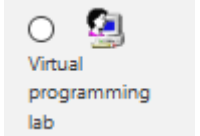




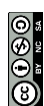
# Virtual Programming Lab

Activité Moodle



## Table des matières

Résumé .....	2
Scénarios pédagogiques .....	2
1. Scénario 1 .....	2
2. Scénario 2 .....	3
3. Scénario 3 .....	4
4. Scénario 4 .....	5
Création de l'activité dans Moodle .....	6
Réglages Moodle de l'activité VPL.....	7
1. Généraux .....	8
2. Submission period .....	8
3. Submission restrictions .....	8
4. Note .....	8
5. Autres .....	8
Introduction au réglage de l'objet VPL.....	9
1. Virtual programming labs.....	9
2. Execution option.....	9
3. Requested files .....	10
4. Réglages avancés.....	10
4.1. Exécution files .....	10
4.2. Maximum execution resources limits .....	10
4.3. Files to keep when running.....	10
4.4. Variations .....	11
4.5. Tester les serveurs d'exécution .....	11
4.6. Local exécution servers .....	11
Réglages types .....	12
Scénario 1 .....	12
Scénario 2 .....	16
Scénario 3 .....	21
Scénario 4 .....	27
Ressources.....	32
Auteurs .....	33



## Résumé

Cet article présente VPL, laboratoire de programmation virtuelle, disponible dans les activités Moodle.

- Pour les étudiants, il s'agit d'un environnement de développement simple avec des capacités d'auto-évaluation.
- Pour les professeurs, c'est un système de gestion du travail des étudiants, avec des fonctionnalités pour faciliter la préparation des travaux, gérer les soumissions, vérifier le plagiat et faire des évaluations basées sur les tests du programme.

## Scénarios pédagogiques

### 1. Scénario 1

- Le professeur décrit textuellement le programme à réaliser

#### Exemple 1

**Due date:** vendredi 17 janvier 2020, 01:00

**Nombre maximal de fichiers:** 1

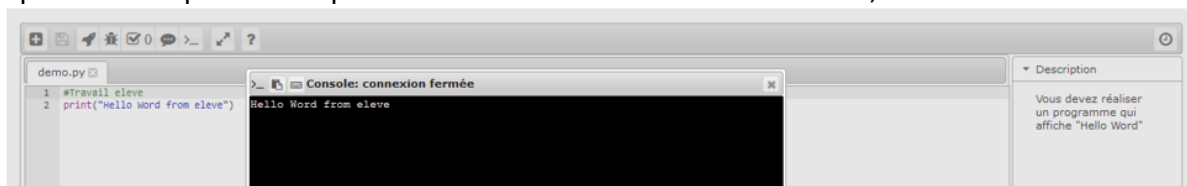
**Type of work:** Individual work

Vous devez réaliser un programme qui affiche "Hello Word"

- L'étudiant dispose d'un environnement qui lui permet de développer son propre programme en ligne. Chaque lancement du programme est considéré comme une soumission.



- A posteriori le professeur peut observer les différentes soumissions, les tester.





## 2. Scénario 2

- Le professeur propose à l'étudiant un programme modèle, par exemple des prototypes de fonctions ...

### Requested files: Exemple 2

```
★ exemple.py 
```

```
1 #voici le fichier à modifier  
2 print("Request file")
```

- L'étudiant complète le programme modèle en ligne. Chaque lancement du programme est considéré comme une soumission.

```
exemple.py 
```

```
1 #voici le fichier à modifier  
2 print("Request file")  
3 print("my proposition")
```

```
>_  Console: connexion fermée  
Request file  
my proposition
```

- A posteriori le professeur peut observer les différentes soumissions, les tester.

```
exemple.py 
```

```
1 #voici le fichier à modifier  
2 print("Request file")  
3 print("my proposition")
```

```
>_  Console: connexion fermée  
Request file  
my proposition
```

Description

Vous devez modifier cet exemple



### 3. Scénario 3

- Le professeur propose à l'étudiant un programme modèle, par exemple des prototypes de fonctions ...

#### Requested files: Exemple 3

```

demo.py
1 #compléter le fichier
2 def main():
3     x = float(input(">"))
4     #
5     z = float(input(">"))
6     print(min([x, y, z]))
7
8 main()
  
```

- Le professeur prépare des cas de tests du programme proposé, basés sur des « input » et des « print » du programme.

#### Test cases: Exemple 3

```

vpl_evaluate.cases
1 case = test un
2 input = 1
3 2
4 3
5 output = 1.0
6
7 case = test deux
8 input = 2
9 1
10 3
11 output = 1.0
12
13 case = test trois
14 input = 2
15 1
16 -3
17 output = -3
  
```

- L'étudiant complète le programme modèle en ligne. Chaque lancement du programme est considéré comme une soumission.
- A posteriori le professeur peut observer les différentes soumissions et sur chacune déclencher les cas de tests. La notation peut être automatique.

Proposed grade: 66,67 / 100

Commentaires

Execution

```

Testing 1/3 : test un
Testing 2/3 : test deux
Testing 3/3 : test trois

<|--
-Test 3: test trois (-33.333)
Incorrect program output
--- Input ---
> 2
>1
>-3

--- Program output ---
>>>-3.0
  
```





## 4. Scénario 4

- Le professeur propose à l'étudiant un programme modèle, par exemple des prototypes de fonctions
- Le professeur a préparé un fichier lanceur de test qui appelle les fonctions à tester avec les arguments voulus et font un « print » du résultat.
- Cette technique universelle permettra de tester toutes sortes de fonctions.
- Comme dans le scénario 3 l'enseignant pourra affiner les réglages pour obtenir une notation de son choix.
- Le professeur donne le prototype du programme, ou demande aux étudiants un programme qui se termine obligatoirement en fin de programme par les lignes :

StudentProduction.py

```
1 #-*-coding=UTF-8 *-
2 # Créé par xavier, le 15/12/2019 en Python 3.4
3 # Fichier modèle
4
5 def somme_ligne(carre, n):
6     """ carre est un tableau carre de nombres
7     n est un nombre entier
8     Calcul la somme des nombres sur la ligne n
9     """
10    return 999
```

```
38 # à ajouter pour la submission
39 if __name__ == "__main__":
40     from TestLauncher import RunTest
41     RunTest(__file__)
```

Ce sont ces lignes qui permettent d'appeler le lanceur de test. Elles sont donc indispensables dans ce scénario.

- Le professeur prépare un jeu de test

vpl\_evaluate.cases

```
1 case = fonction somme_ligne(carré, numéro de ligne)
2 input = u1
3 output = 34
4 case = fonction check_row(carré)
5 input = u2
```

case = description libre du test à effectuer  
input = nom d'une variable qui portera la valeur à tester.  
output = valeur attendue pour valider le test

- Le professeur prépare le fichier lanceur de test

TestLauncher.py

```
1 #TestLauncher.py
2 #-*-coding=UTF-8 *-
3 # Créé par X.CARBONNAUX, le 31/12/2019 en Python 3.4
4
5 def RunTest(FileToTest):
6     # récupération des variations
7     try:
8         import os
9         __VPL_VARIATION__ = os.environ['VPL_VARIATION']
10    except KeyError:
11        pass
12
13    # Import du fichier à tester
14    FileToTest = str(FileToTest.split(".")[0])
15    FileToTest = __import__(FileToTest)
16
17    # Initialisation des jeux de tests
18    # Ajouter ici les jeux de tests nécessaires
19    carre3 = [[2, 7, 6],[9, 5, 1], [4, 3, 8]]
20    carre4 = [[4, 5, 11, 14], [15, 10, 8, 1], [6, 3, 13, 12],[9, 16, 2, 7]]
21
22    # Analyse des paramètres transmis par les Test cases
23    TestParameters = input()
24    TypeOfTest = TestParameters[1]
25    if len(TestParameters) > 2:
26        TestNumber = TestParameters[2]
27    else:
28        TestNumber = ''
29
30    #####
31    # Lancement des tests
32    #####
```

- L'étudiant complète le programme modèle en ligne. Ou dépose son programme.
- A posteriori le professeur peut observer les différentes soumissions, les tester ou bénéficier de l'évaluation automatique.



