Meteo-Copy4

June 20, 2019

1 Partie 1 Les Tableaux (présentation bloc 1)

1.0.1 prérequis

If, for, return, in, def, while

1.0.2 Objectifs

Création de liste (manuelle, automatisée)

```
In [ ]: liste=[1, 2, 3, 4, 5]
        liste=[]*nombre
        liste[3]=int(input())
```

1.0.3 Le contenu d'une liste

- Se repérer dans la liste

```
In [9]: liste=[1, 2, 3, 4, 5]
        print(liste[2])
        print(liste[3])
```

```
3
4
```

- Chercher dans la liste/se référer à la liste

```
In [10]: liste[2]=liste[2]+5
         print(liste)
```

```
[1, 2, 8, 4, 5]
```

- Ajouter/modifier un élément dans la liste

```
liste.append
liste.extend
liste1+liste2
```

- Manipuler liste (retourner, trier, longueur,...)

```
liste.reverse()
liste.sort()
liste.len()
liste.pop()
liste.insert()
```

1.0.4 Activité découverte : De la nécessité d'un tableau.

Dans une librairie, un acheteur décide d'acheter quelques livres pour ses vacances. Aucun prix n'est indiqué sur le livre mais seulement une catégorie. Au bout d'une étagère, les prix sont affichés. Suivant la catégorie croissante, les livres coûtent 4,50€ ; 5,20€ ; 6,10€ ; 7,60€ ou 8,90€.

Pour estimer le coût total de ses achats, il observe la catégorie de chacun des livres. Il a dans ses mains 2 livres de catégorie 3, 1 livre de catégorie 1, 3 livres de catégorie 4, 2 livres de catégorie 2 et un livre de catégorie 5

Quelle somme devra-t-il débourser lors de son passage en caisse ?

Le but de cette activité simple est de proposer une solution claire et détaillée en expliquant votre méthode.

- Discussion : nécessité d'avoir un référencement = une liste ou un tableau
- Quelles autres informations pourrions-nous tirer de l'ensemble de ces données ?

1.1 Cours

- Création de liste (manuelle, automatisée)

```
liste=[1, 2, 3, 4, 5]

liste=[0]*tailleListe génère une liste remplie dun nombre tailleListe de [0]
```

Par exemple :

```
In [11]: jourSemaine=[0]*7
         print(jourSemaine)

[0, 0, 0, 0, 0, 0, 0]
```

- Se référer/remplir une liste

```
liste[i]  :
```

Se réfère à la position i-1 dans la liste car le premier élément de la liste est de rang 0 et non 1. Notre liste de 7 jours possède 7 rangs de 0 à 6

Exemple

```
In [26]: jourSemaine=[0]*7
         jourSemaine[1]=15
         jourSemaine[4]=27
```

- affichage des éléments de la liste :

```
In [27]: for elt in jourSemaine :
             print(elt)
```

```
0
15
0
0
27
0
0
```

afficher le i ème élément de la liste : print (liste[i-1])
Exemple :

```
In [28]: jourSemaine[4]
```

```
Out[28]: 27
```

1.1.1 Activité 1

On a décidé de relever la température de l'air extérieur, durant la journée du jeudi 13 juin 2019, toutes les heures :

0h : 12řC
1h : 11řC
2h : 11řC
3h : 10řC
4h : 9řC
5h : 10řC
6h : 9řC
7h : 10řC
8h : 12řC
9h : 14řC
10h : 16řC
11h : 17řC
12h : 18řC
13h : 20řC
14h : 21řC
15h : 22řC
16h : 23řC
17h : 23řC
18h : 23řC
19h : 23řC

20h : 21řC
21h : 19řC
22h : 17řC
23h : 16řC

On veut pouvoir extraire diverses données de ces informations, comme la température moyenne, la température moyenne pendant le jour, les températures maximale et minimale, les heures correspondant à ces températures, etc.

