CIPoWer v1.0

Guide utilisateur

DESBIN LOU
DIROFF MATHIS
GAUCHET NOÉMIE
HAŸ NATHAN
HEYDEL LILIAN
ZIMMERLIN JULES

École et observatoire
des **sciences de la Terre**
Université de Strasbourg

Table des matières

Table des figures

Liste des tableaux

Pré-requis et dépendances

Hormis Python, l'utilisation de *CIPoWer* nécessite les librairies et dépendances suivantes (la plupart d'entre elles seront automatiquement installées à l'aide de la commande `pip install ...` ou en utilisant le script .cmd dans l'archive) :

— Numpy
— Matplotlib
— Mpltern
— Pyrolite
— PyMuPDF

Chapitre 1 | Introduction générale

1.1 La méthode *CIPW*

La méthode *CIPW*, pour Cross, Iddings, Persson et Washington (Cross et al., 1902) est une méthode de calcul permettant de déterminer de façon expérimentale et mathématique la nature d'une roche.
Concrètement, cette méthode utilise un système de tableau où l'on attribue les éléments disponibles à chaque minéral, tout en prenant en compte les différentes *passes* à effectuer (voir Pétri (2023a) et Pétri (2023b) pour plus de détail sur la méthode).
Plusieurs considérations sont à prendre en compte lors de l'utilisation de cette méthode :

- tous les minéraux sont à l'équilibre ;
- la roche a une composition anhydre (pas de biotite, amphibole…) ;
- elle ne tient compte que des solutions solides majeures et non des mineures ;
- elle prend en compte des minéraux normatifs définis, à composition déterminée et idéale.

1.2 Utilisation des résultats

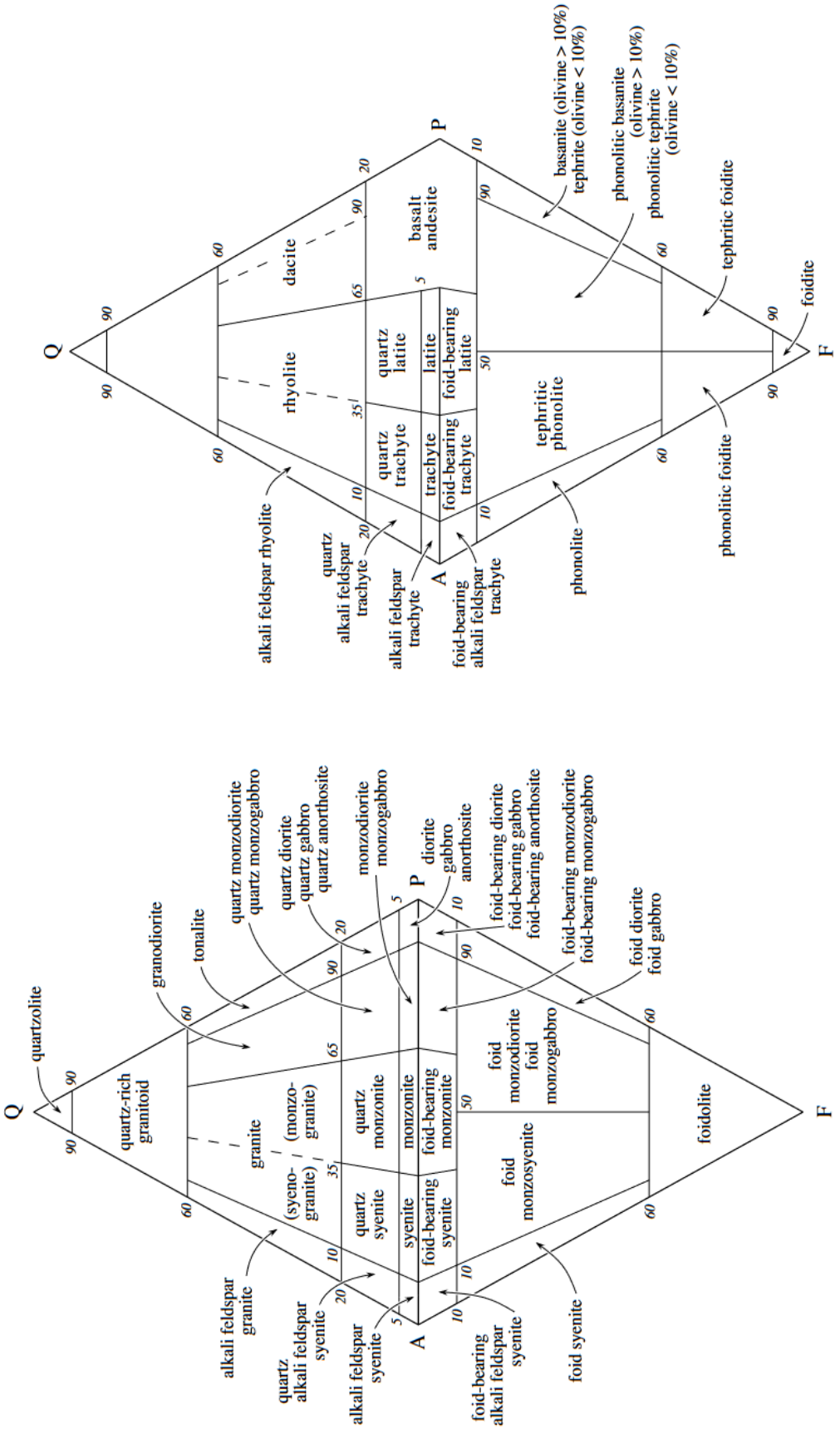
Dès lors que les concentrations minérales sont connues, il est possible de placer la roche dans différents diagrammes selon leur type.

1.2.1 Diagramme de Streckeisen

Le diagramme de Streckeisen, développé par Streckeisen (1974, 1979), est un diagramme utilisable pour les roches plutoniques et volcaniques. Celui-ci présente, pour chaque type de roche, une double pyramide : une pyramide aux pôles QAP (Quartz | Feldspath Alcalin | Feldspath Plagioclase) et FAP (Feldspathoïde | Feldspath Alcalin | Feldspath Plagioclase). Ceux-ci sont présentés sur la figure 1.1. Bien que plutôt sommaire, ce type de diagramme s'associe parfaitement à la méthode *CIPW* grâce au calcul de la concentration minérale de chaque pôle QAPF. La table 1.1 indique les différents minéraux et leur pôle. Ces diagrammes sont utilisés dans *CIPoWer* pour les roches plutoniques.

TABLE 1.1 – Pôles QAPF et minéraux associés.

Pôle	**Q**	**A**	**P**	**F**
Minéraux	Quartz	Orthose	Albite	Néphéline
			Anorthite	Leucite
				Kalsilite



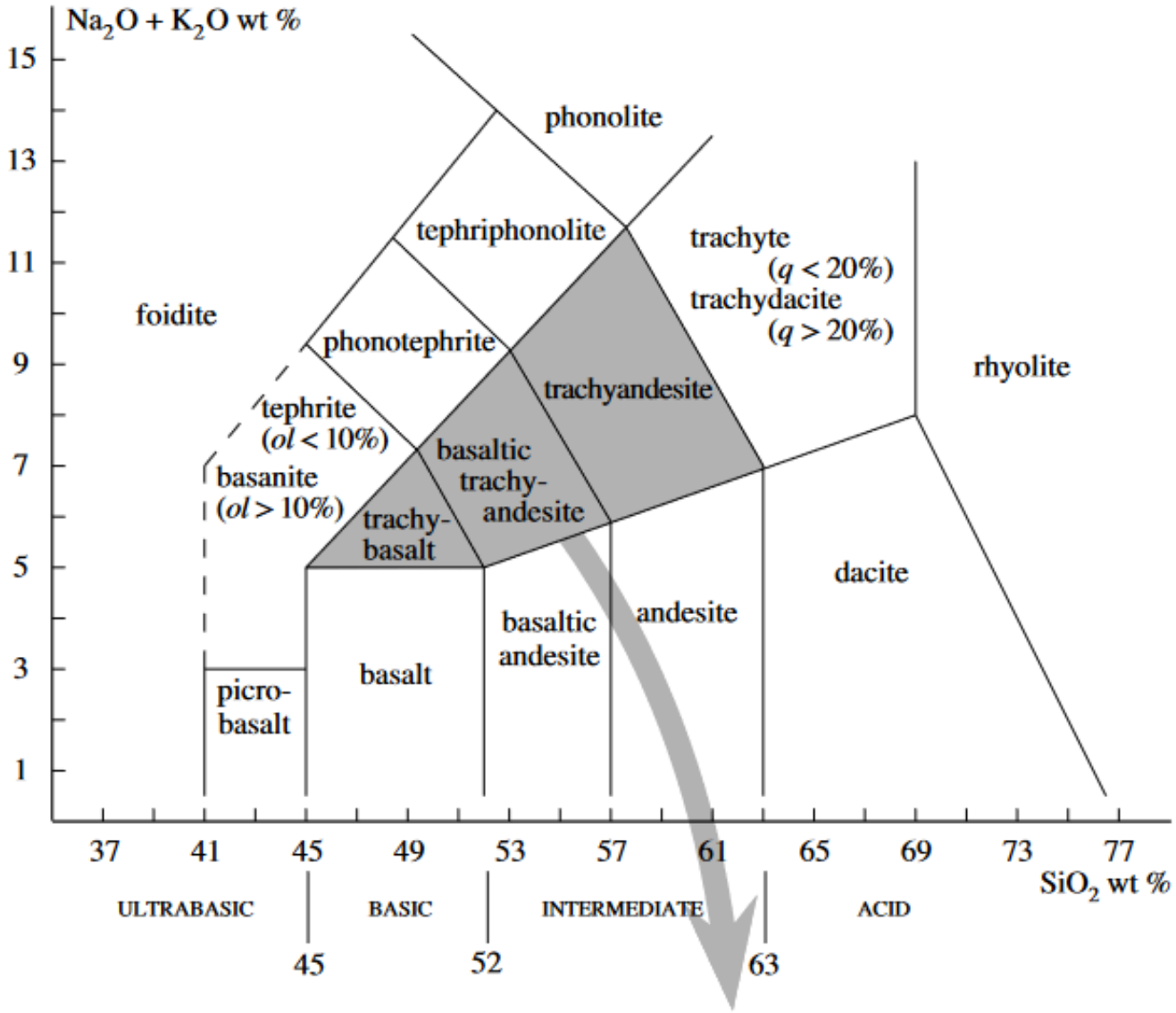
(a) Diagramme pour les roches plutoniques.

(b) Diagramme pour les roches volcanique.

FIGURE 1.1 – Diagrammes de Streckeisen, d'après Maitre et al. (2004).

1.2.2 Diagramme TAS

Le diagramme TAS, pour *Total Alkali Silica*, est une classification développée en 1986 par Le Bas et al. (1986). Cette classification est présentée sur la figure 1.2.



Further subdivisions of shaded fields	trachybasalt	basaltic trachyandesite	trachyandesite
$Na_2O - 2.0 \geq K_2O$	hawaiite	mugearite	benmoreite
$Na_2O - 2.0 < K_2O$	potassic trachybasalt	shoshonite	latite

FIGURE 1.2 – Diagramme TAS, d'après Maitre et al. (2004).

Celle-ci classe les roches volcaniques en fonction de leurs teneurs en silice et en éléments calco-alcalin. Elle met en valeur les séries volcaniques, parfaits indicateurs du contexte géodynamique de la roche. Ce type de diagramme est utilisé dans *CIPoWer* pour les roches volcaniques.

Chapitre 2 | **Pourquoi *CIPoWer* ?**

CIPoWer permet d'automatiser et d'éviter les calculs fastidieux de concentrations molaires et permet d'obtenir rapidement des valeurs QAPF pour une roche. Ceci, en plus des diagrammes mentionnés dans le chapitre 1, permet de nommer la roche et d'afficher des informations concernant celle-ci. Une photographie de la roche s'affiche également.

Chapitre 3 | Utilisation de *CIPoWer*

3.1 Saisie des valeurs

Lorsque lancé, le programme se présente en deux parties : une contenant des cases vides à remplir et une seconde partie de *vide*. La figure 3.1 montre ces cases, où il faut les remplir en fonction de ce qui a été déterminé concernant la concentration moléculaire de la roche. Certaines exceptions (comme le monzogabbro au chapitre 5) de normes considèrent que le fer se trouve uniquement dans le Fe_2O_3 et non dans le FeO ; il faut alors saisir 0 pour FeO.

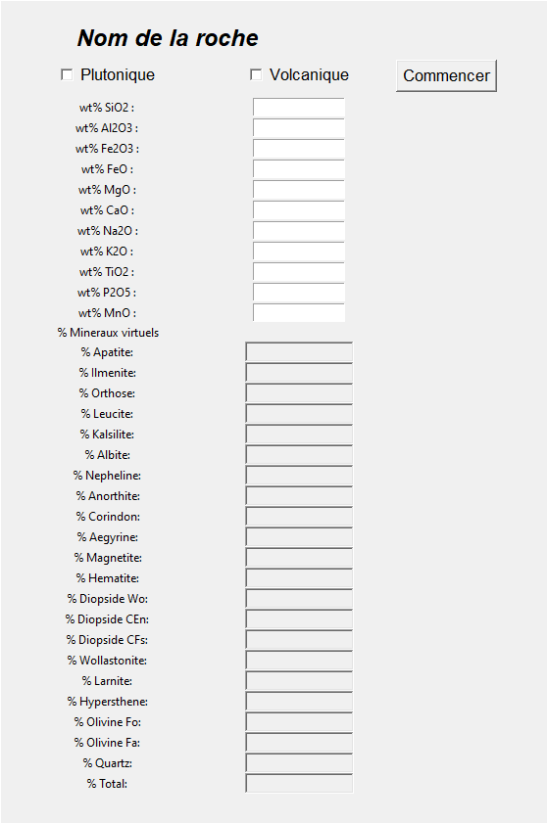


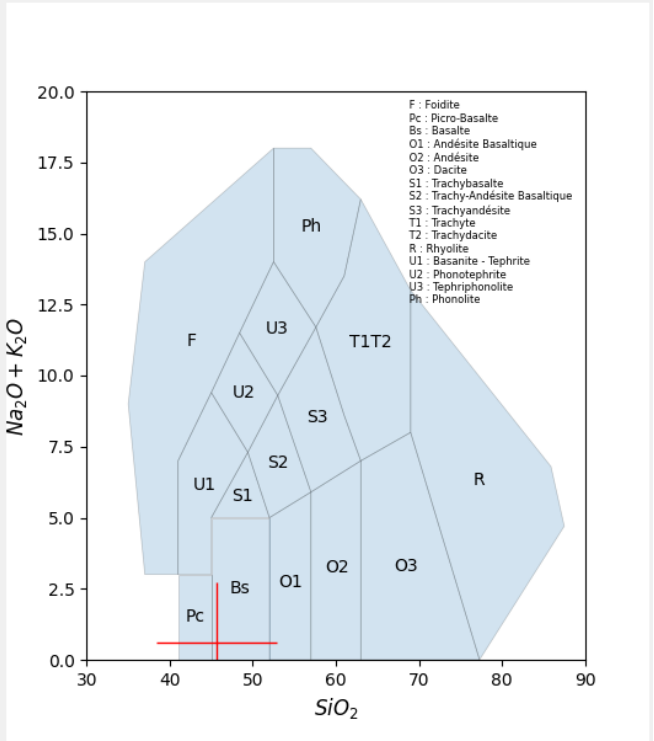
FIGURE 3.1 – Interface de saisie des valeurs lors du démarrage de CIPoWer

3.2 Interprétation

Nous allons ici prendre pour exemple le basalte disponible dans les exemples du programme (chapitre 5). Lorsque celui-ci est chargé et que l'on presse le bouton *Commencer*, plusieurs choses apparaissent, comme visible sur les figures 3.2.

