

# Environnements de travail

# Table des matières

<b>I. IDE (environnement de développement), PyCharm</b>	<b>3</b>
<b>II. Exercice : Quiz</b>	<b>6</b>
<b>III. Environnement virtuel (virtualenv, venv, requirements.txt)</b>	<b>7</b>
<b>IV. Exercice : Quiz</b>	<b>8</b>
<b>V. Convention de code PEP8</b>	<b>8</b>
<b>VI. Exercice : Quiz</b>	<b>10</b>
<b>VII. Essentiel</b>	<b>11</b>
<b>VIII. Auto-évaluation</b>	<b>11</b>
A. Exercice .....	11
B. Test .....	11
<b>Solutions des exercices</b>	<b>12</b>

## I. IDE (environnement de développement), PyCharm

### Contexte

Dans ce cours, nous allons voir différentes manières de travailler avec Python. Tout d'abord Pycharm qui est un IDE (*Integrated Development Environment*), puis l'environnement virtuel, et enfin PEP8 qui représente les bonnes pratiques de code en Python. Un environnement virtuel est simplement un environnement qui est isolé de votre système.

Il y a plusieurs intérêts : le premier est que vous ne voulez probablement pas polluer le Python de votre système avec des bibliothèques qui ne sont utiles que pour un projet en particulier. Le deuxième est que cela évitera que tous vos projets soient pollués par cette même bibliothèque que vous allez utiliser uniquement pour l'un d'entre eux.

### Qu'est-ce que l'IDE ?

Un IDE, abréviation de *Integrated Development Environment*, est l'un des outils de programmation les plus puissants qui combine les différents aspects d'un programme informatique en une seule interface graphique. Il s'agit d'un environnement logiciel robuste qui consolide de nombreuses fonctions telles que la création, la construction et les tests de code, dans un cadre, un service ou une application unique. Ce qu'il fait, c'est qu'il simplifie l'ensemble du processus de développement logiciel en permettant la gestion et le déploiement de plusieurs outils à la fois. Un IDE est un éditeur de texte, un éditeur de code, un débogueur, un compilateur et plus encore. Il existe différents types d'IDE, certains provenant de grandes organisations dont vous avez probablement déjà entendu parler, comme Visual Studio de Microsoft, Xcode d'Apple, etc. En termes simples, un IDE est une suite d'outils de développement logiciel spécialement conçus pour faciliter le codage, vous permettant de travailler sans effort sur différents modules du même projet de manière organisée.

### Qu'est-ce qu'un éditeur de code ?

L'éditeur de code est l'un des outils les plus importants pour les programmeurs, spécialement conçu pour éditer le code source des programmes informatiques. Un éditeur de code est essentiellement un éditeur de texte, mais il est également conçu pour vous aider à écrire du code. Il vous aide à colorer votre code et vous fournit des outils plus avancés pour vous faciliter le codage. Un simple éditeur de code peut être un logiciel autonome ou faire partie d'un IDE ou d'un navigateur Web. Si vous êtes un débutant qui apprend à coder dans n'importe quel langage, vous devez utiliser un éditeur de code. C'est comme un éditeur de texte mais avec plus de fonctionnalités intégrées. Ces fonctionnalités simplifient et accélèrent le processus d'édition et vous aident à écrire de meilleurs programmes logiciels en identifiant les zones problématiques et en déboguant le code. Il existe plusieurs éditeurs de code, tels que Atom, Sublime Text, Visual Studio Code, etc.

### Qu'est-ce que PyCharm ?

Disposer d'un environnement de travail adapté est important dès lors que l'on travaille sur des projets d'une taille moyenne à conséquente.

PyCharm est un IDE qui assure l'essentiel en proposant les fonctionnalités indispensables que sont la coloration syntaxique, l'autocomplétion ainsi que la détection d'erreurs ou d'avertissements (lié à PEP8). Il permet également de formater rapidement et efficacement son code et il propose une multitude de fonctionnalités facilitant l'écriture, la lecture et la maintenance de son code.

Il existe une version communautaire disposant des fonctionnalités essentielles ainsi que d'une version payante disposant de fonctionnalités plus avancées. PyCharm fait partie de la suite D'IDE développée par l'entreprise JetBrains.

