Nom Prénom

Binôme1 HUO Zhengjie

Binôme2 ZHANG Peng

Polytech'Lille

Département IMA , FSA

4ième année (2011-2012)

Correcteur Automatisé Grille TOEIC

Tuteur : Blaise CONRARD

CAHIER DES CHARGES

1.1Objectif :

Il existe une application informatique pour la correction des grilles TOEIC dans le département des langues, qui permet de réaliser le retour du nombre de bonnes et de mauvaises réponses par lire l'image scannée d'une grille. Et le but de notre projet est d'améliorer cette application pour qu'il soit plus simple, plus pratique et plus intelligent à utiliser.

1.2 Logiciel principal choisit:

'cx-freeze'--Faciliter le processus d'installation

'python'avec 'PIL'—Ecrire la programme principale

'HT ML 5', 'avascript' et 'SQLite'—l'interface utilisateur

1.3 Caractéristiques d'acquisition :

Comme l'ancienne version, notre application peut traiter des images en format TIFF, et il peut analyser des images en tous les sens. Les 200 questions doivent être clairement séparées en deux groupes de 100 questions (partie orale et écrite).

1.4Analyse l'ancien application et Gestion des résultats :

En testant l'ancienne application, on a trouvé que la fenêtre HTML permettait afficher que le nombre de Bonne réponses. C'est un peu limite et pas assez pratique pour l'utilisateur. Et puis on a fixé nos tâches à effectuer pour la partie 'l'interface utilisateur' de sorte à réaliser 4 fonctions suivantes :

●Calcul le note final

●Extraire des petits morceaux d'images

●Permettre l'utilisateur à distinguer les confusions en manuel

●Après la validation manuelle, l'application a capable de recalculer le nouvel résultat automatiquement.

En considérant à réaliser tous les fonctions souhaitées, on a dessiné le diagramme du processus suivant pour notre application.

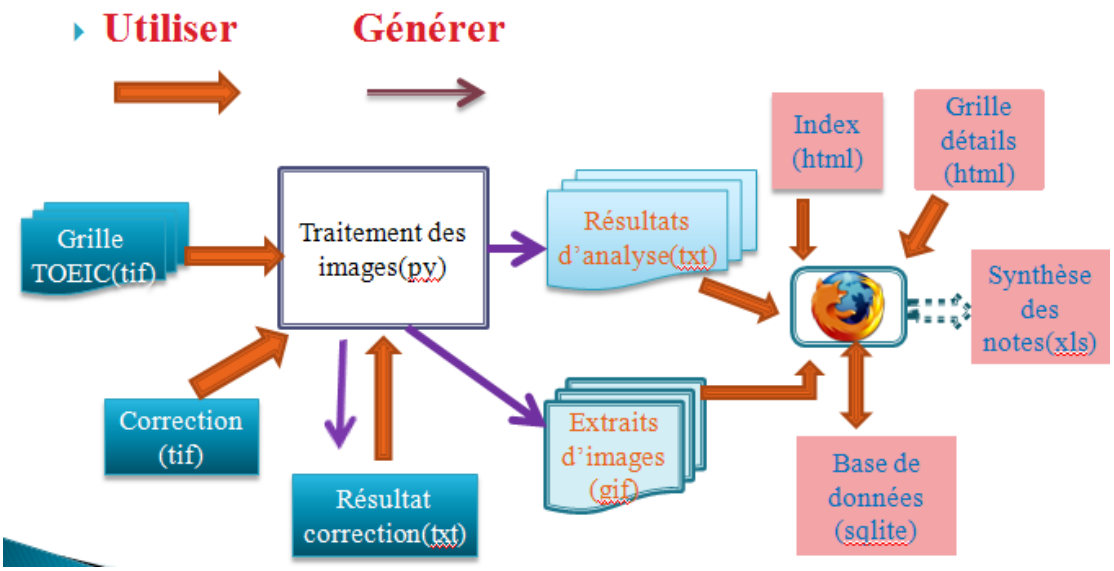


Diagramme du processus de l'application

Comme le diagramme ci-dessus qui nous montre. Notre application est pour deux processus principaux—le traitement d'images et l'interface utilisateur.