% Mineraux virtuels	
% Apatite:	0.336
% Ilmenite:	0.456
% Orthose:	0.556
% Leucite:	0.0
% Kalsilite:	0.0
% Albite:	4.192
% Nepheline:	0.0
% Anorthite:	26.41
% Corindon:	0.0
% Aegyrine:	0.0
% Magnetite:	1.856
% Hematite:	0.0
% Diopside Wo:	10.092
% Diopside CEn:	7.032368896925847
% Diopside CFs:	2.20127305605786
% Wollastonite:	0.0
% Larnite:	0.0
% Hypersthene:	49.59127667269451
% Olivine Fo:	0.9660759493670634
% Olivine Fa:	5.936289330922094
% Quartz:	0.0
% Total:	109.62528390596736

(a) Table des résultats



(b) Diagramme TAS

FIGURE 3.2 – Résultats de *CIPoWer* pour un basalte

Aucune autre manipulation n'est requise par l'utilisateur. Les résultats, les diagrammes, les photos et les descriptifs s'affichent automatiquement.

3.3 Bugs

Le programme contient certains bugs pouvant gêner son utilisation :

— plotter une roche plutonique en premier a tendance à figer la zone de diagramme, quelque soit le type de roche plottée après (même si ceci est plus apparent avec les roches volcaniques). Il faut alors redémarrer le programme.

— il se peut qu'une erreur de calcul du type "division par 0" arrive pour certaines normes. Il n'y a pas de solution à ceci pour l'instant.

— Cocher un des boutons Volcanique ou Plutonique puis dé-sélectionner les deux n'empêche pas le programme de se lancer : celui-ci sauvegardera le dernier choix enregistré par l'utilisateur.

Chapitre 4 | Fonctionnement du logiciel

4.1 Calcul de la norme

On se base ici sur la méthode énoncée dans Pétri (2023b). L'objectif de cette partie essentielle du code est de déterminer les minéraux formés en partant des valeurs des poids d'oxydes résultant de l'analyse de la composition chimique de la roche, puis en répartissant ces poids d'oxydes dans chaque minéral formé selon un ordre bien précis. A partir de là, nous obtiendrons le nombre de chaque minéral formé. De ce nombre, nous pouvons déterminer la proportion de masse de chaque minéral. Cette proportion de masse est essentielle puisqu'elle permet de nous situer dans les différents diagrammes afin d'identifier notre roche.
Nous avons fait le choix de manipuler des listes, car cela permet de faire des boucles lorsque l'on a besoin par exemple de passer de la série de valeurs des poids d'oxydes au nombre d'oxydes. Cela permet aussi de se repérer : par exemple la première valeur de la liste poids d'oxydes est celle de la silice. Pour le premier minéral ce sera l'apatite et ainsi de suite. Les index de chaque liste utilisée sont représentés en rouge dans la table 4.1.

TABLE 4.1 – Index des lignes et colonnes utilisées dans le code

Norme CIPW

Constituants		SiO2 sst.	SiO2	Al2O3	Fe2O3	FeO	MgO	CaO	Na2O	K2O	TiO2	P2O5	CO2		MnO	LOI		Total
% poids oxyd.																		
Poids mol.			60	102	160	72	40	56	62	94	80	142			71			
Proportion mol.			0	1	2	3	4	5	6	7	8	9			10			
					MnO+FeO											Prop. Mol.	Poids mol.	% mnx virt.
Apatite	P2O5 3.3CaO 1/3F2																336	0
Ilménite	TiO2 FeO																152	1
Div.																		
Orthose	K2O Al2O3 6SiO2																556	2
Leucite	K2O Al2O3 4SiO2																436	3
Kalsilite	K2O Al2O3 2SiO2																316	4
Albite	Na2O Al2O3 6SiO2																524	5
Néphéline	Na2O Al2O3 2SiO2																284	6
Anorthite	CaO Al2O3 2SiO2																278	7
Corindon	Al2O3																102	8
Aegyrine	Na2O Fe2O3 4SiO2																462	9
Magnétite	Fe2O3 FeO																232	10
Hématite	Fe2O3																160	11
Diopside	CaO SiO2 Wo																116	12
	MgO SiO2 CEn																100	13
	FeO SiO2 CFs																132	14
Wollastonite	CaO SiO2																116	15
Larnite	2CaO SiO2																172	16
Hypersthène	MgO SiO2 En																100	17
	FeO SiO2 Fs																132	18
Olivine	2MgO SiO2 Fo																140	19
	2FeO SiO2 Fa																204	20
SiO2 restant																Quartz	60	21
		Q		A		P		F			Nom :						Total	

Le programme commence donc par l'initialisation de certaines listes qui contiennent des constantes. Ce sont les valeurs des masses des oxydes pour une mole pour la première liste. La deuxième liste vide est la liste du nombre d'oxydes.

```
1 PoidsMol = [60,102,160,72,40,56,62,94,80,142,71]
2 PropMol = []
```

Nous pouvons ensuite créer cette boucle pour demander à l'utilisateur les valeurs des poids d'oxydes de sa roche et dans la foulée calculer les nombres de chaque oxyde dans la liste `PropMol` :

```
for i in range(len(PoidsOxyd)) :
  PropMol.append(round((PoidsOxyd[i]/PoidsMol[i]*1000)))
```

L'ordre de création des minéraux est discontinu. Nous avons parfois besoin de revenir en arrière, c'est pourquoi nous avons créé une liste composée de 0. Cette liste sera notre résultat final lorsque toutes les étapes de formation de minéraux auront été faites. L'oxyde de magnésium et celui du fer, se comportant de la même manière chimiquement, sont additionnés pour ne former ''qu'un seul oxyde".

```
Mineraux = [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]

# MnO+FeO
PropMol[3] = PropMol[3] + PropMol[10]
```

Une fois que tout a été initialisé, nous pouvons commencer à former les minéraux tout en suivant l'ordre de formation. Chaque minéral se forme jusqu'à épuisement d'un des oxydes qui le compose. Cas spécial pour la silice (PropMol[0]) dont la valeur peut être négative. C'est pourquoi elle n'est pas prise en compte dans les conditions de la boucle. Chaque tour de boucle est compté et *i* nous renvoie le nombre de minéraux d'albite formés dans le cas présent.

```
    # formation albite
i=0
while PropMol[0]!=0 and PropMol[1] != 0 and PropMol[6] != 0 :
  i=i+1
  PropMol[0]=PropMol[0]-6
  PropMol[1]=PropMol[1]-1
  PropMol[6]=PropMol[6]-1
Mineraux[5]=i
```

Cas spécial lorsque le minéral formé est une solution solide comme le cas du diopside. Il faut que les proportions des oxydes disponibles soient conservées au moment de la formation du minéral. Par exemple, s'il y a deux fois plus de fer que de magnésium, il faut former deux fois plus de diopside ferrique que magnésien. C'est pourquoi il y a une étape supplémentaire pour respecter cette condition.

```
# formation Diopside WO
i = 0
MgFe = PropMol[3] + PropMol[4]
while MgFe > 0 and PropMol[5] > 0 :
  PropMol[3] -= PropMol[3]/MgFe
  PropMol[4] -= PropMol[4]/MgFe
  PropMol[5] -= 1
  PropMol[0] -= 1
  MgFe -= 1
  i=i+1
Mineraux[12]=i
Mineraux[13]=i*(PropMol[4]/MgFe)
Mineraux[14]=i*(PropMol[3]/MgFe)
```

Lorsque nous arrivons à la formation du quartz, si la valeur du nombre de Si02 restant est négative, alors nous allons devoir utiliser des passes afin de libérer de la silice à partir de certains minéraux. Une passe consiste à *détruire* un minéral consommant beaucoup de silice pour former un minéral ayant une composition chimique très proche, mais nécessitant moins de silice. Il y a en tout 4 passes que l'on peut écrire sous forme d'équations présentées ci-dessous. Dans le pire des cas, nous avons besoin d'utiliser les 4 passes. Dans le meilleur des cas, nous n'avons même pas besoin de faire de passe, car le système est saturé en silice. La première passe consiste à détruire de l'hypersthène pour former de l'olivine, la deuxième consiste à détruire de l'albite pour former de la néphéline, la troisième à détruire de l'orthose pour former de la leucite et la dernière à former de la kalsilite à partir de la leucite.

Les équations 4.1 à 4.4 sont utilisées pour chaque système, *x* est à chaque fois le minéral *détruit* et *y* le minéral formé :

$$\begin{cases} x + y = \mathrm{SiO_2} \text{ disponible} \\ x + 2y = (\mathrm{FeO} + \mathrm{MgO}) \end{cases} \tag{4.1}$$

$$\begin{cases} x + y = \mathrm{Na_2O} \\ 6x + 2y = 6\mathrm{Na_2O} - \text{ déficit de silice} \end{cases} \tag{4.2}$$

$$\begin{cases} x + y = \mathrm{K_2O} \\ 6x + 4y = 6\mathrm{K_2O} - \text{ déficit de silice} \end{cases} \tag{4.3}$$

$$\begin{cases} x + y = \mathrm{K_2O} \\ 4x + 2y = 4\mathrm{K_2O} - \text{ déficit de silice} \end{cases} \tag{4.4}$$

L'équation permet de trouver les valeurs x et y, si l'une de ces valeurs est négative alors il faut aller à la passe suivante car évidement, on ne peut former un nombre de minéraux négatif. La passe ci-dessous est celle de la néphéline :

```
if deficitsilice < 0 :
    deficit=abs(deficitsilice)
    y = deficit / 4
    x = L2[1]+deficit/6-1/3*y
    if x>=0 and y>=0 :
      Mineraux[5] = x
      Mineraux[6] = y

    else :

      if x<0 or y<0 :

        deficitsilice = deficitsilice + Mineraux[5] * 6- 2 * y
        Mineraux[5] = 0
        Mineraux[6] = y

        # deficit de silice persistant apres la formation de la nephiline =
            formation de leucite

        deficit=abs(deficitsilice)
        y = deficit/2
        x = L3[1] + deficit/6 - 2/3 * y

        if x>=0 and y>=0 :
          Mineraux[2] = x
          Mineraux[3] = y
```

Comme l'oxyde accompagnant la silice n'est utilisé que dans le minéral détruit et le minéral formé, nous pouvons nous permettre de saisir les valeurs dont nous avons besoin à différents endroits dans le code. Cela évite des lignes de calcul pour retrouver ces valeurs d'une manière calculatoire.

```
#Prise d'information dans le cas de la formation de leucite a partir de l'orthose
# L3 = [SiO2 , K2O ]
L3 = [ PropMol[0] , PropMol[7] ]
```

Une fois que tous nos systèmes ont été résolus, nous obtenons enfin la liste des nombres des 22 minéraux formés. Cette liste est rangée selon l'ordre de la première figure. Une fois que nous avons cette liste des valeurs des 22 minéraux, nous allons faire une boucle avec une liste constante contenant les masses molaires de chaque minéral associé. Cela va nous permettre d'obtenir la liste des masses (et non plus le nombre) de chaque minéral formé et ainsi, nous allons pouvoir déterminer les pourcentages massiques de chaque minéral pour placer notre roche dans les différents diagrammes en divisant les valeurs obtenues par 1000 (car elles ont été initialement multipliées par 1000 pour éviter de manipuler des petites valeurs).

```
PoidsMolMineraux = [336,152,556,436,316,524,284,278,102,462,232,160,116,100,132
                    ,116,172,100,132,140,204,60]
PropMin = []
for i in range(len(PoidsMolMineraux)) :
  PropMin.append((Mineraux[i] * PoidsMolMineraux[i]) / 1000)
```

Enfin il ne reste plus qu'à récupérer les valeurs souhaitées, ici les différents pôles du diagramme de Streckeisen.

```
F = PropMin[6] + PropMin[3] + PropMin[4]

A= PropMin[2]

P= PropMin[5] + PropMin[7]

Q = PropMin[21]

CalcoAlcalin =   PoidsOxyd[6] + PoidsOxyd[7]
Neph_Leuc = [PropMin[6], PropMin[3]]
```

4.2 Partie graphique

4.2.1 Affichage des diagrammes

Les parties concernant les graphiques de Streckesein et TAS se basent en grande partie sur les librairies Python `Pyrolite` (Williams et al., 2020) et `Mpltern` (Ikeda, 2023). La première pour le diagramme QAP & le diagramme TAS, et la seconde pour le diagramme FAP.

Concernant les diagrammes de Streckeisen, comme `Pyrolite` n'est pas capable pour l'heure d'afficher un double diagramme QAPF[1] mais uniquement le diagramme ternaire QAP, le diagramme ternaire FAP a été reconstruit à l'aide de `Mpltern`.