## Configuration de Pycharm

Lors du premier démarrage de PyCharm, ce dernier vous demandera si vous souhaitez importer des informations depuis d'autres IDE.

Sous Windows, il saura trouver seul l'ensemble des versions de Python que vous avez installées. Sous Linux et Mac, il trouvera celles qui sont packagées par le système et non celles que vous avez compilées par vous-même. Il faudra donc lui indiquer si vous souhaitez les utiliser (que ce soit pour l'analyse de la syntaxe et pour exécuter un projet).

Pour ce faire, vous devez accéder à la configuration (menu File - Settings) et à la section Project-Python interpréter. Lorsque vous êtes sur cette page, vous pouvez sélectionner un interpréteur pour votre projet parmi ceux présents et en rajouter un nouveau en cliquant sur l'engrenage à droite. Il vous faudra aller chercher le fichier exécutable Python correspondant à la version que vous voulez ajouter.

Notez qu'à l'aide de ce même bouton en forme d'engrenage vous pouvez aussi créer un environnement virtuel.

Dans la seconde partie de l'écran, vous pouvez visualiser l'ensemble des bibliothèques installées pour la version de Python sélectionnée, leur numéro de version ainsi que le numéro de la version la plus récente. Vous pouvez faire la mise à jour d'une bibliothèque en cliquant sur la flèche.

Enfin pour ajouter une bibliothèque, il faut que le « + » soit à droite. Maîtriser cette interface vous permettra d'éviter de toucher au terminal.

## Changez vos raccourcis

PyCharm vous permet d'adapter rapidement votre environnement de développement à vos préférences. Vous pouvez facilement conserver vos raccourcis favoris en recherchant, modifiant et personnalisant les raccourcis de votre IDE. Ainsi, il améliore votre productivité et votre capacité à accéder plus rapidement aux fonctionnalités. Ouvrez la boîte de dialogue « Paramètres/Préférences » (Ctrl + Alt + S) pour afficher la configuration du clavier et sélectionnez Keymap. Cette liste contient les actions et tâches disponibles pour lesquelles vous pouvez ajouter ou personnaliser vos raccourcis ; ils sont ordonnés afin que vous puissiez trouver des actions spécifiques.

### Exemple

L'action Join Lines est associée au raccourci « *Control + Shift + J* » dans Windows. De plus, certains éléments disponibles figurent dans notre liste d'actions ; certains ne sont encore associés à aucun raccourci (par exemple, déplacer le curseur vers l'arrière d'un paragraphe). Cependant, vous pouvez leur attribuer différentes combinaisons de touches en cliquant avec le bouton droit sur les actions et en créant un raccourci clavier ou une combinaison de raccourci souris.

## Personnalisez le thème de votre éditeur

Plus vous utiliserez d'IDE, plus il sera efficace de personnaliser le thème de l'éditeur indépendamment du thème général de l'environnement. La plupart des développeurs aiment personnaliser les fonctionnalités de leur code, telles que la police.

## Explorez les principales fonctionnalités de l'éditeur

Selon le développeur, JetBrains, PyCharm fonctionne sur les deux principes clés suivants :

1. Amélioration de la productivité : PyCharm permet aux data scientists de se concentrer plus facilement sur la construction de leurs modèles en éliminant les aspects répétitifs et non créatifs de la programmation Python.
2. Assistance en temps réel : PyCharm offre aux scientifiques des données, diverses fonctionnalités intelligentes pour résoudre les incohérences en vérifiant en temps réel la logique des erreurs, les styles de code PEP 8 et en suggérant des solutions rapides, entre autres fonctionnalités essentielles. PyCharm propose également diverses fonctionnalités de niveau supérieur qui contribuent à promouvoir ces principes :
  - Prise en charge du calcul scientifique : PyCharm inclut de nombreuses fonctionnalités qui rationalisent le travail des data scientists en facilitant le calcul scientifique via la note IPython et les fonctionnalités de la console interactive.

- Assistance intelligente au codage : PyCharm offre des options de complétion de code, des analyses de syntaxe et d'erreur, ainsi que des moyens automatisés de refactoriser votre code.
- Outils de programmation rationalisés : PyCharm permet aux data scientists de déboguer, tester et gérer efficacement leur code.
- Options de développement Web : PyCharm prend en charge les projets de développement web, tels que les frameworks de développement web Python, et d'autres éléments majeurs tels que JavaScript.

