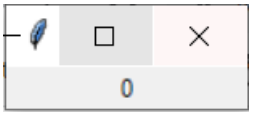
Exercices 1 :
Création du fonction titre :

```
#Exercice 1
print("----------------------------------------")
def titre ():
    zone_texte = tkinter.Label (text = "0")
    zone_texte.pack()
```

On créeait une zone de texte avec la valeur que l'on souhaite (dans notre cas 0) et on pack cette zone pour qu'elle apparaisse.
Résultat :



Pour la seconde partie j'ai décidé de ne pas utiliser le .pack mais le .grid sur chacun de mes éléments .
J'ai crée une fonction button qui me permet d'afficher mes 2 boutons .

```
def titre ():
    zone_texte = tkinter.Label (text = "0").grid(column=2,row=0)
def button ():
    b = tkinter.Button (root, text=" + ").grid(column=1,row=1)
    b1=tkinter.Button(root,text=" - ").grid(column=3,row=1)
```

On possède donc 2 boutons + et - .
Résultat :

Pour réaliser l'addition et la soustraction j'ai crée de nombreuses fonctions et changer mon code.

J'ai crée une variable val qui sera le texte modifié.
Puis je la modifie avec mes fonctions ajouter et soustraire.

```
print("----------------------------------------")
val=0
def titre():
    zone_texte = tkinter.Label (text = val).grid(column=2,row=0)
def button1 ():
    b = tkinter.Button (root, text=" + ",command=ajouter).grid(column=1,row=1)
def button2():
    b1=tkinter.Button(root,text=" - ",command=soustraire).grid(column=3,row=1)

def ajouter ():
    global val
    val=val+1
    zone_texte = tkinter.Label (text = val).grid(column=2,row=0)
def soustraire():
    global val
    val=val-1
    zone_texte = tkinter.Label (text = val).grid(column=2,row=0)
```

Dans le programme j'ai crée 5 fonctions :
- Une fonction titre qui affiche le titre
-Une fonction button1 qui sera le boutton ajouter
-Une fonction button 2 qui sera le bouton soustraire
-Une fonction ajouter qui change la valeur du titre en lui ajoutant 1
- Une fonction soustraire qui change la valeur du titre en lui enlevant 1
Puis je lance les fonctions titre et bouton qui apparaitront sur la page

```
titre()
button2()
button1()
root.mainloop ()
```

Dans mon programme initial je n'avais pas fait de fonction et je n'avais pas compris l'utilité du .pack et je n'arrivais pas à afficher dans la fenêtre le titre et les boutons donc j'ai changé et utilisé le .grid(col,row)

Exercice 2:

```
root = tkinter.Tk ()
val= tkinter.StringVar()
val.trace("w", lambda name, index, mode, val=val: verif())
user = tkinter.Label(root, text = "Veuillez entrer votre Email:",bg='yellow').grid(column=0,row=0)
user_Entry = tkinter.Entry(root,bg="orange",textvariable=val).grid(column=0,row=2)
user_button=tkinter.Button(root,text="Valider",command=valid,state=tkinter.DISABLED,bg='red')
user_button.grid(column=0,row=5)
root.mainloop ()
```

Création d'une variable en stringVar()
val.trace permet qu'à chaque fois qu'un caractère est inscrit dans la saisie alors il va éxecuter la fonction verif()
Création d'une saisie avec Entry qui permet de récupérer les valeurs du texte et de les mettre dans la variable val .
Création du bouton de validation qui renvoie à la commande qui fermera le programme , ce bouton est en DISABLED car sans les conditions de la fonction il ne peut pas être éxécuté.
La couleur de ce bouton (bg) est en rouge elle passera en verte lorsque que l'on aura vérifié toutes les conditions.
J'ai créer 2 fonctions
Une fonction qui va vérifier tant que les conditions ne sont pas valides.

```
def verif():
    saisie=val.get()
    if "@" not in saisie or   "." not in saisie or " " in saisie :
        rep =tkinter.Label().grid(column=0,row=4)
    else :
        user_button['state']=tkinter.NORMAL
        user_button['command']=root.destroy
        user_button['bg']='green'
        print("Vous allez être enregistré avec:",saisie)
```

Elle vérifie si la valeur entrée est conforme aux conditions.
On récupère la valeur de la saisie avec .get()
On vérifie que tant que il n'y a pas d'@ , de . et qu'il n'y aucun espace la saisie reste identique.
Dans le cas contraire note bouton valider va changer.
On va pouvoir cliquer dessus à l'aide du 'state' qui va passer en NORMAL(Up)

