Créer sur la machine le répertoire Documents/ExoPython

Pour tous les exercices les fichiers source commenceront par la ligne

# -*-coding:Latin-1 -*

## Exercice 1

Déterminer si une année est bissextile

Dans le fichier exo1.py coder l'algorithme suivant :

- Si une année n'est pas multiple de 4, elle n'est pas bissextile.
- Si elle est multiple de 4, on regarde si elle est multiple de 100.
    - Si c'est le cas, on regarde si elle est multiple de 400.
        - Si c'est le cas, l'année est bissextile.
        - Sinon, elle n'est pas bissextile.
    - Sinon, elle est bissextile.

Pour saisir une chaîne au clavier et la convertir en un nombre :

```
anneeStr = input("saisir une année : ")
annee = int(anneeStr)
print (annee)
```

Idée : créer une variable booléenne qui indique si l'année est bissextile ou non

```
bissextile = False
```

# Exercice 2 :

Création d'un module au sens de Python

Du code d'affichage de tables d'addition et multiplication va être écrit dans un fichier .py
Ces fonctions vont être utilisées dans un autre programme.

Procéder comme suit :

Créer le fichier exo2Module.py
Définir les fonctions :

- addition(nb, max=10) qui affiche la table d'addition de nb
- multiplication(nb, max=10) qui affiche la table de multiplication de nb

Sauvegarder exo2Module.py

Créer le fichier exo2Emploi.py
Commencer le fichier par

```
import os
from exo2Module import *
```

et le terminer par

```
os.system("pause")
```

Ajouter au centre du code pour afficher la table d'addition de 6 et table de multiplication par 12, pour les valeurs de 1 à 10

Sauvegarder exo2Emploi.py
Double clic sur exo2Emploi.py et vérifier le résultat

Que fait un double clic sur exo2Module.py ?

On souhaite maintenant ajouter dans le fichier exo2Module.py un auto test des fonctions du module.
C'est très simple !
Il suffit d'ajouter en fin du fichier

```
from os import system
if __name__ == "__main__":
```

puis le code de test : faire une table d'addition et une table de multiplication

# Exercice 2 bis :

Créer et utiliser un « Package » au sens Python

Créer le répertoire Documents/ExoPython/outils
Dans ce répertoire créer le fichier vide __init__.py  (2 underscores de chaque côté).
C'est l'existence de ce fichier qui indique à Python qu'il s'agit d'un package .

Créer dans ce répertoire le fichier operations.py et y définir les fonctions soustractions et division qui affichent les tables.

Revenir dans le fichier exo2Emploi.py de l'exercice précédent et y ajouter :

from outils.operations import *

puis utiliser les fonctions soustraction et division

## Exercice 3 :

Utilisez une *exception* pour calculer, dans une boucle évoluant de -3 à 3 compris, la valeur de sin(*x*)/*x*. sin(0)/0 vaut 1

## Exercice 4 :

définir la liste : liste =[17, 38, 10, 25, 72], puis effectuez les actions suivantes :
– triez et affichez la liste ;
– ajoutez l'élément 12 à la liste et affichez la liste ;
– renversez et affichez la liste ;
– affichez l'indice de l'élément 17 ;
– enlevez l'élément 38 et affichez la liste ;
– affichez la sous-liste du 2eau 3eélément ;
– affichez la sous-liste du début au 2eélément ;
– affichez la sous-liste du 3eélément à la fin de la liste ;
- affichez la sous-liste complète de la liste ;

# Exercice 5 :

# Lecture et écriture de fichier.

Le fichier fourni DataExo5.txt contient à partir de la ligne 2, des lignes de type
date ;float ;int
ex 01/01/2016 00:02:00;5.633771;1

En moyenne chaque ligne a 30 s d'écart par rapport à la ligne précédente.
Certaines sont manquantes.
Réaliser un programme qui lit toutes les lignes du fichier, les recopie dans le fichier en sortie DataExo5Out.txt, et quand des dates sont manquantes, elles sont créées dans le fichier de sortie avec pour float et int les dernières valeurs connues.
Ex

| Fichier en entrée | Fichier en sortie |
|---|---|
| 01/01/2016 00:00:30;5.179501;1 | 01/01/2016 00:00:30;5.179501;1 |
| 01/01/2016 00:**01:00**;5.137346;1 | 01/01/2016 00:01:00;5.137346;1 |
| 01/01/2016 00:**02:00**;5.633771;1 | 01/01/2016 00:**01:30;5.137346;1 ligne ajoutée** |
| | 01/01/2016 00:02:00;5.633771;1 |

Dates :
Il faudra utiliser le module datetime et datetime.strptime() pour convertir une chaîne en date.
datetime.strftime() pour convertir une date en chaîne
datetime.timedelta() permet de définir un intervalle de temps.

