

# Utilisation de Caseine et Visual Studio Code dans l'UE MMI Info

Margaux Nattaf

## 1 Virtual Programming Labs (VPL) sur Caseine

Caseine est une plateforme d'apprentissage en ligne qui comporte un environnement de programmation avec un système d'auto-évaluation. Vous y trouverez un ensemble d'exercices qui permettent de s'entraîner sur les notions vues en TD. Il vous est fortement conseillé de travailler en autonomie sur ces exercices avant de venir en TP. Les TP visent en effet à résoudre des problèmes construits et complexes, et non pas à s'entraîner sur des tâches basiques.


Bien que très utile, l'environnement de programmation en ligne proposé par Caseine est très basique (pas d'auto-complétion, pas de correction de syntaxe, pas de debug). Si vous souhaitez développer, faites-le plutôt sous Visual Studio Code (VSCode). La section 2 présente l'utilisation de Caseine depuis VSCode. On retrouvera sur VSCode les mêmes options que décrites ci-dessous.

### 1.1 Edition d'un VPL

1. Accédez au cours d'informatique sur CaseInE.
2. Il y a une section consacrée à chaque TD (aussi appelé séquence) dans laquelle vous trouvez une série d'exercices à faire après chaque TD. Le nombre d'étoiles est une indication de la difficulté.
3. Choisissez un exercice ; ici nous prenons «Premières fonctions» (Prise en main : Caseine). Vous voyez un aperçu des fichiers qu'il vous faut compléter. Pour les compléter, choisissez l'onglet «Edit».



### 1.2.2 Evaluation et vérification de votre code

Après avoir sauvegardé («Save»), cliquez sur «Evaluate»  exécute les tests prédéfinis par les enseignants, et affiche un score sur 100. Le panneau sur la droite vous fournit des informations sur votre code. Les indications sont réparties selon les catégories suivantes :

- Compilation : relève les erreurs graves, notamment de syntaxe telle qu'au moins une partie de votre code n'a aucun sens en python. Si un problème est relevé à ce niveau-là, l'exécution s'arrête. En effet, une seule erreur de ce type, est c'est l'ensemble de votre code qui devient incompréhensible. La Figure 1 montre un exemple de problème lié à la syntaxe. Ici, le problème vient de l'indentation du dernier `return`.

```
## La TVA est de 5,5% si réduit est vrai, 19,6% sinon.  
def tva(prix: float, réduit: bool) -> float:  
    """ Renvoie le montant de la TVA pour le prix (hors-tax) indique."""  
    if réduit: return 0.55*prix  
    else:  
return 0.196*prix
```

>>> COMPILATION: 1 error  
student.py:16:0: E0001: expected an indented block (<unknown>, line 16) (syntax-error)  
Fatal errors -- abort evaluation

FIGURE 1 – Problème de compilation

- Types : relève les erreurs liées aux types, i.e. types de retour, types des arguments, types des paramètres... Votre code peut fonctionner même s'il y a des erreurs dans cette partie là. Cependant, si vous n'êtes pas capable de corriger les problèmes liés à cette section, il y a de fortes chances pour que votre code soit faux ! Un problème relevé ici implique une pénalité sur votre note finale. La Figure 2 montre un exemple de problème lié aux types de retour de la méthode. En effet, d'après la spécification de votre fonction, elle devrait renvoyer un entier mais renvoie un nombre flottant.

```
## La TVA est de 5,5% si réduit est vrai, 19,6% sinon.  
def tva(prix: float, réduit: bool) -> int:  
    """ Renvoie le montant de la TVA pour le prix (hors-tax) indique."""  
    if réduit: return 0.55*prix  
    else: return 0.196*prix
```

>>> TYPES: 2 errors  
student.py:15: error: Incompatible return value type (got "float", expected "int")  
student.py:16: error: Incompatible return value type (got "float", expected "int")  
Penalty: 10 (2 x 5, limited to 20)

FIGURE 2 – Problème de types

- Tests : c'est la partie où votre code est exécuté et testé (les tests sont écrits par vos enseignants). Ici, il s'agit de vérifier que vos méthodes font bien ce qui est demandé. La Figure 3 montre un exemple de problème lié aux tests. En effet, la méthode `tva` ne semble pas renvoyer le bon résultat lorsqu'on l'appelle avec les paramètres 100 et `True`.