Plot du diagramme de Strekeisen

Diagramme QAP Si les conditions sont réunies, on plot le diagramme QAP comme ceci :

```
    diagramme_Streckeisen = QAP(fontsize=7,linewidth=0.5,figsize=(6,6))
     Roche = (Q, A, P)
     df = pd.DataFrame(data=Roche)
     df.pyroplot.scatter(ax=diagramme_Streckeisen, marker = '+',c='r', s = 5000,
         axlabels=False)
     diagramme_Streckeisen.set_title("Streckeisen's diagram for QAP rocks")
     canvas = FigureCanvasTkAgg(plt.gcf(), master=root)
     canvas.draw()
     canvas.get_tk_widget().pack(side=tk.RIGHT, fill=tk.BOTH, expand=1)
     root.update()
```

Diagramme FAP Si les conditions sont réunies, on construit le diagramme FAP à l'aide de `Mpltern` et on plot les résultats comme ceci :

```
    ax = plt.subplot(projection="ternary")
     ligne_horizontale = list(range(2))
     ligne_horizontale[0] = [0.1, 0.8, 0.1]
     ligne_horizontale[1] = [0.6, 0.3, 0.1]
     slope = [0, 1, -1]
     for i in range(len(ligne_horizontale)):
         ax.axline(xy1=ligne_horizontale[i], slope=slope, color='k', linewidth
            =1)
     ligne_droite = [0, .1, .9]
     ligne_droite_stop = [0.6, 0.05, 0.35]
```


[1] Cette fonctionnalité est actuellement en cours de développement: https://github.com/morganjwilliams/pyrolite/issues/49

```
10        ax.plot([ligne_droite[0], ligne_droite_stop[0]],[ligne_droite[1],
              ligne_droite_stop[1]],[ligne_droite[2], ligne_droite_stop[2]],color='k',
               linewidth=1)
11        ligne_droite = [0, .9, .1]
12        ligne_droite_stop = [0.6, 0.35, 0.05]
13        ax.plot([ligne_droite[0], ligne_droite_stop[0]],[ligne_droite[1],
              ligne_droite_stop[1]],[ligne_droite[2], ligne_droite_stop[2]],color='k',
               linewidth=1)
14        ligne_droite = [0.1, .45, .45]
15        ligne_droite_stop = [0.6, 0.2, 0.2]
16        ax.plot([ligne_droite[0], ligne_droite_stop[0]],[ligne_droite[1],
              ligne_droite_stop[1]],[ligne_droite[2], ligne_droite_stop[2]],color='k',
               linewidth=1)
17        ligne_droite = [0, .35, .65]
18        ligne_droite_stop = [0.055, 0.1842 , 0.3158]
19        ax.plot([ligne_droite[0], ligne_droite_stop[0]],[ligne_droite[1],
              ligne_droite_stop[1]],[ligne_droite[2], ligne_droite_stop[2]],color='k',
               linewidth=1)
20        ligne_droite = [0,  .65, .35]
21        ligne_droite_stop = [0.055 , 0.3158, 0.1842]
22        ax.plot([ligne_droite[0], ligne_droite_stop[0]],[ligne_droite[1],
              ligne_droite_stop[1]],[ligne_droite[2], ligne_droite_stop[2]],color='k',
               linewidth=1)
23        pc = ax.scatter(F/100, A/100, P/100,marker = '+',c='r', s = 5000)
24        ax.set_tlabel("Feldspathoid")
25        ax.set_llabel("Alkali␣Feldspar")
26        ax.set_rlabel("Plagioclase")
27        ax.set_title("Streckeisen's␣diagram␣for␣FAP␣rocks")
28        canvas = FigureCanvasTkAgg(plt.gcf(), master=root)
29        canvas.draw()
30        canvas.get_tk_widget().pack(side=tk.RIGHT, fill=tk.BOTH, expand=1)
31        diagramme_est_dessine = True
32        root.update()
```

Plot du diagramme TAS Si les conditions sont réunies, on plot le diagramme TAS comme ceci :

```
1         diagramme_Tas = TAS(add_labels=True, which_model="LeMaitreCombined",
              fontsize=7,linewidth=0.5,figsize=(6,6), fill=True, alpha = 0.2)
2         diagramme_Tas.set_title("TAS␣diagram␣for␣volcanic␣rocks")
3         CalcoAlcalin = ((Na2O*62) + (K2O*94))/1000
4         ValeursTAS = ((SiO2*60)/1000, CalcoAlcalin)
5         tas = pd.DataFrame(data = ValeursTAS)
6         tas.pyroplot.scatter(ax=diagramme_Tas, c="r", s=5000, marker="+", alpha=1,
              axlabels=False)
7         canvas = FigureCanvasTkAgg(plt.gcf(), master=root)
8         canvas.draw()
9         canvas.get_tk_widget().pack(side=tk.RIGHT, fill=tk.BOTH, expand=1)
10        diagramme_est_dessine = True
11        root.update()
```

4.2.2 Affichage des descriptions

Selon la proportion des minéraux trouvés dans chaque échantillon, on obtient une correspondance avec une roche magmatique que l'on stocke dans la variable *def_roche*. Cette variable permet de créer des balises délimitant les fiches descriptives des roches.

```
1         def_roche = ""
2   def afficher_contenu_pdf(def_roche):
3     global scrollbar, label
4     def_roche = nom_roche.cget("text")
```

```
    # Creation des balises pour les roches du fichier
    caractfin = "_"
    balise_debut = "DEBUT_"+def_roche+caractfin
    balise_fin = "FIN_"+def_roche+caractfin
```

Ensuite, dans la structure try, le programme ouvre et parcourt tout le fichier PDF contenant les fiches pour en extraire le texte compris entre les balises.

```
        chemin_pdf = "Data\\fiches_roches.pdf"

    try:
        pdf_doc = fitz.open(chemin_pdf)

        texte_complet = ""
        for page_num in range(pdf_doc.page_count):
            page = pdf_doc[page_num]
            texte_complet += page.get_text("text")

        # Filtrer le texte en fonction du nom de la roche et des balises de debut/fin
        texte_filtr  = filtrer_texte_par_roche(texte_complet, balise_debut, balise_fin)
        text_widget.config(state=tk.NORMAL)
        text_widget.delete("1.0", tk.END)  # Efface le contenu actuel du Text widget
        text_widget.insert(tk.END, texte_filtr )
        text_widget.config(state=tk.DISABLED)

        # Mettre a jour le widget de defilement vertical
        scrollbar.config(command=text_widget.yview)
        text_widget.config(yscrollcommand=scrollbar.set)
        label.config(text="")

    except FileNotFoundError:
        label.config(text="Erreur : Fichier introuvable.")
    except Exception as e:
        label.config(text=f"Erreur : {str(e)}")

# Fonction pour filtrer le texte entre les balises
def filtrer_texte_par_roche(texte_complet, balise_debut, balise_fin):
    # Rechercher les balises de debut et de fin pour le nom de la roche
    debut_index = texte_complet.find(balise_debut)
    fin_index = texte_complet.find(balise_fin, debut_index + len(balise_debut))
```

Enfin, le texte filtré apparaît sur l'interface dans un widget consacré à son affichage, non modifiable par l'utilisateur. Pour chaque nouvelle utilisation, le texte est réinitialisé et une nouvelle fiche apparaît. Si le PDF ou les balises sont erronées, un message d'erreur apparaîtra.

```
# Filtrer le texte en fonction du nom de la roche et des balises de debut/fin
texte_filtr  = filtrer_texte_par_roche(texte_complet, balise_debut, balise_fin)
text_widget.config(state=tk.NORMAL)
text_widget.delete("1.0", tk.END)  # Efface le contenu actuel du Text widget
text_widget.insert(tk.END, texte_filtr )
text_widget.config(state=tk.DISABLED)

# Mettre a jour le widget de defilement vertical
scrollbar.config(command=text_widget.yview)
text_widget.config(yscrollcommand=scrollbar.set)
label.config(text="")
```

4.2.3 Affichage des images

L'affichage des photos d'échantillon de roches repose principalement sur l'utilisation de l'outil canvas de la librairie tkinter. Celui-ci permet de définir une zone dans la fenêtre du programme comme étant une zone dite "de dessin", où l'on peut tracer des

figures (utilisé notamment pour la partie 4.2.1) ou dans le cas présent, importer des fichiers png/jpeg/jpg.
La première étape consiste à créer le canvas tout en définissant sa taille et sa position dans l'affichage, ainsi qu'importer une première image transparente. La présence d'une image affichée dès le lancement du programme (et non pas uniquement après pression du bouton "Commencer") est indispensable pour permettre l'actualisation du canvas par la suite. Ci-dessous le code correspondant à ces actions :

```
affichage_photo = Canvas(root, width=300, height=200)
photo = PhotoImage(file='Data\images\image_rien.png')
Image_Affichee = affichage_photo.create_image(0,0, image=photo, anchor='nw')
affichage_photo.place(x=400, y=100)
affichage_photo.pack(expand=True, side=LEFT)
```

Pour actualiser la photo affichée afin de correspondre à chaque roche calculée lors d'une même utilisation du programme, il faut disposer du dossier contentant la banque de photos. Celles-ci sont toutes au même format, à la même dimension (celle du canvas) et encodées à partir du même logiciel. Ce dossier est joint dans le fichier compressé du programme.
Le programme lui-même contient une fonction Afficher_Photo permettant de récupérer la photo correspondante dans le fichier. Celle-ci se déroule en 3 étapes :

- la récupération de la photo sélectionnée depuis le dossier et la stocker dans une variable. On utilise une deuxième variable, indexphoto, qui contient le nom du fichier image à récupérer.
- demander au canvas d'afficher cette nouvelle photo. On utilise la fonction .itemconfigure pour remplacer l'image précédente par celle voulue.
- conservation de la référence de l'image. Cette dernière ligne permet de garder les modifications effectuées même une fois sorti de la fonction. Sans elle, l'image ne s'actualiserait qu'une seule fois à la première pression du bouton "Démarrer" puis ne fonctionnerait plus.

```
def Afficher_Photo(indexphoto):
      photo2=PhotoImage(file=f'Data\\images\\{indexphoto}.png')
      affichage_photo.itemconfigure(Image_Affichee, image=photo2)
      affichage_photo.image=photo2
```

La variable indexphoto est définie dans la section du code permettant de définir la roche formée, à hauteur d'une ligne par sous-section comme sur cet exemple :

```
if F <= 60 and F > 10:
          pourc_olivine = PropMin[21] + PropMin[20]
          if P_prim <= 10 and P_prim >= 0:
            nom_roche.configure(text="Phonolite")
            indexphoto = 'Phonolite'
          elif P_prim <= 50 and P_prim >= 10:
            nom_roche.configure(text="Tephriphonolite")
            indexphoto = 'Tephriphonolite'
```

Chapitre 5 | Exemples

CIPoWer embarque plusieurs normes pré-enregistrées de roches plutoniques et volcaniques, disponibles en faisant *Fichier* → *Charger une roche*. Ces dernières sont présentées ci-dessous dans la table 5.1.

TABLE 5.1 – Valeurs utilisées pour chacun des exemples. * pour le FeO du monzogabbro, indique que la méthode de calcul considère que tout le fer est dans Fe_2O_3.

Roche / Constituant (% poids oxydant)	SiO_2	Al_2O_3	Fe_2O_3	FeO	MgO	CaO	Na_2O	K_2O	TiO_2	P_2O_5	MnO
Granite [a]	70,83	14,52	0,51	2,5	0,78	0,65	2,48	5,56	0,32	0,14	0,05
Basalte [b]	45,65	10,56	1,26	8,26	17,87	10,34	0,48	0,11	0,26	0,1	0,15
Monzodiorite [c]	53,45	17,9	3,88	3,16	1,82	6,2	6,23	2,46	1,15	0,47	0,21
Monzogabbro [d]	45,45	12,81	11,78*		13,52	10,99	1,25	1,81	1,48	0,55	0,17
Rhyolite [e]	72,34	13,25	1,77	1,03	0,35	0,89	3,96	4,89	0,27	0,15	0,09
Hawaiite [f]	48,9	16,7	4,6	5,2	5,8	7,15	4,8	1,7	2,17	0,69	0,17

a. Von Eller (1961)
b. Everard et al. (1997)
c. Andersen et Sørensen (1993)
d. Lains Amaral et al. (2022)
e. Santos et Hartmann (2021)
f. Pillard et al. (1980)

Bibliographie et références

Andersen, T., & Sørensen, H. O. (1993). Crystallization and metasomatism of nepheline syenite xenoliths in quartz-bearing intrusive rocks in the Permian Oslo rift, SE Norway. *Norsk Geologisk Tidsskrift*, *73*, 250-266. https://api.semanticscholar.org/CorpusID:128356243

Cross, W., Iddings, J. P., Pirsson, L. V., & Washington, H. S. (1902). A Quantitative Chemico-Mineralogical Classification and Nomenclature of Igneous Rocks. *The Journal of Geology*, *10*(6), 555-690. https://doi.org/10.1086/621030

Everard, J., Calver, C., Taheri, J, & Dixon, G. (1997). Geology of the islands of southwestern Bass Strait.

Ikeda, Y. (2023). *yuzie007/mpltern : 1.0.2* (Version 1.0.2). Zenodo. https://doi.org/10.5281/zenodo.8289090

Lains Amaral, J., Mata, J., & Santos, J. F. (2022). The Carboniferous shoshonitic (s.l.) gabbro–monzonitic stocks of Veiros and Vale de Maceira, Ossa-Morena Zone (SW Iberian Massif) : Evidence for diverse subduction-related lithospheric metasomatism. *Geochemistry*, *82*(4), 125917. https://doi.org/https://doi.org/10.1016/j.chemer.2022.125917

Le Bas, M. J., Maitre, R. W. L., Streckeisen, A., Zanettin, B., & on the Systematics of Igneous Rocks, I. S. (1986). A Chemical Classification of Volcanic Rocks Based on the Total Alkali-Silica Diagram. *Journal of Petrology*, *27*(3), 745-750. https://doi.org/10.1093/petrology/27.3.745

Maitre, R., Streckeisen, A., Zanettin, B., Le Bas, M., Bonin, B., & Bateman, P. (2004). Igneous Rocks : A Classification and Glossary of Terms. *Cambridge University Press*, *-1*.

Pillard, F., Maury, R., Tournemire, R., & Massal, P. (1980). Évolution hydrothermale de l'hawaiite d'Espalion (Aveyron). Hydrothermal evolution of the Espalion hawaiite (Aveyron, France). *Bulletin de Minéralogie*, *103*(1), 101-106. https://doi.org/10.3406/bulmi.1980.7379

Pétri, B. (2023a). Diaporama de présentation CIPW, L3 STUE 2023-2024. *Cours de pétrologie magmatique*. https://moodle.unistra.fr/pluginfile.php/672649/mod_resource/content/4/TD6_Petri_CIPW_Presentation.pdf

Pétri, B. (2023b). Guide CIPW, L3 STUE 2023-2024. *Cours de pétrologie magmatique*. https://moodle.unistra.fr/pluginfile.php/672650/mod_resource/content/5/TD6_Petri_CIPW_Guide.pdf

Santos, J. O., & Hartmann, L. A. (2021). Chemical classification of common volcanic rocks based on degree of silica saturation and CaO/K₂O ratio. *Anais da Academia Brasileira de Ciências*, *93*(3), e20201202. https://doi.org/10.1590/0001-3765202120201202

Streckeisen, A. (1974). Classification and nomenclature of plutonic rocks recommendations of the IUGS subcommission on the systematics of Igneous Rocks. *Geologische Rundschau*, *63*(2), 773-786. https://doi.org/10.1007/BF01820841

Streckeisen, A. (1979). Classification and nomenclature of volcanic rocks, lamprophyres, carbonatites, and melilitic rocks : Recommendations and suggestions of the IUGS Subcommission on the Systematics of Igneous Rocks. *Geology*, *7*. https://doi.org/10.1130/0091-7613(1979)7<331:CANOVR>2.0.CO;2

Von Eller, J. (1961). Les gneiss de Sainte-Marie-aux-Mines et les séries voisines des Vosges moyenne. *Service de la carte géologique d'Alsace et de Lorraine*. https://www.persee.fr/doc/sgeol_0080-9020_1961_mon_19_1

Williams, M. J., Schoneveld, L., Mao, Y., Klump, J., Gosses, J., Dalton, H., Bath, A., & Barnes, S. (2020). pyrolite : Python for geochemistry. *Journal of Open Source Software*, *5*(50), 2314. https://doi.org/10.21105/joss.02314

