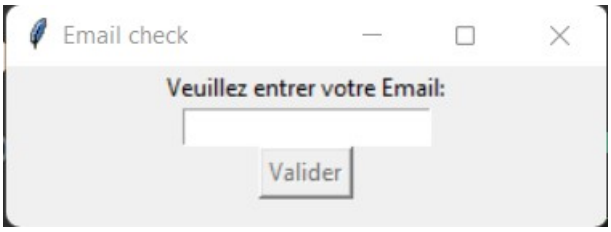
Source :

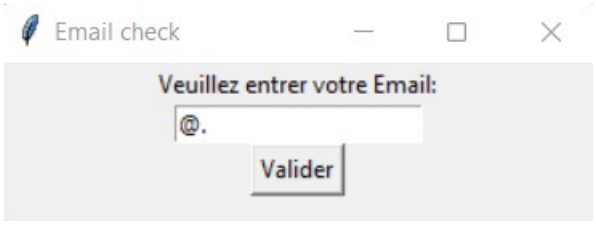
https://openclassrooms.com/forum/sujet/incrementer-variable-lors-d-une-interruption
https://waytolearnx.com/2020/07/comment-desactiver-un-bouton-tkinter-python.html

EXO 1 :

Pour faire le premier programme j'ai crée deux fonction une qui ajoute 1 et l'autre retire 1 a la variable initiale de valeur 0 j'ai attribuer au bouton une « commande » (fonction)
Les fonctions récupère grâce à un .get la valeur de la variable et lui ajoute +1 ou lui retire -1 après une nouvelle variable « v » (str) reçoit le texte (« valeur » + la nouvelle valeur de la variable), ensuite le texte du label se mets à jour et le nouveaux texte devient la variable v

EXO 2 :





Pour faire ce programme j'ai d'abord crée plusieurs choses :
-Un label avec un texte : « Veuillez entrer votre Email »
-Un Entry sur le que j'ai mis un .focus_set() pour que quand on lance le programme on soit directement en écriture sans avoir à cliquer sur le champ
-Un bouton de texte « Valider » , d'état state= DISABLED qui le désactive et qui appelle la fonction « Valider » quand on appuie sur le bouton

```
#//// fonction validation du mail
def Valider():
    print( "Votre Email :" + mail.get())
    gui.destroy()
```

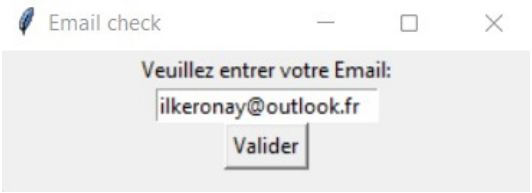
```
# //// Création du boutton et du label state= disabled pour désactiver un boutton
varmail = StringVar()
label = Label(text="Veuillez entrer votre Email:")
label.pack()
mail = Entry(textvariable=varmail )
mail.focus_set()         #//// Permet de lancer une saissie directe
mail.pack()

button = Button(text="Valider", command=Valider,state=DISABLED)
button.pack()
```

Ensuite j'ai crée un bind <Key> qui lance la fonction check quand on appuie sur une touche du clavier

La fonction check :

Elle est appeler à chaque fois qu'on clique sur une touche du clavier, elle est composer de 3 flag.
flag_a = 0
flag_point=0
flag_espace = 1

la valeur de flag_espace et initialement en 1 car par défaut il n'y a pas d'espace
La variable texte récupère la valeur du mail (mail est la variable Entry donc ce qu'on saisie)
On utilise de if pour vérifier si ce qu'on cherche est présent, si le @ et le point et présent alors on donne pour valeur 1 au flag et si tout les flag sont à 1 alors le boutton récupère sont état dit « NORMAL »
Met si l'un des flags a pour valeur 0 le bouton reste désactiver state=DISABLED

