- Il est important de connaître où est l'origine de la matrice 8x8 et comment sont numérotées les lignes et les colonnes pour la suite :



Voici donc comment sont orientés les axes des coordonnées x et y.
Leurs valeurs vont de 0 à 7.

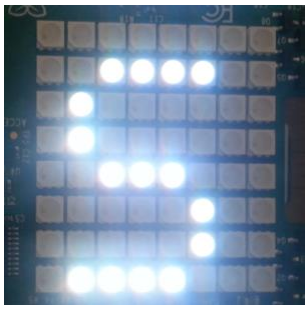
- Que ce soit pour les deux coordonnées **x** et **y** ou pour les trois composantes de la couleur **r**, **g** et **b**, vous aller devoir générer un nombre aléatoire. Pour cela vous utiliserez la méthode '**randint()**'. Je vous invite fortement à revoir son utilisation dans le TP n°2 lorsque par exemple, on devait simuler le lancer d'un dé à jouer. Remarque : à noter que les 3 composantes de la couleur sont comprises entre 0 et 255 !!
- Enfin, pour allumer un pixel aux coordonnées voulues et dans la couleur voulue, on va utiliser la méthode '**set_pixel()**'. A vous de regarder dans la documentation ('Mémento Rpi' page 39) pour savoir comment l'utiliser.
- Vous pouvez ajouter une faible pause (avec '**sleep**') entre chaque pixel allumé de manière à rendre l'animation plus agréable visuellement parlant.
- Veillez toujours à effacer la matrice ('**sense.clear()**') dans l'exception '**KeyboardInterrupt**' de façon à sortir proprement du programme.

Faîtes contrôler le bon fonctionnement de cet exercice par un enseignant.

EXERCICE 2 : 'Ex02.py'

Objectif de l'exercice :

On va poursuivre l'utilisation de cette matrice en y faisant défiler horizontalement un texte saisi dans le terminal par l'utilisateur. Vous devez vous assurer que la longueur du message n'excédera pas 60 caractères !!



Informations utiles :

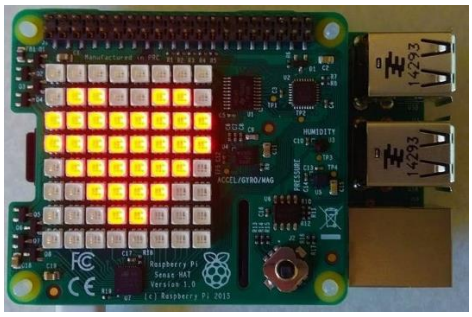
- Dans le terminal, le programme doit demander à l'utilisateur le message qu'il désire afficher et lui redemander un autre message si ce dernier dépasse 60 caractères. Remarque : pour l'information, on obtient la longueur d'une chaîne de caractères par la méthode '**len(chaine)**'.
- On affichera le message en jaune sur un fond bleu ou dans deux autres couleurs de votre choix pourvu qu'elles soient contrastées et que le message reste bien lisible.
- Pour l'affichage et le scrolling du texte, on va utiliser la méthode '**show_message()**' de l'objet '**sense**'. Utiliser la documentation du mémento (page 41) pour voir quels sont ses paramètres et comment l'utiliser.

Faîtes contrôler le bon fonctionnement de cet exercice par un enseignant.

EXERCICE 3 : 'Ex03.py'

Objectif de l'exercice :

On va chercher dans cet exercice à afficher sur la matrice, non plus du texte mais une image que l'on va animer.



Informations utiles :

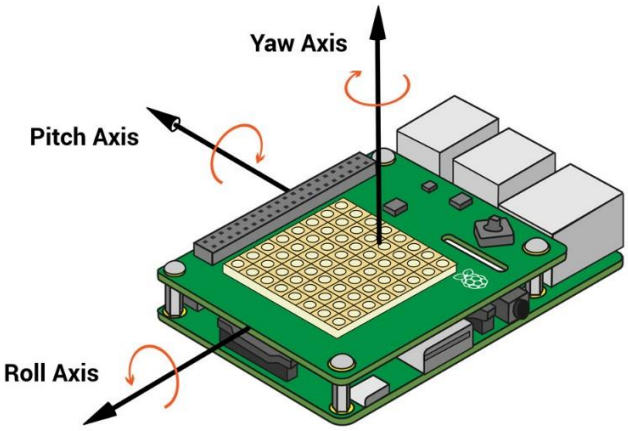
- Le programme incomplet vous est fourni : '**Ex03_incomplet.py**'
- On a défini ici deux tableaux de 64 'triplets' de couleur : '**heart1**' et '**heart2**'.
- Dans l'exercice n°1, on a utilisé la méthode '**set_pixel()**' pour contrôler un pixel à la fois. Ici, on va utiliser la méthode '**set_pixels()**' pour contrôler les 64 pixels de la matrice en une fois. Utilisez le mémento (page 38) pour compléter le programme.