Annexes

Chapitre A | Version précédente du calcul de norme

Dans ses versions antérieure, le programme utilisait ce code pour calcluer la norme (tout en se basant également sur Pétri (2023b)) :

```
import random
def CIPower(pourc_SiO2,pourc_Al2O3,pourc_Fe2O3,pourc_FeO,pourc_MgO,pourc_CaO,
    pourc_Na2O,pourc_K2O,pourc_TiO2,pourc_P2O5,pourc_MnO) :
  SiO2=round(pourc_SiO2/60*1000)
  Al2O3=round(pourc_Al2O3/102*1000)
  Fe2O3=round(pourc_Fe2O3/160*1000)
  FeO=round(pourc_FeO/72*1000)
  MgO=round(pourc_MgO/40*1000)
  CaO=round(pourc_CaO/56*1000)
  Na2O=round(pourc_Na2O/62*1000)
  K2O=round(pourc_K2O/94*1000)
  TiO2=round(pourc_TiO2/80*1000)
  P2O5=round(pourc_P2O5/142*1000)
  MnO=round(pourc_MnO/71*1000)
  MnO_FeO=MnO+FeO

  #apatite
  i3=0
  while CaO>0 and P2O5>0 :
    P2O5=P2O5-1
    CaO=CaO-3.3
    if CaO<0 or P2O5<0 :
      P2O5=P2O5+1
      CaO=CaO+3.3
      break
    i3=i3+1
  CaO=round(CaO)
  apatite=i3

  #ilmenite
  i4=0
  while MnO_FeO>0 and TiO2>0 :
    MnO_FeO=MnO_FeO-1
    TiO2=TiO2-1
    i4=i4+1
  ilmenite=i4

  #orthose
  SiO2_5=SiO2
  K2O_5=K2O
  i5=0
  while Al2O3>0 and K2O>0 :
    SiO2=SiO2-6
    Al2O3=Al2O3-1
    K2O=K2O-1
    i5=i5+1
  orthose=i5

  #albite
  SiO2_6=SiO2
```

```
50    Na2O_6=Na2O
51    i6=0
52    while Al2O3>0 and Na2O>0 :
53      SiO2=SiO2-6
54      Al2O3=Al2O3-1
55      Na2O=Na2O-1
56      i6=i6+1
57    albite=i6
58
59    #anorthite
60    i7=0
61    while Al2O3>0 and CaO>0 :
62      SiO2=SiO2-2
63      Al2O3=Al2O3-1
64      CaO=CaO-1
65      i7=i7+1
66    anorthite=i7
67
68    #corindon
69    corindon=Al2O3
70
71    #aegyrine
72    i8=0
73    while Fe2O3>0 and Na2O>0 :
74      SiO2=SiO2-4
75      Fe2O3=Fe2O3-1
76      Na2O=Na2O-1
77      i8=i8+1
78    aegyrine=i8
79
80    #magnetite
81    i9=0
82    while Fe2O3>0 and MnO_FeO>0 :
83      Fe2O3=Fe2O3-1
84      MnO_FeO=MnO_FeO-1
85      i9=i9+1
86    magnetite=i9
87
88    #hematite
89    hematite=Fe2O3
90
91    #diopside
92    y11=CaO/(MnO_FeO/MgO+1)
93    x11=MnO_FeO/MgO*y11
94    y11=round(y11)
95    x11=round(x11)
96    i11=0
97    while CaO>0 :
98      CaO=CaO-1
99      SiO2=SiO2-1
100     i11=i11+1
101   MgO=MgO-y11
102   MnO_FeO=MnO_FeO-x11
103   SiO2=SiO2-y11-x11
104   diopsideWo=i11
105   diopsideCEn=y11
106   diopsideCFs=x11
107
108   #wollastonite
```

```
  i12a=0
  while CaO>0 :
    CaO=CaO-1
    SiO2=SiO2-1
    i12a=i12a+1
  wollastonite=i12a

  #hypersthene
  SiO2_12=SiO2
  MgO_12=MgO
  i12b=0
  while MgO>0 :
    MgO=MgO-1
    SiO2=SiO2-1
    i12b=i12b+1
  hypersthenEn=i12b
  MnO_FeO_12=MnO_FeO
  i12c=0
  while MnO_FeO>0 :
    MnO_FeO=MnO_FeO-1
    SiO2=SiO2-1
    i12c=i12c+1
  hypersthenFs=i12c

  #quartz ?
  if SiO2<0 :
    #2eme passe
    y14=MnO_FeO_12+MgO_12-SiO2_12
    x14=MnO_FeO_12+MgO_12-2*y14
    y14=round(y14)
    x14=round(x14)
    if x14<0 or y14<0 :
      #olivine
      hypersthenEn=0
      hypersthenFs=0
      SiO2=SiO2_12
      MgO=MgO_12
      MnO_FeO=MnO_FeO_12
      i15a=0
      while MgO>0 :
        MgO=MgO-2
        SiO2=SiO2-1
        if MgO<0 :
          MgO=MgO+2
          SiO2=SiO2+1
          break
        i15a=i15a+1
      olivineFo=i15a
      i15b=0
      while MnO_FeO>0 :
        MnO_FeO=MnO_FeO-2
        SiO2=SiO2-1
        if MnO_FeO<0 :
          MnO_FeO=MnO_FeO+2
          SiO2=SiO2+1
          break
        i15b=i15b+1
      olivineFa=i15b
      if SiO2<0 :
```

```
168           #nepheline
169           y16=-(SiO2)/4
170           x16=Na2O_6+SiO2/6-1/3*y16
171           y16=round(y16)
172           x16=round(x16)
173           albite=x16
174           nepheline=y16
175           SiO2=SiO2_6-6*albite-2*nepheline-2*anorthite-4*aegyrine-diopsideWo-
                 diopsideCEn-diopsideCFs-wollastonite-olivineFo-olivineFa
176           if SiO2<0 :
177             #leucite
178             y17=-(SiO2)/2
179             x17=K2O_5+SiO2/6-2/3*y17
180             y17=round(y17)
181             x17=round(x17)
182             orthose=x17
183             leucite=y17
184             K2O_17=K2O_5-x17
185             SiO2_17=SiO2_5-6*orthose
186             SiO2=SiO2_5-6*orthose-4*leucite-6*albite-2*nepheline-2*anorthite-4*
                   aegyrine-diopsideWo-diopsideCEn-diopsideCFs-wollastonite-olivineFo-
                   olivineFa
187             if SiO2<0 :
188               #kalsilite
189               y18=-(SiO2)/2
190               x18=K2O_17+SiO2/4-y17/2
191               y17=round(y17)
192               x17=round(x17)
193               leucite=x17
194               kalsilite=y17
195               SiO2=SiO2_17-4*leucite-2*kalsilite-6*albite-2*nepheline-2*anorthite
                     -4*aegyrine-diopsideWo-diopsideCEn-diopsideCFs-wollastonite-
                     olivineFo-olivineFa
196             else :
197               kalsilite=0
198           else :
199             leucite=0
200             kalsilite=0
201         else :
202           nepheline=0
203           leucite=0
204           kalsilite=0
205       else :
206         #hypersthene+olivine
207         olivineFa=random.randint(0,y14+1)
208         olivineFo=y14-olivineFa
209         hypersthèneFs=x14-MgO_12+2*olivineFo
210         hypersthèneEn=x14-hypersthèneFs
211         SiO2=SiO2_12-hypersthèneEn-hypersthèneFs-olivineFo-olivineFa
212         nepheline=0
213         leucite=0
214         kalsilite=0
215     else :
216       olivineFo=0
217       olivineFa=0
218       nepheline=0
219       leucite=0
220       kalsilite=0
221
```

```
222    quartz=SiO2
223
224    pourc_apatite=apatite*336/1000
225    pourc_ilmenite=ilmenite*152/1000
226    pourc_orthose=orthose*556/1000
227    pourc_leucite=leucite*436/1000
228    pourc_kalsilite=kalsilite*316/1000
229    pourc_albite=albite*524/1000
230    pourc_nepheline=nepheline*284/1000
231    pourc_anorthite=anorthite*278/1000
232    pourc_corindon=corindon*102/1000
233    pourc_aegyrine=aegyrine*462/1000
234    pourc_magnetite=magnetite*232/1000
235    pourc_hematite=hematite*160/1000
236    pourc_diopsideWo=diopsideWo*116/1000
237    pourc_diopsideCEn=diopsideCEn*100/1000
238    pourc_diopsideCFs=diopsideCFs*132/1000
239    pourc_wollastonite=wollastonite*116/1000
240    pourc_hypersthеneEn=hypersthеneEn*100/1000
241    pourc_hypersthеneFs=hypersthеneFs*132/1000
242    pourc_olivineFo=olivineFo*140/1000
243    pourc_olivineFa=olivineFa*204/1000
244    pourc_quartz=quartz*60/1000
245    pourc_total=pourc_apatite+pourc_ilmenite+pourc_orthose+pourc_leucite+
          pourc_kalsilite+pourc_albite+pourc_nepheline+pourc_anorthite+pourc_corindon+
          pourc_aegyrine+pourc_magnetite+pourc_hematite+pourc_diopsideWo+
          pourc_diopsideCEn+pourc_diopsideCFs+pourc_wollastonite+pourc_hypersthеneEn+
          pourc_hypersthеneFs+pourc_olivineFo+pourc_olivineFa+pourc_quartz
246    Q=round(pourc_quartz/pourc_total*100)
247    A=round(pourc_orthose/pourc_total*100)
248    P=round((pourc_albite+pourc_anorthite)/pourc_total*100)
249    F=round((pourc_nepheline+pourc_leucite+pourc_kalsilite)/pourc_total*100)
250    return Q,A,P,F
251
252 [Q,A,P,F]=CIPower(70.83, 14.52, 0.51, 2.5, 0.78, 0.65, 2.48, 5.56, 0.32, 0.14,
       0.05)
253 print(Q,A,P,F)
```