Exercice 1 :

1. construire manuellement la liste des températures (on ne s'occupe pas de l'heure)
2. afficher les éléments de la liste
3. afficher un rapport indiquant ń à … h il fait … řC ż
4. écrire un programme permettant de retourner la température minimale de cette liste
5. créer une fonction minMax permettant de retourner les températures minimale et maximale
6. créer une fonction moyenneTemp retournant la température moyenne de ce relevé de température
7. créer une fonction moyenneDiurne retournant la température moyenne des températures diurnes

Correction exercice 1 :

1.

```
In [ ]: temperature = [12, 11, 11, 10, 9, 10, 9, 10, 12, 14, 16, 17, 18, 20, 21, 22, 23, 23, 23
```

2.

```
In [ ]:
        heure = 0
        for temp in temperature :
            print("à", heure,"h il fait", temp,"řC")
            heure += 1
```

variante :

```
In [ ]: for heure in range (len(temperature)) :
                print("à", heure,"h il fait", temperature[heure],"řC")
```

3.

```
In [ ]: tempMin = 15e6
        for temp in temperature :
                if temp < tempMin :
                    tempMin = temp
        print (tempMin)
```

4.

```
In [ ]: def minMax(temperature):
            tempMin = 15e6
            tempMax = -273
            for temp in temperature :
                    if temp < tempMin :
                        tempMin = temp
                    if temp > tempMax :
                        tempMax = temp
            return tempMin, tempMax
```

variante :

```
In [ ]: print ("la température minimale est", min(temperature),"řC et la température maximale
```

5.

```
In [ ]: def moyenneTemp(temperature):
        total = 0
        for temp in temperature :
                    total += temp
            return total/len(temperature)
```

6.

```
In [ ]: def moyenneDiurne(temperature):
            total = 0
            dureeJour=1
            for temp in range(6, 22) :
                total += temperature[temp]
                dureeJour += 1
            return total/dureeJour

                def moyenneNocturne(temperature):
                            total = 0
                            dureeNuit=1
                            for temp in range(0, 6) :
                                        total += temperature[temp]
                                        dureeNuit += 1
                            for temp in range(22, 24) :
                                        total += temperature[temp]
                                        dureeNuit += 1
                            return total/dureeNuit
```

Exercice 2 : 1. créer une fonction maxPremiereFois retournant l'heure à laquelle la température atteint son maximum pour la première fois en utilisant une fonction déjà définie ? 2. créer une fonction estSupA20 retournant le nombre d'heures où la température est supérieure ou égale à 20řC ?

Correction exercice 2 : 1.

```
In [ ]: def minMax(temperature):
            tempMin = 15e6
            tempMax = -273
            for temp in temperature :
                    if temp < tempMin :
                        tempMin = temp
                    if temp > tempMax :
                        tempMax = temp
            return tempMin, tempMax

        def maxPremiereFois(temperature):
            tempMax = minMax(temperature)[1]
            heure = 0
            while temperature[heure] < tempMax :
                heure += 1
            return heure
```

2.

```
In [ ]: def estSupA20(temperature):
            nbHeure = 0
            for i in temperature :
                if temperature[i] > 20 :
                    nbHeure +=1
            return nbHeure
```

Exercice 3 : 1. Créer une nouvelle liste ń temperature2 ż pour laquelle le programme demande à l'utilisateur de remplir la liste de 0h à 11h 2. Ajouter un ensemble de valeurs à ń temperature2 ż 3. comparer jour 1 et 2 en utilisant les fonctions précédentes

Corrections à faire

2 Un projet envisageable : Une station météo (Présentation bloc 3)

2.1 Objectifs

Utilisation d'un microcontroleur pour piloter un circuit électronique - affichage de données via la console, via un écran - recueil de données - Sauvegarde des données - Exploitation des données (réinvestir le travail sur les listes) - ...

3 Présentation des capteurs

3.1 Capteurs analogiques et numériques

Un capteur analogique transforme une grandeur physique (pression, température, deformation, etc...) en une autre grandeur physique (en général électrique) facilement mesurable. Le signal reçu en sortie du capteur peut prendre une infinité de valeurs à l'intérieur de sa plage de mesures

Un capteur numérique rend, en sortie, une ou des valeurs finies : Tout Ou Rien, trains d'impulsions, un ensemble de valeurs discretes dont le nombre dépend de l'échantillonage.

