TP

Python

React IT, Loïc Guillois

Version 1.0, 2020-04-15

Table des matières

Chapitre 1. Installation des outils de développement

1.1. Environnement de développement

Avant de commencer le TP, vous devez avoir un environnement Python installé. Après installation, vous pouvez tester le bon fonctionnement en exécutant Python dans un terminal:

```
$ python
```

Pour fermer la console, vous pouvez tapper sur les touches ctrl+D.

Pour être plus efficace en programmation, il faut un bon éditeur. Vous êtes libre d'utiliser celui de votre choix mais il est recommandé d'utiliser un environnement évolué comme Visual Studio code. Prenez le temps de l'installer si ce n'est pas déjà le cas.

1.2. Environnement interactif

Jupyter est un outil qui permet de développer directement depuis le navigateur. Il apporte l'avantage de pouvoir exécuter directement certaines portions de code sans devoir recharger et exécuter à nouveau tout le programme. C'est intéressant si vous voulez vous concentrer sur une partie du code ou si vous souhaitez éviter des traitements long en début de programme: typiquement le chargement de données et les calculs complexes.

La procédure d'installation est un peu plus complexe et dépend de votre système (Linux, Mac, Windows). Pour faire simple et efficace, vous pouvez suivre cette documentation: https://openclassrooms.com/fr/courses/4452741-decouvrez-les-librairies-python-pour-la-data-science/5559646-installez-jupyter-sur-votre-propre-ordinateur

Pour la suite du TP, vous serez libre d'utiliser Jupyter ou un éditeur de texte plus conventionnel.

Chapitre 2. Créer votre premier programme Python

UTF-8 est un encodage universel qui a pour objectif de rénuir les caractères utilisés par toutes les langues. Il n'y a donc en théorie plus de problèmes de communication si tous les programmes sont encodés avec de l' UTF-8.

Par défaut dans python 2.7 l'encoding est ASCII , il est donc préférable d'indiquer l'encodage UTF-8. Pour cela il vous faudra indiquer dans l'entête du fichier la ligne suivante:

```
# -*- coding: utf-8 -*-
```

Sous Linux et Mac, il est préférable de préciser la commande d'exécution du programme. S'agissant d'un commentaire, il n'a aucun impact négatif sous Windows. Par soucis de respect de l'aspect multiplateforme du langage Python, vous positionnerez systématiquement cette ligne en haut de fichier, juste après l'encodage :

```
#!/usr/bin/python
```

Si l'on résume, voici votre premier programme :

```
# -*- coding: utf-8 -*-
#!/usr/bin/python
print("hello world")
```

Vous l'enregistrer avec l'extension .py soit par exemple : hello.py.

Il vous restera a exécuter votre programme:

```
$ python hello.py
```

Sous Linux, vous pouvez donner les droits en exécutions sur le fichier puis l'exécuter simplement:

```
$ chmod a+x hello.py
$ ./hello.py
```

Grâce aux informations de la ligne numéro 2, le shell sait quel programme exécuté pour lancer notre programme Python.

Chapitre 3. TP 1 : Initiation à Python

Pour chaque exercice, prenez soin de conserver un code source séparé.

3.1. Mise en place de notre programme de caisse

A l'aide de ce que nous avons pu voir dans la première partie du cours, vous allez devoir réaliser un programme Python qui permet de traiter les actions suivantes:

- Demander à l'utilisateur de saisir un prix hors taxe (HT) ainsi qu'une quantité
- Calculer et afficher le prix total, sur la base d'une TVA de 20%
- Si le total dépasse les 200€, vous effectuerez une remise de 5%

Vous prendrez soin de vérifier les saisies utilisateurs : un prix et une quantité ne peuvent pas être nul ou négatif.

3.2. Gestion de plusieurs produits

Vous modifierez le programme précédent pour permettre la gestion de plusieurs produits. Il faudra donc demander à l'utilisateur combien de produits différents seront à saisir.

Vous devrez utiliser des boucles pour la saisie (for ou while) et le calcul.

Chapitre 4. TP 2 : Découverte du langage

4.1. Utilisation de données structurées

Cette fois si ce n'est plus l'utilisateur qui devra saisir le prix HT mais les prix seront stockés dans un dictionnaire Python. La correspondance référence / produit est la suivante:

Id	Nom	Prix
1	Banane	4
2	Pomme	2
3	Orange	1.5
4	Poire	3

Modifier le programme précédent pour permettre la saisir de l'id plutôt que du prix HT. Vous traiterez les erreurs de saisies utilisateurs par des exceptions plutôt que par des blocs conditionnels if.