## Création de l'activité dans Moodle

- Dans cours Moodle, cliquer sur le bouton

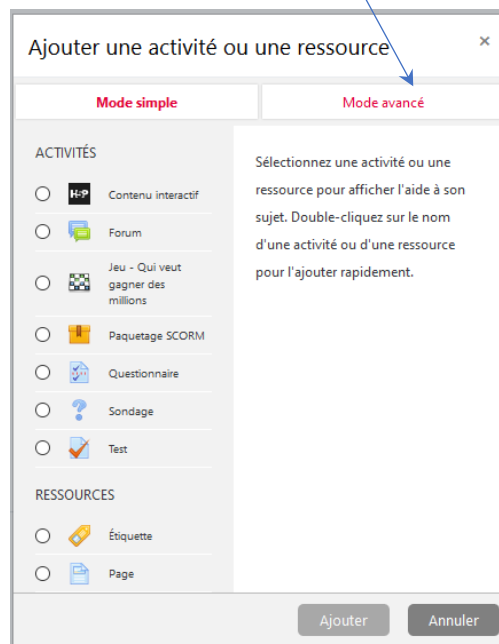
Activer le mode édition



- En bas de la section de votre choix, cliquer sur **+ Ajouter une activité ou une ressource**




- Dans la fenêtre contextuelle cliquer sur **Mode avancé**



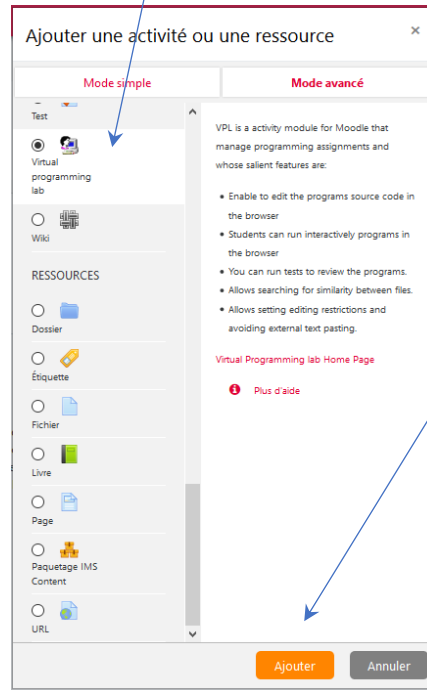


- Puis déplacer l'ascenseur et sélectionner

  
 Virtual  
 programming  
 lab

puis cliquer sur ajouter




Ajouter






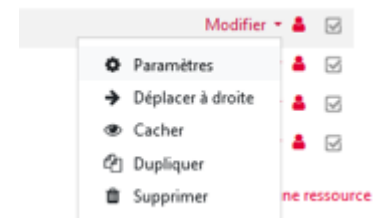
L'activité Moodle est créée et l'écran suivant présente les réglages usuels des activités Moodle.

## Réglages Moodle de l'activité VPL

Si le mode édition est activé, on peut toujours revenir sur l'écran de réglage d'une activité, en cliquant sur

Modifier   à droite de l'écran puis  Paramètres

  Exemple 1 





## 1. Généraux

Modification Virtual programming lab dans VPL

### Généraux

Nom   
 Courte description   
 Full description   
 Vous devez réaliser un programme qui affiche "Hello Word"

La courte description sera visible dans le lab et permettra de trier les activités.

On indique ici la description complète.

## 2. Submission period

On indique ici les date d'ouverture et de fermeture de l'activité.

### Submission period

Disponible à partir de       Activer  
 Due date       Activer

## 3. Submission restrictions

### Submission restrictions

Nombre maximal de fichiers   
 Type of work   
 Dissable external file upload, paste and drop external content   
 This activity acts as example   
 Maximum upload file size   
 Mot de passe   
 Allowed submission from net   
 SEB browser required   
 SEB exam Key/s

Les réglages par défauts peuvent convenir.

Eventuellement, on obliger les étudiants à écrire le code :

Dissable external file upload, paste and drop external content

## 4. Note

### Note

Note   
 Barème   
 Note maximale   
 Catégorie de note   
 Note pour passer   
 Reduction by automatic evaluation   
 Free evaluations   
 Visible

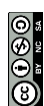
Les réglages sont similaires aux autres activités Moodle.

Seuls les champs suivants sont spécifiques :

Reduction by automatic evaluation   
 Free evaluations   
 Number of automatic evaluations that do not reduce final score

## 5. Autres

Les derniers réglages ne seront pas détaillés ici car communs à toutes les activités Moodle







## Introduction au réglage de l'objet VPL

Une particularité de cette activité est que le bouton Modifier l'activité de Moodle n'influence pas l'accès aux réglages. Il suffit de cliquer sur l'objet d'activité  Exemple 0 pour entrer dans Virtual Programming Lab.


Description **Liste des devoirs rendus** Similarité Activité test

### Exemple 0







Due date: Saturday 1 February 2020, 01:00  
 Nombre maximal de fichiers: 1  
 Type of work: Individual work  
 Réglages des notes: Note maximale: 100  
 Run: Non. Evaluate: Non

Description générale de l'activité Moodle

Pour le professeur l'onglet description contient un résumé de l'activité et de tous les fichiers qui ont été placés dans l'objet VPL.

En cliquant sur l'icone  on accède à TOUS les menus de VPL

Paramètres

-  Test cases
-  Execution options
-  Requested files
-  Réglages avancés
  - Execution files
  - Maximum execution resources limits
  - Files to keep when running
  - Variations
  - Tester les serveurs d'exécution
  - Local execution servers
-  Activité test
  - Devoir rendu
  - Edit
  - Submission view
  - Note
  - Previous submissions list
-  Virtual programming labs
- Permissions
- Voir les permissions
- Filtres
- Journaux
- Sauvegarde
- Restauration

Seule la zone encadrées concerne VPL, les autres menus renvoient vers des fonctionnalités Moodle qui sont généralement accessibles par ailleurs.

 Ce symbole permet de repérer les menus principaux.

## 1. Virtual programming labs

Ce menu donne une vision de toutes les activités VPL au sein d'un même cours.

Virtual programming labs

#	Nom	Courte description	Due date	Devoirs rendus	Graded
1	Exemple 0	Scénario par défaut	samedi 1 février 2020, 01:00	0	0
2	Exemple 1	Scénario 1	vendredi 17 janvier 2020, 01:00	2	0
3	Exemple 2		vendredi 17 janvier 2020, 01:00	2	1
4	Exemple 3		vendredi 17 janvier 2020, 01:00	2	0
5	Exemple 4		samedi 1 février 2020, 01:00	0	0
Total				6	1

## 2. Execution option

Execution options: Exemple 0

Execution options

Basé sur: Sélectionner

Run script: Autodétection

Script de débogage: Autodétection

Run: Non

Débugger: Non

Evaluate: Non

Evaluate just on submission: Non

Note automatique: Non

Il faudrait régler convenablement ces champs car par défaut les étudiants ne peuvent ni exécuter, ni déboguer ni faire évaluer leur code. Ils disposent juste du droit d'éditer un fichier.

Chaque enregistrement constitue une nouvelle soumission.

Description Edit **Submission view**

demo.py

```

1 #Travail eleve
2 a=input("entrer un nombre")
3 print(a,"a work from eleve")
4
        
```





### 3. Requested files

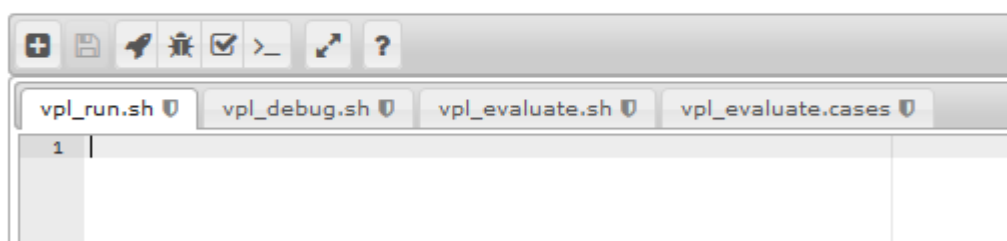
Ce menu permet d'imposer à l'étudiant le nom du fichier remis en créant un fichier vide. Le fichier pourra être aussi être imposé comme squelette de départ à modifier par l'étudiant.