- Une fois le programme terminé et testé, vous pourrez utiliser une autre méthode (‘**load_image()**’ en page 40 du mémento) de plus haut niveau encore qui se charge d’afficher sur la matrice une image issue d’un fichier, à condition que cette dernière soit au format 8x8 pixels. Vous utiliserez les images pour créer une animation dans le dossier ‘sprites’ (fichiers ‘pacman…’). Vous devez fournir en paramètre de la méthode le chemin relatif des images au dossier en cours.

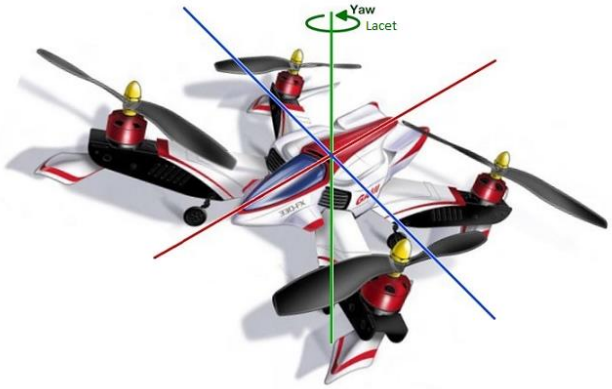
EXERCICE 4 : ‘Ex04.py’

Objectif de l’exercice :

On va maintenant apprendre à exploiter les données des capteurs présents sur la carte à commencer par la centrale inertielle (IMU) qui comprend un accéléromètre, un gyroscope et un magnétomètre.
C’est ce type de centrale que l’on va retrouver par exemple dans les drones pour stabiliser leur vol lors de leurs déplacements.
On peut accéder aux données brutes de ces 3 capteurs (Mémento pages 48 à 50), mais leur exploitation nécessite une mise en forme et un peu de calcul pour aboutir à des informations plus pertinentes comme le cap (ou lacet en aviation), le roulis et le tangage c’est-à-dire les orientations sur les 3 axes. La traduction anglophone de ces trois paramètres est : ‘**yaw**’ (ou ‘**heading**’), ‘**roll**’, et ‘**pitch**’. On les retrouve physiquement sur la carte de la manière suivante :



La flèche indique à chaque fois le sens d’incrémentation de ces angles qui sont exprimés en degrés.
Voici une représentation de ces 3 axes sur un drone :



(source : https://www.bourdons.fr/)

On va donc utiliser une méthode de très haut niveau qui va nous permettre d'accéder directement à ces 3 paramètres.
Dans cet exercice, vous allez donc chercher comment récupérer ces 3 informations et les afficher en continu dans la console du terminal.

Informations utiles :

