Charte de codage

Indentation :

Les tabulations sont interdites il faut utiliser 4 espaces à la place.

identifiant : (noms des variables)

Les noms des variables doivent être en minuscule, éviter la bicapitalisation et préférer ma_variable à maVariable.

Utiliser des noms de variable pluriels pour indiquer les objets conteneurs (list, tuple, ...)

```
personnes = []
[ ... ]
personne = personnes[ 3 ]
```

Les noms de classe doivent commencer par une majuscule.

Pour les noms des fonctions et méthodes, utiliser des groupes verbaux.

```
def ajouter_domaine( ...
def fermer_menu( ...
```

Les espaces :.

Les opérateurs doivent être précédés et suivis d'espaces sauf l'opérateur point.

```
ma_liste[ 3 : 4 ]
a = num_personne + i
consol.get_text()
```

Ne pas faire

```
a=-numPersonne[:2]+i
```

Chaîne de documentation :

Les fonctions et méthodes doivent possèder une chaîne de documentation sauffe si leur nom laisse facilement entendre leurs utilisations.

```
class Text_en_couleur( object ) :
	"""Pas la peine de dire que c'est une classe pour instancier du texte en couleur"""

	def __init__( self, text, color ):
		"""
		Color doit être au format #xxx x étant un chiffre en hexa.
		"""

	def get_color( self ):
		"""
		Retourne la couleur au format #xxx x étant un chiffre en hexa.
		"""
		return self.color

	def get_text( self ) :
		return self.text

	def get_text_coloried( self ) :
		return self.text
```

Quand utiliser des commentaire :

Lorsque une variable contiendra plusieurs données imbriquée, complexe, pour indiquer la forme des données.

```
self.dico_table_config = {}	# De la forme { "table_1" :
				# { "defaut" : { "champ_1" : ( "couleur_1", "font_1" ), ... }
				#   "perso" : { "champ_1" : ( "couleur_2", "font_2" ), ... } ... }
```

Lorsque la signification de la variable ne peut laisser entendre le traitement.
Dans cet exemple, min(personne) est la première personne à arriver, min(personne) +1 la deuxième et num_personne % 2 == 0 désigne une personne de sexe féminin.

```
# Si c'est la première femme à être arrivée.
if num_personne == min( ( personne for personne in personnes if personne % 2 == 0 ) ) :
	...
```

Les IHM.

Utiliser :

```
from tkinter import Label, Tk
```

et non

```
from tkinter import *
```

qui pollue l'espace de nom.

Le gestionnaire de position doit être placé à la fin et précédé du commentaire # GDP puis, suivi du gestionnaire d'événement précédé du commentaire # GDE

Exemple :

```
class Trigger_maker( Frame ) :
        """ IHM pour la création de trigger. """
        def __init__( self, master ) :
                Frame.__init__( self, master )

                # GDP
                widget_1.grid(   row = 0, column = 1 )
                widget_242.grid( row = 1, column = 1 )

                # GDE
                widget_1.bind(<1>, foo )
```

Cas d'une IHM à plusieurs contenants :

```
        frame_1 = Frame( self, ...
        frame_2 = Frame( self, ...
        label_1 = Label( frame_1, ...
        label_2 = Label( frame_1, ...
        label_3 = Label( frame_2, ...

        # GDP
        frame_1.grid( row = 0, column = 1 )
        frame_2.grid( row = 1, column = 1 )

        # GDP frame_1.   ( sous entendu, GDP des esclaves de frame_1 )
        label_1.grid( row = 0, column = 1 )
        label_2.grid( row = 1, column = 1 )

        # GDP frame_2.
        label_3.grid( row = 0, column = 1 )
```