Notre programme de python a la fonction de traiter de grilles. Tout à bord, le programme va lire l'image de correction (tif), et générer un fichier Résultat correction (txt) qui peut garder les références venues de l'image de correction. Et puis, le programme va juger des Grilles TOEIC en lisant le fichier Résultat correction (txt). Ici, on utilise le NIP pour identifier les grilles TOEIC. Et les informations obtenus (le NIP, le nombre de bonnes et mauvais réponses, et le note totale) doit être rangé sous le texte 'Résultats d'analyse.txt'.Pour chaque grille Toeic, on générera un texte sous la forme'NIP.txt' qui peut mémoriser ses informations en détaille. Et tous les informations peut être mémorisé dans le 'Base de données 'par le programme interface utilisateur. Tant que le

programme python rencontreras quelques confusions(le vide ou le double), elle va extraire l'image où il aura apparu des confusions et générer un petit morceau d'image qui le correspondant.

Et pour utilisateur, il peut voir et vérifier par l'interface utilisateur(le page web), qui peut lire les résultats de le texte 'index.html' ,'Grilles détaills .html'. En plus, dans ce cas où il apparait des confusions(le cas de vide ou double), l'utilisateur peut le juger en voyant le petit morceau d'image sur 'Grilles détaills.html'. S'il y a le changement de note, après la validation de l'utilisateur, notre programme de l'interface utilisateur qui est écrit par JavaScript va recalculer le score final et changer la base de données locale de Grille. Et, pour la prochaine fois d'ouverture de l'interface utilisateur, ce sont des résultats corrigés qui vont s'afficher.

Planning

Séances	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1. Analyse l'ancienne application														
2. Faciliter le processus d'installation														
3. Réaliser à calculer la note														
4. Réaliser à générer les morceaux d'images														
5. Ecrire le page HTLM														
6. Ecrire la partie 'Base de données'														

I. Analyser l'ancienne application

Tester l'ancienne application et essayer de comprendre l'ancienne le programme

II. Faciliter le processus d'installation

Trouver les logiciels qui permet de générer le programme exécutable du programme python.

III. Réaliser la fonction de calcul note

Concevoir un algorithme qui permet de calculer la note d'après la table de référence et trouver la meilleure méthode au réaliser

IV. Réaliser la fonction d'extraire des morceaux d'images

Analyser les coordonnées de grille dans notre programme python et concevoir un algorithme qui permet extraire un petit morceau d'images tant qu'il y a une confusion.

V. Remplacer la fenêtre d'affichage de détails '.txt' par '.html'

Faire deux fenêtres d'affichage en HTML : l'une fonctionne à afficher les infos de tous les grilles, et l'autre peut afficher les détails d'infos pour chaque grille

VI. La partie 'base de données'

Employer les bases de données à sauvegarder les changements de résultat après la validation et la correction par l'utilisateur en manuel

I. Analyse l’ancienne application

1. Tester l’ancienne application

Faire quelques tests sur l’ancienne application au but de fixer nos travaux à réaliser

Faciliter l’installation,

Faire la fonction du calcul le score

Faire une interface plus visible

Réaliser le passement à l’utilisateur quand il a besoin.

2. Essayer de comprendre l’ancienne le programme

Essayer de comprendre les différentes fonctions de chaque fonction de l’ancienne application et le lien entre eux.

Les fonctions principales et leurs explications(en vert) sont :

def chargeConf()
Charger le fichier "confFormulaire.txt" qui donne les coordonnées de chaque question

def litPixel(im,pos)
def rechercheObjet(image,zone)
def rechercheRond(image,zone,taille)
def rechercheReperes(im)
def litCase(image,pos,taille)
def litCaseNom(image,repere,nom)
def litZone(image,repere,listeCases)
def marqueZone(image,repere,nom,couleur,forme='X')
def litIdentifiant(image,repere)

def litReponses(image,repere,bonneReponse=None)

Traiter d’image

def changementRepere(pt,repere)

Changer les coordonnées obtenues du fichier "confFormulaire.txt" au format pixel d’image

def litGrilleCorrection(rep,console)
def litFichierCorrection(rep)
Pour obtenir les résultats corrects

def corrigeGrille(repertoire,fic,reponses,console)
Corriger les grilles et créer les fichiers textes pour mettre les détails de chaque grille

def analyseRep(rep,console)
Commencer à analyser le répertoire choisi et utiliser les autres fonctions pour corriger

Partie Graphique
Choisir le répertoire et visualiser les résultats pendant le processus

II. Faciliter le processus d’installation

Pour installer l’ancienne implication, on doit installer le 'python' et le 'PIL' à cause que l’exécution de programme python n’est pas indépendante de le 'python' et le 'PIL'. Le durée d'installation est environs 3 minutes et le processus n’est pas assez facile à prise à la main. On a trouvé les logiciels 'cx-freeze' ,’py2exe’et ‘PyInstaller’ qui nous permettent transférer notre programme 'analyse.py' à un programme exécutable de format'.exe'. Et on a choisi le 'cx-freeze' en raison que son programme emballé est plus petit.