### 4. Réglages avancés

#### 4.1. Exécution files

Ce menu regroupe l'accès à tous les fichiers permettant de créer l'activité.

#### Execution files: Exemple 1



Remarquer que le menu principal « Test Cases » permet d'accéder directement à ce même fichier « vpl\_evaluate.cases ».

#### 4.2. Maximum execution resources limits

#### Resources limits: Exemple 0

##### ▼ Resources limits

Maximum execution time

Maximum memory used

Maximum execution file size

Maximum number of processes

Ces limites sont utilisées lors les fichiers précédents sont exécutés.

Par expérience le réglage suivant semble être satisfaisant.

##### Resources limits

Maximum execution time

Maximum memory used

Maximum execution file size

Maximum number of processes

#### 4.3. Files to keep when running

#### Files to keep when running: Exemple 0

##### ▼ Files to keep when running

- vpl\_run.sh
- vpl\_debug.sh
- vpl\_evaluate.sh
- vpl\_evaluate.cases

Les fichiers nécessaires pendant l'exécution du programme devront être sélectionnés ici.





## 4.4. Variations

Variations: Exemple 0

### Variation options

Use variations  Non

Variation title

### Ajouter

Identification

Description

Cette documentation ne détaille pas l'utilisation de cette fonctionnalité du module VPL.

Elle permettrait, à priori, de faire des variations dans les tests suivants l'étudiant qui soumet son source code.

## 4.5. Tester les serveurs d'exécution

Tester les serveurs d'exécution: Exemple 0

#	Serveur	Statut courant	Last error info	Last error date	Erreurs
1	http://demojail.dis.ulpgc.es	ready	request failed: Resolving timed out after 5515 milliseconds	mercredi 8 janvier 2020, 23:04	57

On peut remarquer que le serveur retenu renvoie régulièrement des erreurs.

## 4.6. Local exécution servers

Local execution servers: Exemple 0

### Local execution servers

Write a line for each server

VPL prévoit la possibilité de choisir soit même un serveur.





## Réglages types

### Scénario 1

- Le professeur décrit textuellement le programme à réaliser en utilisant le menu paramètres

Modification Virtual programming lab dans VPL

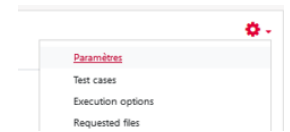
#### ▼ Généraux

Nom

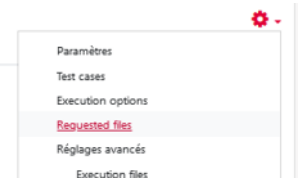
Courte description

Full description

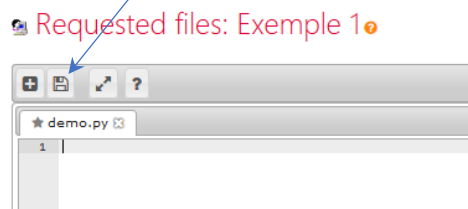
**Vous devez réaliser un programme qui affiche "Hello Word"**



- Le professeur impose le nom du fichier attendu

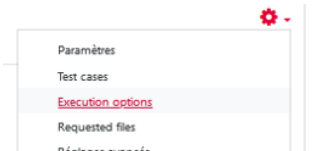


Cliquer sur Ok, puis ne pas oublier d'enregistrer

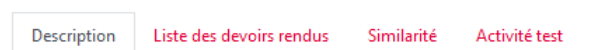


- Le professeur règle les options d'exécution

Run	<input type="button" value="Oui"/>
Dégager	<input type="button" value="Non"/>
Evaluate	<input type="button" value="Non"/>
Evaluate just on submission	<input type="button" value="Non"/>
Note automatique	<input type="button" value="Non"/>



- Le professeur voit un résumé des réglages dans l'onglet description



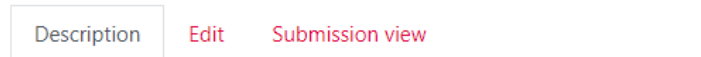
### Exemple 1

**Due date:** samedi 1 février 2020, 01:00  
**Requested files:** demo.py ([Download](#))  
**Type of work:** Individual work  
**Réglages des notes:** Note maximale: 100  
**Dissable external file upload, paste and drop external content:** Oui  
**Run:** Oui. **Evaluate:** Non

Vous devez réaliser un programme qui affiche "Hello Word"



- L'étudiant ouvre l'activité



## Exemple 1

**Due date:** samedi 1 février 2020, 01:00

**Requested files:** demo.py ([Download](#))

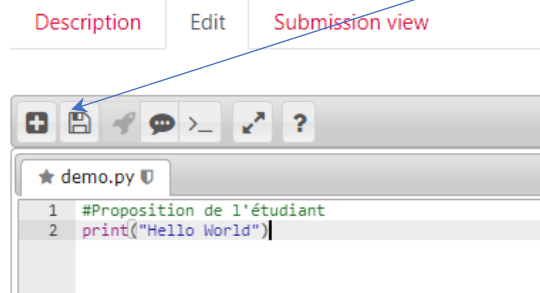
**Type of work:** Individual work

Vous devez réaliser un programme qui affiche "Hello Word"

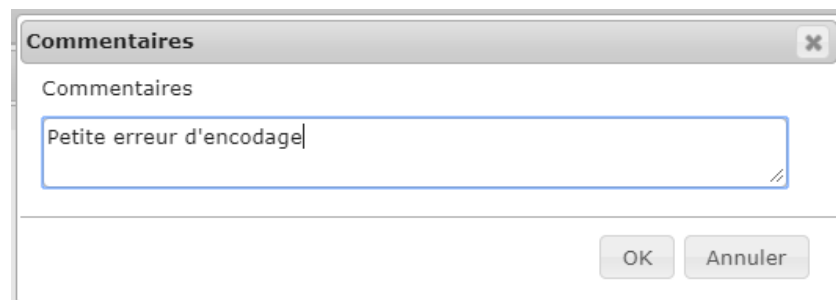
- Par l'onglet Edit, l'étudiant peut éditer son code



- Après avoir codé ou modifié son programme l'étudiant doit sauver son travail

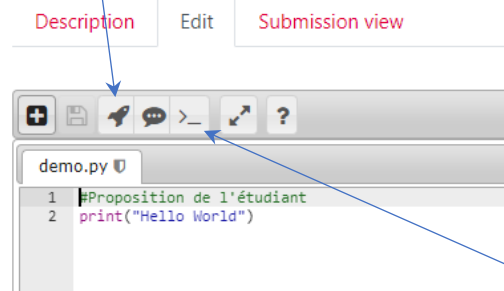


- L'étudiant peut placer des commentaires qui seront perdus dès la fermeture de l'onglet Edit

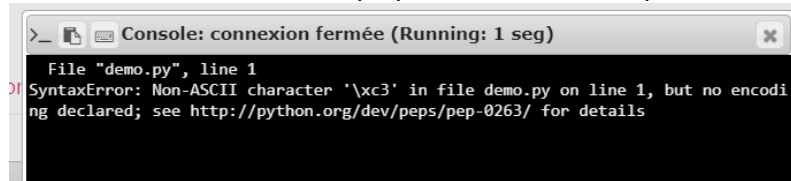




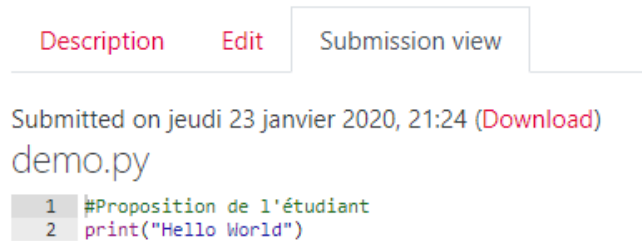
- Un programme sauvé peut être testé



- Le programme est exécuté dans une console qui peut être ouverte par l'icone



- Chaque nouvelle sauvegarde ou exécution enregistrera une nouvelle soumission que l'étudiant pourra observer via l'onglet dédié



