Écriture d'une Fonction d'Évaluation

La fonction *evaluator()* est la fonction qui sera appelé avec la réponse de l'élève afin de la corriger.

La fonction étant déclarer en **python 3**, il est nécessaire d'avoir des connaissances en python.

La fonction *evaluator()*, reçoit, en plus de la réponse de l'élève, le dictionnaire de l'exercice après un passage par la fonction *build()*, et peut donc se servir de l'ensemble des variables déclarées/créées afin de corriger l'élève.

Les Bases

La fonction *evaluator()* doit être écrit en **python 3** et respecter un prototype précis:

```
def evaluator(reponse, dic):
    [...]
    return [True/False], feedback
```

Où *réponse* est le dictionaire contenant l'ensemble des réponses (voir Écrire un Formulaire), et *dic* est le dictionnaire de l'exercice qui contient donc l'ensemble des variables déclaré dans le fichier PL associé.

La fonction doit retourner un tuple contenant un booléen indiquant la réussite de l'élève ainsi qu'un *feedback* qui sera affiché à l'élève (en vert si le booléen est Vrai, rouge si Faux).

Elle doit être déclarée dans le PL grâce à la clé *evaluator*:

```
evaluator==
def evaluator(reponse, dic):
    [...]
    return [True/False], feedback
==
```

Modules et Fonctions Secondaires

N'importe quel module, (excepté *os* et *sys*) peut être importé avant la déclaration de build, de même, plusieurs fonctions annexes peuvenr être déclarées avant la fonction *evaluator()* et être appelée par celle-ci:

```
evaluator==
import math

def is_sqrt(i, j):
```

```
    return math.pow(i,2) == j

def evaluator(reponse, dic):
    if is_sqrt(reponse['answer'], int(dic['n'])):
        return True, "Bonne réponse"
    else:
        return False, "Mauvaise réponse, "+str(reponse['answer'])+"*"+str(reponse['answer'])
==
```

Fonctions d'Évaluation Avancées

Il est ausi possible de déclarer/écraser des clés de l'exercice dans l'évaluator. Cela peut par exemple être utile si on veut par exemple modifié le formulaire de réponse (surligné des champs en vert/rouge, ajouté des indices, etc...).

```
evaluator==
import math

form_fail = """
<div class="input-group">
    <red><span class="input-group-addon">Réponse</span></red>
    <input id="form_txt_answer" type="number" class="form-control" placeholder="" required>
</div>
"""

def is_sqrt(i, j):
    return math.pow(i,2) == j

def evaluator(reponse, dic):
    if is_sqrt(reponse['answer'], int(dic['n'])):
        return True, "Bonne réponse"
    else:
        dic['form'] = form_fail
        return False, "Mauvaise réponse, "+str(reponse['answer'])+"*"+str(reponse['answer'])
==
```

Exemples

Voici divers exemples de fonction *evaluator()*:

Vrai / Faux

```
evaluator==
def evaluator(reponse, dic):
```

```
    if (str(dic['answer']) == reponse['answer']):
        return True, str(dic['feedback_correct'])
    return False, str(dic['feedback_wrong'])
==
```

QCM

```
evaluator==
def evaluator(reponse, dic):
    if not 'answer' in reponse:
        return False, "Merci de cocher au moins une case"
    if len(reponse['answer']) == len(dic['answer']):
        for answer in dic['answer']:
            if not answer in reponse['answer']:
                return False, "Réponse incorrect"
        return True, "Bonne réponse"
    return False, "Réponse incorrect"
==
```

Match

```
evaluator==
def evaluator(reponse, dic):
    if (reponse == dic['answer']):
        return True, "Bien joué"
    return False, "Mauvais matching"
==
```