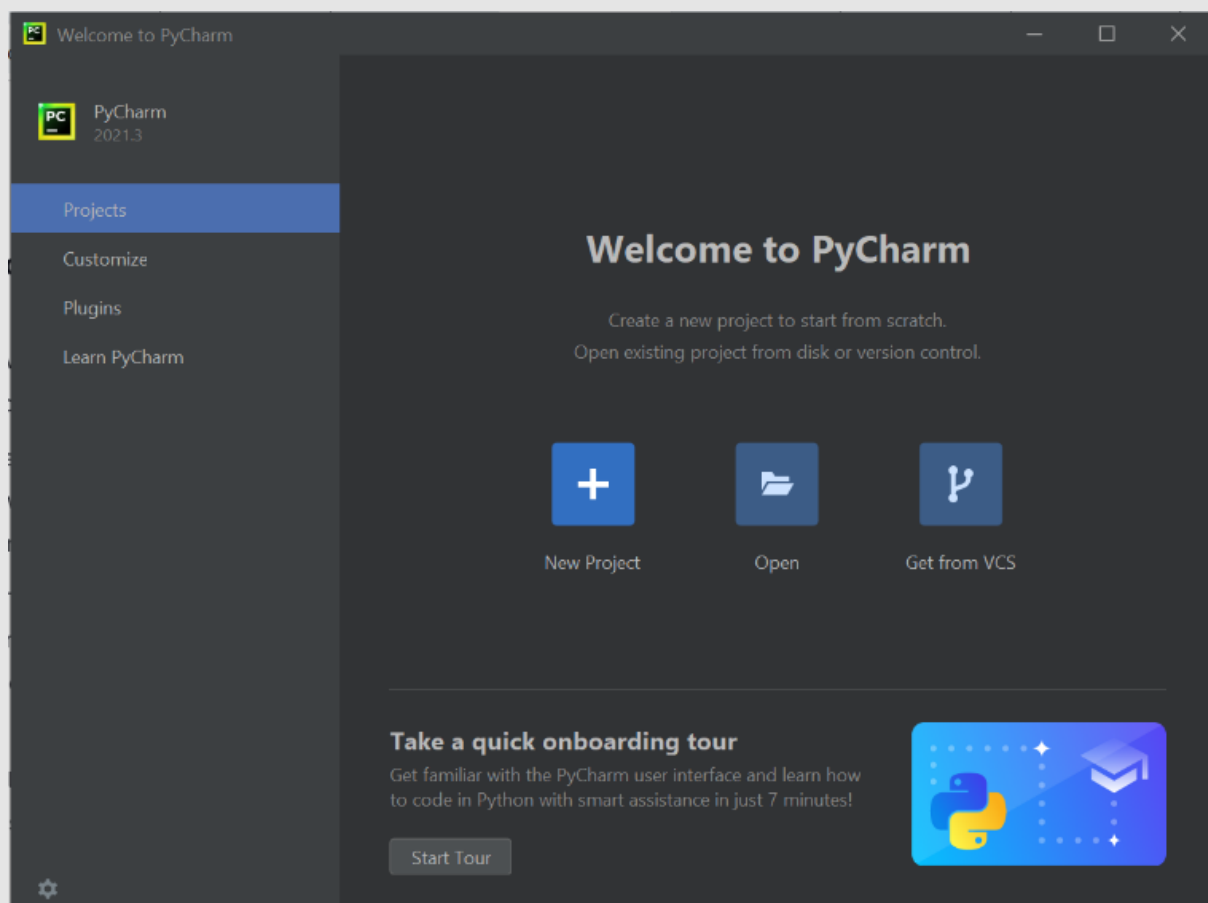
### Méthode Créer un nouveau projet

Maintenant que nous avons vu comment PyCharm a rationalisé votre processus de développement, il est temps de créer notre premier projet.

Si le processus d'installation se déroule avec succès, vous pouvez désormais lancer votre IDE sans erreur.

Votre écran de bienvenue devrait ressembler à ce qui suit.

