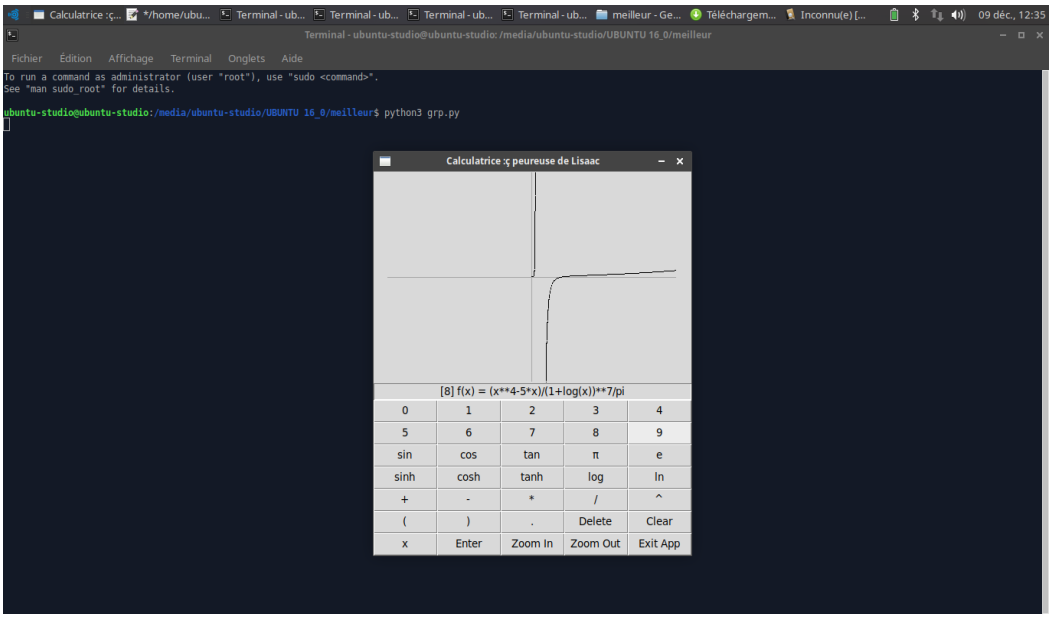

Pourquoi Python :

Coté interface

Python offre une multiple d'option pour le développement GUI (Graphical User Interface).

Et pour toutes ces technologies d'interface graphique, l'une des plus connus est bien TKINTER . Car étant comme une bibliothèque graphique libre et module externe installé avec le langage Python par défaut , rend les sources plus facile à l'utilisation pour créer les applications GUI.

Au tout début je travaillais sur projet avec un parser et autre sur Java mais j'ai renoncé pour en faire celui là

Coté langage orienté objet POO

Au tout début , j'étais assez réticent vis-à-vis à ce langage , sachant que je ne connaissais rien au tout début le concernant juste que c'était langage programmation orienté et que c'est assez utilisé dans les applications et programmation pour le développement logiciel et même web .

```
class Point:
    "Definition d'un point geometrique"
```

Mais après une documentation j'ai appris qu'en Python, toutes les données, y compris les nombres et les chaînes sont des objets. Un nombre est un objet, une chaîne est un objet, une liste est un objet, un dictionnaire est un objet et chaque donnée est un objet. Les objets de même nature ont le même type.

Un objet contient deux parties : les propriétés et les méthodes. Les propriétés permettent de stocker la valeur de l'objet et les méthodes représentent les fonctions permettant de manipuler la valeur de l'objet.

Pour cela donc j'ai préféré y procéder sans pour cela mettre plusieurs class seulement en démarrant d'une class et même d'une unique ,cela nous aiderait a appeler tous les objet avec des méthodes (les fonctions) qu'on leur attribuera, Mais j'ai du importer une librairie Math , exemple de ligne 9 (fichier calculatrice.py)

```
From math import sin
    "exemple pour la fonction sin dans la libraire math"
```

Comme j'aurai pu appeler Mathlotlib.pyplot ou la librairie Numpy… (ex fichier Numpy.py)

```
From numpy import np
    "exemple librairie numpy qui contient un certain nomrbre de fonction que j'uti
liserai avec le mot clé np à chaque appel"
```

Je pouvais aussi utiliser SELF pour faire appel au attributs de l'objet , mais vu les objets n'ont que de simple paramètres qui sont a leur tour des objets , donc comme Habituellement, on crée un objet et l'assigne à une variable. J'ai utilisé la variable pour faire référence à l'objet. Parfois, un objet n'a pas besoin d'être référencé plus tard. Dans ce cas, je pouvais créer un objet sans l'attribuer explicitement à une variable, comme indiqué ci-dessous de ligne 74, (fichier calculatrice.py)

```
« exemple librairie print("NON-INT PW (dbl click^) ")
```

Outils Tkinter (pour interface) et librairie Math :

Pour pouvoir développer cela, j'ai voulu laissé mon idée du **TP3** ou j'ai programmé toutes les classes , afin de faire en plus rapide j'ai juste emporté la librairie même si j'ai dû redéfinir les id car la librairie math devait pêtre complétée par mon code.

Puis pour la partie **graphique :** j'ai dû suivre des cours en ligne sur TKINTER , pour création de fenêtre par exemple:

```
from tkinter import *
fenetre = Tk()
champ_label = Label(fenetre, text="Salut les Zér0s !")
# On affiche le label dans la fenêtre
champ_label.pack()
# On démarre la boucle Tkinter qui s'interompt quand on ferme la fenêtre
fenetre.mainloop()
```

Problème pour la fonction binaire/ résolu avec une idée Morse 'pensant à une liste':

Comme j'ai eu du mal à la coder en faisant de la conversion avec du python , donc j'ai voulu voir le problème autrement , j'ai d'abord essayé de coder une fonction qui fait **morse** crypte et décrypte , disant que le {_} et le {.} faisant référence à 0 et 1, et c'est ce que j'ai pu faire aux derniers moments et qu'il fallait que je pense à une formule mathématique qui m'éviterai d'écrire tout en liste en pensant à sorte de suite d'itération ? Mais là j'ai eu une belle idée de juste faire une permutation de réordonné l'écriture émise par l'utilisateur et et puis faire le décryptage(conversion) , mais cela comme j'ai du vous le reexpliquer à la presentation n'était pas

suffisant donc il fallait que j'utilise du modulot ou …, comme il me restait peu de temps ce qui expliquait la non inclusion dans l'interface graphique :

Mise detexte → puis Inversion de nieme a &ere place ainsi de suite → puis Conversion

Il me restait juste la partie inversion qui me manquait !!

```
MORSE_CODE_DICT = {
    'A':'.-',
    'B':'-...',… à voir le fichier morse.py
```

A partir de dela j'arrive à ceci :

```
BINARY_CODE_DICT = {
    '0':'0000',
    '1':'0001',…  à voir le fichier binary.py
```

Voila un aperçu de l'exécution de fichier binary .