(

3.2 Les capteurs utilisés dans la station météo (à améliorer) :

3.2.1 Température : LM35 DZ

Le capteur de température LM35 est un capteur analogique de température fabriqué par Texas Instruments. Il est extrêmement populaire en électronique, car précis, peu couteux, très simple d'utilisation et d'une fiabilité à toute épreuve. Le capteur de température LM35 DZ est quelque peu obsolète, c'est pourquoi on en trouve beaucoup dans les kits de démarrage Arduino. Il est capable de mesurer des températures allant de 0řC à +100řC . La sortie analogique du capteur est proportionnelle à la température. Il suffit de mesurer la tension en sortie du capteur pour en déduire la température. Pour chaque évolution de 1 degré Celsius correspond une varaition de la tension de +10mV dans le même sens.

La carte Arduino dispose d'un convertisseur analogique/numérique intégré et échantillonne la tension reçue du capteur en 10 bits, la carte pourra donc renvoyer des valeurs comprises entre 0 (qui correspondra à une tension nulle) et 1023 (ce qui correspondra à une tension de 5V).

3.2.2 Luminosité : Photorésistance

Une photorésistance est une résistance dont la valeur change en fonctionn de la quantité de lumière qu'elle reçoit : plus elle est éclairée, plus sa résistance est faible On l'intègre dans un montage diviseur de tension :

La tension mesurée se calcule par :

$$V_{out} = V_{in} \times \frac{R_1}{R_2 + R_1}$$

3.2.3 Humidité : DHT11

Ce capteur permet de mesurer un taux d'humidité relative dans une place de 20% à 80%. Il peut aussi mesurer des températures de 0řC à 100řC mais avec une précision bien moindre que le LM35 DZ, aussi, nous n'utiliserons pas cette fonction ici.

Deux de ces broches concernent son alimentation en 5V, la troisième nous renvoie les données du capteur (taux d'humidité, température)

4 La carte Arduino Uno (à améliorer)

L'Arduino est une plateforme de prototypage électronique open-source, basée d'une part sur du matériel et d'autre part sur un ensemble de logiciels faciles à utiliser. La caractéristique première d'une carte Arduino est le type de MCU ou micro-contrôleur dont elle est équipée Une carte Arduino communique avec son environnement par l'intermédiaire de ses broches d'entrées/sorties. Sur ces broches, des capteurs, dispositifs permettant de transformer une information de l'environnement en signal électrique et des actionneurs, dispositifs permettant de transformer un signal électrique en action mécanique ou lumineuse, vont être connectés. Par conséquent le nombre de broches disponible est un critère de choix important car il détermine le nombre de capteurs et d'actionneurs que l'on va pouvoir connecter. L'Arduino étant un ordinateur spécialisé dans la gestion de capteurs et d'actionneurs, c'est un programme qui va décider de la manière donc les capteurs et les actionneurs sont utilisés. C'est donc très différent de l'électronique traditionnelle où les fonctions qui relient les capteurs aux actionneurs, l'appui d'un bouton qui entraine l'allumage d'un DEL par exemple, sont déterminées ń en dur ż par le câblage

entre les composants, et par les composants eux-mêmes. Ici les fonctions sont déterminées par un programme. Par conséquent les fonctions peuvent être beaucoup plus élaborées.

5 Montage et programme (à améliorer)

Nous voulons recueillir les données des trois capteurs. Le montage à effectuer est le suivant : [station%20meteo.jpg]

5.0.1 Affichage des données dans la console Arduino

In [ ]:
```
#include <dht11.h>
#define DHT11PIN 3 // broche DATA -> broche 3

dht11 DHT11;

void setup()
{
  // Initialise la communication avec le PC
  Serial.begin(9600);

  //Améliore la précision en réduisant la plage de mesure
  analogReference(INTERNAL);


}
void loop()
{
  //TEMPERATURE

  //Mesure la tension sur la branche A0
  int valeur_brute = analogRead(A0);

  //Transforme la valeur brute en température (spécification du composant)
  float tension = valeur_brute * (1.05/1024.0);
  float temperature = tension * 100;
  int valeur_lum = analogRead(A1);
  DHT11.read(DHT11PIN);


  Serial.print("temps (ms) :");
  Serial.print(millis());
  Serial.print(" ");
  Serial.print("temperature (C) : ");
  Serial.print(temperature);
  Serial.print(" ");
  Serial.print("Luminosité (0-1023)")
  Serial.print(valeur_lum);
```

```
  Serial.print(" ");
  Serial.print("Humidité, %");
  Serial.println((float)DHT11.humidity, 2);



delay(1000);
}
```