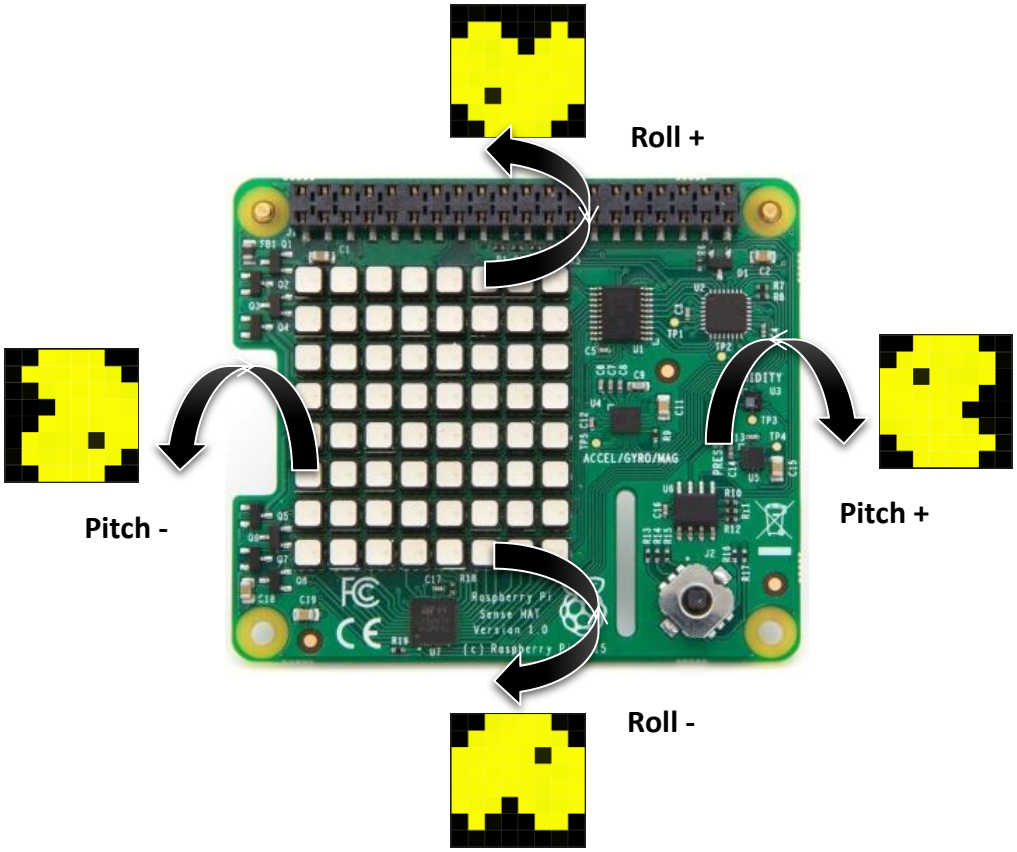
- Consulter le mémento (pages 46 à 50) pour déterminer la méthode à utiliser dans cet exercice.
- Cette méthode renvoie les mesures sous un format assez complet. Utilisez la méthode '**print**' pour comprendre le format des données renvoyées via le terminal. Isolez ensuite chacune des trois valeurs qui nous intéresse ('**yaw/cap**', '**pitch/tangage**' et '**roll/roulis**') afin de les arrondir à deux chiffres après la virgule (utilisation de la méthode '**round()**' : https://www.tutorialspoint.com/python/number_round.htm).

- Affichez enfin dans le terminal ces trois informations avec un message clair permettant de bien identifier chacune des trois valeurs.

Faire valider le bon fonctionnement de votre programme par un enseignant.

EXERCICE 5 : 'Ex05.py'

Objectif de l'exercice :
Maintenant que l'on sait récupérer les informations en degrés de la centrale inertielle, on va mixer le programme de l'exercice n°3 et celui de l'exercice n°4. On va donc faire pivoter l'affichage de l'image suivant le basculement sur les deux axes 'Roll' et 'Pitch' de la carte « Raspberry + Sense Hat » :



Informations utiles :

- Il va falloir tester les valeurs de '**Roll**' et de '**Pitch**' par rapport à leurs valeurs quand la carte est à l'horizontale puis fixer des seuils (basculement de 5° par exemple dans chaque direction) pour déterminer quel basculement est validé pour la rotation de l'image. Encore une fois, utilisez et abusez de la fonction '**print**' pour afficher dans le terminal des variables, des objets ou tout autre chose qui pourrait vous aider dans l'écriture de votre code.
- Vous utiliserez la méthode '**set_rotation()**' pour faire tourner l'image sur la matrice sans avoir à définir une image différente pour chaque position de la Raspberry. Pour cela, vous regarderez dans le mémento page 37. *Remarque :* vous pouvez également utiliser les méthodes '**flip_h()**' et '**flip_v()**' si vous le désirez.