Chapitre B | Code du programme

```
from tkinter import*
from math import*
import numpy as np
import tkinter as tk
import matplotlib.pyplot as plt
import pandas as pd
import pyrolite
from pyrolite.plot import pyroplot
from pyrolite.util.plot.style import color_ternary_polygons_by_centroid
from pyrolite.plot.templates import QAP, TAS,FeldsparTernary
from pyrolite.util.synthetic import normal_frame
import mpltern
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg
import random
import csv
from tkinter.filedialog import asksaveasfile, asksaveasfilename, askopenfilename
import webbrowser
from tkinter import ttk
import fitz
import unittest

Liste_des_elements = ["SiO2","Al2O3", "Fe2O3", "FeO", "MgO", "CaO", "Na2O", "K2O"
    , "TiO2", "P2O5", "MnO"]

root = tk.Tk()
root.title("CIPoWer")
root.geometry("1920x1000")
logo = PhotoImage(file = 'Data\\CIPoWer_logo.png')
root.iconphoto(False, logo, logo)
#root.resizable(width=False, height=False)
# CADRE POUR LA FENETRE
cadre_titre = tk.Frame(root)
cadre_titre.pack(side=TOP)
cadre = tk.Frame(root)
cadre.pack(side=tk.LEFT)
label_width = 15
#ENTREE DES VALEURS PAR L'UTILISATEUR
titrewtSiO2=tk.Label(cadre,text="wt% SiO2 :",width=label_width)
titrewtSiO2.grid(row=2,column=1)
wtSiO2=tk.Entry(cadre,width=label_width)
wtSiO2.grid(row=2,column=2)
#Al2O3
titrewtAl2O3=tk.Label(cadre,text="wt% Al2O3 :",width=label_width)
titrewtAl2O3.grid(row=3,column=1)
wtAl2O3=tk.Entry(cadre,width=label_width)
wtAl2O3.grid(row=3,column=2)
#Fe2O3
titrewtFe2O3=tk.Label(cadre,text="wt% Fe2O3 :",width=label_width)
titrewtFe2O3.grid(row=4,column=1)
wtFe2O3=tk.Entry(cadre,width=label_width)
wtFe2O3.grid(row=4,column=2)
#FeO
```

```
52  titrewtFeO=tk.Label(cadre,text="wt% FeO :",width=label_width)
53  titrewtFeO.grid(row=5,column=1)
54  wtFeO=tk.Entry(cadre,width=label_width)
55  wtFeO.grid(row=5,column=2)
56  #MgO
57  titrewtMgO=tk.Label(cadre,text="wt% MgO :",width=label_width)
58  titrewtMgO.grid(row=6,column=1)
59  wtMgO=tk.Entry(cadre,width=label_width)
60  wtMgO.grid(row=6,column=2)
61  #CaO
62  titrewtCaO=tk.Label(cadre,text="wt% CaO :",width=label_width)
63  titrewtCaO.grid(row=7,column=1)
64  wtCaO=tk.Entry(cadre,width=label_width)
65  wtCaO.grid(row=7,column=2)
66  #Na2O
67  titrewtNa2O=tk.Label(cadre,text="wt% Na2O :")
68  titrewtNa2O.grid(row=8,column=1)
69  wtNa2O=tk.Entry(cadre,width=label_width)
70  wtNa2O.grid(row=8,column=2)
71  #K2O
72  titrewtK2O=tk.Label(cadre,text="wt% K2O :",width=label_width)
73  titrewtK2O.grid(row=9,column=1)
74  wtK2O=tk.Entry(cadre,width=label_width)
75  wtK2O.grid(row=9,column=2)
76  #TiO2
77  titrewtTiO2=tk.Label(cadre,text="wt% TiO2 :",width=label_width)
78  titrewtTiO2.grid(row=10,column=1)
79  wtTiO2=tk.Entry(cadre,width=label_width)
80  wtTiO2.grid(row=10,column=2)
81  #P2O5
82  titrewtP2O5=tk.Label(cadre,text="wt% P2O5 :",width=label_width)
83  titrewtP2O5.grid(row=11,column=1)
84  wtP2O5=tk.Entry(cadre,width=label_width)
85  wtP2O5.grid(row=11,column=2)
86  #MnO
87  titrewtMnO=tk.Label(cadre,text="wt% MnO :")
88  titrewtMnO.grid(row=12,column=1)
89  wtMnO=tk.Entry(cadre,width=label_width)
90  wtMnO.grid(row=12,column=2)
91
92  ListeCasesElements = [wtSiO2, wtAl2O3, wtFe2O3, wtFeO, wtMgO, wtCaO, wtNa2O,
        wtK2O, wtTiO2, wtP2O5, wtMnO]
93  diagramme_est_dessine = False
94
95  affichage_photo = Canvas(root, width=300, height=200) #cr ation de l'espace d'
        affichage de la photo
96  photo = PhotoImage(file='Data\\images\\image_rien.png')
97  Image_Affichee = affichage_photo.create_image(0,0, image=photo, anchor='nw') #
        Affichage de l'image
98  affichage_photo.place(x=300, y=100) #placer l'image a droite du tableau
99  affichage_photo.pack(expand=True, side=LEFT)
100
101 def calcul_norme(PoidsOxyd) :
102   PoidsMol = [60,102,160,72,40,56,62,94,80,142,71]
103   PropMol = []
104   for i in range(len(PoidsMol)) :
105     N = PoidsOxyd[i]/PoidsMol[i]*1000
106     PropMol.append(round(N))
107   Mineraux = [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]
```

```
108
109   # MnO+FeO
110   PropMol[3] = PropMol[3] + PropMol[10]
111   del PropMol[10]
112
113
114
115   # formation apatite
116   i=0
117   while PropMol[9]!=0 and PropMol[5] >= 3.3 :
118     i=i+1
119     PropMol[9]=PropMol[9]-1
120     PropMol[5]=PropMol[5]-3.3
121   Mineraux[0]=i
122
123   # formation ilm nite
124   i=0
125   while PropMol[8]!=0 and PropMol[3] != 0 :
126     i=i+1
127     PropMol[8]=PropMol[8]-1
128     PropMol[3]=PropMol[3]-1
129
130   Mineraux[1]=i
131
132   #Prise d'information dans le cas de la formation de Leucite a partir de l'
          Orthose
133   # L3 = [SiO2 , K2O ]
134   L3 = [ PropMol[0] , PropMol[7] ]
135
136   # formation Orthose
137   i=0
138   while PropMol[1] > 0 and PropMol[7] > 0 :
139     i=i+1
140     PropMol[0]=PropMol[0]-6
141     PropMol[1]=PropMol[1]-1
142     PropMol[7]=PropMol[7]-1
143
144   Mineraux[2]=i
145
146   # prise d'information dans le cas de la 3e passe
147   # L2 = [ SiO2 , Na2O ]
148   L2 = [ PropMol[0] , PropMol[6] ]
149
150   # formation albite
151   i=0
152   while PropMol[1] != 0 and PropMol[6] != 0 :
153     i=i+1
154     PropMol[0]=PropMol[0]-6
155     PropMol[1]=PropMol[1]-1
156     PropMol[6]=PropMol[6]-1
157   Mineraux[5]=i
158
159
160   # formation anorthite
161   i=0
162   while PropMol[1] > 0 and PropMol[5] > 0 :
163     i=i+1
164     PropMol[0]=PropMol[0]-2
165     PropMol[1]=PropMol[1]-1
```

```
    PropMol[5]=PropMol[5]-1

  Mineraux[7]=i

  # formation corindon
  i=0
  while PropMol[1]>0 :
    i=i+1
    PropMol[1]=PropMol[1]-1
  Mineraux[8]=i

  # formation aegyrine
  i=0
  while PropMol[6]!=0 and PropMol[2]!=0 :
    i=i+1
    PropMol[6]=PropMol[6]-1
    PropMol[0]=PropMol[0]-1
    PropMol[2]=PropMol[2]-1
  Mineraux[9]=i

  # formation magnetite
  i=0
  while PropMol[2]!=0 and PropMol[3]!=0 :
    i=i+1
    PropMol[3]=PropMol[3]-1
    PropMol[2]=PropMol[2]-1
  Mineraux[10]=i

  # formation hematite
  i=0
  while PropMol[2]!=0 :
    i += 1
    PropMol[2]=PropMol[2]-1
  Mineraux[11]=i

  # formation Diopside WO
  i = 0
  MgFe = PropMol[3] + PropMol[4]
  while MgFe > 0 and PropMol[5] > 0 :
    PropMol[3] -= PropMol[3]/MgFe
    PropMol[4] -= PropMol[4]/MgFe
    PropMol[5] -= 1
    PropMol[0] -= 1
    MgFe -= 1
    i=i+1
  Mineraux[12]=i
  Mineraux[13]=i*(PropMol[4]/MgFe)
  Mineraux[14]=i*(PropMol[3]/MgFe)

  # formation wollastonite
  i = 0
  while PropMol[5]>0 :
    PropMol[5] -= 1
    PropMol[0] -= 1
    i+=1
  Mineraux[15] = i

  # formation Larnite
```

```
225   # prise d'information dans le cas de la 2e passe
226   # quantit  de SiO, FeO, et MgO
227   # L1 = [ SiO , FeO , MgO ]
228
229   L1 = [PropMol[0],PropMol[3],PropMol[4]]
230
231   # formation hypersth ne Mg
232   i=0
233   while PropMol[4] > 0 :
234     i+=1
235     PropMol[4] -= 1
236     PropMol[0] -= 1
237   Mineraux[17] = i
238
239   # formation hypersth ne Fe
240   i=0
241   while PropMol[3] > 0 :
242     i+=1
243     PropMol[3] -= 1
244     PropMol[0] -= 1
245   Mineraux[18] = i
246
247   # formation silice
248   if PropMol[0] >= 0 :
249     Mineraux[21] = PropMol[0]
250   else :
251
252     #1ere passe
253     #x+y= SiO2 dispo
254     #x+2y=MnO_FeO_3+MgO_1
255     MgFe = L1[1] + L1[2]
256     y = MgFe - L1[0]
257     x = MgFe - 2*y
258
259
260     if x>=0 and y>0 :
261
262       Mineraux[17] = x * (L1[1]/(L1[2]+L1[1]))
263       Mineraux[18] = x * (L1[2]/(L1[2]+L1[1]))
264       Mineraux[19] = y * (L1[1]/(L1[2]+L1[1]))
265       Mineraux[20] = y * (L1[2]/(L1[2]+L1[1]))
266
267     else :
268       #deficit de silice toujours pr sent -> formation d'olivine uniquement
269       Mineraux[17] = 0
270       Mineraux[18] = 0
271       Mineraux[19] = y * (PropMol[3]/(PropMol[3]+PropMol[4]))
272       Mineraux[20] = y * (PropMol[4]/(PropMol[3]+PropMol[4]))
273       deficitsilice = L1[0] - y
274
275   #deficit de silice persistant apres la formation de l'Olivine =  formation de
        nephiline
276       if deficitsilice < 0 :
277           deficit=abs(deficitsilice)
278           y = deficit / 4
279           x = L2[1]+deficit/6-1/3*y
280           if x>=0 and y>=0 :
281             Mineraux[5] = x
282             Mineraux[6] = y
```

```
283
284             else :
285
286               if x<0 or y<0 :
287
288                 deficitsilice = deficitsilice + Mineraux[5] * 6- 2 * y
289                 Mineraux[5] = 0
290                 Mineraux[6] = y # 0% albite d truite, 100% nephiline form e
291
292                 # deficit de silice persistant apres la formation de la nephiline =
                        formation de leucite
293
294                 deficit=abs(deficitsilice)
295                 y = deficit/2
296                 x = L3[1] + deficit/6 - 2/3 * y
297
298                 if x>=0 and y>=0 :
299                   Mineraux[2] = x
300                   Mineraux[3] = y
301
302                 else :
303
304                   if x<0 or y<0 :
305
306                     deficitsilice = deficitsilice + Mineraux[2] * 6 - 4 * y
307                     Mineraux[2] = 0
308                     Mineraux[3] = y
309
310                     # deficit de silice persistant apres la formation de la leucite
                          , derniere passe avec formation de la kalsilite
311
312                     deficit=abs(deficitsilice)
313                     y= deficit/2
314                     x= L3[1] + deficit/4 - y/2
315
316                     Mineraux[3] = x
317                     Mineraux[4] = y
318
319
320
321
322   PoidsMolMineraux =
          [336,152,556,436,316,524,284,278,102,462,232,160,116,100,132,116,172,100,132,140,20
323   PropMin = []
324   for i in range(len(PoidsMolMineraux)) :
325     PropMin.append((Mineraux[i] * PoidsMolMineraux[i]) / 1000)
326
327
328   F = PropMin[6] + PropMin[3] + PropMin[4]
329
330   A = PropMin[2]
331
332   P = PropMin[5] + PropMin[7]
333
334   Q = PropMin[21]
335
336
337
```

```
338
339     return Q,A,P,F,PropMin
340
341 def CIPower():
342     def Calcul_Norme():
343         global diagramme_est_dessine, ax, canvas, indexphoto, nom_roche, P_prim,
                PoidsOxyd
344         indexphoto = ""
345         PoidsOxyd = [
346             float(wtSiO2.get()),
347             float(wtAl2O3.get()),
348             float(wtFe2O3.get()),
349             float(wtFeO.get()),
350             float(wtMgO.get()),
351             float(wtCaO.get()),
352             float(wtNa2O.get()),
353             float(wtK2O.get()),
354             float(wtTiO2.get()),
355             float(wtP2O5.get()),
356             float(wtMnO.get())
357         ]
358
359         Q,A,P,F,PropMin = calcul_norme(PoidsOxyd)
360
361         CalcoAlcalin =   PoidsOxyd[6] + PoidsOxyd[7]
362         Neph_Leuc = [PropMin[6], PropMin[3]]
363         Str_Neph_Leuc = ["N ph line", "Leucite"]
364         print('SiO2 =', PoidsOxyd[0], '| Na2O + K2O =', CalcoAlcalin)
365         print(Q, A, P, F)
366         pourcApatite.configure(text=PropMin[0])
367         pourcIlmenite.configure(text=PropMin[1])
368         pourcOrthose.configure(text=PropMin[2])
369         pourcLeucite.configure(text=PropMin[3])
370         pourcKalsilite.configure(text=PropMin[4])
371         pourcAlbite.configure(text=PropMin[5])
372         pourcNepheline.configure(text=PropMin[6])
373         pourcAnorthite.configure(text=PropMin[7])
374         pourcCorindon.configure(text=PropMin[8])
375         pourcAegyrine.configure(text=PropMin[9])
376         pourcMagnetite.configure(text=PropMin[10])
377         pourcHematite.configure(text=PropMin[11])
378         pourcDiopside_Wo.configure(text=PropMin[12])
379         pourcDiopside_CEn.configure(text=PropMin[13])
380         pourcDiopside_CFs.configure(text=PropMin[14])
381         pourcWollastonite.configure(text=PropMin[15])
382         pourcLarnite.configure(text=PropMin[16])
383         pourcHypersthene.configure(text=PropMin[17]+PropMin[18])
384         pourcOlivine_Fo.configure(text=PropMin[19])
385         pourcOlivine_Fa.configure(text=PropMin[20])
386         pourcQuartz.configure(text=PropMin[21])
387         pourcTotal_valeur = 0
388         for i in range(len(PropMin)):
389             pourcTotal_valeur = pourcTotal_valeur + PropMin[i]
390         pourcTotal.configure(text=pourcTotal_valeur)
391         if roche_volca == True :
392             #CONDITION DES ROCHES VOLCANIQUES
393           P_prim = 100 * P / (A + P)
394           Petit_q = (100*Q)/(Q-(A + P))
395           ##########################################################$ Quartz Dominant (
```

```
            On rajoute le 0.00000000000001 pour les cas o   on a 0 F et 0 Q)
396         if Q+.00001 > F :
397           if Q < 90 and Q > 60 :
398             nom_roche.configure(text="Quartzolite")
399             indexphoto = 'Quartzolite'
400           ##########################################################$
401           if Q <= 60 and Q > 20:
402             if P_prim <= 10 and P_prim >= 0:
403                 nom_roche.configure(text="Rhyolite alcaline")
404                 indexphoto = 'Rhyolite'
405             elif P_prim <= 65 and P_prim > 10:
406                 nom_roche.configure(text="Rhyolite")
407                 indexphoto = 'Rhyolite'
408             elif P_prim <= 90 and P_prim > 65:
409                 nom_roche.configure(text="Dacite")
410                 indexphoto = 'Dacite'
411             elif P_prim <= 100 and P_prim > 90:
412                 nom_roche.configure(text="Dacite")
413                 indexphoto = 'Dacite'
414           ##########################################################
415           elif Q <= 20 and Q > 5:
416             if P_prim <= 10 and P_prim >= 0:
417                 nom_roche.configure(text="Trachyte alcaline et   quartz")
418                 indexphoto = 'trachyte'
419             elif P_prim <= 35 and P_prim >=10:
420                 nom_roche.configure(text="Trachyte    quartz")
421                 indexphoto = 'trachyte'
422             elif P_prim <= 65 and P_prim > 35:
423                 nom_roche.configure(text="Latite    quartz")
424                 indexphoto = 'latite'
425             elif P_prim <= 90 and P_prim > 65:
426                 if PoidsOxyd[0] <= 52 : #Normalement c'est 52 mais on fait
