

Selection

```
In [ ]: ''' Visualisation tri par sélection '''
```

```
In [ ]: import turtle
        from random import randint
        import time
        import argparse
```

```
In [ ]: # Les constantes
        LARGEUR_ECRAN = 1000
        HAUTEUR_ECRAN = 600
        MARGE = 40
        ECART = 20
        LARGEUR_CARTE = (LARGEUR_ECRAN-MARGE*2)//10 - ECART
        HAUTEUR_CARTE = HAUTEUR_ECRAN // 3 - 2 * MARGE
        VMIN = 100
        VMAX = 200
        COULEUR_BASE = "black"
        COULEUR_SPOT = "red"
        LIGNE_CARTE = MARGE + HAUTEUR_CARTE*2
        LIGNE_SPOT = LIGNE_CARTE - 20
        WAIT_BLINK = 0.3
        MOVE = 0.0005
        PARCOURS = 0.4
```

```
In [ ]: # La tortue et l'écran de jeu
        papier = turtle.Screen()
        crayon = turtle.Turtle()
        tortue_spot = turtle.Turtle()
        tortue_carte = [turtle.Turtle() for _ in range(26)]
        tortue_ligne = turtle.Turtle()
        tortue_pause = turtle.Turtle()
```

```
In [ ]:
```

```
In [ ]: # Taille et couleur du papier + coordonnées
        papier.bgcolor("lightgray")
        papier.title("Observer attentivement !")
        papier.setup(width=LARGEUR_ECRAN,height=HAUTEUR_ECRAN)
        papier.setworldcoordinates(0,0,LARGEUR_ECRAN,HAUTEUR_ECRAN)
```

```
In [ ]: # Accélération des dessins
        crayon.speed(0)
        papier.tracer(400)
```

```
In [ ]: def origine(tortue,x,y):  
        tortue.penup()  
        tortue.goto(x,y)  
        tortue.pendown()
```



```
In [ ]: def ecrit(tortue,x,y,texte,fonte):  
        origine(tortue,x,y)  
        tortue.write(texte,align="center",font=fonte)
```



```
In [ ]: def ligne(tortue,x,y,l,angle):  
        '''Trace le segment de droite d'origine (x,y) et de longueur l dans la  
        direction angle'''  
        origine(tortue,x,y)  
        tortue.setheading(angle)  
        tortue.forward(l)
```



```
In [ ]: def rectangle(tortue,x,y,lx,ly):  
        origine(tortue,x,y)  
        tortue.begin_fill()  
        for _ in range(2):  
            tortue.forward(lx)  
            tortue.left(90)  
            tortue.forward(ly)  
            tortue.left(90)  
        tortue.end_fill()
```



```
In [ ]: def cercle(tortue,x,y,r,angle=360):  
        origine(tortue,x+r,y)  
        tortue.setheading(90)  
        tortue.pendown()  
        tortue.begin_fill()  
        tortue.circle(r)  
        tortue.end_fill()
```



```
In [ ]: def  
set_crayon(tortue,epaisseur=1,couleur="black",remplissage="white",visible=False)  
        tortue.pensize(epaisseur)  
        tortue.color(couleur)  
        tortue.fillcolor(remplissage)  
        if visible:  
            tortue.showturtle()  
        else:  
            tortue.hideturtle()
```



```
In [ ]: def dessine_carte(index,liste,couleur=None,visible=True,posx=0,posy=0):
    if couleur==None:
        couleur = tortue_carte[index].pencolor()
    tortue_carte[index].reset()
    if visible:

set_crayon(tortue_carte[index],epaisseur=5,couleur=couleur,remplissage="white")
    rectangle(tortue_carte[index],MARGE+
(LARGEUR_CARTE+ECART)*index+posx,LIGNE_CARTE+posy,LARGEUR_CARTE,HAUTEUR_CARTE)
    ecrit(tortue_carte[index],MARGE+
(LARGEUR_CARTE+ECART)*index+LARGEUR_CARTE//2+posx,LIGNE_CARTE+HAUTEUR_CARTE//
2+posy,liste[index],fonte=("Arial",24,"bold"))
```

```
In [ ]: def dessine_liste(liste):
    set_crayon(crayon,epaisseur=10,couleur="darkblue",remplissage="lightgray")
    rectangle(crayon,0,0,LARGEUR_ECRAN-10,HAUTEUR_ECRAN-10)
    couleur = COULEUR_BASE
    for ind in range(len(liste)):
        dessine_carte(ind,liste,couleur)
    pointille(0)
```

```
In [ ]: def pointille(index):
    tortue_ligne.reset()
    set_crayon(tortue_ligne,epaisseur=4,couleur="brown")
    start = MARGE*2
    inc = 10
    while start+inc<HAUTEUR_ECRAN-MARGE*2:
        ligne(tortue_ligne,MARGE+(LARGEUR_CARTE+ECART)*index-ECART//
2,start,inc//4,90)
        start=start+inc
```

```
In [ ]: def dessine_spot(index,visible=True):
    tortue_spot.reset()
    if visible:
        set_crayon(tortue_spot,couleur=COULEUR_SPOT,remplissage=COULEUR_SPOT)
        tortue_spot.shapesize(2)
        origine(tortue_spot,MARGE+(LARGEUR_CARTE+ECART)*index+LARGEUR_CARTE//
2,LIGNE_SPOT)
        tortue_spot.setheading(90)
        tortue_spot.stamp()
```

```
In [ ]: def deplace_vertical(liste,i,vy,env = False):
        signe = -int(abs(vy)/vy)
        if env:
            for y in range(vy*signe,0):
                dessine_carte(i,liste,posy=y*signe)
                time.sleep(MOVE)
        else:
            for y in range(0,-vy*signe):
                dessine_carte(i,liste,posy=-y*signe)
                time.sleep(MOVE)
```



```
In [ ]: def deplace_horizontal(liste,i,j):
        signe = int(abs(i-j))/(i-j)
        for x in range(0,signe*(i-j)*(LARGEUR_CARTE+ECART)):
            dessine_carte(i,liste,posy=(-HAUTEUR_CARTE-ECART)*signe,posx=-
x*signe)
            time.sleep(MOVE)
```



```
In [ ]: def show_echange(liste,i,j):
        if i!=j:
            deplace_vertical(liste,i,HAUTEUR_CARTE+ECART)
            deplace_vertical(liste,j,-HAUTEUR_CARTE-ECART)
            deplace_horizontal(liste,i,j)
            deplace_horizontal(liste,j,i)
            liste[i],liste[j] = liste[j],liste[i]
            tortue_carte[i],tortue_carte[j] = tortue_carte[j],tortue_carte[i]
            deplace_vertical(liste,i,-HAUTEUR_CARTE-ECART,env=True)
            deplace_vertical(liste,j,HAUTEUR_CARTE+ECART,env=True)
```



```
In [ ]: def pause_tri(x,y):
        global pause
        pause = not pause
        papier.update()
        while pause:
            set_crayon(tortue_pause)
            origine(tortue_pause,LARGEUR_ECRAN//2,HAUTEUR_ECRAN-MARGE*2)
            tortue_pause.color("red")
            tortue_pause.write(chr(0x23F8),align="center",font=("Arial",
42,"bold"))
            time.sleep(WAIT_BLINK)
            tortue_pause.color("black")
            tortue_pause.write(chr(0x23F8),align="center",font=("Arial",
42,"bold"))
            time.sleep(WAIT_BLINK)
```



```
In [ ]: pause = False
        papier.onscreenclick(pause_tri)
```



```
In [ ]: parser =
    argparse.ArgumentParser(description="Visualisation du fonctionnement de
    l'algorithme du tri par insertion")
    parser.add_argument('-l', type=str, help="Lettres à trier dans l'ordre
    alphabétique")
    args = vars(parser.parse_args())
    liste = list(args['l'])
    nb=len(liste)
    dessine_liste(liste)
    papier.update()
    time.sleep(3)
    for index in range(nb-1):
        dessine_spot(index)
        cmin = index
        dessine_carte(index,liste,"orange")
        for j in range(index+1,nb):
            tortue_pause.reset()
            dessine_carte(j,liste,"lime")
            time.sleep(PARCOURS)
            if liste[j]<liste[cmin]:
                dessine_carte(cmin,liste,COULEUR_BASE)
                cmin = j
                dessine_carte(cmin,liste,"orange")
            else:
                dessine_carte(j,liste,COULEUR_BASE)
        for _ in range(3):
            dessine_carte(cmin,liste,"red")
            dessine_spot(index,visible=False)
            time.sleep(WAIT_BLINK)
            dessine_carte(cmin,liste,"orange")
            dessine_spot(index,visible=True)
            time.sleep(WAIT_BLINK)
            dessine_carte(cmin,liste,"red")
            show_echange(liste,index,cmin)
            dessine_carte(index,liste,"green")
            pointille(index+1)
        dessine_carte(nb-1,liste,"green")
        dessine_spot(nb-1)
        pointille(nb)
        papier.update()
```

```
In [ ]:
```

```
In [ ]: papier.exitonclick()
```