5.0.2 On peut également afficher les données sur un écran LCD

In [ ]:
```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
#include <dht11.h>
#define DHT11PIN 4 // broche DATA -> broche 4

dht11 DHT11;
LiquidCrystal_I2C lcd(0x27,16,2);  // set the LCD address to 0x27 for a 16 chars and 2


void setup()
{
  // Initialise la communication avec le PC
  Serial.begin(9600);

  //Améliore la précision en réduisant la plage de mesure
  analogReference(INTERNAL);

  // initialise le lcd
  lcd.init();

}
void loop()
{
  //TEMPERATURE

  //Mesure la tension sur la branche A0
  int valeur_brute = analogRead(A0);

  //Transforme la valeur brute en température (spécification du composant)
  float tension = valeur_brute * (1/1024.0);

  //Transforme la tension en température (étalonnage possible en modifiant 100 (augment

  float temperature = tension * 100;

  //LUMINOSITE

  //Mesure l'intensité lumineuse sur la broche A1
```

```
  int valeur_lum = analogRead(A1);

  //HYGROMETRIE

  //Mesure l'hygrométrie seulement (le capteur de température du DHT11 n'est pas préci
  DHT11.read(DHT11PIN);

  //Affiche le temps en millisecondes
  Serial.print(millis());

  //Génère un espace pour séparer les données et les enregistrer façilement
  Serial.print(" ");

  //Affiche la température
  Serial.print(temperature);
  Serial.print(" ");

  //Affiche la luminosité
  Serial.print(valeur_lum);
  Serial.print(" ");

  //Affiche l'humidité
  Serial.println((float)DHT11.humidity);
  // AFFICHAGE LCD
  lcd.clear();
  lcd.backlight();
  lcd.setCursor(0,0);
  lcd.print("temp: ");
  lcd.setCursor(6,0);
  lcd.print(temperature);
  lcd.setCursor(12,0);
  lcd.print("C");
  lcd.setCursor(0,1);
  lcd.print("lum: ");
  lcd.setCursor(5,1);
  lcd.print(valeur_lum);
  lcd.setCursor(10,1);
  lcd.print("H: ");
  lcd.setCursor(13,1);
  lcd.print((float)DHT11.humidity,0);
  lcd.setCursor(15,1);
  lcd.print("%");

//Temps entre 2 mesures en millisecondes (à paramétrer)
delay(1000);
}
```

6 Récupérer les données - Communication Arduino-Python

Au moyen d'un programme en Python, on peut récupérer les données des capteurs. Pour cela, il faut faire lire au programme ce que la carte envoie sur le port serie. Le tableau final est enregistré sous la forme d'un fichier .csv

On va épurer le programme arduino de manière à ne lui faire afficher que des valeurs séparées par un espace : ### Programme Arduino

7 include <dht11.h>

8 define DHT11PIN 3 // broche DATA -> broche 3

dht11 DHT11;

void setup() { // Initialise la communication avec le PC Serial.begin(9600);

//Améliore la précision en réduisant la plage de mesure analogReference(INTERNAL);

} void loop() { //TEMPERATURE

//Mesure la tension sur la branche A0 int valeur_brute = analogRead(A0);

//Transforme la valeur brute en température (spécification du composant) float tension = valeur_brute * (1.05/1024.0); float temperature = tension * 100; int valeur_lum = analogRead(A1); DHT11.read(DHT11PIN);

Serial.print(millis()); Serial.print(" "); Serial.print(temperature); Serial.print(" "); Serial.print(valeur_lum); Serial.print(" "); Serial.println((float)DHT11.humidity, 2);