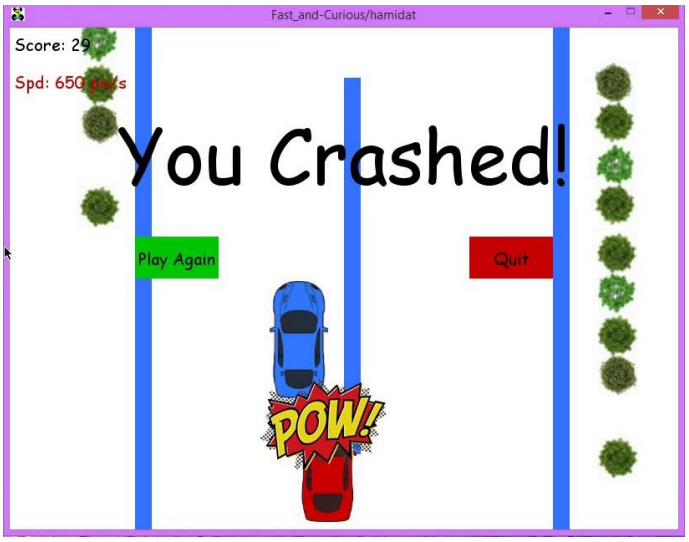
EXERCICE 6 : 'Ex06.py'

Objectif de l'exercice :

A partir du moment où l'on sait déterminer la position 3D d'un élément physique que l'on manipule, celui-ci peut servir à beaucoup de choses, notamment faire office de contrôleur de jeu (il manquerait juste quelques boutons poussoirs).

C'est l'utilisation de la centrale inertielle comme « joystick » que l'on va exploiter dans cet exercice. Toutefois, la création d'un jeu avec une interface graphique consommerait beaucoup trop de temps dans ce module.

Aussi, nous allons partir d'un jeu existant, développé par Mohamed Hamidat (disponible sur Github à l'adresse suivante : https://github.com/mohamedhamidat/car_game). Le jeu original se joue au clavier. Vous allez le modifier pour le piloter avec la centrale inertielle.



Informations utiles :

- Ouvrir le fichier '**Ex06_original.py**' et enregistrez-le sous le nom '**Ex06.py**'.
- La gestion des mouvements du véhicule au clavier se fait entre les lignes 232 et 250. Observez bien la structure de ces quelques lignes (elles sont faciles à comprendre). La variable '**x_change**' représente le déplacement en pixels du véhicule sur l'axe x à chaque appui sur une touche de direction.
- Inspirez-vous de l'exercice précédent pour gérer le déplacement du véhicule avec l'orientation '**roll**'.

Pensez à faire valider votre programme dès qu'il fonctionne.

Remarque :

- Vous disposez d'un autre code python pour cet exercice '**Ex06_variante.py**'. Il s'agit juste d'une démonstration visant à vous montrer les possibilités 3D de la Raspberry et prendre ainsi conscience que l'on est très loin des possibilités d'un microcontrôleur !! Toutefois pour cette démo, il vous faudra installer le package suivant via un terminal : **$ sudo pip install pi3D**

EXERCICE 7 : 'Ex07.py'

Objectif de l'exercice :

Cet exercice ne va pas vous demander beaucoup d'effort car une seule ligne de code vous sera demandée. Il s'agit ici de créer une interface graphique qui affiche une photographie. A chaque endroit de la photo où l'on va cliquer avec la souris, on va récupérer la valeur de la couleur du pixel correspondant et afficher cette couleur sur la matrice de la carte Sense Hat.

Informations utiles :

- Vous allez pouvoir ouvrir le programme '**Ex07.py**' qui est quasiment complet. Ce dernier utilise le package '**Tkinter**' qui permet de créer des interfaces graphiques. Là encore, il serait trop long de présenter cette librairie, mais cet exemple pourra vous aider ultérieurement.

- La seule ligne de code que vous devez ajouter est celle de la ligne n°25. L'instruction à ajouter est celle qui va permettre de modifier la totalité de la couleur de la matrice. A vous de trouver laquelle (en vous aidant du mémento ou d'internet si nécessaire).
- Vous pouvez aussi lors de vos tests, changer de photographies pour vérifier que votre programme fonctionne quel que soit le fichier utilisé. Il suffit juste de modifier le chemin du fichier de la ligne n°39 (plusieurs photographies sont fournies dans le dossier '**images**' du TP3)

Pensez à faire valider le fonctionnement de votre programme avant de passer à la suite.

EXERCICE 8 : 'Ex08.py'

Objectif de l'exercice :

On va maintenant s'intéresser à un autre capteur disponible sur la carte Sense Hat. En effet, outre l'IMU, il y a des capteurs environnementaux donnant des informations telles que la température, l'humidité et la pression. On va se concentrer sur la mesure de la température.
Le but va donc être d'afficher en message défilant sur la matrice 8x8 la température mesurée par le capteur.
La couleur de fond de la matrice dépendra de la valeur de la température.