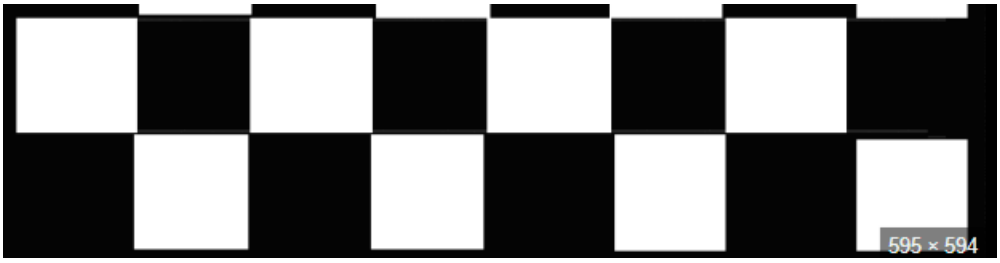



```
/Prog événementielles/TP 1/exo2.py"
Votre Email :ilkeronay@outlook.fr
```

```
def check(key):
    # ////////////// MES FLAGS DE VERIFICATIONS POUR LE @ ;" " ; .
    flag_a= 0
    flag_point=0
    flag_espace=1
    texte = mail.get()
    if "@" in texte :
        flag_a = 1
    if "." in texte:
        flag_point = 1
    if " " in texte :
        flag_espace = 0
    if flag_a == 1 and flag_point == 1 and flag_espace == 1 :
        button.configure(state=NORMAL)
    if flag_a == 0 or flag_point ==0 or flag_espace == 0 :
        button.configure(state=DISABLED)


gui.bind("<Key>", check)
gui.mainloop()
```

EXO 3 :

Un damier est fait de 2 motifs répéter 4 fois :



Donc j'ai crée une fonction qui créent la première ligne est l'autre la deuxième puis j'ai fait une boucle for pour crée grâce au fonction tout le damier on se base sur les nombres paires et impaire pour les canvas noir ou blanc

```
###############################################################################################
#               FONCTION POUR AFFICHER LE DAMIER EN CREANT LES CANVAS                         #
###############################################################################################

def damier_un():
    largeur,hauteur= 75,75
    num=[1,2,3,4,5,6,7,8]
    numb=[2,4,6,8]
    for i in range(len(num)):
        for j in range(len(numb)):
            if (i % 2) == 0 :
                color = "black"
            else:
                color= "white"
            canvas= Canvas(gui, bg=[color],width=largeur,height=hauteur,border=False).grid(row=numb[j],column=num[i])

def damier_deux():
    largeur,hauteur= 75,75
    num=[1,2,3,4,5,6,7,8]
    numb=[1,3,5,7]
    for i in range(len(num)):
        for j in range(len(numb)):
            if (i % 2) == 0 :
                color = "white"
            else:
                color= "black"
            canvas= Canvas(gui, bg=[color],width=largeur,height=hauteur).grid(row=numb[j],column=num[i])
```

```
###############################################################################################
#                              BOUCLE POUR AFFICHER DE LA  GRILLE                             #
###############################################################################################


    #//// Affichage
for i in range(7):
    if (i % 2) == 0:
        damier_deux()
    else:
        damier_un()
```

J’ai utiliser pour les positionner le .grid(row= x, column= y) , car un canvas peut être écrasser par un notre par exemple :

canvas = Canvas(gui,bg=’red’).grid(row=1,column=1)
canvas = Canvas(gui,bg=’bleu’).grid(row=1,column=1)
se sera le dernier pris en compte donc le canvas sera bleu

Pour aller plus vite dans mes essaies j’ai crée un raccourcie via bind sur la touche Echap qui ferme directement le jeu

```
#############################################################################################
#               FONCTION ET BIND POUR QUITTER LORS DE L'APPUIE SUR LA TOUCHE ECHAP           #
#############################################################################################

def destroy(event):
    gui.destroy()

gui.bind("<Escape>",destroy)                #//// touche échap pour quitter

#############################################################################################
```

La fonction pion :
Après la fonction principal prend le plus de place car après un clique gauche , il permet de capter les coordonnées (en grid c’est-à-dire row et column), il permet de capter si un pion a déjà était jouer a cette case pour éviter de le prendre en compte 2 fois , il permet que le nbr de pion max soit 8, permet de définir si derrière la dame poser la couleur et noir ou blanche , et il permet de verifier si un autre pion se trouve dans les diagonales, à la verticale ou à l’horizontal de celui qu’on veut poser pour définir si c’est une dame poser correctement ou une erreur et donc d’afficher la case en rouge (à 3 case rouge on perd, c’est comme si on avait 3 vie et qu’à chaque erreur on en perd une)

