Python higer order functions and event listeners

.listen() is used to make the turtle to listen from the user

turtle.onkey(fun, key)

turtle.onkeyrelease(fun, key)

Parameters:

fun : a function with no arguments or None

key : a string: key (e.g. "a") or key-symbol (e.g. "space")

Bind fun to key-release event of key.

If fun is None, event bindings are removed.

Remark: in order to be able to register key-events, TurtleScreen must have the focus

In [ ]:
```
from turtle import Turtle, Screen

pointer = Turtle()
screen = Screen()

def move_forward():
    pointer.forward(10)

screen.listen()
screen.onkey(key = "space", fun = move_forward)
screen.exitonclick()
```

Task:

To build a Etch-a-sketch app using turtle module

In [ ]:
```
from turtle import Turtle, Screen

pointer = Turtle()
screen = Screen()

def move_forward():
    pointer.forward(50)

def move_backward():
    pointer.backward(50)

def clock_wise():
    pointer.right(10)

def anti_clock_wise():
    pointer.left(10)

screen.listen()
screen.onkey(move_backward,"s")
screen.onkey(move_forward,"w")
screen.onkey(anti_clock_wise,"a")
screen.onkey(clock_wise,"d")
screen.exitonclick()
```

Task:

To build a Etch-a-sketch app using turtle module.

In [ ]:
```
from turtle import Turtle, Screen

pointer = Turtle()
screen = Screen()

def move_forward():
    pointer.forward(50)

def move_backward():
    pointer.backward(50)

def clock_wise():
    new_heading = pointer.heading() - 10
    pointer.setheading(new_heading)

def anti_clock_wise():
    new_heading = pointer.heading() + 10
    pointer.setheading(new_heading)

def clear():
    pointer.clear()
    pointer.penup()
    pointer.home()
    pointer.pendown()

screen.listen()
screen.onkey(move_backward,"s")
screen.onkey(move_forward,"w")
screen.onkey(anti_clock_wise,"a")
screen.onkey(clock_wise,"d")
screen.onkey(clear,"c")
screen.exitonclick()
```

Object State and Instances

We can create many object form the same classes.

point_1 = turtle()

point_2 = turtle()

here two objects were created by using same class.

Task:

To create a turtle race.

In [ ]:
```
'''
from turtle import Turtle, Screen
import random as rd

screen = Screen()
screen.setup(width=500, height=400)
user_bet = screen.textinput(title="make your bet", prompt="which turtle will win the race ?/nEnter the color ?")
colors = ["red","orange","yellow","green","blue","purple"]
y_axis = [-70,-40,-10,20,50,80]
# print(user_bet)

is_race_on = False

#understanding the co ordinates  x and y axis
#pointer = Turtle(shape = "turtle")
#pointer.penup()
#pointer.goto(x = -230, y = 0)

all_turtles=[]
for i in range(0,6):
    new_turtle = Turtle(shape = "turtle")
    new_turtle.color(colors[i])
    new_turtle.penup()
    #understanding the co ordinates  x and y axis
    new_turtle.goto(x = -230, y = y_axis[i])
    all_turtles.append(new_turtle)

if user_bet:
    is_race_on=True

while is_race_on:
    for turtle in all_turtles:
        if turtle.xcor() > 230:
            is_race_on = False
            winner = turtle.pencolor()
            if user_bet == winner:
                print(f"you won, the {winner} is the winner")
            else:
                print(f"you lose, the {winner} is the winner")
            is_race_on = False
        speed = rd.randint(0,10)
```

```
        turtle.forward(speed)

screen.exitonclick()
'''
```