Informations utiles :

- On va utiliser, comme nous l'avons déjà fait, la méthode '**Show_message()**' pour afficher les différentes informations provenant des capteurs.
- Le capteur de température est câblé sur le bus I2C de la Raspberry, mais on dispose toujours ici d'une librairie de très haut niveau qui rend la communication sur ce bus totalement transparente pour l'utilisateur. On va utiliser la méthode '**get_temperature()**' de la librairie '**Sense Hat**'. Veuillez-vous reporter au mémento (page 44) pour voir son utilisation.
- Pour améliorer la lecture sur la matrice 8x8, on va arrondir ces informations à un seul chiffre après la virgule. On a déjà utilisé cette fonction native de Python, la fonction '**round()**' notamment pour la centrale inertielle. Vous pouvez toujours vous référer à la documentation officielle de Python en ligne : https://docs.python.org/2/library/functions.html
- Pour l'affichage, il va falloir combiner dans une chaîne de caractères un message au texte évocateur (exemple 'temperature = ' ou 'temperature : ').
- Vous pouvez bien sûr utiliser à tout moment la fonction '**print**' dans le terminal pour contrôler le contenu des objets que vous manipulez.
- Pour la couleur de fond de la matrice, les seuils de température vont être un peu étonnants. En effet, le capteur de température étant situé juste au-dessus de la Raspberry, la chaleur qui se dégage de cette dernière fausse forcément la valeur obtenue. On va donc choisir :
  - Couleur bleue : température < 25°
  - Couleur magenta : température comprise entre 25° et 35°
  - Couleur rouge : température > 35°

  [Veillez à utiliser une couleur de texte compatible avec les trois couleurs de fond]
- Remarque : En fait, la carte '**Sense Hat**' dispose de deux capteurs de température, un intégré au capteur d'humidité, l'autre au capteur de pression. Il peut être intéressant de comparer leurs deux résultats. Jetez un coup d'œil page 45 du mémento pour savoir comment vous y prendre.

Faire valider le fonctionnement de votre programme par un enseignant.

EXERCICE 9 : 'Ex09.py'

Objectif de l'exercice :
On va tenter d'améliorer un peu notre mesure de température. Comme celle-ci dépend de la chaleur générée par la carte Raspberry Pi juste en-dessous, on va chercher à corriger la température en utilisant la température du CPU.

Informations utiles :

- La librairie '**GPIOzero**' nous fournit une méthode pour accéder à cette température. Vous pouvez vous référer à la page web suivante pour son utilisation : https://gpiozero.readthedocs.io/en/stable/api_other.html#cputemperature
- Plusieurs formules empiriques sont disponibles sur le net. On va utiliser celle-ci dans notre code **T_corrigee = T_mesuree – ((T_CPU – T_mesuree)/0.75)**
- Modifiez votre programme précédent pour assurer cette correction.

Faire valider le fonctionnement de votre programme par un enseignant.

EXERCICE 10 : 'Ex10.py'

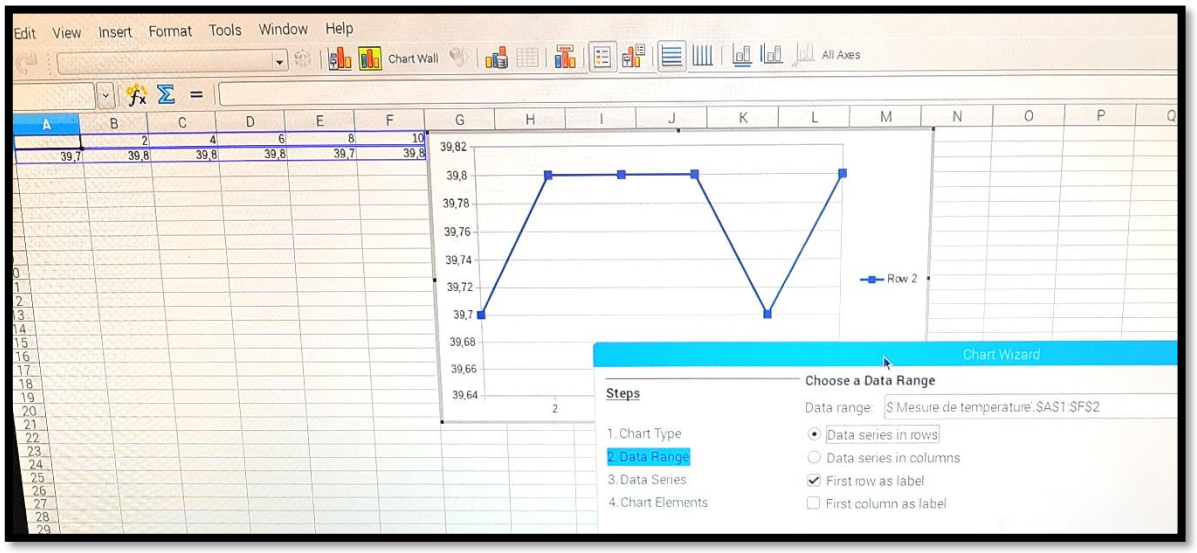
Objectif de l'exercice :
On va ici étudier comment on peut stocker dans un fichier tableur toute une série de mesures de température. Cette opération de 'data-logging' est très courante dans l'industrie sur des bancs de test notamment.
On va voir que ce type de traitement de données avec la Raspberry est relativement simple à mettre en œuvre.