Cliquez sur le bouton « **Nouveau projet** » pour finaliser le processus. PyCharm vous amènera à une autre fenêtre pour saisir les détails de votre nouveau projet. Par défaut, il suggère les paramètres dans la capture d'écran ci-dessous :



### Écran de lancement Pycharm

Cliquez sur le bouton « **Nouveau projet** » pour finaliser le processus, Ajustez les paramètres du projet à vos besoins, puis cliquez sur « **Créer** ». Si vous souhaitez ajouter un nouveau fichier à votre projet, vous pouvez cliquer avec le bouton droit sur le projet dans le panneau de gauche et choisir « **Nouveau** » → « **Fichier Python** ». Vous pouvez créer plusieurs types de fichiers. Le « .py » n'est pas nécessaire dans le nom du fichier puisque vous avez déjà spécifié que vous créez un fichier Python. Cliquez sur « **OK** » pour continuer.

Une fois le fichier créé avec succès, il s'ouvre par défaut. Il ne vous reste plus qu'à coder.

### Codage productif

De nombreux aspects sont impliqués dans le développement de logiciels ou dans la mise en œuvre d'un projet de science des données qui va au-delà du codage ; ces activités peuvent impliquer des tests, du débogage et du profilage. Cependant, compte tenu de leur complexité et de la profondeur des connaissances requises pour effectuer de telles tâches, de nombreux développeurs sont intimidés et ont donc tendance à les ignorer dans leurs projets. Heureusement, c'est là que PyCharm intervient pour offrir un moyen pratique de suivre les meilleures pratiques de programmation en temps réel pendant que vous construisez vos projets. De plus, les tâches de gestion ennuyeuses d'organisation des packages, des interpréteurs et des environnements virtuels sont également prises en charge de manière simple. PyCharm fournit non seulement des méthodes intuitives et graphiques pour effectuer cela, mais il fournit également une assistance de codage intelligente, une coloration syntaxique et des suggestions en parallèle.

### Exercice : Quiz

[solution n°1 p.13]

#### Question 1

Quelle est la norme qui régit la correction de code sur PyCharm ?

- ☐ PEP8
- ☐ PEP11
- ☐ PY12

#### Question 2

Pycharm est disponible sur tous les systèmes d'exploitation.

- ☐ Vrai
- ☐ Faux

#### Question 3

Pycharm est développé par JetBrains.

- ☐ Vrai
- ☐ Faux

#### Question 4

PyCharm est :

- ☐ Un éditeur de code
- ☐ Un éditeur de texte
- ☐ Un IDE

#### Question 5

Pycharm permet de télécharger et d'installer Python.

- ☐ Vrai
- ☐ Faux

### III. Environnement virtuel (virtualenv, venv, requirements.txt)

Vous allez peut-être devoir développer un nouveau site internet avec Django en même temps que vous assurez la maintenance de deux autres sites avec d'autres versions de Django. Vous n'allez pas passer votre temps à changer de version.

Vous allez créer un environnement virtuel pour chaque nouveau projet que vous conduirez, de préférence avec des versions identiques à celles que vous utiliserez en production. L'environnement virtuel est un élément indispensable dans le cadre professionnel.

La possibilité de créer des environnements virtuels est livrée d'office à partir de Python 3.3 et avec les versions suivantes, sous la forme du module venv. Le script pyenv a été déprécié dans la version 3.6 de Python. On utilise la commande venv directement dans son terminal.

```
1 venv path/to/my/env
```

Une fois l'environnement créé, il est activable ainsi :

```
1 path/to/my/env/bin/activate
```

Pour Windows c'est un peu différent :

```
1 C : \\path\tp\my\env\Scripts\activate.bat
```

Une fois l'environnement activé, l'exécutable Python utilisé par le terminal devient alors celui de l'environnement virtuel (mais juste pour le terminal courant). De la même manière, toutes les bibliothèques externes sont celles de l'environnement local. Cela permet de ne pas polluer notre environnement Python habituel.