- Le professeur se rend dans l'onglet Liste des devoirs rendus



- Le professeur peut consulter chaque devoir rendu





- Le professeur pourra ensuite voir, exécuter

[Devoir rendu](#) [Edit](#) [Submission view](#) [Note](#) [Previous submissions list](#)

---

Submitted on jeudi 23 janvier 2020, 21:37 ([Download](#))

demo.py

```

1 #Proposition de l'etudiant
2 print("Hello World")

```

- Le professeur peut noter manuellement le travail et valide par

[Devoir rendu](#) [Edit](#) [Submission view](#) [Note](#) [Previous submissions list](#)

---

Note  [Note](#) [Remove grade](#)

[Copier](#) [Calculer](#)

Commentaires

Deux essais nécessaires

---

[Description](#) [Liste des devoirs rendus](#) [Similarité](#) [Utilisateur eleve graves](#)

---

[Devoir rendu](#) [Edit](#) [Submission view](#) [Note](#) [Previous submissions list](#)

---

Graded

- Attention il faut s'assurer que l'étudiant ne peut plus compléter son devoir, mais uniquement avoir accès à la correction. La date limite de dépôt doit être dépassée avant de commencer à noter.

Le professeur voit :

[Description](#) [Edit](#) [Submission view](#)

---

Note

Reviewed on jeudi 23 janvier 2020, 22:12 by LUC VINCENT  
**grade:** 100,00 / 100,00

**Assessment report[-]**  
 Plusieurs essais

Submitted on jeudi 23 janvier 2020, 22:02 ([Download](#))

demo.py

```

1 #Proposition de l'etudiant
2 print("Hello World")
3

```

L'étudiant voit :

[Description](#) [Submission view](#)

---

Note

Reviewed on jeudi 23 janvier 2020, 22:12 by LUC VINCENT  
**grade:** 100,00 / 100,00

**Assessment report[-]**  
 Plusieurs essais





## Scénario 2

- Le professeur propose à l'étudiant un programme modèle, par exemple un prototype de fonction. Il donne la consigne dans le menu paramètre ?

➤ [Ajout Virtual programming lab à VPL](#)

### ▼ Généraux

Nom

Courte description

Full description

Compléter le fichier qui vous est proposé

Paramètres

Test cases

Execution options

Requested files

- Le fichier modèle est fourni aux étudiants

Requested files: Exemple 2

Créer un nouveau fichier

Nom du nouveau fichier (modele.py)

OK Annuler

Paramètres

Test cases

Execution options

**Requested files**

Réglages avancés

Execution files

- Le professeur donne le prototype du programme

➤ [Requested files: Exemple 2](#)

```

1 def peri_poly(*args):
2     """ Return perimeter of polygon
3     input : some int
4     output : int
5     """
6     pass # A remplacer par votre code
7
8
9
10 if __name__=="__main__":
11     print(peri_poly())
12     print(peri_poly(1))
13     print(peri_poly(1, 2))
14     print(peri_poly(1, 2, 3))
15     print(peri_poly(1, 2, 3, 4))
  
```

- Le professeur règle les options d'exécution

Run

Débugger

Evaluate

Evaluate just on submission

Note automatique

Paramètres

Test cases

**Execution options**

Requested files

Réglages avancés

- Le professeur voit un résumé des réglages dans l'onglet description

Description [Liste des devoirs rendus](#) [Similarité](#) [Activité test](#)

### Exemple 2

Due date: samedi 1 février 2020, 01:00  
 Requested files: [modele.py \(Download\)](#)  
 Type of work: Individual work  
 Réglages des notes: Note maximale: 100  
 Disable external file upload, paste and drop external content: Oui  
 Run: Oui. Evaluate: Non

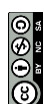
Compléter le fichier qui vous est proposé

### Requested files

modele.py

```

1 def peri_poly(*args):
2     """ Return perimeter of polygon
3     input : some int
4     output : int
5     """
6     pass # A remplacer par votre code
7
8
9
10 if __name__=="__main__":
11     print(peri_poly())
12     print(peri_poly(1))
13     print(peri_poly(1, 2))
14     print(peri_poly(1, 2, 3))
15     print(peri_poly(1, 2, 3, 4))
  
```







- L'étudiant ouvre l'activité

[Description](#)   [Edit](#)   [Submission view](#)

## Exemple 2

**Due date:** samedi 1 février 2020, 01:00

**Requested files:** modele.py ([Download](#))

**Type of work:** Individual work

Compléter le fichier qui vous est proposé

## Requested files

modele.py

```

1 def peri_poly(*args):
2     """ Return perimeter of polygon
3     input : some int
4     output : int
5     """
6     pass # A remplacer par votre code
7
8
9
10 if __name__=="__main__":
11     print(peri_poly())
12     print(peri_poly(1))
13     print(peri_poly(1, 2))
14     print(peri_poly(1, 2, 3))
15     print(peri_poly(1, 2, 3, 4))
  
```

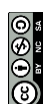
La suite de ce scénario est identique au scénario 1, elle est redétaillée ici pour faciliter l'utilisation de cette fiche par un lecteur non aguerri.

- Par l'onglet Edit, l'étudiant peut éditer/compléter le code

[Description](#)   [Edit](#)   [Submission view](#)

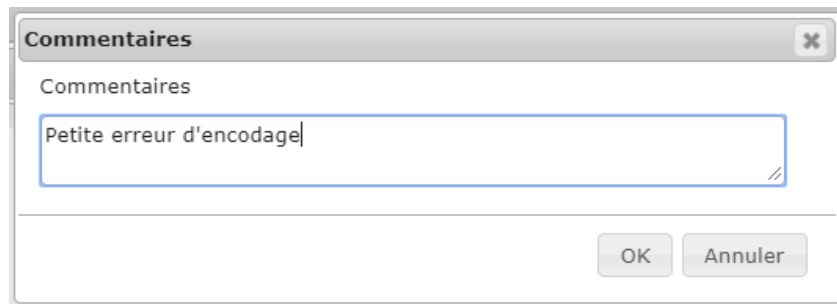
- Après avoir codé ou modifié son programme l'étudiant doit sauvegarder son travail

[Description](#)   [Edit](#)   [Submission view](#)

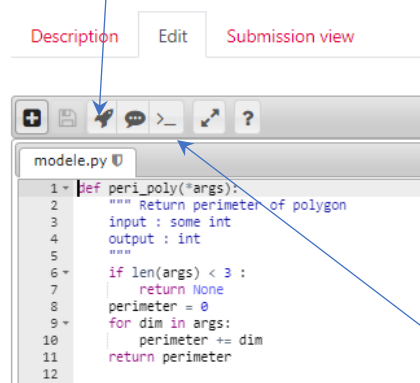




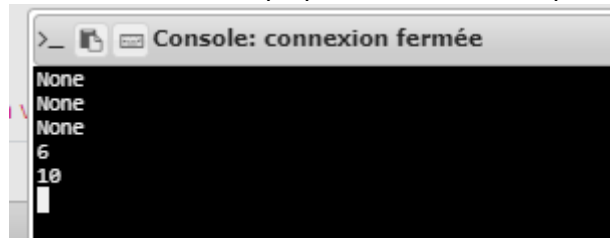
- L'étudiant peut placer des commentaires qui seront perdus dès la fermeture de l'onglet Edit



- Un programme sauvé peut être testé



- Le programme est exécuté dans une console qui peut être ouverte par l'icone



- Chaque nouvelle sauvegarde ou exécution enregistrera une nouvelle soumission que l'étudiant pourra observer via l'onglet dédié





- Le professeur se rend dans l'onglet Liste des devoirs rendus

[Description](#) | [Liste des devoirs rendus](#) | [Similarité](#) | [Activité test](#)

Submission selection: Tous les envois

Evaluate: Choisir...