Tout a bord, on doit rajouter les commandes dans notre programme pour indiquer le type d'image ce qu'on a besoin. Parce que le PIL (Python Image Library) utilise un dynamique mécanisme à charger plugins, mais cx-freeze oublie d'inclure ce que le plugin a besoins parfois.

import Tiff Image Plugin
import Png Image Plugin
import Bmp Image Plugin
import Gif Image Plugin
import Jpeg Image Plugin
import Image Palletle
import Image File

Ensuite, on a installé le 'cx-freeze' sous le répertoire'\Python 27\Script’, et puis on a passé le command ‘cxfreeze-h’ pour vérifier le réussit d’installation de 'cx-freeze' :

```
C:\Python27\Scripts>cxfreeze -h
Usage: cxfreeze [options] [SCRIPT]

Freeze a Python script and all of its referenced modules to a base
executable which can then be distributed without requiring a Python
installation.

Options:
  --version             show program's version number and exit
  -h, --help            show this help message and exit
```

En fin, on a passé le command ’cxfreeze analyse.py—traget-dir dist’ au but de générer l’application.

III. Réaliser la fonction de calcul note

Réalisation1:

On a trouvé le table de référence pour calcul le note d'après le nombre de bonne réponse de l'enseignant de l'anglais. Et on a ajouté la fonction de calcul le note d'écrit et le note d'orale et le note total.

Le programme et son explication(en vert) :

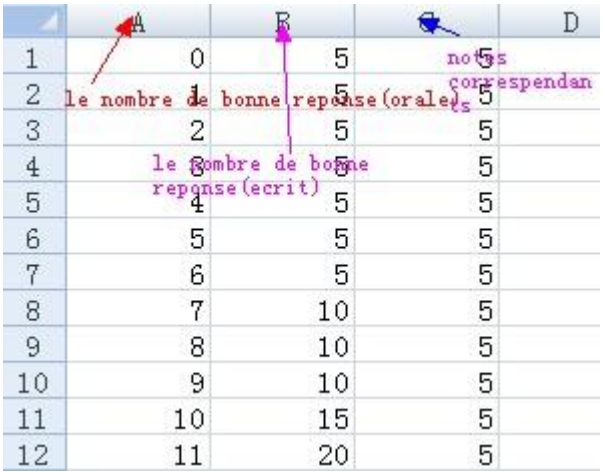
```
defcalculeorale(bonneRep0):#on définit deux fonction pour calculer les notes.
    if bonneRep0<=6 & bonneRep0>=0: noteorale=5
    elif bonneRep0<=9 & bonneRep0>=7: noteorale=10
    .
    .
    else: noteorale=495
    return noteorale

defcalculeecrit(bonneRep1):
    if bonneRep1<=15 & bonneRep1>=0: noteecrit=5
    elif bonneRep1<=17 & bonneRep1>=16: noteecrit=10
    .
    .
    else: noteecrit=495
    return noteecrit
```

Réalisation2:

Pour changer le table facilement et avoir une plus vaste application de notre programme, on a amélioré la fonction par créer le table de note par EXECEL de format 'csv'. Et puis on remplacer notre précédent calcul de note par une fonction qui peut consulter sur la table 'csv'.

Le fichier ‘.csv’ :



	A	B	C	D
1	0	5	5	
2	1	5	5	
3	2	5	5	
4	3	5	5	
5	4	5	5	
6	5	5	5	
7	6	5	5	
8	7	10	5	
9	8	10	5	
10	9	10	5	
11	10	15	5	
12	11	20	5	

Le programme et son explication(en vert) :

```
defcalculetext(rep,bonneRep0,bonneRep1):#on utilise cet fonction pour donner les notes correspondants. C’est plus facile à changer des notes sur un fichier '.csv'
    import csv
    f=open(rep+'/table.csv', "rb")
    reader = csv.reader(open(rep+'/table.csv', "rb"))
    note=[]
    n=1
    for row in reader:
        iflen(row)==3:
            if int(row[0])==bonneRep0:# tout à bord, on a écrit "row[0]==bonneRep0",mais row[0] n'est pas un entier, il peut être '10'.
            note.append(int(row[1]))# tout à bord, on a écrit "note[0]=int(row[1])",il y a un erreur sur l'index de note. Pour ajouter des chiffres sous Python, on peut utiliser 'note.append': ajouter à la fin d’une liste.
            if bonneRep1==int(row[0]):
                note.append(int(row[2]))
    f.close()
```

```
    return note
```

