3.4.5 La programmation orientée objet (classes et objets)

La programmation orientée objet (ou POO en abrégé) est une autre manière de construire et d'organiser son code.

Python est un langage orienté objet, ce qui signifie que le langage tout entier est construit autour de la notion d'objets.

Par exemple, les types **str**, **int**, **list**,... sont des objets, les fonctions sont des objets, etc…

La programmation orientée objet repose sur le concept d'objets qui sont des blocs de code qui possède un ensemble de variables (appelées **attributs** en python) et de fonctions qui leur sont propres (appelées **méthodes** en python).

Les objets sont créés à partir de "modèles" appelés **classes** qui sont également des ensembles de codes qui contiennent des variables et des fonctions. Les objets créés possèdent alors un même ensemble d'attributs et de méthodes que les classes, les attributs étant des variables accessibles depuis toute méthode de la classe où elles sont définies.

On dit qu'un objet est une instance d'une classe. On peut créer autant d'objets que l'on désire avec une classe.

. Les classes

Une classe est équivalente à un type de données et créer une nouvelle classe en Python correspond à définir un nouveau type de données.

Sans le savoir, nous avons déjà vu des classes :

- le type de donnée **str** est une classe dont l'objet **ch = "chaine "** est une instance et par exemple, la fonction **upper()** est une de ses méthodes,

- le type de donnée **list** est une classe dont l'objet **liste=[1,2,3,4]** est une instance et par exemple, la fonction **sort()** est une de ses méthodes,

- …

Pour créer une nouvelle classe Python, on utilise le mot clef **class** suivi du nom de la classe.

Nous allons créer une classe nommée **inventaire** à partir du programme d'inventaire déjà vu en tant qu'exemple d'application de la manipulation des fichiers.

```
class inventaire:
    def __init__(self):
        self.invent={}
        self.inventpath=""
        self.finprog=False
```

Les instructions ci-dessus crée une classe nommée **inventaire** dont les attributs sont :

. un dictionnaire nommé **self.invent** initialement vide,

. une chaine de caractères nommée **self.inventpath** correspondant au chemin de l'inventaire initialement vide,

. une variable booléenne nommée **self.finprog** initialisée en **False** afin de pouvoir mettre fin à l'exécution du programme.

La méthode **_init_(self)** dans laquelle les attributs sont définis est appelée lors de la création d'un objet à partir de la classe. Cette méthode est nommée un **constructeur**.

Pour créer un objet à partir de la classe **inventaire**, il suffira dans le programme principal d'appeler la classe à l'aide de l'instruction : **nom_objet = inventaire()**

La méthode **_init_(self)** est alors appelée et les variables du constructeur sont attribuées à l'objet créé.

La méthode prend en paramètre la variable **self**. Cette variable représente l'objet lui-même (d'où le nom **self**…) et c'est pourquoi les attributs sont de type : **self.variable** (car ce sont les variables de l'objet !).

De cette façon, il n'y a plus besoin de **variables globales** dans des fonctions du programme ayant besoin de modifier des variables externes à la fonction (les attributs **self.variable** étant des variables accessibles depuis toute méthode de la classe).

Après la définition des attributs, il faut définir les méthodes de la classe **inventaire** :

- Méthode pour ouvrir un inventaire :

```
def OuvreInventaire(self):
    self.InventPath=input("Indiquez le nom de l'inventaire:")
    self.invent = {}
    try:
        with open(self.InventPath, 'r') as fichier:
            for line in fichier:
                listline=line.split(";")
                self.invent[listline[0]]=listline[1].strip()

    except Exception as message:
        print(message)
```

Cette méthode demande à l'utilisateur le chemin d'un fichier d'inventaire (**self.InventPath**) et tente de l'ouvrir. Un message est affiché si le fichier n'existe pas, sinon l'inventaire est chargé dans le dictionnaire **self.invent**.

- Méthode pour ajouter un élément à l'inventaire :

```
def AjoutMatos(self):
    Matos=input("saisissez le type de matériel à ajouter à l'inventaire:")
    Quant=input("saisissez la quantité de ce matériel:")
    self.invent[Matos]=Quant
```

Cette méthode demande à l'utilisateur de saisir les données (nom et quantité) de l'élément à ajouter à l'inventaire et l'ajoute au dictionnaire **self.invent**.

- Méthode pour effacer un élément de l'inventaire :

```
    def EffaceMatos(self):
        element=input("saisissez l'élément à supprimer dans l'inventaire:")
        try:
            del self.invent[element]
        except:
            print("L'élément que vous voulez supprimer n'existe pas!")
```

Cette méthode demande à l'utilisateur de saisir le nom de l'élément à supprimer de l'inventaire et le supprime du dictionnaire **self.invent**.

- Méthode pour afficher l'inventaire :

```
    def ReadInventaire(self):
        for cle, valeur in self.invent.items():
            print("{} : {}".format(cle, valeur))
```

Cette méthode affiche la liste des éléments de l'inventaire **self.invent** dans la fenêtre Python shell.

- Méthode pour sauvegarder l'inventaire :

```
    def SaveInventaire(self):
        self.InventPath=input("Indiquez le nom de sauvegarde de l'inventaire:")
        with open(self.InventPath, 'w') as fichier:
            for cle, valeur in self.invent.items():
                fichier.write("{};{}".format(cle, valeur))
                fichier.write("\n")
```

Cette méthode demande à l'utilisateur de saisir le chemin d'enregistrement de l'inventaire (**self.InventPath**). L'inventaire est parcouru et sauvegarder dans le fichier ouvert.