>>> TESTS (80):  
est\_positif (test 1)  
tva (test 2)  
Incorrect program result  
AssertionError: Err. pour tva(100,True)  
min2 (test 3)  
min3 (test 4)  
somme (test 5)

FIGURE 3 – Problème lié aux tests

- Bonnes pratiques : cette partie permet de vérifier que votre code est sain, i.e. tout simplement que vous n'avez pas de pratiques dangereuses dans votre façon de coder. Un problème relevé ici implique une petite pénalité sur votre note finale. La Figure 4 montre un exemple de problème lié aux bonnes pratiques de codage. En effet, deux choses posent problème dans ce code et les deux sont liées à la variable `tva`. D'une part, cette

variable n'est jamais utilisée dans votre code, elle n'est donc pas nécessaire et d'autre part, elle porte le même nom que la méthode dans laquelle elle est définie!!! Ce dernier problème peut amener à des comportements plus que surprenant...

```
## La TVA est de 5,5% si reduit est vrai, 19,6% sinon.
def tva(prix: float, reduit: bool) -> float:
    """ Renvoie le montant de la TVA pour le prix (hors-tax) indique. """
    tva: float = 0
    if reduit: return 0.55*prix
    else: return 0.196*prix
```

```
>>> GOOD PRATICES: 2 errors
student.py:14:4: W0621: Redefining name
'tva' from outer scope (line 12) (redefined-
outer-name)
student.py:14:4: W0612: Unused variable
'tva' (unused-variable)

Your code has been rated at 9.17/10

Penalty: 2 (2 x 1, limited to 5)
```

FIGURE 4 – Problème lié aux bonnes pratiques

- Qualité du code : cette partie permet de vérifier la qualité de votre code. En effet, pour pouvoir travailler à plusieurs en informatique et notamment lors d'énormes projets (imaginer vos applications préférées), des conventions d'écritures sont nécessaires. Si vous ne respectez pas ces conventions, votre code pourra fonctionner mais sera plus difficile à utiliser par une personne tierce (voir projet info au 2<sup>ème</sup> semestre).

Dans cette partie, les remarques seront souvent (pas nécessairement) faites à titre informatif.

La Figure 5 montre un exemple de problème lié aux conventions d'écriture. En effet, dans le code de la méthode `est_positif`, il n'y a aucun espace ce qui rend la lecture plus difficile. De plus, le code pourrait être simplifié puisque on pourrait directement écrire `return n>=0`.

```
>>> CODE QUALITY: 5 errors
student.py:5:17: warning C0326: Exactly
one space required after :
def est_positif(n:int)->bool:
^ (bad-whitespace)
student.py:6:8: warning C0326: Exactly one
space required around comparison
if n>=0:return True
^^ (bad-whitespace)
student.py:6:11: warning C0326: Exactly
one space required after :
if n>=0:return True
^ (bad-whitespace)
```

```
student.py:7:8: warning C0326: Exactly one
space required after :
else:return False
^ (bad-whitespace)
student.py:6:4: warning R1703: The if
statement can be replaced with 'return
bool(test)' (simplifiable-if-statement)
```

```
def est_positif(n:int)->bool:
    if n>=0:return True
    else:return False
```

FIGURE 5 – Problème lié aux conventions


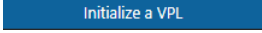
## 2 Plug-in Caseine sur VSCode

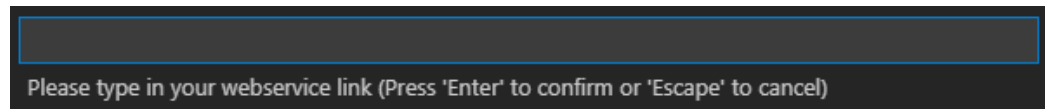
Une autre manière de travailler (et que l'on utilisera obligatoirement dans ce cours) est de travailler directement sur un environnement de développement intégré. Dans le cadre de ce cours, nous utiliserons VSCode car il a l'avantage de posséder un plug-in Caseine. Dans cette partie, nous vous montrons comment utiliser le plug-in Caseine sur VSCode, VSCode étant directement disponible sur Citrix. Si vous souhaitez installer VSCode sur votre machine personnelle, reportez vous à l'annexe A.

1. Allez sur Citrix Receiver ou allumer une machine de l'école.
2. Lancez Visual Studio Code (vous pouvez faire une recherche dans la barre située en haut à droite).
3. Une fois VSCode ouvert, vous devez ouvrir un dossier (dans lequel sera sauvegarder votre travail). Deux possibilités :
  - Allez dans l'explorateur de fichiers de Windows et créez le dossier à la main. Ensuite, l'ouvrir en passant par File -> Open Folder.