On va lui faire détruire la page et on va aussi lui faire passer sa couleur en verte.
On affiche ensuite dans le terminal la valeur qui va être enregistrée.
La seconde fonction permet de quitter simplement le programme c'est ce qui s'éxecute si le bouton Valider est en Up .(soit toutes les conditions réunies)

```
def valid():
    rep = tkinter.Label(root).grid(column=0,row=5)
    root.quit
```

Le programme comporte un problème:
Lorsque les conditions sont réunies, la boucle if est passée donc il ne va pas revenir en arrière et on peut donc supprimer les caractères et valider sans les conditions.
Exemple:

Les conditons sont réunies: un (@) , un (.) , sans espace mais si je supprime les carractères alors on pourra toujours valider. Et cela ne doit pas être possible.

Les conditions ont été passées au dessus donc on peut maintenant modifier nos caractères et valider .
Donc une personne pourrait envoyer sans les points.

Exercice 3 :

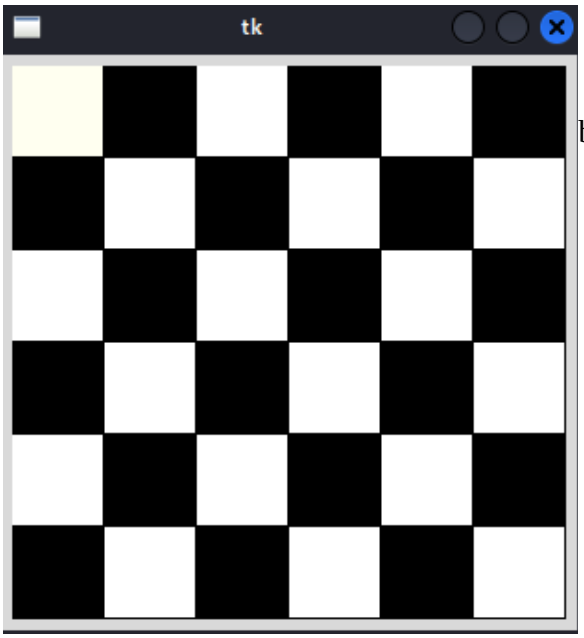
```
x1,y1,x2,y2=0,0,50,50
couleur= ()
fen=tkinter.Tk()
can=tkinter.Canvas(fen,width=400,heigh=400,bg='ivory')
Commencer()
can.pack(side=TOP,padx=5,pady=5)
fen.mainloop()
```

Nous allons créer des variables de coordonées , une varibale couleur qui sera la couleur des rectangles (50 par 50 de notre damier)
Puis choisir la taille de la fenêtre qui sera de 400 à 400 pour permettre d'avoir seulement 8 lignes et colones.400/8 =50
On va ensuite lancer la fonction du programme et afficher notre Canvas.
La fonction s'occupe de tracer les rectangles. Attention si on change la valeur de la fenêtre alors on aura moins de lignes ex : 300 = 6 lignes.
La fonction :

```
def Commencer():
    global x1,x2,y1,y2,couleur
    ite=0
    i=1
    while x1 < 400 and y1<400:
        can.create_rectangle(x1,y1,x2,y2,fill=couleur)
        i+=1
        ite+=1
        x1+=50
        x2+=50
        if ite== 8 :
            y1+=50
            y2+=50
            i+=1
            ite=0
            x1=0
            x2=50
        if i%2==0:
            couleur='black'
        else:
            couleur='white'
```

On se sert des variables de coordonées et de la couleur du carré.
Nous allons créer une variable qui va permettre d'avancer sur les colones(carré) et les lignes(carré) (ite,i))
Puis tant que notre ordonnée et notre abscice sont dans la fenêtre alors nous allons créer des rectagles de 50 par 50 .
Puis nos itérateurs vont augmenter de 1 et former d'autres carrés.
Si la valeur de i modulo 2 est pair alors c'est un carré noir et c'est pour cela que notre premier carré et donc blanc.

Résultat :



Un damier constitué de rectangle (50x50)alternant noir blanc dans une fenêtre 400x400.