Réalisation3(le final):

Pour diminuer le temps de calcul, on définit une mémoire dynamique qui peut lire le table 'csv' seulement une fois, et comme ça, quand on faire le calcul de note, le programme n'a pas besoin d'ouvrir le 'csv' tous les temps.

Le programme et son explication(en vert) :

```
note=[]
definitNote(rep):
    global note
    import csv
    note=[]
    reader = csv.reader(open(rep+'/table.csv', "rb"))#on ouvert seulement une fois la fichier texte. C'est plus rapide.
    for row in reader:
        iflen(row)==3:
            note.append((int(row[1]),int(row[2])))

defcalculetext(rep,bonneRep0,bonneRep1):#on utilise cet fonction pour donner les notes correspondants. C'est plus facile à changer des notes sur un fichier '.csv'
    return (note[bonneRep0][0],note[bonneRep1][1])
```

On a essaie trois fois pour améliorer notre programme de calcul. Après l'amélioration, on peut changer la table facilement et exécuter plus rapidement. Tous les trois réalisations peut nous fournir le note correct, et

puis, si on modifie le page HTML, on peut voir le note qui est affiché sur le web comme ci-dessous :

Afficher sur le page index.html:

Id	OK 1a100	OK 101a200	Double	R. Vide	Notetotal	De
04002030	060	063	003	004	595	Pas de
64852730	058	063	003	006	585	Pas de

IV.Réaliser la fonction de extraire des morceaux d'images pour le vide et le double

Réalisation1:

Après bien compris l'ancienne programme, on a connu que l'ancien programme a choisi le repère1 (0,1) *(0,1) à traiter une feuille de taille A4, cependant, on doit traiter la feuille A4 sur 'PIL' en choisissant le repère 2(0,2338)*(0,1645). On a défini une fonction qui a permis de transfère le coordonnées du repère1 au repère2 --c*hangementRepere*. Et en savant les coordonnées précisément créé la fonction 'genererMorceauImage' qui peut lire les coordonnées où il y a un confuse, et puis il peut générer l'image en prenant

Le programme :

```
defgenererMorceauImage(i,imageGrille,fic,rep):
        im=imageGrille
        l=["Q%03dA"%i,'Q%03dB'%i,'Q%03dC'%i,'Q%03dD'%i]
        pt1=listePoints[l[0]]
        pt2=listePoints[l[3]]
        image=Image.open(rep+'/'+fic[:-4]+'.tif')
        repere=rechercheReperes(image)
        (x1,y1)=changementRepere(pt1,repere)
        (x2,y2)=changementRepere(pt2,repere)
        pasx=repere[4]
        pasy=repere[5]
        box=(int(x1-4.5*pasx),int(y1-2*pasy),int(x2+2*pasx),int(y2+2*pasy))
        region = im.crop(box)
        nom=("Q%03d"%i)
```

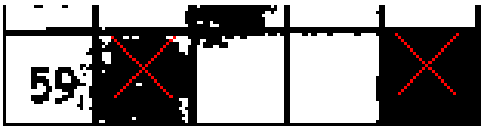
region.save(rep+'/'+nom+'.gif')

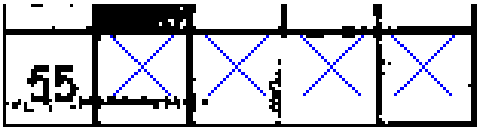
Réalisation2(le final):

Pour le programme précédant, on a besoin de calculer chaque fois le repère (ce calcul est assez long à cause de lecture des coordonnées d'images).
Donc pour déduire le temps de générer les morceaux d'images, on a modifié 'genererMorceauImage' qui passe le repère comme un paramètre.

-- *“defgenererMorceauImage(i,imageGrille,fic,rep,repere):”*

Par exemple : les images obtenues sont comme ci-dessous :





V. Remplacer la fenêtre d'affichage de détail '.txt'par'.html'

Réalisation1:

On a fait une fonction dans le programme python pour créer les fichiers des détails '.html' qui est plus lisible que '.txt', et aussi pour pouvoir faire visualiser les petits morceaux d'image crée par la fonction 'genererMorceauImage'. Sur la visualisation de cette page web, on peut visualiser les détails qui comprennent le nombre de bonne réponse orale (écrit), les notes correspondants, la note totale et les questions doubles ou vides avec les images générées.