                        52*1000/60
427                     nom_roche.configure(text="Trachybasalte")
428                     indexphoto = 'TrachyBasalte'
429                 elif PoidsOxyd[0] > 52 :
430                     nom_roche.configure(text="Trachyand site")
431                     indexphoto = 'TrachyAndesite'
432             elif P_prim <= 100 and P_prim > 90:
433                 if PoidsOxyd[0] <= 52 : #Normalement c'est 52 mais on fait
                        52*1000/60
434                     nom_roche.configure(text="Basalte")
435                     indexphoto = 'Basalte'
436                 elif PoidsOxyd[0] > 52 :
437                     nom_roche.configure(text="And site")
438                     indexphoto = 'Andesite'
439           ##########################################################
440           elif Q <= 5 and Q+1 >= 0:
441             if P_prim <= 10 and P_prim >= 0:
442                 nom_roche.configure(text="Trachyte alcaline")
443                 indexphoto = 'trachyte'
444             elif P_prim <= 35 and P_prim > 10:
445                 if Petit_q > .2 :
446                   nom_roche.configure(text="Trachydacite")
447                   indexphoto = 'trachyte'
448                 elif Petit_q < .2 :
449                   nom_roche.configure(text="Trachyte")
450                   indexphoto = 'trachyte'
451             elif P_prim <= 65 and P_prim > 35:
```

```
452                 nom_roche.configure(text="Latite")
453                 indexphoto = 'latite'
454             elif P_prim <= 90 and P_prim > 65:
455                 if PoidsOxyd[0] <= 52 :
456                     if (PoidsOxyd[6]) - 2 >= (PoidsOxyd[7]):
457                         nom_roche.configure(text="Hawaiite")
458                         print('hawaiite')
459                         indexphoto = 'Hawaiite'
460                     if (PoidsOxyd[6]) - 2 < (PoidsOxyd[7]):
461                         nom_roche.configure(text="Trachybasalte␣potassique")
462                         indexphoto = 'TrachyBasalte'
463                 if PoidsOxyd[0] > 52 and PoidsOxyd[0] <= 57:
464                     if (PoidsOxyd[6]) - 2 >= (PoidsOxyd[7]):
465                         nom_roche.configure(text="Mugearite")
466                         indexphoto = 'Mugearite'
467                     if (PoidsOxyd[6]) - 2 < (PoidsOxyd[7]):
468                         nom_roche.configure(text="Shoshonite")
469                         indexphoto = 'Shoshonite'
470                 if PoidsOxyd[0] > 57 :
471                     if (PoidsOxyd[6]) - 2 >= (PoidsOxyd[7]):
472                         nom_roche.configure(text="Benmoreite")
473                         indexphoto = 'Benmoreite'
474                     if (PoidsOxyd[6]) - 2 < (PoidsOxyd[7]):
475                         nom_roche.configure(text="Latite")
476                         indexphoto = 'latite'
477             elif P_prim <= 100 and P_prim > 90:
478               if PoidsOxyd[0] <= 52 :
479                 nom_roche.configure(text="Basalte")
480                 indexphoto = 'Basalte'
481                 print('basalte')
482               elif PoidsOxyd[0] > 52 :
483                 nom_roche.configure(text="And site")
484                 indexphoto = 'Andesite'
485         ########## Bon faut remettre les variables pour que  a  fonctionne jsp pq
486         if diagramme_est_dessine == True:
487           ax.clear()
488           canvas.draw()
489           canvas.get_tk_widget().destroy()
490         diagramme_est_dessine = True
491         ax = TAS(add_labels=True, which_model="LeMaitreCombined", fontsize=7,
               linewidth=0.5,figsize=(6,6), fill=True, alpha = 0.2)
492         ValeursTAS = (PoidsOxyd[0], CalcoAlcalin)
493         tas = pd.DataFrame(data = ValeursTAS)
494         tas.pyroplot.scatter(ax=ax, c="r", s=5000, marker="+", alpha=1, axlabels=
               False)
495         #Ajout de la l gende d'apr s LeMaitre(2004)
496         plt.plot([], [], ' ', label="␣F␣:␣Foidite␣\n␣Pc␣:␣Picro-Basalte␣\n␣Bs␣:␣
               Basalte␣\n␣O1␣:␣And site␣Basaltique␣\n␣O2␣:␣And site␣\n␣O3␣:␣Dacite␣
               \n␣S1␣:␣Trachybasalte␣\n␣S2␣:␣Trachy-And site␣Basaltique␣\n␣S3␣:␣
               Trachyand site␣\n␣T1␣:␣Trachyte␣\n␣T2␣:␣Trachydacite␣\n␣R␣:␣Rhyolite␣
               \n␣U1␣:␣Basanite␣-␣Tephrite␣\n␣U2␣:␣Phonotephrite␣\n␣U3␣:␣
               Tephriphonolite␣\n␣Ph␣:␣Phonolite")
497         plt.legend(loc = "upper␣right", bbox_to_anchor=(1,1),prop={'size': 6})
498         canvas = FigureCanvasTkAgg(plt.gcf(), master=root)
499         canvas.draw()
500         canvas.get_tk_widget().pack(side=tk.RIGHT,  expand=1)
501         root.update()
502       ################################################## FELDSPATHOIDES
            DOMINANTS
```

```
503         elif F > Q+.00001:
504           if F <= 10 and F >= 0 :
505             if P_prim <= 10 and P_prim >= 0:
506               nom_roche.configure(text="Trachyte␣  ␣feldpsaths␣alcalins␣  ␣"+
                      Str_Neph_Leuc[Neph_Leuc.index(max(Neph_Leuc))])
507               indexphoto = 'trachyte'
508             elif P_prim <= 35 and P_prim >= 10:
509               nom_roche.configure(text="Trachyte␣  ␣"+ Str_Neph_Leuc[Neph_Leuc.
                      index(max(Neph_Leuc))])
510               indexphoto = 'trachyte'
511             elif P_prim <= 65 and P_prim >= 35:
512               nom_roche.configure(text="Latite␣  ␣"+ Str_Neph_Leuc[Neph_Leuc.index(
                      max(Neph_Leuc))])
513               indexphoto = 'latite'
514             elif P_prim <= 100 and P_prim >= 65:
515               if PoidsOxyd[0] <= 52 :
516                 nom_roche.configure(text="Basalte")
517                 indexphoto = 'Basalte'
518               elif PoidsOxyd[0] > 52 :
519                 nom_roche.configure(text="And site")
520                 indexphoto = 'Andesite'
521         ############################################################ * Correspond
              au fait que la roche d pends du color index
522           if F <= 60 and F > 10:
523             pourc_olivine = PropMin[21] + PropMin[20]
524             if P_prim <= 10 and P_prim >= 0:
525               nom_roche.configure(text="Phonolite")
526               indexphoto = 'Phonolite'
527             elif P_prim <= 50 and P_prim >= 10:
528               nom_roche.configure(text="Tephriphonolite")
529               indexphoto = 'Tephriphonolite'
530             elif P_prim <= 90 and P_prim >= 50:
531                 if pourc_olivine <= 10 :
532                   nom_roche.configure(text="Phonotephrite")
533                   indexphoto = 'Tephriphonolite'
534                 if pourc_olivine > 10 :
535                   nom_roche.configure(text="Phonobasanite")
536                   indexphoto = 'Phonolite'
537             elif P_prim <= 100 and P_prim >= 90:
538                 if pourc_olivine <= 10 :
539                   nom_roche.configure(text="Tephrite")
540                   indexphoto = 'Tephrite'
541                 elif pourc_olivine > 10 :
542                   nom_roche.configure(text="Basanite")
543                   indexphoto = 'basanite'
544           ##########################################################
545           elif F <= 90 and F > 60:
546             Str_Neph_Leuc = ["N ph lite", "Leucitite"]
547             if P_prim <= 50 and P_prim >= 0:
548               nom_roche.configure(text=Str_Neph_Leuc[Neph_Leuc.index(max(Neph_Leuc)
                      )]+ "Phonolitique")
549               indexphoto = 'Phonolite'
550             elif P_prim <= 50 and P_prim >= 10:
551               nom_roche.configure(text=Str_Neph_Leuc[Neph_Leuc.index(max(Neph_Leuc)
                      )]+ "Tephritique")
552               indexphoto = 'Tephrite'
553           ##########################################################
554           elif F <= 100 and F > 90:
555             nom_roche.configure(text="Fo dite")
```

```
556             indexphoto = 'Fo dite'
557           ########################################################
558           if diagramme_est_dessine == True:
559             ax.clear()
560             canvas.draw()
561             canvas.get_tk_widget().destroy()
562           diagramme_est_dessine = True
563           ax = TAS(add_labels=True, which_model="LeMaitreCombined", fontsize=7,
                  linewidth=0.5,figsize=(6,6), fill=True, alpha = 0.2)
564           ValeursTAS = (PoidsOxyd[0], CalcoAlcalin)
565           tas = pd.DataFrame(data = ValeursTAS)
566           tas.pyroplot.scatter(ax=ax, c="r", s=5000, marker="+", alpha=1, axlabels=
                  False)
567           #Ajout de la l gende d'apr s LeMaitre(2004)
568           plt.plot([], [], ' ', label="␣F␣:␣Foidite␣\n␣Pc␣:␣Picro-Basalte␣\n␣Bs␣:␣
                  Basalte␣\n␣O1␣:␣And site␣Basaltique␣\n␣O2␣:␣And site␣\n␣O3␣:␣Dacite␣
                  \n␣S1␣:␣Trachybasalte␣\n␣S2␣:␣Trachy-And site␣Basaltique␣\n␣S3␣:␣
                  Trachyand site␣\n␣T1␣:␣Trachyte␣\n␣T2␣:␣Trachydacite␣\n␣R␣:␣Rhyolite␣
                  \n␣U1␣:␣Basanite␣-␣Tephrite␣\n␣U2␣:␣Phonotephrite␣\n␣U3␣:␣
                  Tephriphonolite␣\n␣Ph␣:␣Phonolite")
569           plt.legend(loc = "upper␣right", bbox_to_anchor=(1,1),prop={'size': 6})
570           canvas = FigureCanvasTkAgg(plt.gcf(), master=root)
571           canvas.draw()
572           canvas.get_tk_widget().pack(side=tk.RIGHT, fill=tk.BOTH, expand=0)
573           root.update()
574         #GENERER LE DIAGRAMME DE STREKEISEN (Roches plutoniques)
575       elif roche_pluto == True :
576         #CONDITION DES ROCHES PLUTONIQUES
577         P_prim = 100 * P / (A + P)
578         if Q+0.00001 > F :
579         ########################################################## Quartz Dominant (
                On rajoute le 0.00000000000001 pour les cas o  on a 0 F et 0 Q)
580           if Q <= 60 and Q > 20:
581             if P_prim <= 10 and P_prim >= 0:
582               nom_roche.configure(text="Granite␣  ␣feldpsaths␣alcalins")
583               indexphoto = 'Granite_a_fds_alcalin'
584             elif P_prim <= 65 and P_prim >= 10:
585               nom_roche.configure(text="Granite")
586               indexphoto = 'Granite_a_fds_alcalin'
587             elif P_prim <= 90 and P_prim >= 65:
588               nom_roche.configure(text="Granidiorite")
589               indexphoto = 'Granidiorite'
590             elif P_prim <= 100 and P_prim >= 90:
591               nom_roche.configure(text="Tonalite")
592               indexphoto = 'Tonalite'
593           ########################################################
594           elif Q <= 20 and Q > 5:
595             if P_prim <= 10 and P_prim >= 0:
596               nom_roche.configure(text="Sy nite␣  ␣feldpsaths␣alcalins␣et␣  ␣
                      quartz")
597               indexphoto = 'Syenite'
598             elif P_prim <= 35 and P_prim >= 10:
599               nom_roche.configure(text="Sy nite␣  ␣quartz")
600               indexphoto = 'Syenite'
601             elif P_prim <= 65 and P_prim >= 35:
602               nom_roche.configure(text="Monzonite␣  ␣quartz")
603               indexphoto = 'Monzonite'
604             elif P_prim <= 90 and P_prim >= 65:
605                 if PropMin[7] > 50 :
```

```
606                 nom_roche.configure(text="Monzodiorite␣  ␣quartz")
607                 indexphoto = 'Monzodiorite'
608               elif PropMin[7] <= 50 :
609                 nom_roche.configure(text="Monzogabbro␣  ␣quartz")
610                 indexphoto = 'Monzogabbro'
611           elif P_prim <= 100 and P_prim >= 90:
612               if PropMin[7] <= 50 :
613                 nom_roche.configure(text="Diorite␣  ␣quartz")
614                 indexphoto = 'Diorite'
615               elif PropMin[7] > 50 :
616                 nom_roche.configure(text="Gabbro␣  ␣quartz")
617                 indexphoto = 'gabbro'
618         ##########################################################
619           elif Q <= 5 and Q >= 0 :
620             if P_prim <= 10 and P_prim >= 0:
621               nom_roche.configure(text="Sy nite␣  ␣feldpsaths␣alcalins")
622               indexphoto = 'Syenite'
623             elif P_prim <= 35 and P_prim >= 10:
624               nom_roche.configure(text="Sy nite")
625               indexphoto = 'Syenite'
626             elif P_prim <= 65 and P_prim >= 35:
627               nom_roche.configure(text="Monzonite")
628               indexphoto = 'Monzonite'
629             elif P_prim <= 90 and P_prim >= 65:
630               if PropMin[7] > 50 :
631                 nom_roche.configure(text="Monzodiorite")
632                 indexphoto = 'Monzodiorite'
633               elif PropMin[7] <= 50 :
634                 nom_roche.configure(text="Monzogabbro")
635                 indexphoto = 'Monzogabbro'
636             elif P_prim <= 100 and P_prim >= 90:
637               if PropMin[7] > 50 :
638                 nom_roche.configure(text="Diorite")
639                 indexphoto = 'Diorite'
640               elif PropMin[7] <= 50 :
641                 nom_roche.configure(text="Gabbro")
642                 indexphoto = 'gabbro'
643           if diagramme_est_dessine == True:
644             ax.clear()
645             canvas.draw()
646             canvas.get_tk_widget().destroy()
647           diagramme_est_dessine == True
648           ax = QAP(fontsize=5,linewidth=0.5,figsize=(6,6))
649           Roche = (Q, A, P)
650           df = pd.DataFrame(data=Roche)
651           df.pyroplot.scatter(ax=ax, marker = '+',c='r', s = 5000, axlabels=False)
652           canvas = FigureCanvasTkAgg(plt.gcf(), master=root)
653           canvas.draw()
654           canvas.get_tk_widget().pack(side=tk.LEFT, fill=tk.BOTH, expand=0)
655           root.update()
656         elif F > Q+.00001 :
657         ########################################################## Feldspatho des
              dominant
658           if F <= 10 and Q >= 0 :
659             if P_prim <= 10 and P_prim >= 0:
660               nom_roche.configure(text="Sy nite␣  ␣feldpsaths␣alcalins␣  ␣"+
                    Str_Neph_Leuc[Neph_Leuc.index(max(Neph_Leuc))])
661               indexphoto = 'Syenite'