Le code suivant peut vous aider :

```
        date1 = d.datetime.strptime("01/01/2016 00:03:00","%d/%m/%Y %H:%M:%S")
        date2 = d.datetime.strptime("01/01/2016 00:04:00","%d/%m/%Y %H:%M:%S")
        diff = date2 - date1

if (diff > d.timedelta(seconds=30)):
    print("plus de 30s")
```

Le format du fichier en entrée est en unicode.
Il faut importer le module codecs et utiliser, pour ouvrir le fichier, codecs.open() avec l'encoding 'utf-16'

# Exercice 6:

# Parcours récursif de répertoires

Réaliser un script  exo6.py  qui accepte en paramètre un chemin vers un répertoire et qui liste tous les fichiers et répertoires sous ce chemin, récursivement.
exo6.py lancé sans paramètre affiche son mode d'emploi
exo6.py <chemin>  teste si la chaîne <chemin> caractérise un chemin vers un répertoire

pistes :
Récupération de paramètres d'entrée (arguments d'entrée) :

import sys
nbArg = len(sys.argv)  #  nbre de parametres
prog= sys.argv[0]  # le nom du fichier .py
param1= sys.argv[1]  # le 1er parametre

test si un chemin désigne un répertoire existant  os.path.isdir(path)

Pour un parcours récursif de répertoires, article http://apprendre-python.com/page-gestion-fichiers-dossiers-python

```
for path, dirs, files in os.walk(folder_path):
    for filename in files:
        print(filename)
```

# Exercice 7

Création d'un fichier Excel
Article sur http://xlsxwriter.readthedocs.io/

extraire le fichier XlsxWriter-master.zip
ouvrir une fenêtre cmd Windows et se déplacer dans le répertoire XlsxWriter-master avec la commande cd

exécuter la commande `python setup.py install`

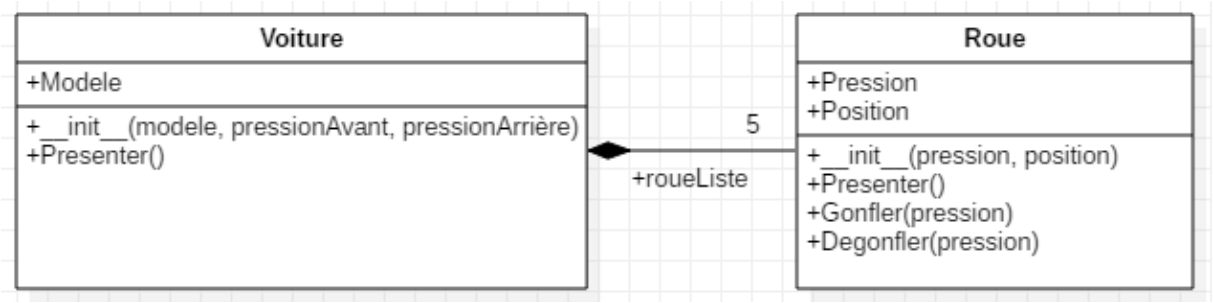
Essayer le code du site http://xlsxwriter.readthedocs.io/example_demo.html#ex-demo

Reprendre ensuite le code de l'exercice 5 et, au lieu de produire le fichier DataExo5Out.txt, produire le fichier Excel DataExo5Out.xlsx

# Exercice 8

Les premiers pas en conception Objet

Le diagramme UML de classe suivant représente la conception et relation entre une voiture et ses roues




Une Voiture est caractérisée par son modèle (ex : C3, 207, SMART …).
Elle possède 5 roues (roue de secours comptée).
Les roues avant sont gonflées à la pressionAvant, idem pour l'arrière.
La roue de secours est gonflée avec la plus forte des 2 pressions.
La méthode Presenter() réalise simplement un print de son modèle et de ses 5 roues.