Informations utiles :
Python gère les fichiers en natif, mais on va utiliser ici une librairie qui permet de s'intéresser plus particulièrement aux fichiers de type tableur et donc de faire des opérations diverses sur les cellules (tri, formules, …) directement depuis Python. Cette librairie se nomme '**pyexcel-ods**'. Elle s'installe via un terminal par la commande :

```
$ sudo pip install pyexcel-ods
```

- Le programme vous est fourni de manière incomplète via le fichier '**Ex10_incomplet.py**'
- Dans l'exercice n°2 du TP2, on a utilisé à la fois des dictionnaires et des listes pour stocker les dessins des différentes faces d'un dé. Le dictionnaire contenait une liste de caractères pour chaque face du dé et chaque liste était identifiée par une clef représentant le chiffre de la face du dé. Ici aussi, les données permettant de stocker des informations dans le fichier tableur, le seront sous forme de dictionnaire de listes.
- Le fichier '**Ex07_incomplet.py**' intègre déjà les importations de librairies nécessaires dont '**pyexcel-ods**'
- Pour créer un fichier tableur, il nous faut donc utiliser des listes en Python (équivalentes à des tableaux, on en a parlé à l'exercice n°2 du TP n°2). Vous allez donc créer deux listes vides en début de programme (après vos importations) :
  - **liste_temporelle = []**
  - **liste_temperature = []**
- Vous continuerez par demander à l'utilisateur les trois informations suivantes :
  - Le nom du fichier qu'il désire
  - La durée totale des mesures (en secondes)
  - Le pas entre chaque mesure (en secondes)

- Vous devez utiliser les deux dernières informations pour déterminer le nombre de mesures à effectuer qui constituera l'indice d'itération maximal de votre boucle.
- Vous gérez ensuite votre boucle de mesures. Pour la partie 'détermination de la température', vous utiliserez le code que vous avez mis au point dans les exercices précédents.
- Dans cette boucle, vous aurez à chaque itération la nécessité de compléter les deux listes de mesures, l'une avec la durée en secondes, l'autre avec la valeur de la température. Pour ajouter une valeur à la fin d'une liste, pas besoin de connaître l'indice, vous avez juste à appeler la méthode suivante : '**liste.append(valeur_a_ajouter)**'
- La fin du programme est déjà écrite, elle met à jour les données du dictionnaire avec les deux listes de mesures et sauvegarde le fichier.
- Pour obtenir des informations supplémentaires pour compléter le programme, vous pouvez consulter la page web suivante : https://pythonhosted.org/pyexcel-ods/
- Vous pouvez une fois votre programme exécuté, lancer le fichier '**ods**' ainsi créé. Vous en profiterez pour créer un graphique dans Calc pour visualiser vos mesures :



Faire valider le fonctionnement de votre programme par un enseignant.

EXERCICE 11 : 'Ex11.py'

Objectif de l'exercice :

On va terminer cette séance de TP avec deux exercices qui utilisent toujours le capteur de température, mais dont le but est d'afficher un graphe de la température mesurée au cours du temps.