Le programme et son explication(en vert):

```
DefecritDetail(rep,fic,listeReponses,imageGrille,numId,note,bonneRep0,bonneRep1,repere):
    liste=listdir(rep)
    var=[]
    im=imageGrille
    f=open(rep+'/'+fic[:-4]+'.html','w')#on crée un fichier '.html' pour écrire les contenus.
    f.write("<HTML><HEAD><TITLE>Resulat grid number%s</TITLE>"%numId)

    #on utilise le javascript pour créer une fonction qui peut distinguer cette question est une réponse orale ou une réponse écrit puis ajouter un sur le nombre de bonne réponse et changer le nom de ce bouton.
    f.write('''
                <SCRIPT language="javascript">
    functionModifNote(i)
                {
```

```
                    var n1=document.getElementById("resultat1");

                    var n2=document.getElementById("resultat2");

            if(i<=100)

                    {

                        n1.innerHTML=parseInt(n1.innerHTML)+1;'")

    f.write('"    document.getElementById("i").innerHTML="<INPUT type=button
                value=OK/>";}'")

        f.write('"          if(i>100)

                    {

                        n2.innerHTML=parseInt(n2.innerHTML)+1;'")

    f.write('"   document.getElementById("i").innerHTML="<INPUT type=button
                value=OK/>";}'")

        f.write('"};

        </SCRIPT>'")

        #afficher les informations générales

    f.write("</HEAD><BODY><H1>Grid number%s</H1>"%numId)

        f.write("<BR>Resultat Listening :<LI>Right response : <SPAN
id=resultat1>%d</SPAN>/100</LI>"%bonneRep0)

        f.write("<LI>Note TOEIC : %s</LI>"%note[0])

        f.write("<BR>Resultat Writing :<LI>Right response : <SPAN
id=resultat2>%d</SPAN>/100</LI>"%bonneRep1)

        f.write("<LI>Note TOEIC : %s</LI>"%note[1])

        notetotal=note[0]+note[1]

        f.write("<BR><BR>Resultat Total :<LI>Note TOEIC :<DIV
id=resultotal>%d</DIV</LI>"%notetotal)

        #afficher les questions doubles ou vides avec les images générées

    i=0

        for e in listeReponses:

            i=i+1

            iflen(e)>1 : #pour générer les cas qui ont plus de 2 réponses
```

```
                    f.write('<HR><BR>Problem 2 responses :<BR>')

                    f.write("There is a double in this question %s<BR/>"%str(e))

                    #on utilise la fonction 'genererMorceauImage' pour générer les
petits morceaux et afficher sur le page web.

                    genererMorceauImage(i,fic,imageGrille,rep,repere,numId)

                    nom=("Q%03d"%i)

                    f.write('<IMG SRC=%s%s.gif><LI>If this question
isright,clickhere!</LI>'%(str(numId),str(nom)))#on ajoute le nom de fichier pour éviter
les conflits(rep+'/'+fic[:-4]+nom+'.gif' il y a que le nom de la question)

                    f.write('''<span id="%d"><INPUT id=%d type=button value="La
Reponseest OK" onclick="ModifNote(%d)"/></span>'''%(i,i,i))

                iflen(e)==0 :#pour générer les cas qui ont 0 réponse

                    f.write('<HR><BR><BR>Problem empty :<BR>')

                    f.write("There is no response in this question
['Q%03d']<BR/>"%i)

                    genererMorceauImage(i,fic,imageGrille,rep,repere,numId)

                    nom=("Q%03d"%i)

                    f.write('<IMG SRC=%s%s.gif><LI>If this question is right,click
here!</LI>'%(str(numId),str(nom)))

                    f.write('''<span id="%d"><INPUT id=%d type=button value="La
Reponseest OK" onclick="ModifNote(%d)"/></span>'''%(i,i,i))

        foot='''<HR></BODY></HTML>'''

        f.write(foot)

        f.close()
```

Le résultat visuel :

Grid number04002030

Resultat Listening :
- Right response : 63/100
- Note TOEIC : 310

Resultat Writing :
- Right response : 63/100
- Note TOEIC : 285

Resultat Total :
- Note TOEIC :

595

Problem empty :
There is no response in this question ['Q025']

- If this question is right,click here!