	Prénom	Nom	Submitted on	Devoirs rendus	Note	Evaluator	Evaluated on
1	élève graves		lundi 27 janvier 2020, 22:23	2	Pas de note		

- Le professeur peut consulter chaque devoir rendu

# Date de soumission

- 2 lundi 27 janvier 2020, 22:23
- 1 jeudi 23 janvier 2020, 22:44

- En cliquant sur une des dates, le professeur pourra ensuite voir, exécuter

[Description](#) | [Liste des devoirs rendus](#) | [Similarité](#) | Utilisateur élève graves

[Devoir rendu](#) | [Edit](#) | [Submission view](#) | [Note](#) | [Previous submissions list](#)

Submitted on lundi 27 janvier 2020, 22:23 (Download)

modele.py

```

1 def peri_poly(*args):
2     """ Return perimeter of polygon
3     input : some int
4     output : int
5     """
6     if len(args) < 3 :
7         return None
8     perimetre = 0
9     for dim in args:
10        perimetre += dim
11    return perimetre
12
13
14 if __name__ == "__main__":
15    print(peri_poly())
16    print(peri_poly(1))
17    print(peri_poly(1, 2))
18    print(peri_poly(1, 2, 3))
19    print(peri_poly(1, 2, 3, 4))
  
```

- Onglet Note, le professeur peut noter manuellement le travail et valide par

[Description](#) | [Liste des devoirs rendus](#) | [Similarité](#) | Utilisateur élève graves

[Devoir rendu](#) | [Edit](#) | [Submission view](#) | [Note](#) | [Previous submissions list](#)

Note: 100 [Note](#) [Remove grade](#)

[Copier](#) [Calculer](#)

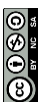
Commentaires

Excellent travail

[Description](#) | [Liste des devoirs rendus](#) | [Similarité](#) | Utilisateur élève graves

[Devoir rendu](#) | [Edit](#) | [Submission view](#) | [Note](#) | [Previous submissions list](#)

Graded





- Attention la date limite de dépôt doit être dépassée avant de commencer à noter. C'est la seule façon de s'assurer que l'étudiant ne peut plus compléter son devoir, mais uniquement avoir accès à la correction.

## Coté professeur

[Description](#)
[Liste des devoirs rendus](#)
[Similarité](#)
[Utilisateur eleve graves](#)

### Exemple 2

**Due date:** dimanche 26 janvier 2020, 01:00  
**Requested files:** modele.py ([Download](#))  
**Type of work:** Individual work  
**Réglages des notes:** Note maximale: 100  
**Dissable external file upload, paste and drop external content:** Oui  
**Run:** Oui. **Evaluate:** Non

Compléter le fichier qui vous est proposé

### Requested files

modele.py

```

1 def peri_poly(*args):
2     """ Return perimenter of polygon
3     input : some int
4     output : int
5     """
6     pass # A remplacer par votre code
7
8
9
10 if __name__=="__main__":
11     print(peri_poly())
12     print(peri_poly(2))
13     print(peri_poly(2, 2))
14     print(peri_poly(2, 2, 3))
15     print(peri_poly(2, 2, 3, 4))
  
```

Si la date de remise est dépassée

Alors coté étudiant, l'onglet « edit » a disparu. L'onglet Submission view donne accès au document corrigé, note et commentaires.

[Description](#)
[Submission view](#)

## Note

Reviewed on lundi 27 janvier 2020, 22:43 by LUC VINCENT  
**grade:** 100,00 / 100,00

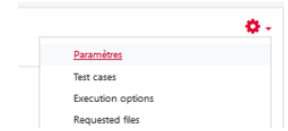
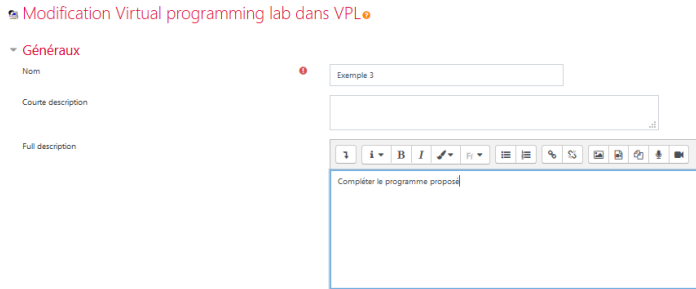
### Assessment report[-]

Excellent travail

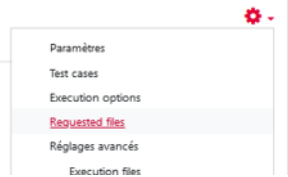
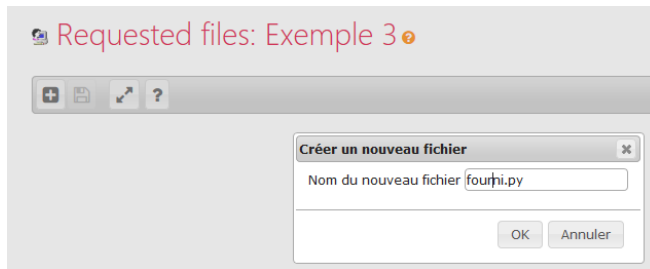


### Scénario 3

- Le professeur propose à l'étudiant un programme modèle, par exemple un prototype de fonction.

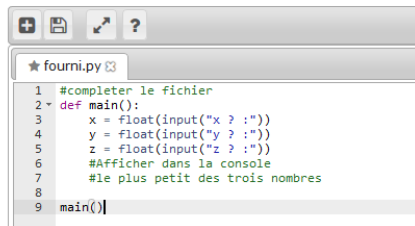


- Le fichier modèle est fourni aux étudiants



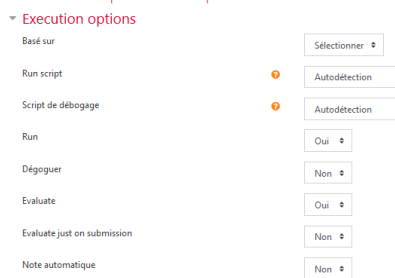
- Le professeur donne le prototype du programme

Requested files: Exemple 3

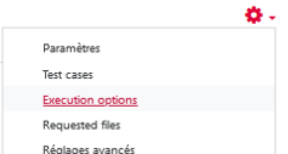


- Le professeur règle les options d'exécution

Execution options: Exemple 3

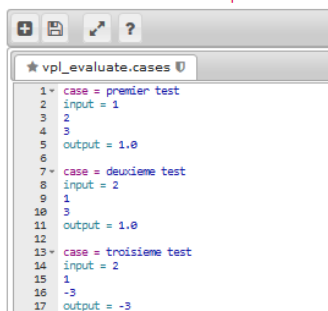


On laisse la possibilité à l'étudiant de tester son programme Et de tester la réponse de son programme au jeu de test.

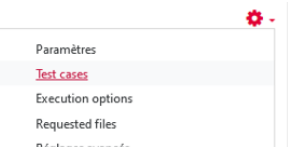


- Le professeur prépare un jeu de test

Test cases: Exemple 3



case = description libre du test à effectuer  
input = valeurs successives soumises  
output = réponse attendue pour valider le test





- Le professeur précise les réglages du serveur

### 📁 Ressources limits: Exemple 3

#### ▼ Ressources limits

Maximum execution time	<input type="text" value="Sélectionner"/>
Maximum memory used	<input type="text" value="512 MiB"/>
Maximum execution file size	<input type="text" value="Sélectionner"/>
Maximum number of processes	<input type="text" value="30"/>

La mémoire et les nombres de process semblent avoir une influence sur le comportement du serveur.