- Méthode pour déterminer l'action à exécuter :

```
    def ChoixAction(self):
        print("/////////////// INVENTAIRE MATERIEL ///////////////")
        print("appuyer sur O pour ouvrir un inventaire:")
        print("appuyer sur A pour ajouter un matériel à l'inventaire:")
        print("appuyer sur S pour supprimer un matériel de l'inventaire:")
        print("appuyer sur V pour afficher la liste de matériel:")
        print("appuyer sur E pour enregistrer l'inventaire:")
        print("appuyer sur Q pour quitter:")
        print("////////////////////////////////////////////////")
        print(" ")

        while self.finprog == False:
            choix = " "
            while choix.upper()!="A" or "S" or "Q" or "P" or "O" or "E":
                choix=input()
```

```
            # Action en fonction de l'entrée clavier:
            if choix.upper()=="O": self.OuvreInventaire()
            if choix.upper() == "A": self.AjoutMatos()
            if choix.upper() == "V": self.ReadInventaire()
            if choix.upper() == "S": self.EffaceMatos()
            if choix.upper()=="E": self.SaveInventaire()
            if choix.upper() == "Q":
                print("Fin du programme")
                self.finprog = True
                break
```

Cette méthode affiche la liste des actions disponibles et demande en boucle à l'utilisateur de faire un choix.

Les attributs et les méthodes de la classe **inventaire** ont maintenant tous été définis. Le programme va créer un objet nommé **mon_inventaire** qui est une instance de la classe **inventaire** :

mon_inventaire=inventaire()

Puis on appelle la méthode **ActionChoix()** de l'objet créé pour lancer la boucle des actions :

mon_inventaire.ChoixAction()

Résultats dans le fenêtre Python Shell :

```
============ RESTART: D:\Travail\ArdPyLab\Docs\Python\inventclass.py ===========
/////////////// INVENTAIRE MATERIEL ///////////////
appuyer sur O pour ouvrir un inventaire:
appuyer sur A pour ajouter un matériel à l'inventaire:
appuyer sur S pour supprimer un matériel de l'inventaire:
appuyer sur V pour afficher la liste de matériel:
appuyer sur E pour enregistrer l'inventaire:
appuyer sur Q pour quitter:
////////////////////////////////////////////////

o
Indiquez le nom de l'inventaire:invent2
v
bécher : 10
erlenmeyer : 20
a
saisissez le type de matériel à ajouter à l'inventaire:eprouvette
saisissez la quantité de ce matériel:5
e
Indiquez le nom de sauvegarde de l'inventaire:invent2
v
bécher : 10
erlenmeyer : 20
eprouvette : 5
o
Indiquez le nom de l'inventaire:invent
v
bécher : 10
eprouvette : 15
o
Indiquez le nom de l'inventaire:invent2
v
bécher : 10
erlenmeyer : 20
eprouvette : 5
q
Fin du programme
```

Voici Le programme complet avec la classe **inventaire** qui pourra bien-sûr se situé dans un module à part qui le cas échéant devra être importé avant utilisation :

inventclass.py - D:\Travail\ArdPyLab\Docs\Python\inventclass.py (3.7.7)

File Edit Format Run Options Window Help

```
class inventaire:
    def __init__(self):
        self.invent={}
        self.inventpath=""
        self.finprog=False

    def OuvreInventaire(self):
        self.InventPath=input("Indiquez le nom de l'inventaire:")
        self.invent = {}
        try:
            with open(self.InventPath, 'r') as fichier:
                for line in fichier:
                    listline=line.split(";")
                    self.invent[listline[0]]=listline[1].strip()

        except Exception as message:
            print(message)

    def AjoutMatos(self):
        Matos=input("saisissez le type de matériel à ajouter à l'inventaire:")
        Quant=input("saisissez la quantité de ce matériel:")
        self.invent[Matos]=Quant

    def EffaceMatos(self):
        element=input("saisissez l'élément à supprimer dans l'inventaire:")
        try:
            del self.invent[element]
        except:
            print("L'élément que vous voulez supprimer n'existe pas!")

    def ReadInventaire(self):
        for cle, valeur in self.invent.items():
            print("{} : {}".format(cle, valeur))

    def SaveInventaire(self):
        self.InventPath=input("Indiquez le nom de sauvegarde de l'inventaire:")
        with open(self.InventPath, 'w') as fichier:
            for cle, valeur in self.invent.items():
                fichier.write("{};{}".format(cle, valeur))
                fichier.write("\n")
    def ChoixAction(self):
        print("/////////////// INVENTAIRE MATERIEL ///////////////")
        print("appuyer sur O pour ouvrir un inventaire:")
        print("appuyer sur A pour ajouter un matériel à l'inventaire:")
        print("appuyer sur S pour supprimer un matériel de l'inventaire:")
        print("appuyer sur V pour afficher la liste de matériel:")
        print("appuyer sur E pour enregistrer l'inventaire:")
        print("appuyer sur Q pour quitter:")
        print("////////////////////////////////////////////////")
        print(" ")

        while self.finprog == False:
            choix = " "
            while choix.upper()!="A" or "S" or "Q" or "P" or "O" or "E":
                choix=input()

                # Action en fonction de l'entrée clavier:
                if choix.upper()=="O": self.OuvreInventaire()
                if choix.upper() == "A": self.AjoutMatos()
                if choix.upper() == "V": self.ReadInventaire()
                if choix.upper() == "S": self.EffaceMatos()
                if choix.upper()=="E": self.SaveInventaire()
                if choix.upper() == "Q":
                    print("Fin du programme")
                    self.finprog = True
                    break

mon_inventaire=inventaire()
mon_inventaire.ChoixAction()
```

Tout ce qui a été vu concernant la programmation en Python ne représente que des bases, mais cela va nous permettre d'aborder le thème suivant à savoir la communication entre un Arduino Uno et un programme Python à des fins de contrôle et d'exploitation de données.