Pour toutes les versions de Python inférieure à la 3.2 il faut utiliser le module virtualenv, il faut donc installer virtualenv.

```
1 $ sudo apt install python3-virtualenv
```

On rappelle qu'il faut installer l'exécutable pip quand on utilise virtualenv, il est installé par défaut avec venv. Cette opération n'est à faire qu'une seule fois pour se donner la possibilité de créer des environnements virtuels ainsi :

```
1 virtualenv -p python3.5 path/to/my/env
```

Une fois l'environnement virtuel activé on constate que l'invite de commande a changé.

```
1 source ~ /.virtualenvs/path/to/my/env/bin/activate
2 (env) user@localhost:~
```

On constate que l'exécutable Python qu'on lance n'est pas celui de notre système mais bien celui de notre environnement virtuel, preuve que ce dernier est bien lancé.

```
1 user@localhost :~ $ which python
2 usr/bin/python
3 source ~ /.virtualenvs/path/to/my/env/bin/activate
4 (env) user@localhost:~ which python
5 home/user/.virtualenvs/path/to/my/env/bin/
```

Ainsi, on peut maintenant installer toutes les bibliothèques souhaitées dans notre environnement virtuel.

Pour quitter ce dernier, il suffit d'utiliser la commande suivante :

```
1 (env) user@localhost:~ deactivate
```

Sachez que pour répliquer une installation d'un environnement à un autre sur la même machine, vous pouvez utiliser cette commande de l'environnement source. L'option « -l » permet de ne sélectionner que les paquets locaux (et non les paquets issus du système qui peuvent également être accessibles depuis les environnements virtuels).

```
1 pip freeze -l > requirements.txt
```

On peut ensuite le copier dans l'environnement cible depuis le requirements.txt avec cette commande :

```
1 $ pip install -r requirements/base.txt
```

**Attention**

Lorsque vous avez fini, pensez à bien désactiver votre environnement virtuel grâce à la commande :

```
1 (env) user@localhost:~ deactivate
```

## Exercice : Quiz

[solution n°2 p.14]

### Question 1

Quelles sont les commandes qui permettent de créer un environnement virtuel ?

- ☐ Venv
- ☐ pvenv
- ☐ virtualenv

### Question 2

Il faut installer la commande virtualenv avant de l'utiliser.

- ☐ Vrai
- ☐ Faux

### Question 3

On peut activer ou désactiver un environnement virtuel.

- ☐ Vrai
- ☐ Faux

### Question 4

Que fait l'activation d'un environnement virtuel ?

- ☐ Elle désactive l'exécutable Python système
- ☐ Elle crée un lien entre l'exécutable Python système et celui de l'environnement virtuel
- ☐ Elle crée un nouveau dossier où elle stocke des fichiers de configuration locaux

### Question 5

Quand on utilise venv, le gestionnaire de paquets pip est fourni par défaut.

- ☐ Vrai
- ☐ Faux

## V. Convention de code PEP8

### Qu'est-ce que PEP8

PEP8 (pour *Python Extension Proposal*) est un ensemble de règles qui permet de standardiser le code et d'appliquer les bonnes pratiques. L'exemple le plus connu est la guerre entre les développeurs pour savoir s'il faut indenter leur code avec des espaces ou avec des tabulations. Le PEP8 décide : ce sont les espaces qui gagnent, au nombre de 4. Fin du débat.

C'est une logique très intéressante pour Python qui se veut un langage « *lisible* » plutôt qu'« *un langage écrit* ». Écrire un code compréhensible pour une machine est plutôt facile, mais écrire un code compréhensible par le plus grand nombre de codeurs (de tout niveau) est une autre histoire...



Comme l'a dit Guido van Rossum, « *le code est lu beaucoup plus souvent qu'il n'est écrit* ». Vous pouvez passer quelques minutes, voire une journée entière, à écrire un morceau de code pour traiter l'authentification de l'utilisateur. Une fois que vous l'avez écrit, vous n'allez plus jamais l'écrire. Mais vous devrez certainement le relire. Ce morceau de code peut continuer à faire partie d'un projet sur lequel vous travaillez. Chaque fois que vous revenez à ce fichier, vous devrez vous rappeler ce que fait ce code et pourquoi vous l'avez écrit, donc la lisibilité est importante.

Si vous débutez avec Python, il peut être difficile de se souvenir de ce que fait un morceau de code quelques jours ou semaines après l'avoir écrit. Si vous suivez la PEP8, vous pouvez être sûr que vous avez bien nommé vos variables. Vous saurez que vous avez ajouté suffisamment d'espaces blancs pour qu'il soit plus facile de suivre les étapes logiques de votre code. Vous aurez également bien commenté votre code. Tout cela signifie que votre code est plus lisible et plus facile à consulter. En tant que débutant, suivre les règles de PEP8 peut rendre l'apprentissage de Python beaucoup plus agréable.