Une roue est caractérisée par sa pression en bars et sa position : avant gauche , avant droit, arrière gauche, arrière droit, roue de secours.
La méthode Presenter() réalise simplement un print de sa pression et de sa postion

Ecrire le code Python dans le fichier exo8.py :

Pour décrire la classe Voiture
Pour décrire la classe Roue

Puis créer un objet Voiture  C3 , pression avant à 2.5 et pression arrière à 2.2
Afficher les caractéristiques de cette voiture pour obtenir quelque chose comme ceci :
Voiture C3
Pneu Avant Gauche pression 2.5
Pneu Avant Droite pression 2.5
Pneu Arrière Gauche pression 2.2
Pneu Arrière Droite pression 2.2
Pneu De secours pression 2.5

Pour démarrer il est possible d’utiliser le squelette de code suivant :

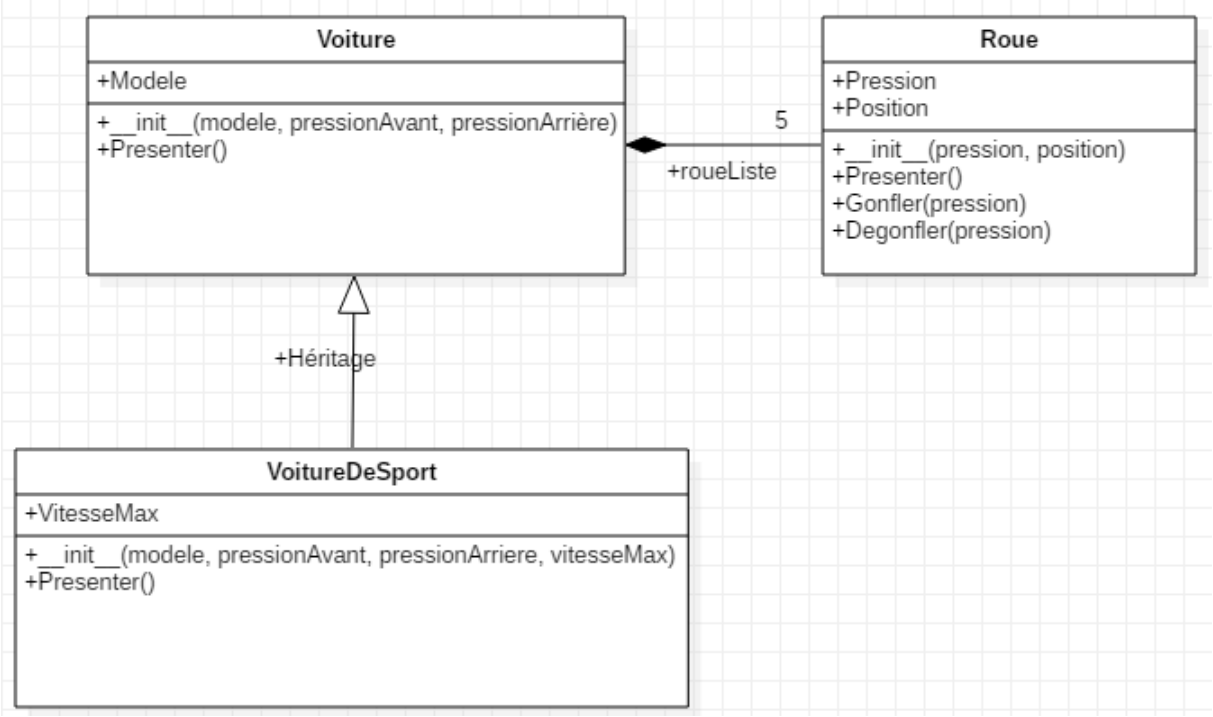
```
#!/usr/bin/python
#-*- coding: utf-8 -*-

class Voiture:

    def __init__(self, modele, pressionAvant, pressionArrière):
        # to do

    def Presenter(self, ):
        # to do

class Roue:
        self.Pression = None
        self.Position = None

    def __init__(self, pression, position):
        # to do

    def Presenter(self, ):
        # to do

    def Gonfler(self, pression):
        # to do

    def Degonfler(self, pression):
        # to do
```

# Exercice 9

Héritage

Copier le fichier Exo8.py en Exo9.py

Notre conception fait apparaître les Voitures de sport comme étant un cas particulier de Voiture. Le diagramme de classes UML en fait état :




Dans Exo9.py coder la classe dérivée VoitureDeSport.
Le constructeur __init__() de Voiture de sport doit appeler le constructeur de Voiture par Voiture.__init__()
Presenter() de VoitureDeSport  doit afficher en plus la vitesse max.