- Le professeur voit maintenant un résumé de tous les réglages

### Exemple 3

**Due date:** samedi 1 février 2020, 01:00  
**Requested files:** fourni.py ([Download](#))  
**Type of work:** Individual work  
**Réglages des notes:** Note maximale: 100  
**Reduction by automatic evaluation:** 5 Free evaluations: 2  
**Dissable external file upload, paste and drop external content:** Oui  
**Run:** Oui. **Evaluate:** Oui  
**Maximum memory used:** 512 MiB. **Maximum number of processes:** 30.

Compléter le programme proposé

#### Requested files

fourni.py

```

1 #compléter le fichier
2 def main():
3     x = float(input("x ? :"))
4     y = float(input("y ? :"))
5     z = float(input("z ? :"))
6     #Afficher dans la console
7     #le plus petit des trois nombres
8
9 main()
  
```

#### Execution files

- L'étudiant ouvre l'activité

### Exemple 3

**Due date:** samedi 1 février 2020, 01:00  
**Requested files:** fourni.py ([Download](#))  
**Type of work:** Individual work

Compléter le programme proposé

#### Requested files

fourni.py

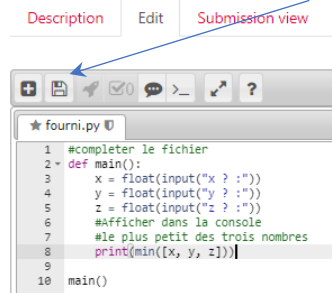
```

1 #compléter le fichier
2 def main():
3     x = float(input("x ? :"))
4     y = float(input("y ? :"))
5     z = float(input("z ? :"))
6     #Afficher dans la console
7     #le plus petit des trois nombres
8
9 main()
  
```

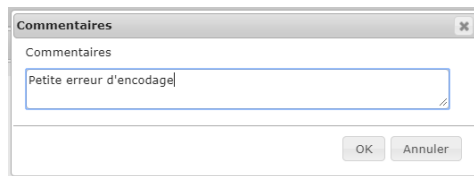




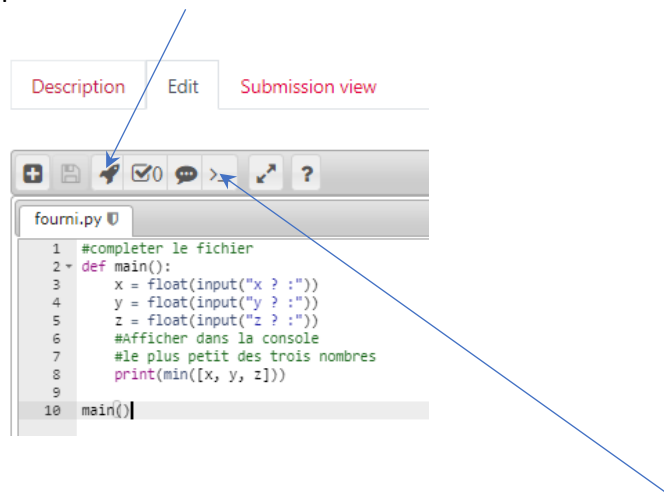
- Par l'onglet Edit, l'étudiant peut éditer compléter le code
- Après avoir codé ou modifié son programme l'étudiant doit sauver son travail



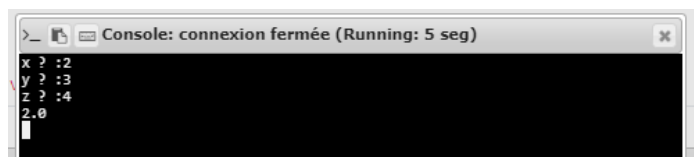
- L'étudiant peut placer des commentaires qui seront perdus dès la fermeture de l'onglet Edit



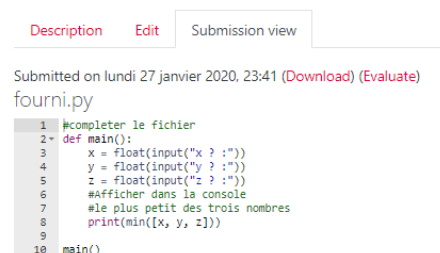
- Un programme sauvegardé peut être testé



- Le programme est exécuté dans une console qui peut être rouverte par l'icone



- Chaque nouvelle sauvegarde enregistrera une nouvelle soumission que l'étudiant pourra observer via l'onglet dédié





- L'étudiant peut évaluer son programme à l'aide du jeu de test fourni

Description Edit Submission view

```
fourni.py
1 #compléter le fichier
2 def main():
3     x = float(input("x ? :"))
4     y = float(input("y ? :"))
5     z = float(input("z ? :"))
6     #Afficher dans la console
7     #le plus petit des trois nombres
8     print(min([x, y, z]))
9
10 main()
```

Description

Compléter le programme proposé

- Sur la droite de l'écran l'étudiant voit le résultat

Proposed grade: 66,67 / 100

Commentaires

**Test 3: troisieme test**  
Incorrect program output  
--- Input ---  
2  
1  
-3  
--- Program output ---  
x ? :y ? :z ? :-3,0  
--- Expected output (numbers)---  
-3

**Summary of tests**  
/ 3 tests run/ 2 tests passed /

- A chaque nouvelle évaluation, l'écran submission view évolue

Description Edit Submission view

Submitted on lundi 27 janvier 2020, 23:54 (Download) (Evaluate)  
Évaluation automatique[-]

Proposed grade: 66,67 / 100

Commentaires[-]  
[+]Test 3: troisieme test  
[+]Summary of tests

```
fourni.py
1 #compléter le fichier
2 def main():
3     x = float(input("x ? :"))
4     y = float(input("y ? :"))
5     z = float(input("z ? :"))
6     #Afficher dans la console
7     #le plus petit des trois nombres
8     print(min([x, y, z]))
9
10
11 main()
```

On peut voir l'intérêt du réglage professeur : dans propriétés notes

Note

Type Point

Barème  
Appropriation du savoir liée ou détachée

Note maximale  
100

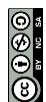
Catégorie de note  
Sans catégorie

Note pour passer  
0,00

Reduction by automatic evaluation  
5

Free evaluations  
2

Visible  
Oui







- La conséquence de ce réglage pour l'étudiant est visible

Description Edit Submission view

Submitted on lundi 27 janvier 2020, 23:54 (Download) (Evaluate)

Évaluation automatique

Proposed grade: 66,67 / 100  
Final reduction: 0 [2 / 2 -5]

Commentaires[-]  
[+]Test 3: troisieme test  
[+]Summary of tests

```
fourni.py
1 #completer le fichier
2 def main():
```

FR [NE/FE R]  
FR Final grade reduction.  
NE Automatic evaluations requested by the student.  
FE Free evaluations allowed.  
R Grade reduction by evaluation. If it is a percent, it is apply over previous result.

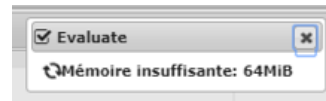
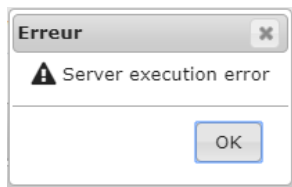
Description Edit Submission view

Submitted on mardi 28 janvier 2020, 00:06 (Download) (Evaluate)

Évaluation automatique[-]

Proposed grade: 61,67 / 100  
Final reduction: 5 [3 / 2 -5]

- Quelques messages d'erreurs peuvent être présents qui semble être liés à une certaine indisponibilité du serveur.



- Le professeur se rend dans l'onglet Liste des devoirs rendus

Description Liste des devoirs rendus Similarité Activité test

Submission selection Tous les envois

Evaluate Choisir...