Pour avoir les coordonnées :

```
def pion(event):
    global pion_encours,nbrdepion,erreur,pion
    x = event.x_root - gui.winfo_rootx()            #permet d'avoir les coordonnées
    y = event.y_root - gui.winfo_rooty()
    z= gui.grid_location(y,x)
    x_new,y_new= str(z),str(z)
    x_new=int(x_new[1:2])                           # int row
    y_new=int(y_new[4:5])                           #int column
    #print("x_new =", x_new,"       y_new = ", y_new)
```

Pour déterminer la couleur de la case derrière l'image de la dame :

```
for i in range(1,9):                                #permet de savoir si la case est blanche si oui derrière la dame la couleur sera bl
    if x_new % 2 == 0:
        if x_new == i and y_new == 2 or x_new == i and y_new == 4 or x_new == i and y_new == 6 or x_new == i and y_new == 8:
            tks=Canvas(gui,bg="white",width=75,height=75)
for i in range(1,9):
    if x_new % 2 == 1:
        if x_new == i and y_new == 1 or x_new == i and y_new == 3 or x_new == i and y_new == 5 or x_new == i and y_new == 7:
            tks=Canvas(gui,bg="white",width=75,height=75)
```

Permet de savoir si on a déjà jouer sur cette case (mémorisation des cases jouer):

```
z=str(z)
z=str(z[1:5])
pion=len(pion_encours)
if pion < 8 :                                       #PERMET DE SAVOIR SI LE PION ( dame ) a déjà était jouer a ces coordonnées
    if z in pion_encours:
        return
    else:                                           # si il n'a pas été jouer alors on le place
        tks.grid(row=x_new,column=y_new)
        nbrdepion+=1
        tks.create_image(10,10,anchor=NW,image=new_image)
        pion_encours.append(z)
```

Permet de savoir si notre coup est juste s'il ne l'est pas on dit qu'on a fait une erreur donc la variable erreur récupère un +1 à 3 erreur le jeu s'arrête :

```
############################
# Diagonale haut droite    #
############################
for check in range(pion):
    for value in range(1,9):
        flag =0
        check_upx = pion_encours[check]
        check_upx = check_upx[:1]
        check_upx = int(check_upx) - value
        check_upy = pion_encours[check]
        check_upy = check_upy[3:4]
        check_upy = int(check_upy) + value
        check_up= str(check_upx) + espace + str(check_upy)
        if  pion_encours[check] == check_up :
            flag=-1
        if check_up in pion_encours :
            flag = flag +1
        if flag == 1:
            tks=Canvas(gui,bg="red",width=75,height=75)
            tks.grid(row=x_new,column=y_new)
            erreur +=1
            pion_encours.pop()
```

```
############################
# Diagonale bas gauche     #
############################
for check in range(pion):
    for value in range(1,9):
        flag =0
        check_upx = pion_encours[check]
        check_upx = check_upx[:1]
        check_upx = int(check_upx) + value
        check_upy = pion_encours[check]
        check_upy = check_upy[3:4]
        check_upy = int(check_upy) - value
        check_up= str(check_upx) + espace + str(check_upy)
        if  pion_encours[check] == check_up :
            flag=-1
        if check_up in pion_encours :
            flag = flag +1
        if flag == 1:
            tks=Canvas(gui,bg="red",width=75,height=75)
            tks.grid(row=x_new,column=y_new)
            erreur+=1
            pion_encours.pop()
```

```
############################
# Diagonale haut gauche    #
############################

for check in range(pion):
    for value in range(1,9):
        flag =0
        check_upx = pion_encours[check]
        check_upx = check_upx[:1]
        check_upx = int(check_upx) - value
        check_upy = pion_encours[check]
        check_upy = check_upy[3:4]
        check_upy = int(check_upy) - value
        check_up= str(check_upx) + espace + str(check_upy)
        if  pion_encours[check] == check_up :
            flag=-1
        if check_up in pion_encours :
            flag = flag +1
        if flag == 1:
            tks=Canvas(gui,bg="red",width=75,height=75)
            tks.grid(row=x_new,column=y_new)
            erreur +=1
            pion_encours.pop()
```

```
############################
# Diagonale bas droite     #
############################
for check in range(pion):
    for value in range(1,9):
        flag =0
        check_upx = pion_encours[check]
        check_upx = check_upx[:1]
        check_upx = int(check_upx) + value
        check_upy = pion_encours[check]
        check_upy = check_upy[3:4]
        check_upy = int(check_upy) + value
        check_up= str(check_upx) + espace + str(check_upy)
        if  pion_encours[check] == check_up :
            flag=-1
        if check_up in pion_encours :
            flag = flag +1
        if flag == 1:
            tks=Canvas(gui,bg="red",width=75,height=75)
            tks.grid(row=x_new,column=y_new)
            erreur +=1
            pion_encours.pop()
```

```
############################
# DE DROITE A GAUCHE       #
############################
for check in range(pion):
    for column in range(1,9):
        flag =0
        check_up = pion_encours[check]
        check_up = check_up[:1]
        check_up = check_up + espace + str(column)
        if  pion_encours[check] == check_up :
            flag=-1
        if check_up in pion_encours :
            flag = flag +1
        if flag == 1:
            tks=Canvas(gui,bg="red",width=75,height=75)
            tks.grid(row=x_new,column=y_new)
            erreur +=1
            pion_encours.pop()
```

```
############################
# DE BAS EN HAUT           #
############################
for check in range(pion):
    for row in range(1,9):
        flag =0
        check_up = pion_encours[check]
        check_up = check_up[3:4]
        check_up = str(row) + espace + check_up
        if  pion_encours[check] == check_up :
            flag=-1
        if check_up in pion_encours :
            flag = flag +1
        if flag == 1:
            tks=Canvas(gui,bg="red",width=75,height=75)
            tks.grid(row=x_new,column=y_new)
            erreur = erreur +1
            pion_encours.pop()
```