# Exercice 10

Polymorphisme de classe

Continuer le code de l’exercice 9.

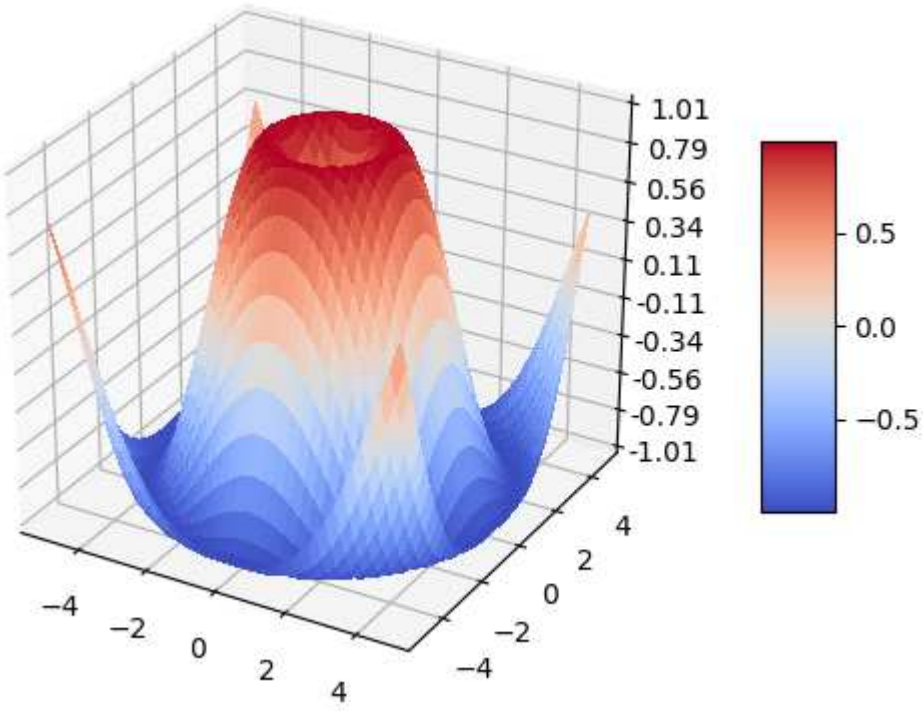
En dehors des codes de classe, créer une liste qui contient 2 objets Voiture et 2 objets VoitureDeSport.
Ecrire ensuite une boucle for sur cette liste pour Presenter chaque objet

# Exercice 11

Utilisation de MatPlotLib

Parcourir la page http://apprendre-python.com/page-creer-graphiques-scientifiques-python-apprendre
Essayer et comprendre le code de certains graphiques de cette page.
Si le composant MatPlotLib est absent, depuis une fenêtre commande de Windows, taper la commande :
pip install matplotlib

Essayer également le code de https://matplotlib.org/gallery/mplot3d/surface3d.html



# Exercice 12

Première utilisation de wxPython

Il faut d'abord importer le librairie wx :
Soit par  pip install wxPython  depuis une fenêtre commande Windows

Sur le site  https://wxpython.org/pages/overview/  récupérer le code helloworld2.py  et l'exécuter.

Lire le code.

# Exercice 13

Première utilisation de Glade : nous allons créer un fenêtre avec 2 boutons et une zone de saisie de texte. L'appui sur le bouton « Rouge » change la couleur de la zone de saisie en rouge. Idem pour le bouton « Bleu ».

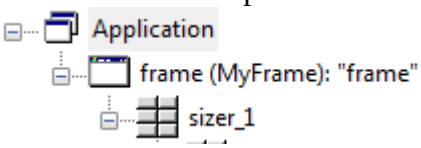
Installer Glade
**https://sourceforge.net/projects/wxglade/files/wxglade/0.8.0/**

La doc de Glade se trouve en local à l'endroit où s'est installé Glade : docs/htmt/html.index

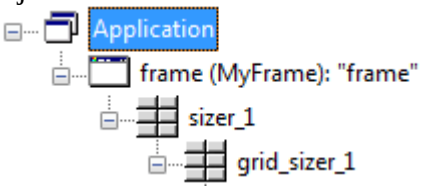
Glade permet de générer du code Python et d'utiliser la bibliothèque wxPython. La doc de wxPython est sur https://docs.wxpython.org

Réaliser une petite application avec Glade.