delay(1000); }

8.0.1 Programme Python

```
In [34]: import serial as sr     # bibliothèque permettant de reconnaître les ports
         import time                                  # permet de compter le temps

         port_serie = sr.Serial(port = "COM5", baudrate = "9600")
                                                  # définit le port (à trouver en bas à droite
                                                  # la fenêtre Arduine) ainsi que le taux de
                                                  # transfert

         liste_temperature = []                   # génère une liste de temperature
         liste_temps = []                         # génère une liste de temps
         liste_luminosite=[]                         # génère une liste de luminosite
         liste_humidite=[]                        # génère une liste de humidite

         nbMesures = int(input('Nombre de mesures souhaité :'))
                                                  # demande à l'utilisateur le nombre de mesure
         print('Patientez...')

         for i in range(nbMesures):               # prends les nbMesures premières valeurs
             val = port_serie.readline().split() # place dans la variable "val" les lignes déc
             print(val)                                  # contrôle d'acquisition
             try:                                 # permet de sauter à la valeur suivante en ca
```

```
                temps = float(val[0])           # temps prend la valeur du rang [0] de la lis
                temperature = float(val[1])        # temperature prend la valeur du rang [1]
                luminosite= float(val[2])        # luminosite prend la valeur du rang [2] de
                humidite = float(val[3])        # humidite prend la valeur du rang [3] de la
                liste_temperature.append(temperature)           # on ajoute temperature à la
                liste_temps.append(temps)                        # on ajoute temps à la liste
                liste_luminosite.append(luminosite)                # on ajoute luminosite à l
                liste_humidite.append(humidite)                        # on ajoute humidite à

            except:
                print("error")
                pass                            # sauf en cas d'erreur
        print('Acquisition terminée')
        port_serie.close()                      # on ferme le port série
        mesures=[[0,0,0,0]]*(len(liste_temps))
        for i in range (len(liste_temps)):        # génère un tableau regroupant toutes les d
            mesures[i]=[liste_temps[i], liste_temperature[i], liste_luminosite[i], liste_humi

        print(mesures)
```

```
Nombre de mesures souhaité :10
Patientez...
[b'73', b'24.10', b'97', b'0']
[b'2096', b'24.30', b'100', b'60']
[b'4119', b'24.51', b'33', b'60']
[b'6141', b'25.12', b'29', b'60']
[b'8165', b'25.63', b'29', b'60']
[b'10187', b'26.25', b'29', b'60']
[b'12211', b'26.76', b'29', b'60']
[b'14234', b'26.87', b'29', b'59']
[b'16257', b'31.89', b'39', b'59']
[b'18280', b'27.48', b'39', b'62']
Acquisition terminée
[[73.0, 24.1, 97.0, 0.0], [2096.0, 24.3, 100.0, 60.0], [4119.0, 24.51, 33.0, 60.0], [6141.0, 25
```

On peut vouloir sauvegarder les données dans un fichier .csv en vue d'une exploitation ultérieure

```
In [41]: import pandas as pd # bibliothèque permettant de lire et d'enregistrer des fichiers .
         df = pd.DataFrame(mesures, columns=["temps", "temperature", "luminosite", "humidite"])
         df.to_csv("meteo.csv", index=False) #enregistre dans un fichier meteo.csv
```

Le fichier .csv peut être lu dans un tableur grapheur comme regressi, excel...
Le fichier csv se présente sous la forme suivante : temps,temperature,luminosite,humidite
1134.0,35.64,320.0,0.0
2201.0,27.93,357.0,0.0
3268.0,27.93,367.0,0.0
4335.0,27.93,353.0,0.0

5401.0,27.83,457.0,0.0
6468.0,28.03,444.0,0.0
7535.0,27.93,412.0,0.0
8602.0,27.93,460.0,0.0
9668.0,27.93,434.0,0.0
10736.0,27.93,416.0,0.0

Les grandeurs sont séparées, ici, par des virgules. Suivant l'outil utilisé pour créer le fichier .csv, les séparateurs peuvent être divers : virgule, point virgule, point, tabulation (/t), ... On peut ouvrir le fichier csv dans un éditeur de texte afin d'observer le type de séparateur entre les données.

9 Extraction des données d'un fichier .csv

On se propose, à partir du fichier .csv généré précédemment, de représenter l'évolution dans le temps des diverses grandeurs mesurées.