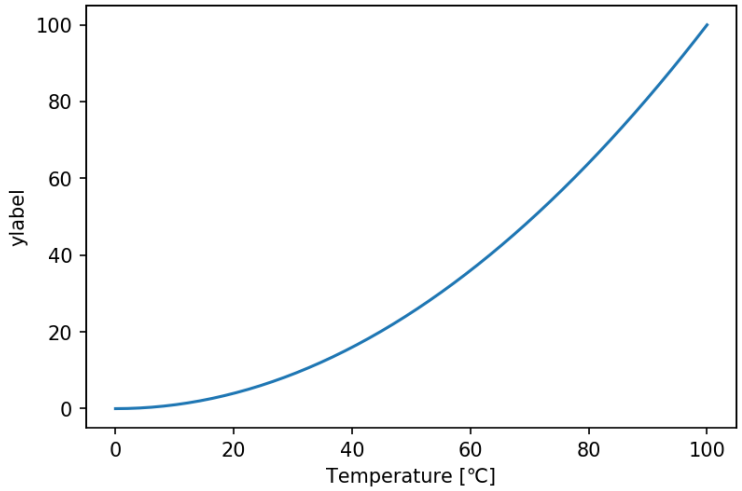
Là encore, on utilise une librairie de haut niveau permettant la création d'une fenêtre avec un graphe affichant la ou les courbes de données souhaitées. Cette librairie s'appelle '**matplotlib**' et est installée par défaut dans l'OS.

Remarque : Sachez toutefois que cette librairie est disponible en version 3.x uniquement pour Python 3. Pour nous, utilisateurs de Python 2.x, il nous faut la dernière version stable 2.2.3. Mais celle-ci ne sera plus mise à jour !!

On voit donc bien que la tendance soit à Python 3.x mais très souvent pour les librairies Python les plus populaires et les plus anciennes. Pour les librairies dédiées à la Raspberry (souvent celles communicantes avec du hardware), elles sont encore 'jeunes' et certaines n'existent qu'en Python 2.x, d'où notre choix pour ces TPs.

Pour s'assurer d'avoir tout de même la dernière version à jour, ouvrez un terminal pour y taper la commande suivante :

$ sudo apt-get install python-matplotlib



Informations utiles :

- Pour l'utilisateur, il faut comme dans l'exercice précédent lui demander la durée et le pas de la mesure (mais bien sûr le nom du fichier devient ici inutile).
- Comme lors de la création d'un fichier 'ods' dans le précédent exercice, nous avons aussi besoin de listes de valeurs pour créer un graphique :
  - 'liste_temporelle' pour l'axe X
  - 'liste_temperature' pour l'axe Y
- Ces listes vont être remplies comme précédemment par une boucle ayant comme nombre d'itérations le nombre de mesures à effectuer.
- La création du graphique ne se fera qu'à la fin du programme. Vous pouvez vous référer à la documentation de la librairie (https://matplotlib.org/contents.html ou encore celui-ci https://matplotlib.org/tutorials/introductory/pyplot.html#sphx-glr-tutorials-introductory-pyplot-py). Toutefois, la documentation d'une telle librairie est souvent dense et il convient plutôt de rechercher un tutoriel expliquant les bases comme celui-ci : https://www.raspberrypi.org/learning/visualising-sorting-with-python/lesson-1/worksheet/ Aidez-vous donc de cette page pour écrire la fin de votre programme

Faire valider le bon fonctionnement de votre programme par un enseignant.

EXERCICE 12 : 'TP4_ex12.py'

Objectif de l'exercice :

On va terminer ce TP en modifiant légèrement le programme précédent. Supposant que la durée de la mesure soit longue, il peut être plus intéressant de voir le graphe s'afficher progressivement plutôt que de le voir s'afficher intégralement à la fin de la série de mesures.

C'est ce que l'on va faire ici. La librairie '**matplotlib**' propose un mode interactif dans lequel le graphique s'affiche de manière dynamique au cours du temps.

Informations utiles :

- Corrigez votre programme pour que le graphe s'affiche au fur et à mesure.
- Les modifications à apporter sont mineures et encore une fois, des tutoriels en ligne pourront grandement vous aider. Je vous en propose deux (les méthodes diffèrent légèrement mais les deux fournissent le résultat escompté) :
  - https://projects.raspberrypi.org/en/projects/temperature-log/6
  - https://learn.sparkfun.com/tutorials/graph-sensor-data-with-python-and-matplotlib/all

Faire valider le bon fonctionnement de votre programme par un enseignant.