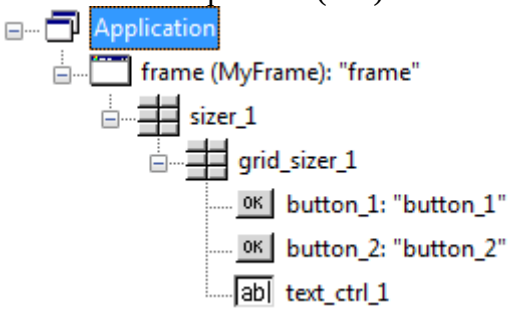
- File / New
- Dans le TreeView placer une frame sous le nœud Application



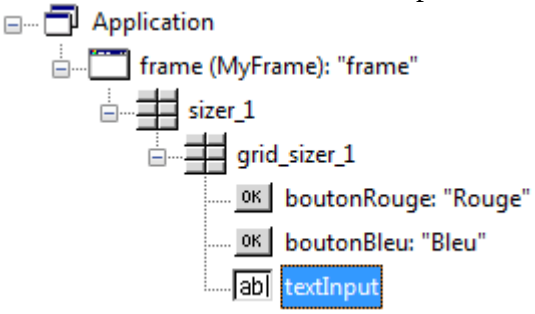

- Ajouter un Grid Sizer sous le nœud sizer1 . 1 ligne 3 colonnes



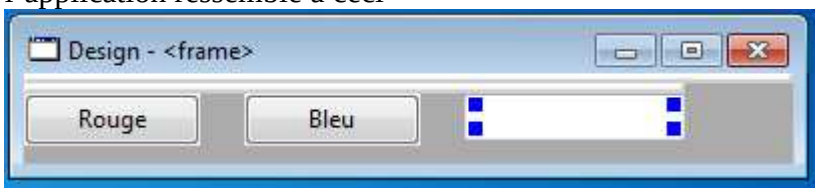

- Placer dans chaque case (slot) un bouton, un bouton, un text input



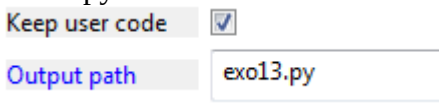

- Renommer les boutons et text input de cette façon



- l'application ressemble à ceci




- Pour générer le code Python :
  - Cliquer sur le noeud Application
  - Dans la fenêtre Properties cocher Keep user code et choisir un nom de fichier : exo13.py

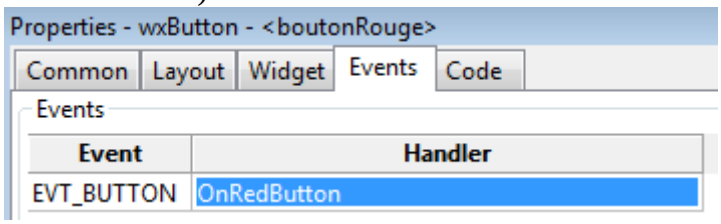



  - Cliquer sur Generate Source

Ouvrir le code dans Wing. Analysons ensemble le code

Il faut maintenant associer du code à l'appui du bouton rouge et bleu : déclarer des événements :

- Fenêtre Properties du bouton rouge, onglet Event
  Pour l'événement souris EVT_BUTTON, déclarer le handler (la méthode appelée sur événement clic) OnRedButton




- Idem pour bouton bleu
- Sauvegarder Ctrl S . Générer le code Ctrl G
- Regarder le code dans Wing
- Remplacer le code

```
print("Event handler 'OnRedButton' not implemented!")
event.Skip()
```

- Par le code qui change la couleur du texte input. Ne pas oublier de terminer par self.textInput.Refresh()

- Essayer le résultat

- Ensuite l'appui sur les boutons doit changer un message bulle (tooltip) sur le text input (« fond rouge » et « fond bleu »)

# Exercice 14

A partir de l’exercice 6 (recherche de répertoires) constituer une ihm utilisant un treeview.

Ce treeview montre les répertoires et sous répertoires à partir d’un chemin de base (racine) ;