Exercice 4 :
J'importe une image d'une reine trouvée sur internet.J'ai essayé de l'avoir en transparent avec la méthode Anchor mais mon image n'était pas bonne alors je l'ai laissé comme elle était.

```
fen=tkinter.Tk()
can=tkinter.Canvas(fen,width=400,heigh=400,bg='ivory')
can.pack(side=TOP,padx=5,pady=5)
photo=reine()
Commencer()
can.bind('<B1-Motion>',drag)
fen.mainloop()
```

Je vais créer une variable qui vaut la fonction reine.
Je vais me servir de la fonction .bind sur mon canvas pour y associer une fonction.
J'ai crée l'image reine pour afficher l'image et la fonction drag pour qu'elle suive l'image.
La fonction reine :

```
def reine ():

    photo=PhotoImage(file='img/reine.png')
    photo=resizeImage(photo,45,45)
    return photo
```

La fonction drag :

```
def drag(event):
    global photo
    photo = PhotoImage(file = "img/reine.png")
    photo=resizeImage(photo,45,45)
    x=event.x
    y=event.y
    #Impossible de sortir
    if x <=0 or x>=400 or y <=0 or y>=400 : #Empeche la photo de sortir
        if x<0 :
            x=0
        elif x>400 :
            x=400
        if y<0:
            y=0
        elif y>400:
            y=400
    my_img = can.create_image(x, y,  image=photo)
```

Elle permet en fonction de event.x et event?y de suivre la souris si on clique (.bind) sur la reine.
J'ai mis une petite condition pour que si après avoir cliqué sur la reine on la déplace hors du canvas alors elle reste dedans.

J'ai recontré un problème avec la taille de l'image qui n'étais pas dans la bonne dimension de ma fenêtre.J'avais 2 options soit redimentionner l'image moi même ou coder une fonction qui change l'image.
J'ai donc trouvé sur internet avec un petit code qui permet de changer la taille d'une image et je l'ai donc appliqué à ma photo

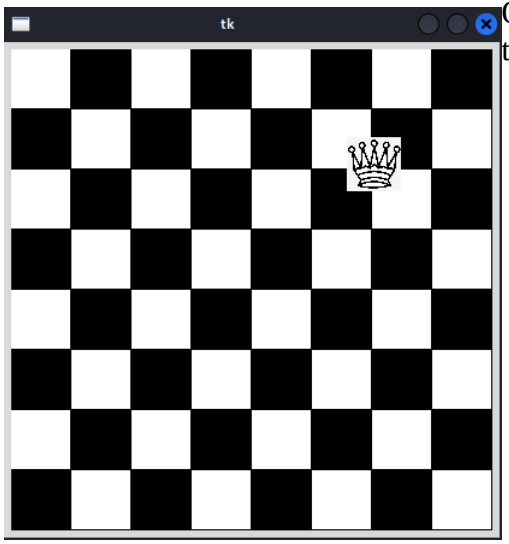
```
photo=resizeImage(photo,45,45)
```

La photo qui correspond à mon image reine.png fera donc 45x45.donc on poura remarquer la couleur de la case, elle est au millieu.J'ai choisi cette dimension car je n'ai pas reussi à la mettre en transparent.

Fonction resizeIamage (stackoverflow) je précise que je n'ai pas écrit cette fonction.

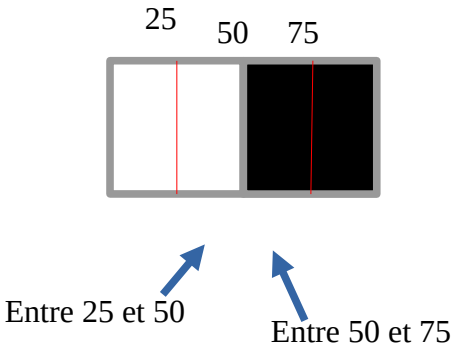
```
# Fonction pour modifier la taille de l'image
def resizeImage(img, newWidth, newHeight):
    oldWidth = img.width()
    oldHeight = img.height()
    newPhotoImage = PhotoImage(width=newWidth, height=newHeight)
    for x in range(newWidth):
        for y in range(newHeight):
            xOld = int(x*oldWidth/newWidth)
            yOld = int(y*oldHeight/newHeight)
            rgb = '#%02x%02x%02x' % img.get(xOld, yOld)
            newPhotoImage.put(rgb, (x, y))
    return newPhotoImage
```

Résultat :



On peut la déplacer en cliquant dessus et elle suivra la souris tant que l'on a le clique activé.