— Dans VSCode, cliquez sur File -> Open Folder. Une fenêtre s'affiche dans laquelle vous pouvez choisir le dossier dans lequel vous voulez travailler. Vous pouvez créer de nouveaux dossiers à tout moment en faisant un clic-droit sur la fenêtre puis Nouveau -> Dossier.

**ATTENTION ! Vous devrez ouvrir un nouveau dossier pour chaque activité sur Caseine !**

4. Dans la barre de gauche de l'IDE (VSCode), cliquez sur la brique de lait  puis sur *Initialize a VPL* : . Une barre s'ouvre alors vous demandant le lien du web service :



Nous allons aller chercher cette information sur Caseine.

5. Sur Caseine, cliquez sur le lien de l'activité VPL que vous voulez réaliser. Vous devez alors être redirigé sur l'onglet **Description** de cette activité. Tout en bas de la page vous trouverez un lien appelé *Web service* (voir Figure 6).

Cliquez sur ce lien (obligatoire). Une page s'ouvre contenant plusieurs adresse. Copiez l'adresse

```
31
32  ## Ce qui suit est le "programme principal", c'est-a-dire le point d'entree
33  ## de votre code. C'est la que vous pouvez utiliser vos fonctions et en
34  ## particulier ecrire vos tests.
35  if __name__ == "__main__": # NE PAS SUPPRIMER CETTE LIGNE
36      print(absolute(5)) # 5 ## un exemple de test, avec l'affichage attendu
37      print(absolute(-5)) # 5 ## autre exemple ; il faut essayer d'etre exhaustif
38      print(absolute(0)) # 0 ## sans oublier les cas limites
```

## Web service


◀ Micro-programme

Jump to...

FIGURE 6 – Lien Web service

"Full URL" de la section "Global VPL webservice" (voir Figure 7). Collez cette adresse dans la barre demandant le lien web service de VSCode.

ATTENTION ! Cette adresse est différente pour chaque activité et chaque personne.

6. Allez dans l'onglet *Explorer*  (barre latérale gauche). Vous devriez voir les fichiers requis de l'activité. Vous pouvez les ouvrir, les modifier... (cf. Figure 8). Si vous ne voyez pas les fichiers, c'est probablement que vous avez copié le mauvais lien web service. Avez vous pensé à cliquer sur le lien ?

## Global VPL webservice

Personal token: [redacted]

Webservice URL: <https://moodle.caseine.org/webservice/rest/server.php>

Full URL: <https://moodle.caseine.org/webservice/rest/server.php?moodlewsrestformat=json&id=27239&wstoken=.....&wsfunction=>

## Local VPL webservice

Personal token: [redacted]

Webservice URL: <https://moodle.caseine.org/mod/vpl/webservice.php>

Full URL: <https://moodle.caseine.org/mod/vpl/webservice.php?moodlewsrestformat=json&id=27239&wstoken=.....&wsfunction=>

Continue

FIGURE 7 – URL à mettre dans VSCode.