La Reponse est OK

Réalisation2:

Pour facilement changer le contenu du fichier '.html' et utilise la base de donnée après, on n'a écrit plus les deux fichiers 'index.html' et 'lectureGrille.html' dans le programme python. On a met les deux fichiers en forme HTML en dehors du programme python.

VI. La partie ‘base de données’

Pour réaliser le changement de résultat après le validation et la correction par l’utilisateur en manuel, on a trouvé le ’HTML5’ qui nous permet d’utiliser des bases de données locales. On a écrit un programme qui peut créer la table correspondue aux résultats obtenus dans les bases de données locales. Et comme ce diagramme de ‘Index.html’ qui nous montre. Tout à bord,
le ’notetotal’ a la même valeur que le ‘Notefinal’.

Index des grilles

fichier	bonne_orale	bonne_ecrit	double	pasrep	notetotal	detail	Notefinal
04002030.txt	60	63	3	4	595	voir	595
64852730.txt	58	63	3	6	585	voir	585

Et quand il y a du changement de nombre de bonne réponses d’orale et d’écrit en le ’lectureGrille.html’, le programme va utiliser le nouvel nombre de bonne réponses d’orale et d’écrit à relire le table de référence ’.csv’ en recalculant le score. La partie ‘Notefinal’ de la table va prendre le résultat actuel et le sauvegarder.

Et puis, la fois prochaine, quand utilisateur ouvre le ‘index.html’, le’index.html’ vas lire le résultat actuel du table.

Index des grilles

fichier	bonne_orale	bonne_ecrit	double	pasrep	notetotal	detail	Notefinal
04002030.txt	60	63	3	4	595	voir	605
64852730.txt	58	63	3	6	585	voir	585

Et on a trouvé un outil SQLite qui peut nous aider à vérifier les étés de tous les bases de données:

TABLE testTOEIC | Recherche | Tout voir | Ajouter une nouvelle entrée | Duplicate | Éditer les entrées sélectionnées | Supprimer les entrées sélectionnées

id	nomGrille	note1	note2	notetotal
1	04002030.txt	62	63	605
2	64852730.txt	58	63	585

Enfin, on a embelli la page web et on a obtenu les résultats comme suivant :

- Index.html

Index des grilles

fichier	bonne_orale	bonne_ecrit	double	pasrep	notetotal	detail	Notefinal
04002030.txt	60	63	3	4	595	voir	600
64852730.txt	58	63	3	6	585	voir	585

- lectureGrille.html

Visualiser une grille

Nom de la grille:*04002030.txt*

Numero Grille: *04002030*

Resultat Orale

Bonne response	Note TOEIC
60	310

Resultat Ecrit

Bonne response	Note TOEIC
63	285

Resultat Total

Note TOEIC : 595

Probleme de lecture

Numero de question	MorceauxImage	BoutonModif
Q025		La Reponse est OK
Q026		La Reponse est OK

Conclusion

Résultat final

On a simplifié le processus d'installation d'implication à l'aide de 'cx_freeze', et on a ajouté des fonctions qui permettent l'utilisation plus intelligente et plus pratique grâce à l'utilisation d'un navigateur HTLM

Améliorations possibles

A cause de le temps limite, il reste quelques améliorations possibles :

1. Accélérer le temps de traitement d'images (peut-être en C++)
2. Génération du fichier synthèse des notes

Apports personnels

Ce projet nous permet découvrir un langage qui peut s'utiliser dans de nombreux contextes et s'adapter à tout type d'utilisation—python. Et pour notre projet, sa bibliothèque de traitement d'image—le PIL(Python Imaging Library) nous offert une méthode pratique à réaliser la partie de traitement d'images dans notre projet . On a appris aussi appliquer la partie 'base de donnée locale ' de HTML5 à satisfaire le sauvegarde et le changement locale sur le page HTML.

Par ce projet, on connait qu'il est nécessaire de réfléchir sur des aspects suivants avant et en train de faire un projet :

1) Faire des effets sur déduire le durée de traitement
2) Considérer les problèmes possibles rencontrés quand l'utilisateur applique notre application, et trouver les solutions
3) Quand on réfléchit sur la réalisation de chaque étape de projet, on doit considérer la réalisation de projet entièrement. Ça veut dire qu'on doit laisser la possibilité pour continuer l'étape suivante de projet.