Suivre le PEP8 est particulièrement important si vous recherchez un emploi en développement. Rédiger un code clair et lisible fait preuve de professionnalisme. Cela indiquera à un employeur que vous comprenez comment bien structurer votre code.

Si vous avez plus d'expérience dans l'écriture de code Python, vous devrez peut-être collaborer avec d'autres. L'écriture de code lisible ici est cruciale. D'autres personnes, qui ne vous ont peut-être jamais rencontré ou qui n'ont jamais vu votre style de codage auparavant, devront lire et comprendre votre code. Avoir des directives que vous suivez et reconnaissez facilitera la lecture de votre code par les autres.

## Règles fondamentales

Ici nous allons décrire quelques règles fondamentales de la norme PEP8. Cette norme est très longue nous n'allons donc pas la décrire entièrement. Libre à vous d'aller consulter l'entièreté de la norme sur le site officiel :

PEP 8 – Style Guide for Python Code<sup>1</sup>

- **Comment choisir des noms**

Choisir des noms pour vos variables, fonctions, classes, etc., peut être difficile. Vous devez réfléchir sérieusement à vos choix de nommage lors de l'écriture du code, car cela rendra votre code plus lisible. La meilleure façon de nommer vos objets en Python est d'utiliser des noms descriptifs pour indiquer clairement ce que l'objet représente.

Lorsque vous nommez des variables, vous pouvez être tenté de choisir des noms simples à une lettre minuscule, comme `x`. Mais, à moins que vous n'utilisiez `x` comme argument d'une fonction mathématique, ce que `x` représente n'est pas clair.

- **Lignes vides**

Les espaces blancs verticaux, ou les lignes vides, peuvent grandement améliorer la lisibilité de votre code. Le code regroupé peut être écrasant et difficile à lire. De même, trop de lignes vides dans votre code le rendent très clairsemé et le lecteur peut avoir besoin de faire défiler plus que nécessaire. Vous trouverez ci-dessous trois directives clés sur la façon d'utiliser les espaces blancs verticaux.

- Entourer les fonctions et les classes de niveau supérieur avec deux lignes vides,
- Entourer les définitions des méthodes dans les classes avec un seul espace blanc,
- Utiliser des lignes vides avec parcimonie dans les fonctions pour montrer des étapes claires.

- **Longueur de ligne maximale et rupture de ligne**

PEP8 suggère que les lignes soient limitées à 79 caractères. En effet, cela vous permet d'avoir plusieurs fichiers ouverts les uns à côté des autres, tout en évitant le retour à la ligne.

Bien entendu, il n'est pas toujours possible de limiter les déclarations à 79 caractères ou moins.

---

<sup>1</sup> <https://peps.python.org/pep-0008/>

- **Indentation**

L'indentation, ou espace blanc de début, est extrêmement importante en Python. Le niveau d'indentation des lignes de code en Python détermine la manière dont les instructions sont regroupées.

Les principales règles d'indentation énoncées par la PEP8 sont les suivantes :

- Utiliser 4 espaces consécutifs pour indiquer l'indentation,
- Préférer les espaces aux tabulations.

- **Tabulation ou Espaces**

Comme mentionné ci-dessus, vous devez utiliser des espaces au lieu des tabulations lors de l'indentation du code. Vous pouvez ajuster les paramètres de votre éditeur de texte pour afficher 4 espaces au lieu d'un caractère de tabulation, lorsque vous appuyez sur la touche Tab.

## Exercice : Quiz

[solution n°3 p.15]

### Question 1

La norme PEP8 permet...

- ☐ D'écrire du code de façon lisible pour tous
- ☐ De donner une taille minimum aux fichiers Python
- ☐ De définir un cadre de travail commun

### Question 2

L'encodage par défaut défini par PEP8 est UTF-8.

- ☐ Vrai
- ☐ Faux

### Question 3

L'indentation doit être faite avec des tabulations.

- ☐ Vrai
- ☐ Faux

### Question 4

L'indentation doit comporter combien de caractères ?