662             elif P_prim <= 35 and P_prim >= 10:
```

```
663             nom_roche.configure(text="Sy nite␣  ␣"+ Str_Neph_Leuc[Neph_Leuc.
                    index(max(Neph_Leuc))])
664             indexphoto = 'Syenite'
665           elif P_prim <= 65 and P_prim >= 35:
666             nom_roche.configure(text="Monzonite␣  ␣"+ Str_Neph_Leuc[Neph_Leuc.
                    index(max(Neph_Leuc))])
667             indexphoto = 'Monzonite'
668           elif P_prim <= 90 and P_prim >= 65:
669               if PropMin[7] <= 50 :
670                 nom_roche.configure(text="Monzodiorite␣  ␣"+ Str_Neph_Leuc[
                        Neph_Leuc.index(max(Neph_Leuc))])
671                 indexphoto = 'Monzodiorite'
672               elif PropMin[7] > 50 :
673                 nom_roche.configure(text="Monzogabbro␣  ␣"+ Str_Neph_Leuc[
                        Neph_Leuc.index(max(Neph_Leuc))])
674                 indexphoto = 'Monzogabbro'
675           elif P_prim <= 100 and P_prim >= 90:
676               if PropMin[7] <= 50 :
677                 nom_roche.configure(text="Diorite␣  ␣"+ Str_Neph_Leuc[Neph_Leuc.
                        index(max(Neph_Leuc))])
678                 indexphoto = 'Diorite'
679               elif PropMin[7] > 50 :
680                 nom_roche.configure(text="Gabbro␣  ␣"+ Str_Neph_Leuc[Neph_Leuc.
                        index(max(Neph_Leuc))])
681                 indexphoto = 'gabbro'
682       ############################################################ * Correspond
              au fait que la roche d pends du color index
683         if F <= 60 and F > 10:
684           Str_Neph_Leuc = ["N ph linitique", "Leucitique"]
685           if P_prim <= 10 and P_prim >= 0:
686             nom_roche.configure(text="*Shokinite␣-␣Malignite␣-␣Sy nite"+
                    Str_Neph_Leuc[Neph_Leuc.index(max(Neph_Leuc))])
687             indexphoto = 'Shokinite_Malignite_Syenite'
688           elif P_prim <= 50 and P_prim >= 10:
689             nom_roche.configure(text="Monzosy nite"+ Str_Neph_Leuc[Neph_Leuc.
                    index(max(Neph_Leuc))])
690             indexphoto = 'Monzosyenite'
691           elif P_prim <= 90 and P_prim >= 50:
692               if PropMin[7] <= 50 :
693                 nom_roche.configure(text="Monzodiorite"+ Str_Neph_Leuc[Neph_Leuc.
                        index(max(Neph_Leuc))])
694                 indexphoto = 'Monzodiorite'
695               if PropMin[7] > 50 :
696                 nom_roche.configure(text="Monzogabbro"+ Str_Neph_Leuc[Neph_Leuc.
                        index(max(Neph_Leuc))])
697                 indexphoto = 'Monzogabbro'
698           elif P_prim <= 100 and P_prim >= 90:
699               if PropMin[7] <= 50 :
700                 nom_roche.configure(text="Diorite"+ Str_Neph_Leuc[Neph_Leuc.index
                        (max(Neph_Leuc))])
701                 indexphoto = 'Diorite'
702               elif PropMin[7] > 50 :
703                 nom_roche.configure(text="Gabbro"+ Str_Neph_Leuc[Neph_Leuc.index(
                        max(Neph_Leuc))])
704                 indexphoto = 'gabbro'
705         ########################################################
706         elif F <= 100 and F > 60:
707           if PropMin[6] > PropMin[3] :
708             nom_roche.configure(text="*Melteigite␣-␣Ijolite␣-␣Urtite")
```

```
709             indexphoto = 'Ijolite'
710           if PropMin[3] > PropMin[6]:
711             nom_roche.configure(text="*Missourite - Fergusite - Italite")
712             indexphoto = 'Missourite'
713
714         ## On g n re les diagramme avec FDLS en haut
715         if diagramme_est_dessine == True:
716           ax.clear()
717           canvas.draw()
718           canvas.get_tk_widget().destroy()
719         diagramme_est_dessine == True
720         ax = plt.subplot(projection="ternary")
721         ligne_horizontale = list(range(2))
722         ligne_horizontale[0] = [0.1, 0.8, 0.1]
723         ligne_horizontale[1] = [0.6, 0.3, 0.1]
724         slope = [0, 1, -1]
725         for i in range(len(ligne_horizontale)):
726             ax.axline(xy1=ligne_horizontale[i], slope=slope, color='k', linewidth
                =1)
727         ligne_droite = [0, .1, .9]
728         ligne_droite_stop = [0.6, 0.05, 0.35]
729         ax.plot([ligne_droite[0], ligne_droite_stop[0]],[ligne_droite[1],
              ligne_droite_stop[1]],[ligne_droite[2], ligne_droite_stop[2]],color='k
              ', linewidth=1)
730         ligne_droite = [0, .9, .1]
731         ligne_droite_stop = [0.6, 0.35, 0.05]
732         ax.plot([ligne_droite[0], ligne_droite_stop[0]],[ligne_droite[1],
              ligne_droite_stop[1]],[ligne_droite[2], ligne_droite_stop[2]],color='k
              ', linewidth=1)
733         ligne_droite = [0.1, .45, .45]
734         ligne_droite_stop = [0.6, 0.2, 0.2]
735         ax.plot([ligne_droite[0], ligne_droite_stop[0]],[ligne_droite[1],
              ligne_droite_stop[1]],[ligne_droite[2], ligne_droite_stop[2]],color='k
              ', linewidth=1)
736         ligne_droite = [0, .35, .65]
737         ligne_droite_stop = [0.055, 0.1842 , 0.3158]
738         ax.plot([ligne_droite[0], ligne_droite_stop[0]],[ligne_droite[1],
              ligne_droite_stop[1]],[ligne_droite[2], ligne_droite_stop[2]],color='k
              ', linewidth=1)
739         ligne_droite = [0,  .65, .35]
740         ligne_droite_stop = [0.055 , 0.3158, 0.1842]
741         ax.plot([ligne_droite[0], ligne_droite_stop[0]],[ligne_droite[1],
              ligne_droite_stop[1]],[ligne_droite[2], ligne_droite_stop[2]],color='k
              ', linewidth=1)
742         pc = ax.scatter(F/100, A/100, P/100,marker = '+',c='r', s = 5000)
743         ax.set_tlabel("Feldspathoid")
744         ax.set_llabel("Alkali Feldspar")
745         ax.set_rlabel("Plagioclase")
746         canvas = FigureCanvasTkAgg(plt.gcf(), master=root)
747         canvas.draw()
748         canvas.get_tk_widget().pack(side=tk.RIGHT, fill=tk.BOTH, expand=0)
749
750         root.update()
751       print(P_prim)
752       return Q,A,P,F
753   def_roche = ""
754   def afficher_contenu_pdf(def_roche):
755       global scrollbar, label
756       def_roche = nom_roche.cget("text")
```

```
757
758         # Cr ation des balises pour les roches du fichier
759         caractfin = "_"
760         balise_debut = "DEBUT_"+def_roche+caractfin
761         balise_fin = "FIN_"+def_roche+caractfin
762
763         chemin_pdf = "Data\\fiches_roches.pdf"
764
765         try:
766             pdf_doc = fitz.open(chemin_pdf)
767
768             texte_complet = ""
769             for page_num in range(pdf_doc.page_count):
770                 page = pdf_doc[page_num]
771                 texte_complet += page.get_text("text")
772
773             # Filtrer le texte en fonction du nom de la roche et des balises de
                    d but/fin
774             texte_filtr  = filtrer_texte_par_roche(texte_complet, balise_debut,
                    balise_fin)
775             text_widget.config(state=tk.NORMAL)
776             text_widget.delete("1.0", tk.END)  # Efface le contenu actuel du Text
                    widget
777             text_widget.insert(tk.END, texte_filtr )
778             text_widget.config(state=tk.DISABLED)
779
780             # Mettre    jour le widget de d filement vertical
781             scrollbar.config(command=text_widget.yview)
782             text_widget.config(yscrollcommand=scrollbar.set)
783             label.config(text="")
784
785         except FileNotFoundError:
786             label.config(text="Erreur␣:␣Fichier␣introuvable.")
787         except Exception as e:
788             label.config(text=f"Erreur␣:␣{str(e)}")
789
790     # Fonction pour filtrer le texte entre les balises
791     def filtrer_texte_par_roche(texte_complet, balise_debut, balise_fin):
792         # Rechercher les balises de d but et de fin pour le nom de la roche
793         debut_index = texte_complet.find(balise_debut)
794         fin_index = texte_complet.find(balise_fin, debut_index + len(balise_debut))
795
796         # V rifier si les balises ont   t   trouv es
797         if debut_index != -1 and fin_index != -1:
798             texte_filtr  = texte_complet[debut_index + len(balise_debut):fin_index
                    ]
799             return texte_filtr .strip()
800         else:
801             return "Balises␣introuvables␣pour", def_roche
802     def Afficher_Photo(indexphoto):
803         photo2=PhotoImage(file=f'Data\\images\\{indexphoto}.png')
804         affichage_photo.itemconfigure(Image_Affichee, image=photo2)
805         affichage_photo.image=photo2
806     Calcul_Norme()
807     afficher_contenu_pdf(nom_roche)
808     Afficher_Photo(indexphoto)
809
810
811
```

```
812 def type_pluto(): #fonction pour que la roche affich e au final soit la roche
    plutonique
813     type2.deselect() #d coche le deuxi me choix
814     global roche_pluto, roche_volca
815     roche_pluto=True
816     roche_volca=False
817 def type_volca(): #fonction pour que la roche affich e au final soit la roche
    plutonique
818     type.deselect() #d coche le deuxi me choix
819     global roche_pluto, roche_volca
820     roche_pluto=False
821     roche_volca=True
822 #TABLEAU RESULTATS
823
824 #Titre
825 sous_titre=tk.Label(cadre, borderwidth=1, text="␣␣␣␣␣␣%␣Mineraux␣virtuels␣␣␣␣␣␣")
826 sous_titre.grid(row=17,column=1)
827 #Apatite
828 titreApatite=tk.Label(cadre, text="%␣Apatite:",width=label_width)
829 titreApatite.grid(row=18,column=1)
830 pourcApatite=tk.Label(cadre, relief="sunken", borderwidth=2, text="",width=
    label_width)
831 pourcApatite.grid(row=18,column=2)
832 #Ilmenite
833 titreIlmenite=tk.Label(cadre, text="%␣Ilmenite:",width=label_width)
834 titreIlmenite.grid(row=19,column=1)
835 pourcIlmenite=tk.Label(cadre, relief="sunken", borderwidth=2, text="",width=
    label_width)
836 pourcIlmenite.grid(row=19,column=2)
837 #Orthose
838 titreOrthose=tk.Label(cadre, text="%␣Orthose:",width=label_width)
839 titreOrthose.grid(row=20,column=1)
840 pourcOrthose=tk.Label(cadre, relief="sunken", borderwidth=2, text="",width=
    label_width)
841 pourcOrthose.grid(row=20,column=2)
842 #Leucite
843 titreLeucite=tk.Label(cadre, text="%␣Leucite:",width=label_width)
844 titreLeucite.grid(row=21,column=1)
845 pourcLeucite=tk.Label(cadre, relief="sunken", borderwidth=2, text="",width=
    label_width)
846 pourcLeucite.grid(row=21,column=2)
847 #Kalsilite
848 titreKalsilite=tk.Label(cadre, text="%␣Kalsilite:",width=label_width)
849 titreKalsilite.grid(row=22,column=1)
850 pourcKalsilite=tk.Label(cadre, relief="sunken", borderwidth=2, text="",width=
    label_width)
851 pourcKalsilite.grid(row=22,column=2)
852 #Albite
853 titreAlbite=tk.Label(cadre, text="%␣Albite:",width=label_width)
854 titreAlbite.grid(row=23,column=1)
855 pourcAlbite=tk.Label(cadre, relief="sunken", borderwidth=2, text="",width=
    label_width)
856 pourcAlbite.grid(row=23,column=2)
857 #Nepheline
858 titreNepheline=tk.Label(cadre, text="%␣Nepheline:",width=label_width)
859 titreNepheline.grid(row=24,column=1)
860 pourcNepheline=tk.Label(cadre, relief="sunken", borderwidth=2, text="",width=
    label_width)
861 pourcNepheline.grid(row=24,column=2)
```

```
862 #Anorthite
863 titreAnorthite=tk.Label(cadre, text="% Anorthite:")
864 titreAnorthite.grid(row=25,column=1)
865 pourcAnorthite=tk.Label(cadre, relief="sunken", borderwidth=2, text="",width=
        label_width)
866 pourcAnorthite.grid(row=25,column=2)
867 #Corindon
868 titreCorindon=tk.Label(cadre, text="% Corindon:")
869 titreCorindon.grid(row=26,column=1)
870 pourcCorindon=tk.Label(cadre, relief="sunken", borderwidth=2, text="",width=
        label_width)
871 pourcCorindon.grid(row=26,column=2)
872 #Aegyrine
873 titreAegyrine=tk.Label(cadre, text="% Aegyrine:",width=label_width)
874 titreAegyrine.grid(row=27,column=1)
875 pourcAegyrine=tk.Label(cadre, relief="sunken", borderwidth=2, text="",width=
        label_width)
876 pourcAegyrine.grid(row=27,column=2)
877 #Magnetite
878 titreMagnetite=tk.Label(cadre, text="% Magnetite:")
879 titreMagnetite.grid(row=28,column=1)
880 pourcMagnetite=tk.Label(cadre, relief="sunken", borderwidth=2, text="",width=
        label_width)
881 pourcMagnetite.grid(row=28,column=2)
882 #Hematite
883 titreHematite=tk.Label(cadre, text="% Hematite:")
884 titreHematite.grid(row=29,column=1)
885 pourcHematite=tk.Label(cadre, relief="sunken", borderwidth=2, text="",width=
        label_width)
886 pourcHematite.grid(row=29,column=2)
887 #Diopside_Wo
888 titreDiopside_Wo=tk.Label(cadre, text="% Diopside Wo:",width=label_width)
889 titreDiopside_Wo.grid(row=30,column=1)
890 pourcDiopside_Wo=tk.Label(cadre, relief="sunken", borderwidth=2, text="",width=
        label_width)
891 pourcDiopside_Wo.grid(row=30,column=2)
892 #Diopside_CEn
893 titreDiopside_CEn=tk.Label(cadre, text="% Diopside CEn:",width=label_width)
894 titreDiopside_CEn.grid(row=31,column=1)
895 pourcDiopside_CEn=tk.Label(cadre, relief="sunken", borderwidth=2, text="",width=
        label_width)
896 pourcDiopside_CEn.grid(row=31,column=2)
897 #Diopside_CFs
898 titreDiopside_CFs=tk.Label(cadre, text="% Diopside CFs:",width=label_width)
899 titreDiopside_CFs.grid(row=32,column=1)
900 pourcDiopside_CFs=tk.Label(cadre, relief="sunken", borderwidth=2, text="",width=
        label_width)
901 pourcDiopside_CFs.grid(row=32,column=2)
902 #Wollastonite
903 titreWollastonite=tk.Label(cadre, text="% Wollastonite:",width=label_width)
904 titreWollastonite.grid(row=33,column=1)
905 pourcWollastonite=tk.Label(cadre, relief="sunken", borderwidth=2, text="",width=
        label_width)
906 pourcWollastonite.grid(row=33,column=2)
907 #Larnite
908 titreLarnite=tk.Label(cadre, text="% Larnite:",width=label_width)
909 titreLarnite.grid(row=34,column=1)
910 pourcLarnite=tk.Label(cadre, relief="sunken", borderwidth=2, text="",width=
        label_width)
```

```
911 pourcLarnite.grid(row=34,column=2)
912 #Hypersthene
913 titreHypersthene=tk.Label(cadre, text="% Hypersthene:",width=label_width)
914 titreHypersthene.grid(row=35,column=1)
915 pourcHypersthene=tk.Label(cadre, relief="sunken", borderwidth=2, text="",width=