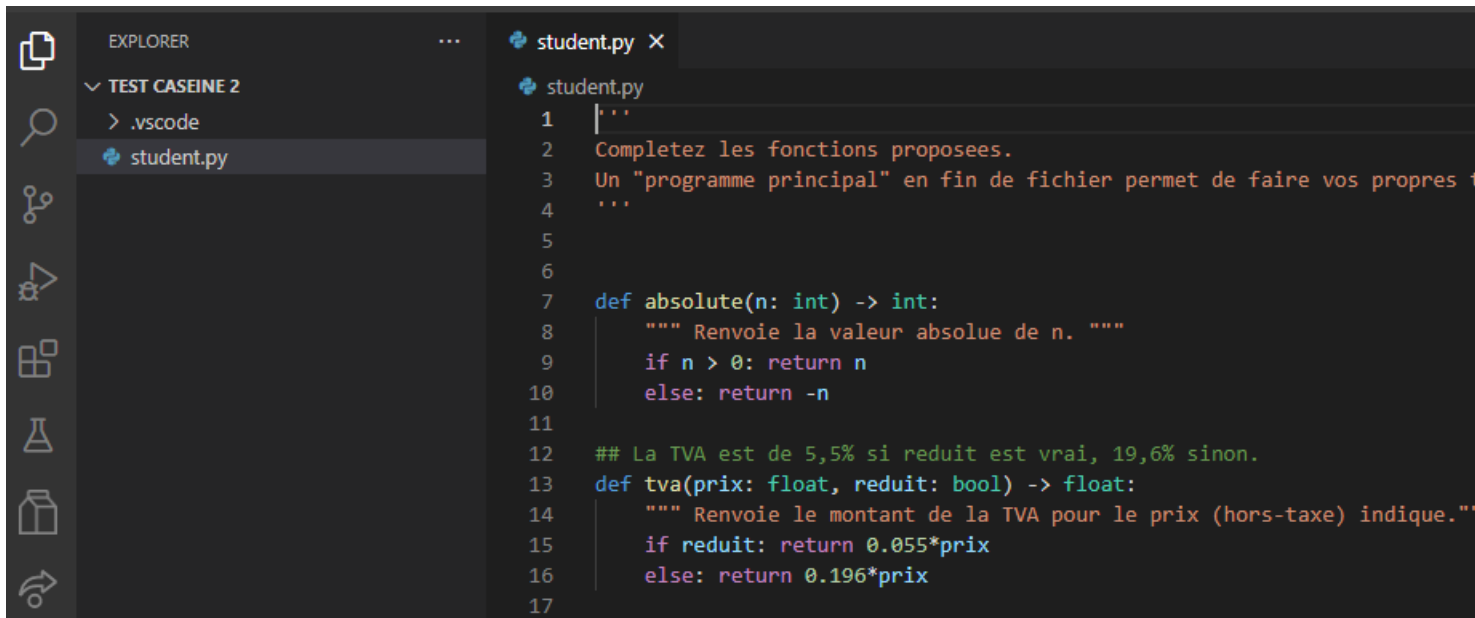
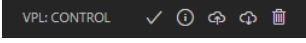


FIGURE 8 – L'activité est importée dans VSCode.

Dans ce cas, il faut tout recommencer : créer un nouveau dossier, cliquer sur Initialize a VPL, aller chercher le lien web service... Vous pourrez bien sur supprimer le dossier créer inutilement !

7. Retournez dans l'onglet VPL (brique de lait) pour évaluer ou soumettre votre code sur Caseine à l'aide des boutons  (cf. Figure. 9). Parmi ces boutons, vous trouverez un bouton "evaluate" qui envoie votre code sur Caseine et l'évalue, un bouton "push" permettant d'envoyer votre code sur Caseine sans l'évaluer et un bouton "pull" vous permettant de récupérer votre code Caseine directement sur votre machine.

N'oubliez pas que vous êtes encouragé à écrire vos propres tests à chaque fois que vous rédigez une méthode. Pour tester vos méthodes, il faut les appeler avec différentes valeurs des paramètres pour vérifier que la méthode fait bien ce que l'on veut (une erreur est vite arrivée)! Pour rappel, ces tests doivent être écrits dans le bloc "principal".

Dans VSCode, pour exécuter votre programme, utilisez la "flèche" située dans le coin supérieur droit

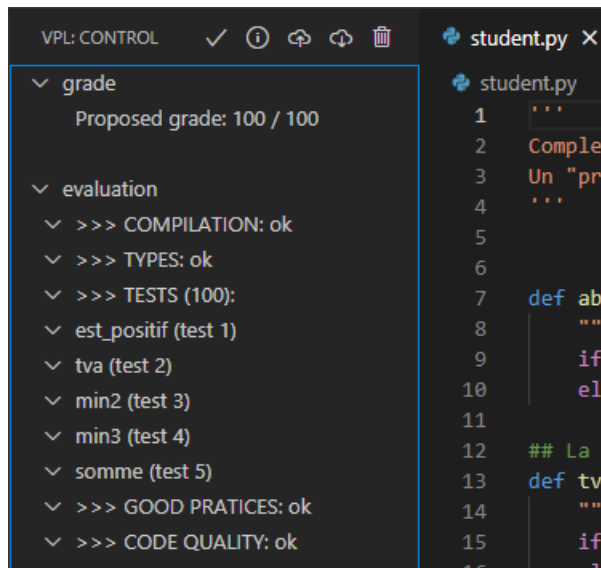



FIGURE 9 – Évaluation de code via VSCode.

de la fenêtre .

## A VSCode sur votre machine personnelle

Si vous souhaitez installer VSCode sur votre machine rendez vous sur la page suivante : <https://code.visualstudio.com/> et téléchargez le Visual Studio Code correspondant à votre système d'exploitation (Windows, Linux, Mac OS...).

Une fois cela fait, lancez Visual Studio. Dans l'onglet *Extensions*  (barre latérale gauche), cherchez et installez les applications Python (celle en haut de la liste qui s'appelle Python) et VPL Client.