- ☐ 2
- ☐ 3
- ☐ 4

### Question 5

Les commentaires doivent obligatoirement être écrit en anglais.

- ☐ Vrai
- ☐ Faux

## VII. Essentiel

Vous êtes maintenant capable d'installer un environnement de travail confortable avec Python. Vous avez vu l'IDE le plus répandu pour travailler avec Python, comment le configurer et l'utiliser. Vous avez vu qu'il est possible de segmenter ses espaces de travail pour ne pas polluer l'espace de travail principal et installer des packages Python sur le binaire système. Et pour finir vous avez reçu une présentation succincte de PEP8 la norme de code de Python. Vous êtes maintenant capable de coder de manière propre et lisible tout en ayant un environnement de développement sain.

## VIII. Auto-évaluation

### A. Exercice

Vous voulez développer votre propre module Python, vous décidez donc de réaliser l'action confortablement et par conséquent cela passe par l'installation d'un bon environnement de développement. Vous utiliserez une version de Python supérieure à la 3.3.

#### Question 1

[solution n°4 p.16]

Vous décidez d'utiliser PyCharm, comment aller vous procéder après son installation ?

#### Question 2

[solution n°5 p.16]

Pour segmenter votre travail vous décidez de vous créer un environnement virtuel comment allez-vous vous y prendre ?

### B. Test

#### Exercice 1 : Quiz

[solution n°6 p.16]

##### Question 1

Quel est le meilleur endroit pour stocker un environnement virtuel lorsque vous travaillez sur un projet ?

- ☐ N'importe où dans le dossier du projet
- ☐ Au même endroit que tous vos autres environnements virtuels
- ☐ Dans la racine du projet

##### Question 2

Comment nommer vos environnements virtuels ?

- ☐ Venv
- ☐ <nom\_du\_projet>
- ☐ env

##### Question 3

Quel est l'intérêt d'un environnement virtuel ?

- ☐ Protéger les fichiers Python système
- ☐ Gérer différents projets sur différentes versions de Python
- ☐ Accéder à d'autres bibliothèques

##### Question 4

PyCharm nécessite de configurer :

- ☐ Le compilateur
- ☐ L'interpréteur
- ☐ Les deux

Question 5

Quelle commande permet de désactiver un environnement virtuel ?


- ☐ Deactivate
- ☐ Clear
- ☐ Stop

## Solutions des exercices

**Exercice p. 6 Solution n°1****Question 1**

Quelle est la norme qui régit la correction de code sur PyCharm ?


- ☒ PEP8
- ☐ PEP11
- ☐ PY12

 La norme qui régit la correction de code sur PyCharm est la norme PEP8, c'est aussi elle qui régit l'écriture de code en général sur Python.

**Question 2**

Pycharm est disponible sur tous les systèmes d'exploitation.


- ☒ Vrai
- ☐ Faux

 PyCharm est téléchargeable sur Windows, Mac et Linux.

**Question 3**

Pycharm est développé par JetBrains.


- ☒ Vrai
- ☐ Faux

 PyCharm fait partie de la suite D'IDE développée par l'entreprise JetBrains.

**Question 4**

PyCharm est :


- ☐ Un éditeur de code
- ☐ Un éditeur de texte
- ☒ Un IDE

 PyCharm est un IDE car il inclut des fonctions d'édition mais aussi des fonctions qui permettent de lancer une application, la déboguer ou encore lancer une suite de test.

**Question 5**

Pycharm permet de télécharger et d'installer Python.

- ☐ Vrai
- ☒ Faux


 PyCharm ne permet pas d'installer Python, il faut le faire au préalable.

## Exercice p. 8 Solution n°2

### Question 1

Quelles sont les commandes qui permettent de créer un environnement virtuel ?


- ☒ Venv
- ☐ pvenv
- ☒ virutalenv

 Pour les versions de Python supérieur à la 3.3, la commande est venv, pour celles qui sont inférieures la commande est virtualenv.

### Question 2

Il faut installer la commande virtualenv avant de l'utiliser.


- ☒ Vrai
- ☐ Faux

 La commande virtualenv fait partie d'un module qu'il faut installer avant de pouvoir l'utiliser.