        label_width)
916 pourcHypersthene.grid(row=35,column=2)
917 #Olivine_Fo
918 titreOlivine_Fo=tk.Label(cadre, text="% Olivine Fo:",width=label_width)
919 titreOlivine_Fo.grid(row=36,column=1)
920 pourcOlivine_Fo=tk.Label(cadre, relief="sunken", borderwidth=2, text="",width=
        label_width)
921 pourcOlivine_Fo.grid(row=36,column=2)
922 #Olivine_Fa
923 titreOlivine_Fa=tk.Label(cadre, text="% Olivine Fa:",width=label_width)
924 titreOlivine_Fa.grid(row=37,column=1)
925 pourcOlivine_Fa=tk.Label(cadre, relief="sunken", borderwidth=2, text="",width=
        label_width)
926 pourcOlivine_Fa.grid(row=37,column=2)
927 #Quartz
928 titreQuartz=tk.Label(cadre, text="% Quartz:",width=label_width)
929 titreQuartz.grid(row=38,column=1)
930 pourcQuartz=tk.Label(cadre, relief="sunken", borderwidth=2, text="",width=
        label_width)
931 pourcQuartz.grid(row=38,column=2)
932 #Total
933 titreTotal=tk.Label(cadre, text="% Total:",width=label_width)
934 titreTotal.grid(row=39,column=1)
935 pourcTotal=tk.Label(cadre, relief="sunken", borderwidth=2, text="",width=
        label_width)
936 pourcTotal.grid(row=39,column=2)
937 #Nom de la roche
938 nom_roche = tk.Label(cadre,text="Nom de la roche",width=label_width*2,font=("
        Arial", 18, "bold", "italic"))
939 nom_roche.grid(row=0, columnspan=3)
940
941 #LISTE DES BOUTONS ET TITRES
942 bouton_calculs = tk.Button(cadre, text="Commencer", command=CIPower, font=("Arial
        ", 13)) #bouton pour lancer les calculs
943 bouton_calculs.grid(row=1, column=3, padx=5, pady=5)  # Place le bouton    c  t
        du premier tableau
944 type = Checkbutton(cadre, text="Plutonique", command=type_pluto, font=("Arial",
        13)) #case    cocher pour le type de roche
945 type.grid(row=1, column=1, padx=5, pady=5) #positionne le bouton au dessus de "
        d marrer"
946 type2 = Checkbutton(cadre, text="Volcanique", command=type_volca, font=("Arial",
        13)) #deuxi me case    cocher pour le type de roche
947 type2.grid(row=1, column=2, padx=5, pady=5) #positionne le bouton    cot  du
        premier
948 label_titre = tk.Label(cadre_titre, text="CIPoWer", font=("Georgia", 30, "bold"))
949 label_titre.grid(row=0, column=0, columnspan=14, pady=10)
950
951 # LANCEMENT DE LA BOUCLE PRINCIPALE
952
953 ###Accessoires
954 barremenu=Menu(root)
955 Fichier=Menu(barremenu,tearoff=0)
956
957 #Fichier
```

```
958 barremenu.add_cascade(label="Fichier",menu=Fichier)
959 menuTypeRoche=Menu(barremenu, tearoff=0)
960
961 def ChargerExemple(indexroche): #ON VA ICI FAIRE DES UNITTESTS POUR VOIR SI LE
        FICHIER CHARGE N'A PAS ETE ALTERE
962     if indexroche == 1:
963       csv_file_path = 'Data\\Exemples\\Granite.csv'
964       #Von Eller 1961 : https://www.persee.fr/doc/sgeol_0080-9020_1961_mon_19_1
965     elif indexroche == 2:
966       csv_file_path = 'Data\\Exemples\\Basalte.csv'
967       #Everard 1997 : https://www.researchgate.net/publication/274712082
            _Geology_of_the_islands_of_southwestern_Bass_Strait
968     elif indexroche == 3 :
969       csv_file_path = 'Data\\Exemples\\Monzodiorite.csv'
970       #Andersen 1993 : https://api.semanticscholar.org/CorpusID:128356243
971     elif indexroche == 4 :
972       csv_file_path = 'Data\\Exemples\\Monzogabbro.csv'
973       #Amaral 2022 : https://doi.org/10.1016/j.chemer.2022.125917
974     elif indexroche == 5:
975       csv_file_path = 'Data\\Exemples\\Rhyolite.csv'
976       # : https://doi.org/10.1590/0001-37652021202012021
977     elif indexroche == 6:
978       csv_file_path = 'Data\\Exemples\\Hawaiite.csv'
979       # Pillard 1980 : https://www.persee.fr/doc/bulmi_0180-9210
            _1980_num_103_1_7379
980     donnees_chargees = pd.read_csv(csv_file_path, delimiter=",")
981     for i in range(len(ListeCasesElements)):
982       ListeCasesElements[i].delete(0,END)
983       ListeCasesElements[i].insert(0, str(donnees_chargees[Liste_des_elements[i
            ]][0]))
984
985 Fichier.add_cascade(label="Charger␣un␣exemple", menu=menuTypeRoche)
986 menuTypeRoche.add_command(label="P␣-␣Granite", command=lambda: ChargerExemple(1))
987 menuTypeRoche.add_command(label="V␣-␣Basalte", command=lambda: ChargerExemple(2))
988 menuTypeRoche.add_command(label="P␣-␣Monzodiorite", command=lambda:
        ChargerExemple(3))
989 menuTypeRoche.add_command(label="P␣-␣Monzogabbro", command=lambda: ChargerExemple
        (4))
990 menuTypeRoche.add_command(label="V␣-␣Rhyolite", command=lambda: ChargerExemple(5)
        )
991 menuTypeRoche.add_command(label="V␣-␣Hawaiite", command=lambda: ChargerExemple(6)
        )
992
993
994 def Charger():
995   csv_file_path = askopenfilename()
996   donnees_chargees = pd.read_csv(csv_file_path, delimiter=",")
997   for i in range(len(ListeCasesElements)):
998     ListeCasesElements[i].delete(0,END)
999     ListeCasesElements[i].insert(0, str(donnees_chargees[Liste_des_elements[i
          ]][0]))
1000 Fichier.add_command(label="Charger...",command=Charger)
1001
1002 def Sauvegarder():
1003   data = [("fichier␣CIPoWer␣csv(*.csv)","*.csv"),('All␣tyes(*.*)', '*.*')]
1004   listeElementswt = [wtSiO2.get(),wtAl2O3.get(), wtFe2O3.get(),wtFeO.get(),wtMgO.
        get(),wtCaO.get(), wtNa2O.get(), wtK2O.get(), wtTiO2.get(), wtP2O5.get(),
        wtMnO.get()]
1005   file = asksaveasfilename(filetypes = data, defaultextension = data)
```

```
  if file:
      with open(file, "w", newline='') as f:
          writer = csv.writer(f)
          writer.writerow(Liste_des_elements)
          writer.writerow(listeElementswt)
Fichier.add_command(label="Sauvegarder sous...",command=Sauvegarder)

Fichier.add_separator()

Fichier.add_command(label="Fermer le programme",command=root.destroy)

#Edition
Edition=Menu(barremenu,tearoff=0)
barremenu.add_cascade(label="Edition",menu=Edition)

def Reinitialiser():
  for i in range(len(ListeCasesElements)):
    ListeCasesElements[i].delete(0,END)
Edition.add_command(label="R initialiser les valeurs",command=Reinitialiser)
#Aide
Aide=Menu(barremenu,tearoff=0)
barremenu.add_cascade(label="Aide",menu=Aide)

def Manuel_Utilisateur():
  webbrowser.open('Notice_Info.pdf')
Aide.add_command(label="Ouvrir la notice utilisateur",command=Manuel_Utilisateur)
Aide.add_separator()
def Fenetre_Liens():
    # Cr ation de la fen tre Tkinter
    fen_Liens = Toplevel()
    fen_Liens.geometry("1000x600")
    fen_Liens.title("Liens des images")
    text_box = tk.Text(fen_Liens, height=12800, width=720, font=("Arial", 10), bg
       ='grey94')
    text_box.pack()
    liste_des_liens = [
        "Miracosta (2021) Every Rock Tells A Story , https://gotbooks.miracosta.
           edu/rocks/igneous%20rocks/43.html",
        "Miracosta (2021) Every Rock Tells A Story , https://gotbooks.miracosta.
           edu/rocks/igneous%20rocks/39.html",
        "Miracosta (2021) Every Rock Tells A Story , https://gotbooks.miracosta.
           edu/rocks/igneous%20rocks/36.html",
        "ViaGallica ([...]) Le trachyte , https://viagallica.com/auvergne/trachyte
           .htm",
        "Alex Strekeizen (2020) Latite , https://www.alexstrekeisen.it/english/
           vulc/latite.php",
        "G oForum (2007) Trachyand site basaltique (Le Mont-Dore / P de D)), 
           https://www.geoforum.fr/gallery/image/415-trachyand site-basaltique-
           le-mont-dore-p-de-d/",
        "G oForum (2006) Trachyand site (La Bourboule - P de D), https://www.
           geoforum.fr/gallery/image/270-trachyand site-la-bourboule-p-de-d/",
        "Virtual Microscope ([...]) Olivine Hawaiite - Isle of Rum , https://www.
           virtualmicroscope.org/content/olivine-hawaiite-isle-rum",
        "Virtual Microscope ([...]) Mugearite , https://www.virtualmicroscope.org/
           content/mugearite",
        "Comparer Roches (2024) shoshonite et shoshonite Types et Faits , https://
           rocks.comparenature.com/fr/shoshonite-et-shoshonite-types-et-faits/
           comparison-107-107-9",
        "Comparer Roches (2024) Formation de Benmoreite et Gr s , https://rocks.
```

```
            comparenature.com/fr/formation-de-benmoreite-et-gres/comparison
            -114-8-8",
1052        "B g nat␣([...])␣basalte␣ ␣olivine,https://www.begenat.com/produit/
            basalte-a-olivine-5/",
1053        "Coll ge␣Colette␣Sartrouville␣(2011)␣ L andsite ␣:␣une␣roche␣
            volcanique,␣https://clg-colette-sartrouville.ac-versailles.fr/spip.php
            ?article22",
1054        "G oForum␣(2006)␣Phonolite␣(La␣Roche␣Tulili re␣-␣P␣de␣D),␣https://www.
            geoforum.fr/gallery/image/215-phonolite-la-roche-tuili re-p-de-d/",
1055        "Mindat␣()␣Tephritic-phonolite,␣https://www.mindat.org/min-48547.html",
1056        "G oForum␣(2007)␣Basanite␣(Cantal),␣https://www.geoforum.fr/gallery/
            image/960-basanite-cantal/",
1057        "Pinterest␣(([...])␣no␣name␣provided,␣https://www.pinterest.fr/pin
            /274508539772482745/",
1058        "Mineralienatlas␣-␣Fossilienatlas␣(2006)␣foidite,␣https://www.
            mineralienatlas.de/lexikon/index.php/RockData?lang=en&rock=foidite",
1059        "Wikipedia␣(2022)␣Fichier:Granit␣(RK␣2206␣P1890210),␣https://fr.wikipedia
            .org/wiki/Fichier:Granit_%28RK_2206_P1890210%29.jpg",
1060        "Geologysciences␣(2023)␣Granodiorite,␣https://fr.geologyscience.com/
            roches/roches-ign es/granodiorite/",
1061        "Wiktionnaire␣(2023)␣tonalite,␣https://fr.wiktionary.org/wiki/tonalite",
1062        "Virtual␣Microscope␣([...])␣Syenite␣-␣Ben␣Loyal,␣https://www.
            virtualmicroscope.org/content/syenite-ben-loyal",
1063        "Alex␣Strekeisen␣(2020)␣Monzonite,␣https://www.alexstrekeisen.it/english/
            pluto/monzonite.php",
1064        "Alex␣Strekeisen␣(2020)␣Diorite,␣https://www.alexstrekeisen.it/english/
            pluto/diorite.php",
1065        "Youtube␣(2023)␣Trajet␣ d un ␣gabbro␣dans␣la␣cro te␣oc anique,␣https://
            www.youtube.com/watch?v=d8qIbCKlp-A",
1066        "Alex␣Strekeisen␣(2020)␣Monzodiorite,␣https://www.alexstrekeisen.it/
            english/pluto/monzodiorite.php",
1067        "Northern␣Geological␣Supplies␣Limited␣([...])␣Essexite␣(olivine␣
            monzogabbro),␣https://www.geologysuperstore.com/product/essexite-
            olivine-monzogabbro/",
1068        "Science␣Photo␣Library␣(2024)␣Nepheline␣monzosyenite␣igneous␣rock,␣https
            ://www.sciencephoto.com/media/169461/view/nepheline-monzosyenite-
            igneous-rock",
1069        "Sandatlas␣([...])␣Rock␣Types,␣https://www.sandatlas.org/rock-types/",
1070        "Fossiliraptor␣([...])␣Roches␣et␣ph nom nes␣ign s␣4,␣http://www.
            fossiliraptor.be/rochesetphenomenesignes4.htm"
1071    ]
1072    for link in liste_des_liens:
1073        text_box.insert(tk.END, link + "\n\n")
1074    text_box.config(state=tk.DISABLED)
1075    fen_Liens.resizable(width=False, height=False)
1076    fen_Liens.iconphoto(False, logo, logo)
1077 Aide.add_command(label="Liens␣des␣images", command=Fenetre_Liens)
1078
1079 def Fenetre_A_Propos():
1080    fen_Apropos = Toplevel()
1081    fen_Apropos.title("A␣propos␣de␣CIPoWer")
1082    canvas_logo = Canvas(fen_Apropos, width = 500, height = 400)
1083    canvas_logo.pack(expand = YES, fill = BOTH)
1084    canvas_logo.create_image(700/2, 200, image = logo)
1085    canvas_logo.logo = logo
1086    Noms_apropos = Label(fen_Apropos, text ="CIPoWer␣est␣un␣logiciel␣open-source␣
        d velopp ␣par␣l' quipe ␣CIPoWer␣d pendante␣de␣l'Ecole␣et␣Observatoire␣
        des␣Sciences␣de␣la␣Terre␣(EOST)\n␣Cr dit␣:␣GAUCHET␣No mie,␣DIROFF␣Mathis
        ,␣DESBIN␣Lou,␣HAY␣Nathan,␣HEYDEL␣Lilian,␣ZIMMERLIN␣Jules.␣\n\n\n\n␣CIPoWer
```

```
        ␣1993-1994")
1087      Noms_apropos.pack(expand = YES,side=TOP)
1088      fen_Apropos.iconphoto(False, logo, logo)
1089      fen_Apropos.resizable(width=False, height=False)
1090  Aide.add_command(label="A␣propos␣de␣CIPoWer", command=Fenetre_A_Propos)
1091
1092
1093  root.config(menu=barremenu)
1094
1095
1096  label = tk.Label(root, text="")
1097  label.pack(pady=10)
1098
1099  scrollbar = tk.Scrollbar(root, orient="vertical")
1100  text_widget = tk.Text(root,font=("Arial", 10),  wrap='word', width=60, height
        =700, yscrollcommand=scrollbar.set, state=tk.DISABLED)
1101  text_widget.pack(pady=10, padx=50, side=RIGHT)
1102
1103  class TestFonctionsCIPOWER(unittest.TestCase):
1104    #ON TEST LA FONCTION DE CHARGEMENT DES EXEMPLES - Ici, les fichiers dans le
          dossier du programme doivent avoir exactement les m mes valeurs que celles
          que l'on veut. Ceci permet donc de voir si rien n'a   t   corrompu.
1105    def test_ChargerExemples(self):
1106      valeurs_attendues = [
1107      [70.83, 14.52, 0.51, 2.5, 0.78, 0.65, 2.48, 5.56, 0.32, 0.14, 0.05],
1108      [45.65, 10.56, 1.26, 8.26, 17.87, 10.34, 0.48, 0.11, 0.26, 0.10, 0.15],
1109      [53.45, 17.9, 3.88, 3.16, 1.82, 6.2, 6.23, 2.46, 1.15, 0.47, 0.21],
1110      [45.45, 12.81, 11.78, 0, 13.52, 10.99, 1.25, 1.81, 1.48, 0.55, 0.17],
1111      [72.34, 13.25, 1.77, 1.03, 0.35, 0.89, 3.96, 4.89, 0.27, 0.15, 0.09],
1112      [48.90, 16.70, 4.60, 5.20, 5.80, 7.15, 4.80, 1.70, 2.17, 0.69, 0.17]
1113      ]
1114      for i in range(1,6):
1115         ChargerExemple(i)
1116         for k in range(len(ListeCasesElements)):
1117            self.assertEqual(float(ListeCasesElements[k].get()), valeurs_attendues[
                i-1][k])
1118    #Test case pour tester la fonction 'calcul_norme()'
1119    def test_calcul_norme(self):
1120        #Test le fonctionnement de la fonction trapz.
1121        #Liste L = valeur issue du cours de p trologie magmatique consid r e
              comme valeurs vraies des min raux pour un granite
1122        PoidsOxyd = [70.83, 14.52, 0.51, 2.5, 0.78, 0.65, 2.48, 5.56, 0.32, 0.14,
              0.05]
1123        L = [0.336, 0.608, 32.804, 0.0, 0.0, 20.96, 0.0, 2.502, 3.468, 0.0, 0.696,
              0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 2.0, 3.828, 0.0, 0.0, 31.14]
1124        Q,A,P,F,PropMin_calcul = calcul_norme(PoidsOxyd)
1125        self.assertAlmostEqual(PropMin_calcul,L)
1126
1127  unittest.main(argv=[''], verbosity=3, exit=False)
1128
1129
1130  Reinitialiser()
1131  root.mainloop()
```