Votre programme devra afficher ce type de résultat:

Nom	Prix	Quantité	Total HT
Banane	4	2	8
Pomme	2	1	2
Poire	3	5	15

```
+---------------------------------+---------+----------------------+----------------+
|          Nom                    |  Prix   |     Quantité         |     Total HT   |
+---------------------------------+---------+----------------------+----------------+
| Banane                          | 4       | 2                    |          8     |
| Pomme                           | 2       | 1                    |          2     |
| Poire                           | 3       | 5                    |         15     |
+---------------------------------+---------+----------------------+----------------+

Total HT = 25€
Total TTC = 30€
```

ou

```
+--------------------------------+---------+----------------------+---------------+
|            Nom                 |  Prix   |      Quantité        |     Total HT  |
+--------------------------------+---------+----------------------+---------------+
| Banane                         | 4       | 20                   |          80   |
| Pomme                          | 2       | 10                   |          20   |
| Poire                          | 3       | 50                   |         150   |
+--------------------------------+---------+----------------------+---------------+

Sous-Total HT = 250€
Remise 5% = 12.5€
Total HT = 237.5€
Total TTC = 285€
```

Vous n'y avez peut être pas penser, mais que se passe-t-il si les prix sont en centimes ? Si vous appliquer une remise, vous aurez un problème sur le nombre de chiffres après la virgules. Utilisez la bibliothèque math pour effectuer un arrondi au centime le plus proche.

4.2. Structuration de votre programme

Nous avons terminé d'ajouter des fonctionnalités à notre programme. Désormais vous aller devoir faire un peu de refactoring de votre code pour que celui ci soit plus facilement lisible.

Pour se faire, vous découperez votre code en plusieurs fonctions et en plusieurs modules. L'idée est de rendre votre programme réutilisable facilement.

Chapitre 5. TP 3 : Programmation orientée objet

Reprenez le TP précédent avec pour objectif d'effectuer un refactoring de votre code. Afin de le rendre encore plus facilement réutilisable, nous allons modifier notre module avec les méthodes de calcul pour les regrouper au sein d'une classe Caisse. Votre programme devra être capable de gérer plusieurs caisse.

Vous devrez modéliser plusieurs passage en caisse et ajouter une méthode permettant d'obtenir le total encaissé. Vous devrez ainsi créer une nouvelle classe Passage.

Les produits passés en caisse devront eux aussi être des objets. Vous créerez pour l'occasion une classe Produit.

Chapitre 6. TP 4 : Programmation orientée objet avancée

Nous allons créer une nouvelle application. Le but est de créer un jeu d'aventure simpliste dont voici le gameplay.

Vous êtes un personnage qui se déplace dans un chateau, de pièce en pièce. Le but du jeu est de sortir du chateau. Vous démarrez le jeu dans une prison. Chaque pièce peut être lié à une pièce ou plusieurs pièces. Chaque pièce peut éventuellement comporté une sortie et aura un nom et un type (prison, salle de garde, cuisine, chambre etc.)

Votre personnage peut se déplacer de pièce en pièce librement. Il possède des points de vie. Le jeu se déroule tour par tour et à chaque fois il peut se déplacer d'une pièce à une autre.

Quand il se trouve dans une pièce, il y a un événement aléatoire ou pas parmis les suivant: il ne se passe rien, il se fait attaquer (et perd un certain nombre de point) ou il trouve une fiole de vie qui lui recrédite des points de vie.

Le joueur gagne quand il sort du chateau et il perd quand il n'a plus de point de vie.

Sur cette base de gameplay, on peut imaginer complexifier le jeu avec notamment l'ajout d'objet et de leur utilisation (par exemple: combattre, inspecter une pièce etc.). On peut imaginer croiser d'autres personnages (allié ou ennemi etc.). On peut aussi imaginer le calcul d'un Score basé sur le nombre de déplacement et le nombre de points de vie restant du joueur. Mais avant celà, veillez à obtenir une première version fonctionnelle.

Pour parvenir à implémenter ce jeu. Vous devrez créer une classe Personnage et une classe Pièce. Vous utiliserez l'héritage pour modéliser les différents types de pièces. Vous êtes libre de créer d'autres classes si vous le souhaiter (Chateau, Evenement…). La boucle principale de votre jeu pourra être une boucle while qui demande à chaque itération quelle action réaliser, avec une liste de choix.

Vous utiliserez au maximum ce qui a été vu jusqu'à présent: listes, tuples, dictionnaires, exceptions… Vous pouvez vous appuyer sur la bibliothèque standard dont le module math pour les nombres aléatoire.