	Prénom / Nom	Submitted on	Devoirs rendus	Note	Evaluator	Evaluated on
1	eleve graves	mardi 28 janvier 2020, 00:27	7	Proposed grade: 56,67 / 100		

- Le professeur peut consulter chaque devoir rendu et en cliquant sur une des dates, le professeur pourra ensuite voir, exécuter le programme de l'étudiant.
- La note est visible et le professeur peut éventuellement la modifier

#	Date de soumission
6	mardi 28 janvier 2020, 00:06
5	lundi 27 janvier 2020, 23:54
4	lundi 27 janvier 2020, 23:41
3	lundi 27 janvier 2020, 23:39
2	lundi 27 janvier 2020, 23:38
1	lundi 27 janvier 2020, 23:35

Note 66,67 -5 Note Grade & next Remove grade

Copier Evaluate Calculer

Commentaires

```
-Test 3: troisieme test (-33.333)
Incorrect program output
--- Input ---
> 2
> 1
> -3
--- Program output ---
>x ? :y ? :z ? :-3.0
```

L'action sur Grade & Next valide la note

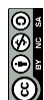
Note	Evaluator	Evaluated on
61,67 / 100,00	LUC VINCENT	mardi 28 janvier 2020, 00:22

Submitted on mardi 28 janvier 2020, 00:06 (Download) (Evaluate)

Évaluation automatique[-]

Proposed grade: 61,67 / 100  
Final reduction: 5 [3 / 2 -5]

Commentaires[-]  
[+]Test 3: troisieme test  
[+]Summary of tests





- On peut observer ici l'intérêt du réglage professeur

Run	<input type="button" value="Oui"/>
Dégager	<input type="button" value="Non"/>
Evaluate	<input type="button" value="Oui"/>
Evaluate just on submission	<input type="button" value="Oui"/>
Note automatique	<input type="button" value="Oui"/>

Le choix de la note automatique

Evitera ensuite la validation du professeur

⚙️


Paramètres

Test cases

**Execution options**

Requested files

Réglages avancés

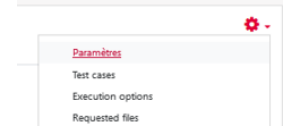
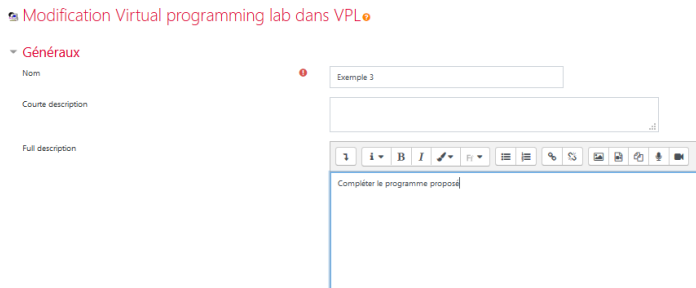
	Prénom ↓ /	Nom ↓	Submitted on ↓	Devoirs rendus ↓	Note ↓	Evaluator ↓	Evaluated on ↓
1		eleve graves	mardi 28 janvier 2020, 00:31	8	51,67 / 100,00	Note automatique	mardi 28 janvier 2020, 00:32



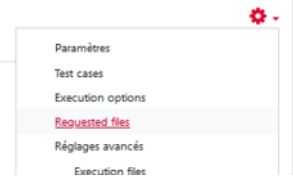
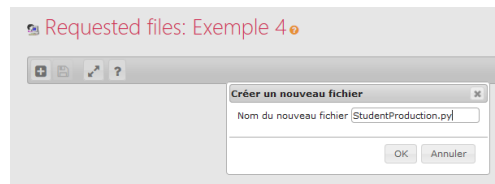


## Scénario 4

- Le professeur propose à l'étudiant un programme modèle, par exemple un prototype de fonction.



- Le fichier modèle est fourni aux étudiants



- Le professeur donne le prototype du programme

À titre d'exemple, on prend comme consigne la réalisation de fonctions utiles dans la manipulation de carré magiques. Voici le code source (fichier StudentProduction.py), fourni à l'étudiant lors de la création du « requested file » :

### StudentProduction.py

```

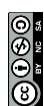
1  #-*-coding=UTF-8 *-
2  # Créé par xavier, le 15/12/2019 en Python 3.4
3  # Fichier modèle
4
5  def somme_ligne(carre, n):
6      """ carre est un tableau carre de nombres
7      n est un nombre entier
8      Calcul la somme des nombres sur la ligne n
9      """
10     return 999
11
12  def check_row(carre):
13     """ carre est un tableau carre de nombres
14     Vérifie si les sommes sur chaque ligne donnent le même résultat
15     """
16     return False
17
18  def somme_col(carre, n):
19     """ carre est un tableau carre de nombres
20     n est un nombre entier
21     Calcul la somme des nombres sur la colonne n
22     """
23     return 999
24
25  def check_col(carre):
26     """ carre est un tableau carre de nombres
27     Vérifie si les sommes sur chaque colonne donnent le même résultat
28     """
29     return False
30
31  def is_magic(carre):
32     """ carre est un tableau carre de nombres
33     Vérifie si c'est un carré magique
34     """
35     return False
36
37
38  # à ajouter pour la submission
39  if __name__ == "__main__":
40     from TestLauncher import RunTest
41     RunTest(__file__)
  
```

Observer l'importance en fin de programme des lignes

```

38  # à ajouter pour la submission
39  if __name__ == "__main__":
40     from TestLauncher import RunTest
41     RunTest(__file__)
  
```

Ce sont ces lignes qui permettent d'appeler le lanceur de test. Elles sont donc indispensables dans ce scénario.





- Le professeur règle les options d'exécution

Execution options: Exemple 4

Execution options

Basé sur Sélectionner

Run script PYTHON-3: Using python3 with the first file

Script de débogage Autodétection

Run Non

Débugger Non

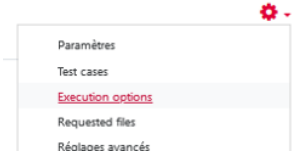
Évaluer Oui

Evaluate just on submission Oui

Note automatique Oui

Ici, on ne laisse pas la possibilité à l'étudiant de tester son programme (Run) A chaque soumission le programme sera évalué

Le champ Run script est indispensable !

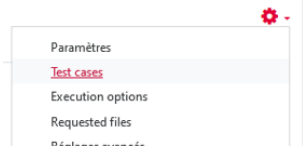


PYTHON-3: Using python3 with the first file

- Le professeur prépare un jeu de test

```
vpl_evaluate.cases
1 #Test Case
2 case = fonction somme_ligne(carré, numéro de ligne)
3 input = u1
4 output = 34
5 case = fonction check_row(carré)
6 input = u2
7 output = 'True'
8 case = fonction somme_col(carré, numéro de ligne)
9 input = u3
10 output = 34
11 case = fonction check_col(carré)
12 input = u4
13 output = 'True'
14 case = fonction is_magic(carré)
15 input = u5
16 output = 'True'
```

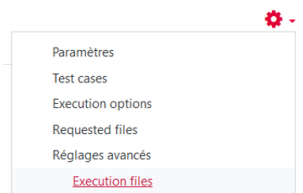
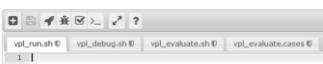
case = description libre du test à effectuer  
input = nom d'une variable qui portera la valeur à tester.  
output = valeur attendue pour valider le test



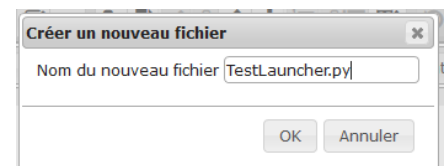
- Le professeur prépare le fichier lanceur de test