Dans un premier temps, on extrait chacune des grandeurs temps, temperature, luminosité et humidité du fichier csv.

```
In [42]: import pandas as pd #permet de lire ou d'enregistrer des fichiers .csv
         nom_fichier = 'meteo.csv'                  # Attention le fichier .csv doit être placé
         data = pd.read_csv(nom_fichier, sep = ',') # ici on place les données du fichier .csv
         print(type(data))                           #  un dataframe est un tableau dans lequel
         data.head()
         temps = data['temps']/1000
         temperature = data['temperature']
         luminosite = data['luminosite']
         humidite= data['humidite']
         print(temperature)
         print(luminosite)
         print(humidite)
         print(temps)
```

```
<class 'pandas.core.frame.DataFrame'>
0    24.10
1    24.30
2    24.51
3    25.12
4    25.63
5    26.25
6    26.76
7    26.87
8    31.89
9    27.48
Name: temperature, dtype: float64
0     97.0
1    100.0
2     33.0
3     29.0
```

```
4     29.0
5     29.0
6     29.0
7     29.0
8     39.0
9     39.0
Name: luminosite, dtype: float64
0     0.0
1    60.0
2    60.0
3    60.0
4    60.0
5    60.0
6    60.0
7    59.0
8    59.0
9    62.0
Name: humidite, dtype: float64
0     0.073
1     2.096
2     4.119
3     6.141
4     8.165
5    10.187
6    12.211
7    14.234
8    16.257
9    18.280
Name: temps, dtype: float64
```

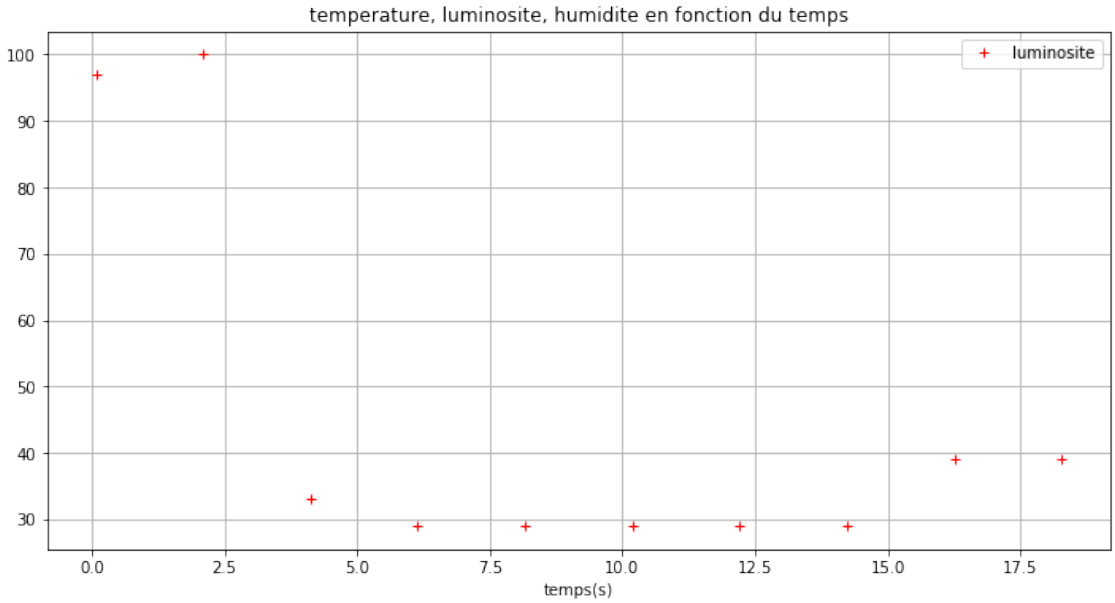
10 Représentation graphique de l'évolution des grandeurs mesurées dans le temps

A partir des listes générées au moyen du fichier .csv, on peut représenter l'évolution de n'importe quelle grandeur en fonction d'une autre.

```
In [46]: import matplotlib.pyplot as plt
         plt.rcParams['figure.figsize'] = [12,6]
         #plt.plot(temps,temperature,'bo', label='temperature')
         plt.plot(temps,luminosite,'r+', label='luminosite')
         #plt.plot(temps,humidite, 'gx', label='humidite')
         plt.legend()
         plt.grid()
         plt.xlabel("temps(s)")

         plt.title("temperature, luminosite, humidite en fonction du temps")
```

```
plt.show()
```


temperature, luminosite, humidite en fonction du temps
luminosite
100
90
80
70
60
50
40
30
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
temps(s)

In [ ]: