Université Paul Valéry Montpellier 3
Antenne universitaire de Béziers
L3 AES parcours MISASHS
ECUE «Logiciels spécialisés »

# Exemple de projet

# « Gestion de contacts »

G. Richomme

## Table des matières

# 1. Introduction

Ce rapport est un exemple partiel, à critiquer (plein de choses sont, parfois volontairement, mal exprimées ou insuffisamment décrites), de ce que peut être un rapport de projet. Il doit servir de source d'inspiration. Pour cela, il conviendra de relever les points positifs et les points négatifs de ce rapport tout aussi bien dans sa forme que dans son fond.

Un rapport s'envisage, se construit dès le début de l'activité à relater. Tout d'abord prévoir dès le début sa forme permet ensuite de se concentrer uniquement sur le fond. Concernant ce dernier, certaines rubriques peuvent dès le départ être envisagées (analyse et algorithmes, limites de l'application, …). D'autres s'imposeront d'elles mêmes en fonction des informations pertinentes notées. Pour cela, il est conseillé de se faire dès le départ une annexe « Chronologie » qui ne sera peut-être pas rendue mais qui permettra de noter très rapidement les points importants à ne pas oublier de mentionner. L'expérience montre qu'attendre la fin du projet pour commencer son rapport n'est pas la bonne solution.

Den la souite de ce raport, nous alons suxessivemant présenter le cadre du projet, les différents phases d'analyse... A COMPLETER plus tard pour présenter le plan définitif et les points importants sur lesquels nous désirons dès le départ attirer l'attention du lecteur. En fonction de la longueur du rapport et de la qualité de la rédaction, le lecteur (dont le temps est compté) va peut-être survoler certaines parties si on n'attire pas son attention sur leurs intérêts.

# 2. Le cadre du projet

La réalisation du programme correspond au besoin exprimé suivant par l'utilisateur qui nous a contacté, en nous écrivant :

*Je gère actuellement ma liste de contact à l'aide d'un fichier « tableur » une liste de contacts. Son utilisation est parfois assez pénible. Serait il possible de créer un programme qui utiliserait ce fichier et qui me faciliterait, sans ouverture d'un tableur, les opérations d'édition (création, suppression, modification, affichage, recherche) de mes contacts ?*

Nous nous sommes mis d'accord pour :

- qu'un programme en langage Python soit réalisé
- qu'une interface textuelle soit suffisante
- que le fichier « tableur » soit au format CSV (*Comma-separated values)*. Ce fichier est constitué d'une suite de lignes contenant chacune 4 valeurs, dans l'ordre *nom, prénom, adresse, téléphone*. Chaque valeur est optionnelle. Il s'agit pour chacune de texte sans format particulier.

# 3. Grandes lignes de l'analyse fonctionnelle et algorithmique

Nous avons décidé de modéliser naturellement les données à l'aide de deux classes d'objet.

Objet Personne : recense les données d'un contact avec 4 attributs nom, prenom, adresse, telephone.

Objet Carnet_adresse : recensera l'ensemble des contacts sous forme d'une liste d'objets Personne.

L'interface graphique fera l'objet d'un autre composant séparé. Les fonctionnalités attendues de cet interface sont :

- lecture du fichier en début d'exécution
- sauvegarde du fichier en début d'exécution
    - Notons que le nom du fichier doit être centralisé dans le programme interface pour être sûr que ce soit le même utilisé.
- arrêt de l'exécution
- sauvegarder en cours d'exécution les données
- ajouter d'un contact
- suppression d'un contact selectionné préalablement
- modification d'un contact selectionné préalablement
- affichage de la liste des contacts
- recherche de contacts vérifiant des critères.

Les critères de recherche seront des parties de nom, prénom, adresse, téléphone. L'utilisateur pourra fournir une information pour chaque caractéristique du contact. La recherche fournira la liste de tous les contacts qui vérifient ces critères.

Pour chaque fonctionnalité, il a été fait le choix de programmer une méthode du carnet d'adresse prévoyant le traitement complet de la fonctionnalité.

Lecture du fichier : parcours ligne à ligne du fichier, instanciation d'un objet Personne avec la ligne et ajout dans la liste des contacts. Pour permettre, ce traitement facilement il est choisi d'avoir un constructeur permettant de donner les valeurs de chaque attribut de l'objet Personne.

Ecriture du fichier : parcours de la liste de contact et écriture au fur et à mesure dans le fichier.

Ajout d'un contact : outre l'insertion dans la liste d'un contact, il faut prévoir une méthode de l'objet Personne permettant la saisie des valeurs d'une personne.

Affichage du carnet d'adresse : parcours de la liste des personnes et affichage

Le fait que les critères de recherche sont les mêmes pour une recherche, une modification ou une suppression amène à créer :

- une méthode de l'objet Personne permettant de chercher les personnes correspondantes à des critères de recherche définis. En résultat de cette méthode, nous avons besoin de la liste des personnes vérifiant un des critères de recherche, mais aussi des indices de ces personnes dans la liste complète de contact.
- une autre méthode effectuant en plus la saisie
- une méthode effectuant saisie, recherche, affichage pour répondre à la fonctionnalité de recherche
- une méthode permettant d'effectuer la saisie des critères de recherche, puis un choix de personne à modifier, puis la modification de la personne. Cette méthode de l'objet Carnet_Adresse nécessite l'usage d'un appel à une méthode de modification de l'objet Personne.
- une méthode permettant d'effectuer la saise de critères de recherche, puis un choix de persone à supprimer, puis la suppresion en elle-même.

Remarque : les analyses sur des projets de plus grande envergure nécessitent beaucoup plus de détails. Plus les spécifications de ce qui va être programmée sont précises, plus le programmeur fera

ce qu'on attend de lui.

## 3.1 Spécifications détaillées

### 3.1.1 Spécification du format du fichier CSV

Le nom actuel du fichier csv traité par le programme est « carnet_adresse.csv ». Ce nom est défini dans le fichier interface.py.

Ce fichier doit être constitué de 4 colonnes, correspondant dans l'ordre au nom, au prénom, à l'adresse, au téléphone de chaque contact.

### 3.1.2 Classe Personne

4 attributs : nom, prenom, adresse, telephone

un constructeur avec des paramètres permettant l'instanciation d'un objet en précisant les valeurs de chaque attribut.

une méthode d'affichage

une méthode saisie des informations sur une personne déjà instanciée

une méthode modification des valeurs via une interface avec l'utilisateur

### 3.1.3 Classe Carnet_adresse

1 attribut : la liste des personnes contact

constructeur permettant la lecture des données du carnet d'adresse dans un fichier dont le nom est donnée en paramètre (appelera la méthode de lecture dans un fichier)

méthode de lecture dans un fichier de nom donné en paramètre

méthode de sauvegarde dans un fichier de nom donné en paramètre

méthode d'affichage : les valeurs devront être triées

méthode d'ajout d'un contact

méthode de recherche de personnes : en paramètre, les parties de nom, prenom, adresse, téléphone à chercher. La recherche fournira la liste de tous les contacts qui vérifient ces critères, ainsi que la liste de leurs indices dans la liste complète de contact.

méthode de recherche de personnes avec saisie des critères : appel de la méthode précédente après saisie des critères de recherche

méthode effectuant saisie, recherche, affichage pour répondre à la fonctionnalité de recherche

méthode permettant la modification d'un contact

méthode permettant la suppression d'un contact

### 3.1.4 Programme interface

Permet via un menu interactif la gestion des choix successifs de l'utilisateur.

### 3.2 Mise en œuvre

L'implémentation a suivi les étapes suivantes :

1. programmation de la classe Personne avec création dans l'ordre :
   - du constructeur
   - de la méthode d'affichage
   - de la méthode de saisie
   - de la méthode de modification

Des tests via un programme local ont permis de vérifier le bon fonctionnement de chaque méthode.

2. classe Carnet_adresse, constructeur, lecture dans un fichier, affichage, tests
3. sauvegarde dans un fichier
4. ajout nouveau contact (avec programme ad hoc permettant de voir la liste de contact avant et après)
5. fonctions de recherche sans puis avec saisie, puis avec affichage
6. fonctions de suppression

A partir de l'étape 3, le programme Interface a progressivement été mis en place.

Il reste à faire des tests d'envergure de bon fonctionnement global.

## 4. Comment lancer le programme

Ce paragraphe sera à rédiger dans le rapport de votre projet. Prévoir de mettre l'environnement de test, les logiciels nécessaires. Expliquer les logiciels à utiliser ou à installer. Comment ? L'emplacement des fichiers. Pensez que l'utilisateur n'a pas suivi votre travail et peut ne pas être un spécialiste de l'informatique.

## 5. Exploitation de l'application

Ici ce ne sera pas le cas, mais dans le projet il faudra prévoir des exemples de graphes sur lesquels vous avez lancé l'application.

## 6. Informations diverses

### 6.1 Non prise en compte d'un retour chariot dans l'interface

Pendant les tests de l'interface, il est apparu que l'utilisateur pouvait être amené à faire parfois des retours chariots trop rapprochés. Certains pouvaient être considérés par le programme comme un choix de fonctionnalité à utiliser et était suivi du message « Choix incompréhensible ». Ce message n'est à présent plus affiché : le retour chariot est ignoré.

### 6.2 Trier une liste d'objets

Trier une liste est une opération prévue par la méthode sort() de la classe list. Toutefois quand les éléments de la liste sont des objets (dans notre cas des objets de type Personne), le critère de tri doit

se baser sur les attributs et l'utilisation de la méthode sort() s'est avérée ne pas le faire. Le recoupement d'informations trouvées sur les sites http://wiki.python.org/moin/HowTo/Sorting/ et http://docs.python.org/2/library/operator.html#module-operator ont permis de trouver les solutions suivante :

- importer attrgetter du module operatore
- si l est la liste à trier, l'instruction suivante permet d'effectuer le tri :
  - l.sort(key = attrgetter( liste des noms d'attributs devant servir au tri

Dans notre application, le tri du carnet d'adresse devient :

- self.personnes.sort(key=attrgetter('nom','prenom', 'telephone', 'adresse'))

Le carnet d'adresse est alors trié par nom croissant, en cas d'homonymie par prénom croissant, puis pour une même personne, les informations sont rangées par ordre alphabétique croissant des adresses puis des numéros de téléphone

## 6.3 Traitement en cas d'inexistence de fichier

Le nom du fichier contenant le carnet d'adresse est défini dans le fichier interface. Lors d'une première utilisation, ou suite à une suppression, ce fichier peut ne pas exister, dans ce cas l'application fonctionne en partant d'un carnet d'adresse vide.

# 7. Limites et améliorations potentielles du projet

**Taille des données gérés**

Les données de contact sont entièrement stockées dans la mémoire vive de l'ordinateur pendant l'exécution du programme. Cela limite, de facto, le nombre de contacts que peut gérer le programme. Néanmoins, la taille importante des mémoires vives de nos ordinateurs actuels permet de penser qu'au regard du faible nombre de contacts d'une personne le logiciel serait sans souci opérationnelle (même avec 10000 contacts (chiffres qui est très largement au dessus de la taille de la liste de contacts actuelle du client), il ne devrait pas y avoir de souci). Des tests d'envergure seraient très certainement nécessaire pour avoir une idée du nombre de données embarcables. A noter, que la présence de tris réguliers sur les données peuvent être de nature à provoquer des ralentissements. Si cela était le cas, le programme devrait être optimisé.

**Déclaration CNIL**

Le logiciel créé permet la gestion d'un fichier contenant des données à caractère personnel. Le client devra envisager une déclaration à la CNIL de son traitement.

**Amélioration interface**

Il a été choisi de faire une interface textuelle. Une interface graphique serait sans doute à envisager. Bien que nous ayons pensé à avoir une interface à minima ergonomique avec des messages explicites, des saisies non redondantes, des affichages temporisés, il faudrait sans doute encore effectuer des tests.

**Approfondissements des tests à réaliser**

La version actuelle est encore en test.

**Fichier utilisé**

On pourrait prévoir le changement du nom du fichier contenant le carnet d'adresse (une fonctionnalité de plus dans le programme interface). Il faudrait toutefois prévoir un fichier

supplémentaire pour conserver le nom du fichier réellement exploité.

**Sauvegarde en fin d'exécution**

Afin d'être sûr de ne pas perdre de données, ces dernières sont systématiquement sauvegardées. Cela peut poser un problème si l'utilisateur a fait des modifications qu'il ne désire pas enregistrer. Il pourrait être intéressant de maintenir un attribut du carnet d'adresse indiquant si les données ont été modifiées ou non depuis la dernière sauvegarde, et demander en fin d'exécution si l'utilisateur veut effectuer une sauvegarde ou non.

# 8. Conclusion

L'apport du projet sur le plan collectif, personnel, ...

# 9. Bibliographie

G. Richomme, Supports de cours de l'ECUE Informatique Programmation, site Misashs, vu sur le web en août 2013

G. Swinnen, *Apprendre à programmer avec Python 3,* Eyrolles, 2ème édition, 2010

Site www.python.org

# 10. Annexe : chronologie

Partie à remplir au fur et à mesure de l'avancée, éventuellement sous forme de notes rapides, en notant les éléments importants pour la suite. Dater chaque événement (date + heure éventuellement). Préciser qui a fait quoi.

Exemple :

**1/07/2013 :** réception de la demande

**3/07/2013 :** rendez-vous avec le client pour une première analyse de ces besoins réels et un début de l'étude de faisabilité

…

**24/08/2013 :** recherche d'une méthode pour savoir si une chaîne ne contient que des nombres. Réponses trouvées (méthode isnumeric() de la classe str) sur la page :

http://docs.python.org/3.3/library/stdtypes.html?highlight=str#str.isnumeric

**25/08/2013.** Recherche pour trier une liste d'objet par G. Richomme. Site ayant permis de trouver une solution :

http://wiki.python.org/moin/HowTo/Sorting/ : toutes les solutions proposées de ce site ne fonctionne pas avec les éléments déjà développés (peut-être un problème de version), mais le site donne des indications intéressantes, notamment sur l'existence d'operator

http://docs.python.org/2/library/operator.html#module-operator : l'étude de cette aide à permis de voir que attrgetter permet facilement de définir les critères de tri.

En résumé, trier une liste d'objets peut se faire simplement en

- important attrgetter du module operatore
- si l est la liste à trier, l'instruction suivante permet d'effectuer le tri :

- l.sort(key = attrgetter( liste des noms d'attributs devant servir au tri

- Dans notre application, le tri du carnet d'adresse devient :
  - self.personnes.sort(key=attrgetter('nom','prenom', 'telephone', 'adresse'))
  - Le carnet d'adresse est alors trié par nom croissant, en cas d'homonymie par prénom croissant, puis pour une même personne, les informations sont rangées par ordre alphabétique croissant des adresses puis des numéros de téléphone

**25/08/2013.** G. Emmohcir. Mise en œuvre d'un affichage trié. Test avec le fichier « essai13.csv » créé manuellement à des fins de tests, les données étant initialement non triées.

**28/08/2013 :** relecture des codes sources, consolidation des commentaires ; amélioration du rapport

# 11. Annexe : messages d'erreur rencontrées

A rendre le 7 octobre : Pour chaque message d'erreur (un par type de message) rencontré pendant la partie 1 du projet, vous copierez-collerez le message d'erreur et expliquerez brièvement ce qu'il se passe et la correction à apporter.