Permet de savoir si on a gagné ou perdu :

```
if pion == 8 :
    victoire()
if erreur == 3 :
    perdu()
```

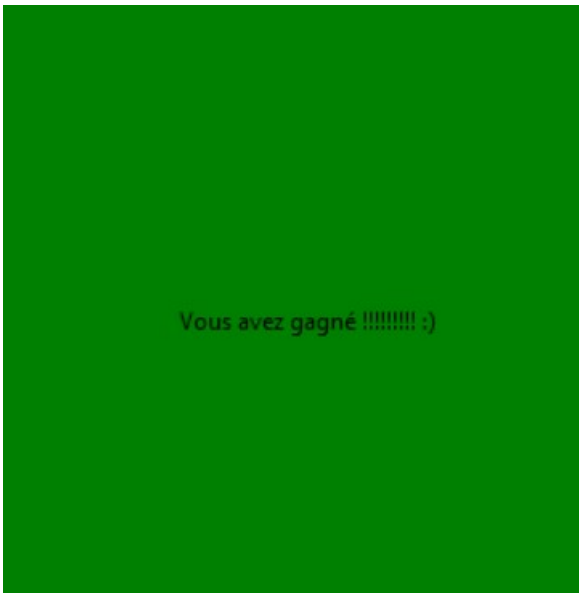
```
###############################################################################################
#         FONCTION QUI ANNONCE LA DEFAITE (vous le verez pas si vous êtes fort au jeu)        #
###############################################################################################

def perdu():
    gui.destroy()
    gui_loose = Tk()
    label= Label(gui_loose,text="Vous avez perdu ! :(",bg="red",width=50,height=25).pack()

###############################################################################################


###############################################################################################
#         FONCTION QUI ANNONCE LA VICTOIRE (POUR LES PGM)                                     #
###############################################################################################

def victoire():
    gui.destroy()
    gui_victoire = Tk()
    label= Label(gui_victoire,text="Vous avez gagné !!!!!!!!! :)",bg="green",width=50,height=25).pack()

###############################################################################################
```

Les variables par défaut (certain sont utilisée dans la fonction principale):

```
# import variable global resized de l'image
from tkinter import *
from PIL import Image,ImageTk

erreur=0
espace = ", "
gui = Tk()
gui.title("Eight Queen")
pion_encours=[]
nbrdepion=0
img=(Image.open("TP 1/dame.png"))
resized_image= img.resize((60,65))
new_image= ImageTk.PhotoImage(resized_image)
```

Quand on perd :

Quand on gagne :



Problème de mon programme que j'ai constater :

Quand nous avons fini de placer les 8 dames il faut recliquer sur n'import quel case pour savoir si on a gagné ou pas car la vérification se fait dans une fonction