Exercice 5 :
Pour cette exercice j'ai amélioré ma fonction drag pour qu'elle détecte simplement la case la plus proche du bouton de ma souris et j'ai centré la photo dessus.
Pour cela j'ai crée 2 variables qui récupèrent les coordonées en string.(avec un nom différents du haut pour ne pas s'embrouiller)
Puis je peux les séparer en dizaine et centaine.
Cette manière va me permettre de tester si la valeur de la souris (dizaine) est supérieure ou inférieure à 25 ou 75 pour placer la reine.Pour expliquer je réalise un petit schéma :



Il est évident que si c'est compris entre 75 et 100 alors c'est sur la cas et pareillement pour 0 et 25 grâce à ce découpage.
Exemple dans le cas des X mais c'est identique à l'horizontal pour les Y.

Ensuite je passe les valeurs en int pour pouvoir afficher l'image avec les bonnes coordonées.
Je recontre un problème si la valeur des x est à 100 comment choisir la case ? Droit ou gauche ?
J'ai donc changé mon positionement, de -1 pour mes X et Y et donc comme ça, l'image par défaut sera à droite sauf si elle est à 400.
Dans tous les cas l'image reste dans le damier sur une case .
Le code :

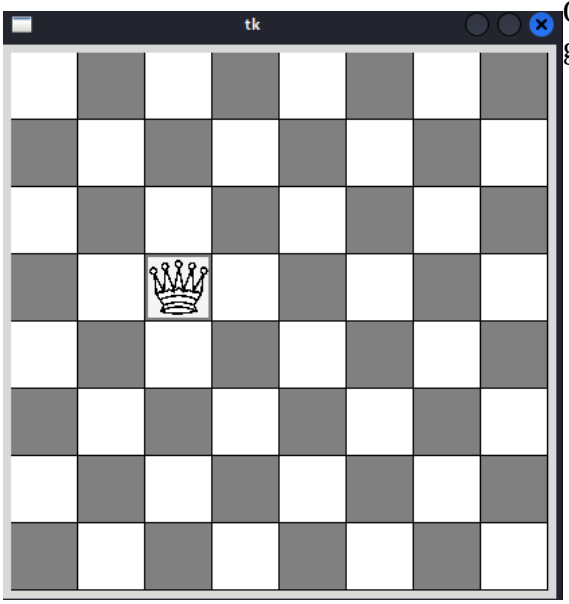
```
#Formatage des coordonnées en String pour récupérer les dizaines et les centaines
cooX = str(x)
cooY = str(y)
#Utilisation de ternaire
centX = cooX[:1] if len(cooX)==3 else "0" #Centaine X
dizX = int(cooX[1:]) if len(cooX)==3 else int(cooX) #Dizaine X
centY = cooY[:1] if len(cooY)==3 else "0" #Centaine Y
dizY = int(cooY[1:]) if len(cooY)==3 else int(cooY) #Dizaine Y

#Mise en place des Pas de la piece
if dizX < 50:
    dizX = 25
elif dizX >= 50:
    dizX = 75
if dizY < 50:
    dizY = 25
elif dizY >= 50:
    dizY = 75
```

```
    dizY = 75
#Reformatage Complet des Coordonnées String en Int
cooX = int(centX+str(dizX))
cooY = int(centY+str(dizY))
#Affichage de l'image aux bonnes Coordonnées
my_img = can.create_image(cooX, cooY,  image=photo)
```

Et pour finir ma petite modification dans le début de mon drag pour que l'image choisisse la droite en cas de chiffre rond au millieu.

```
elif x>=400 :
    x=399
if y<=0:
    y=1
elif y>=400:
    y=399
```

Résultat :



On remarque la petite zone autour de la case qui reste grise, grâce à cela on peut la voir.

Exerice 6 :
Je n'ai pas réussi cet exerice car mon .bind est associé à mon Canvas du coup je ne peux pas déplacer plusieurs reine.Il faudrait créer un bind sur un bouton qui modifiera un objet et pourra placer une reine suplémentaire jusqu'a 8 maximal.
Ensuite il faudrait créer une fonction qui perméttra de déteminer les zones d'attaques de la reine posée et vérifier si aucune reine n'est dessus.Pour cela vérifier la couleur de la case au millieu suffirait si je prends une reine coloré. Puis une fois les 8 reines posées lancer la fonction .Si alors une reine peut manger une autre alors la partie se relance et indique dans l'invite de commande que la partie est perdue .Si alors tout les reines ne peuvent pas se manger la partie est gagnée.

Lors de ce TP je me suis rendu compte de la présence de l'algorithmie dans la programmation. Pour découper les cases pour afficher la reine à certain endroit et prévoir les éventualltités du déplacement de la pièce j'ai du faire des brouillons avec des coordonées aléatoires.Tous les codes sont dans le dossier ci-joint et fonctionne comme présenté sur les captures.