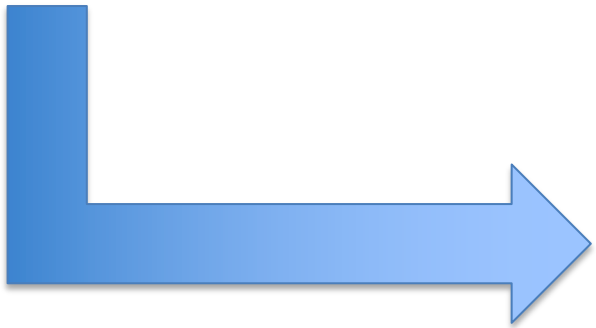
Création d'une librairie en Python.

```
class libPythonClasse():
    def addition(self,a,b):
        return  int(a)+int(b)
```



```
*** Settings ***
Library           libPythonClasse

*** Test Cases ***
TP_addition
    ${res}    addition    0    6
    Should Be Equal As Integers   ${res}     6
```

Il faut indiquer le chemin dans la librairie : PYTHONPATH
Robot –P <chemin> <nom du fichier>

Création d'une librairie

- On peut avoir plusieurs classe dans le même fichier:

```python
class MaLibrairie():
    def calcul_tarif(self,age):
        if int(age) > 12 :
            return  "plein tarif"
        else :
            return "demi tarif"
class MaLibrairie2():
    def calcul_tarif2(self,age):
        if int(age) > 12 :
            return  "plein tarif"
        else :
            return "demi tarif"
```

```
*** Settings ***
Documentation   Ma suite qui fait pas grand chose
Library  LibCalcul.MaLibrairie

*** Test Cases ***
Util LibCalcul
  ${x}  Calcul Tarif   20
  Should Be Equal  ${x}  plein tarif
```

- Library <nom du fichier>.<nom de la classe>

Création d'une librairie

- Inclure des paramètres à la librairie

```
class MaLibrairie():
    limit=12
    def __init__(self, ageLimit=12):
        self.limit = int(ageLimit)
    def calcul_tarif(self,age):
        if int(age) > self.limit :
            return  "plein tarif"
        else :
            return "demi tarif"
```

- L'import de la librairie peut se faire avec un paramètre age.

Création d'une librairie

- Renommage des mots clés

```
from robot.api.deco import keyword, library
class MaClasse:
    ROBOT_AUTO_KEYWORDS = False
    @keyword('Nouveau Nom')
    def fct1(self,Val1,Val2):
        return Val1 + ' ' + Val2
    def fct2(self,Val1,Val2):
        return Val1 + '-' + Val2
```

Seules les méthodes avec l'annotation seront considérées comme des keywords

Le keyword est renommé

Fct2 ne sera pas un mot clef

Création d'une librairie

- Renommage des mots-clés : robot_name

```
class MaLibrairie():
    ROBOT_LIBRARY_SCOPE = 'SUITE'
    limit=12
    def __init__(self, ageLimit=12):
        self.limit = int(ageLimit)
    def calcul_tarif(self,age):
        if int(age) > self.limit :
            return  "plein tarif"
        else :
            return "demi tarif"
    calcul_tarif.robot_name= "nom"
```

Création d'une librairie

- Multiple arguments:

```
def any_arguments(*args):
    print("Got arguments:")
    for arg in args:
        print(arg)
```

- Utilisation dans un test

```
Util plusieurs arguments
   any_arguments   un   deux   trois
```

Création d'une librairies

- Gestion du typage:

```
def calcul_tarif(self,age):
    if int(age) > self.limit :
        return  "plein tarif"
    else :
        return "demi tarif"
```

- Équivalent à:

```
def calcul_tarif(self,age:int):
    if age > self.limit :
        return  "plein tarif"
    else :
        return "demi tarif"
```

ou:

```
@keyword(types={'age': int})
def calcul_tarif(self,age):
    if age > self.limit :
        return  "plein tarif"
    else :
        return "demi tarif"
```

Création d'une librairie

- On peut décider qu'à l'exécution une seule instance de la librairie sera créée pour:
  - Tous les tests : GLOBAL
  - A chaque suite de test : SUITE
  - A chaque test : TEST

```
class MaLibrairie():
    ROBOT_LIBRARY_SCOPE = 'SUITE'
    limit=12
    def __init__(self, ageLimit=12):
        self.limit = int(ageLimit)
    def calcul_tarif(self,age):
        if int(age) > self.limit :
            return  "plein tarif"
        else :
            return "demi tarif"
    calcul_tarif.robot_name= "nom"
```

Création d'une librairie

- On peut définir la version:

```
class MaLibrairie2():
    __version__ = '0.1'
```

- Utilisation de l'annotation `@library`

```
@library(scope='GLOBAL', auto_keywords=False)
class MaLibrairie2():
    __version__ = '0.1'
    limit=12
    def __init__(self, ageLimit):
        self.limit = int(ageLimit)
    @keyword('Nouveau Nom')
    def calcul_tarif2(self,age):
        if int(age) > self.limit :
            return  "plein tarif"
        else :
            return "demi tarif"
```

Création d'une librairie

- Communiquer avec robotframework
- exceptions possibles
  - Failure
  - Error
  - ContinuableFailure
  - SkipExecution
  - FatalError

```
from robot.api import SkipExecution


def example_keyword(self):
    raise SkipExecution('Cannot proceed, skipping test.')
```

Création d’une librairie

- Les bibliothèques python peuvent être packagées pour être installées à l’aide d’une commande de type pip install.
- Cela peut être utile si la bibliothèque a un périmètre plus large que le projet lui-même.