### Question 3

On peut activer ou désactiver un environnement virtuel.


- ☒ Vrai
- ☐ Faux

 On peut l'activer ou le désactiver à notre convenance, cependant attention il n'est actif que pour le terminal en cours.

### Question 4

Que fait l'activation d'un environnement virtuel ?


- ☒ Elle désactive l'exécutable Python système
- ☐ Elle crée un lien entre l'exécutable Python système et celui de l'environnement virtuel
- ☒ Elle crée un nouveau dossier où elle stocke des fichiers de configuration locaux

 Elle crée un lien symbolique entre l'exécutable système et l'exécutable local et elle crée un dossier contenant des fichiers de configuration locaux. Cette action permet de désolidariser l'exécutable local de celui du système.

### Question 5


Quand on utilise venv, le gestionnaire de paquets pip est fourni par défaut.

- ☒ Vrai
- ☐ Faux

 Lors de l'utilisation de venv, pip est fourni par défaut cependant ce n'est pas le cas avec virtualenv.


**Exercice p. 10 Solution n°3****Question 1**

La norme PEP8 permet...

- ☒ D'écrire du code de façon lisible pour tous
- ☐ De donner une taille minimum aux fichiers Python
- ☒ De définir un cadre de travail commun
-  Elle régit les normes d'écritures en Python.


**Question 2**

L'encodage par défaut défini par PEP8 est UTF-8.

- ☒ Vrai
- ☐ Faux
-  PEP8 précise que l'encodage doit être en UTF-8.


**Question 3**

L'indentation doit être faite avec des tabulations.

- ☐ Vrai
- ☒ Faux
-  L'indentation doit être faite avec des espaces.


**Question 4**

L'indentation doit comporter combien de caractères ?

- ☐ 2
- ☐ 3
- ☒ 4
-  L'indentation doit comporter 4 caractères espace.

**Question 5**

Les commentaires doivent obligatoirement être écrit en anglais.

- ☐ Vrai
- ☒ Faux
-  Ce n'est pas obligatoire mais c'est fortement conseillé pour la compréhension internationale surtout si vous codez un module.

#### p. 11 Solution n°4

Il faut commencer par savoir quel système d'exploitation vous utilisez pour pouvoir faire un choix dans la méthode de configuration. En effet, si vous êtes sous Windows, toutes les versions de Python seront trouvées cependant si vous êtes sous Mac ou Linux, il faudra indiquer à PyCharm ou sont les autres versions de Python sur votre système.

#### p. 11 Solution n°5

Vous utilisez Python supérieur à la 3.3, il faudra donc utiliser la commande venv pour créer votre environnement et appliquer la procédure suivante :


```
1 venv path/to/my/env
2 source path/to/my/env/activate
```

Une fois que cette suite de commande sera effectuée, vous serez prêt à travailler.

#### Exercice p. 11 Solution n°6


##### Question 1

Quel est le meilleur endroit pour stocker un environnement virtuel lorsque vous travaillez sur un projet ?

- ☐ N'importe où dans le dossier du projet
- ☐ Au même endroit que tous vos autres environnements virtuels
- ☒ Dans la racine du projet
-  Le meilleur endroit pour stocker un environnement virtuel lorsque l'on travaille sur un projet est de le mettre dans la racine du projet.


##### Question 2

Comment nommer vos environnements virtuels ?

- ☐ Venv
- ☐ <nom\_du\_projet>
- ☒ env
-  Le meilleur nom pour vos environnements est env.

##### Question 3

Quel est l'intérêt d'un environnement virtuel ?


- ☒ Protéger les fichiers Python système
- ☒ Gérer différents projets sur différentes versions de Python
- ☐ Accéder à d'autres bibliothèques
-  L'intérêt est de protéger les fichiers Python système et de se désolidariser de la version système Python.



**Question 4**

---


PyCharm nécessite de configurer :

- ☐ Le compilateur
- ☒ L'interpréteur
- ☐ Les deux
-  PyCharm a juste besoin de l'interpréteur pour fonctionner correctement.

**Question 5**

---

Quelle commande permet de désactiver un environnement virtuel ?

- ☒ Deactivate
- ☐ Clear
- ☐ Stop
-  C'est la commande deactivate qui permet de désactiver l'environnement de travail.