On souhaite créer le fichier lanceur **TestLauncher.py**

Execution files: Exemple 4

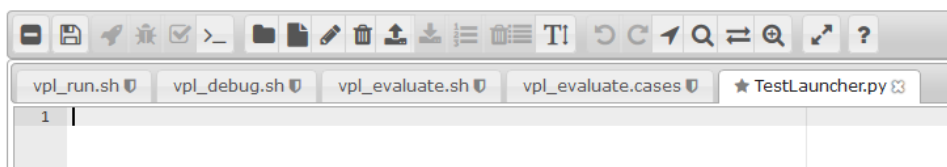


On crée un nouveau fichier



Le fichier est créé

Execution files: Exemple 4





- Le professeur prépare le code nécessaire et enregistre son travail

```
vpl_run.sh vpl_debug.sh vpl_evaluate.sh vpl_evaluate.cases TestLauncher.py
1 #TestLauncher.py
2 # -*- coding=UTF-8 -*-
3 # Créé par X.CARBONNAUX, le 31/12/2019 en Python 3.4
4
5 def RunTest(FileToTest):
6     # récupération des variations
7     try:
8         import os
9         __VPL_VARIATION__ = os.environ['VPL_VARIATION']
10    except KeyError:
11        pass
12
13    # Import du fichier à tester
14    FileToTest = str(FileToTest.split(".")[0])
15    FileToTest = __import__(FileToTest)
16
17    # Initialisation des jeux de tests
18    # Ajouter ici les jeux de tests nécessaires
19    carre3 = [[2, 7, 6],[9, 5, 1], [4, 3, 8]]
20    carre4 = [[4, 5, 11, 14], [15, 10, 8, 1], [6, 3, 13, 12],[9, 16, 2, 7]]
21
22    # Analyse des paramètres transmis par les Test cases
23    TestParameters = input()
24    TypeOfTest = TestParameters[1]
25    if len(TestParameters) > 2:
26        TestNumber = TestParameters[2]
27    else:
28        TestNumber = ''
29
```

Zone 17 à 20 ajouter les jeux de tests supplémentaires

```
29
30 #####
31 # Lancement des tests
32 #####
33
34 # Tests unitaires
35 if(TypeOfTest == 'u'):
36     # Créer un paragraphe par test unitaire (correspondant à un TestNumber particulier)
37     # Utiliser la fonction "print" pour renvoyer la réponse de chaque test
38
39     # 1 : Test somme d'une ligne
40     # réponse attendue = 34
41     if(TestNumber == '1'):
42         print(FileToTest.somme_ligne(carre4, 2))
43
44     # 2 : Test si toutes les lignes donnent le même résultat
45     # réponse attendue = 'True'
46     if(TestNumber == '2'):
47         print(FileToTest.check_row(carre3))
48
49     # 3 : Test somme d'une colonne
50     # réponse attendue = 34
51     if(TestNumber == '3'):
52
53
54     # 4 : Test si toutes les colonnes donnent le même résultat
55     # réponse attendue = 'True'
56     if(TestNumber == '4'):
```



Il faudra ajouter un bloc « if » pour chaque test prévu dans les Test cases.

```
# 1 : Test somme d'une ligne
#   réponse attendue = 34
if(TestNumber == '1'):
    print(FileToTest.somme_ligne(carre4, 2))
```

Dans ce bloc, on appelle la fonction à tester avec les arguments voulus, et on fait un « print » du résultat.

Dans cet exemple, si le test case a comme valeur input = u1 alors c'est la fonction somme\_ligne() qui sera testée avec carre4 et 2 comme arguments.

```
34 # Tests unitaires
35 if(.TypeOfTest == 'u'):
36     # Créer un paragraphe par test unitaire (correspondant à un TestNumber particulier)
37     # Utiliser la fonction "print" pour renvoyer la réponse de chaque test
```

- Le professeur indique dans VPL le nom du fichier à lancer

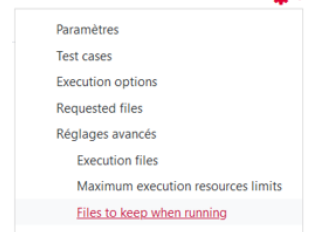
Files to keep when running: Exemple 4

Files to keep when running

- vpl\_run.sh
- vpl\_debug.sh
- vpl\_evaluate.sh
- vpl\_evaluate.cases
- TestLauncher.py

Enregistrer les options

Cocher TestLauncher.py



- Le professeur précise les réglages du serveur

Resources limits: Exemple 3

Resources limits

Maximum execution time: Sélectionner

Maximum memory used: 512 MiB

Maximum execution file size: Sélectionner

Maximum number of processes: 30

La mémoire et le nombres de process semblent avoir une influence sur le comportement du serveur



- Le professeur voit maintenant un résumé de tous les réglages

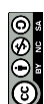
Description Liste des devoirs rendus Similarité Activité test

Exemple 4

**Due date:** samedi 1 février 2020, 01:00  
**Requested files:** StudentProduction.py (Download)  
**Type of work:** Individual work  
**Réglages des notes:** Note maximale: 100  
**Dissable external file upload, paste and drop external content:** Oui  
**Run:** Non. **Evaluate:** Oui. **Evaluate just on submission:** Oui  
**Note automatique:** Oui.

Les trois fichiers sont visibles :

- StudentProduction.py
- vpl\_evaluate.cases
- TestLauncher.py





- L'étudiant ouvre l'activité

Description Edit Submission view

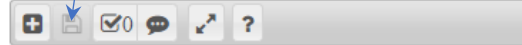
Exemple 4

Due date: samedi 1 février 2020, 01:00  
 Requested files: StudentProduction.py (Download)  
 Type of work: Individual work  
 Requested files  
 StudentProduction.py

Seul le fichier modèle du professeur est visible.  
L'étudiant ne voit pas les tests, ni le launcher.

- L'étudiant démarre l'activité, il modifie le modèle et enregistre son travail. Cela lui donne accès au bouton d'évaluation.

Description Edit Submission view


  
 StudentProduction.py

```

1 #-*-coding=UTF-8 *-
2 # Créé par xavier, le 15/12/2019 en Python 3.4
3 # Fichier modèle
4
5 def somme_ligne(carre, n):
6     """ carre est un tableau carre de nombres
7     n est un nombre entier
  
```

- L'étudiant voit le résultat de ses tests

- Dans l'onglet « Submission » l'étudiant retrouve les tests qui ont échoué

Accueil / Mes cours / Stage Parcours / VPL / Exemple 4

Description Edit Submission view

### Note

Reviewed on mardi 28 janvier 2020, 14:59 by Note automatique  
grade: 20,00 / 100,00

**Assessment report[-]**

**[-]Failed tests**

Test 2: fonction check\_row(carré)  
 Test 3: fonction somme\_col(carré, numéro de ligne)  
 Test 4: fonction check\_col(carré)  
 Test 5: fonction is\_magic(carré)

**[+]Test 2: fonction check\_row(carré)**  
**[+]Test 3: fonction somme\_col(carré, numéro de ligne)**  
**[+]Test 4: fonction check\_col(carré)**  
**[+]Test 5: fonction is\_magic(carré)**  
**[+]Summary of tests**

Proposed grade: 20 / 100

Commentaires

**Failed tests**

Test 2: fonction check\_row(carré)  
 Test 3: fonction somme\_col(carré, numéro de ligne)  
 Test 4: fonction check\_col(carré)  
 Test 5: fonction is\_magic(carré)  
**Test 2: fonction check\_row(carré)**  
 Incorrect program output  
 --- Input ---  
 u2  
 --- Program output ---  
 False  
 --- Expected output (text)---  
 "True"

- Comme dans les autres scénario le professeur peut observer les travaux rendus.

Description Liste des devoirs rendus Similarité Activité test

Submission selection Tous les envois

Evaluate Choisir...

	Prénom	Nom	Submitted on	Devoirs rendus	Note	Evaluator	Evaluated on
1	eleve graves		mardi 28 janvier 2020, 14:01	2	0,00 / 100,00	Note automatique	mardi 28 janvier 2020, 14:10





## Ressources

- <https://vpl.dis.ulpgc.es/>
- [http://www.science.smith.edu/dftwiki/index.php/Moodle\\_VPL\\_Tutorials](http://www.science.smith.edu/dftwiki/index.php/Moodle_VPL_Tutorials)





## Auteurs

- [Xavier.carbonnaux@ac-bordeaux.fr](mailto:Xavier.carbonnaux@ac-bordeaux.fr)
- [Luc.vincent@ac-bordeaux.fr](mailto:Luc.vincent@ac-bordeaux.fr)