**Exercise:**

1. Check if x > 10, retrun T/F
2. Define a function called square that returns the squared value of input X

**Library Importing**

**Different ways to import a library**

**Import numpy**

Numpy.absoulte(-7)

7

numpy.sqrt(8)

2.8284271247461903

**from numpy import \* (no need to add library as prefix)**

absolute(-7)

sqrt(8)

**from numpy import absolute**

absolute(-7)

sqrt(8)

**import numpy as np**

np.absolute(-7)

np.sqrt(8)

**Turtle**

**import turtle as t**

**important functions**

**move forward:**

t.forward(distance) / t.fd(distance)

**move backward**

t.backward(distance) / t.bk(distance) / t.back(distance)

**move to the left**

set angle first then forward

t.left(degree) / t.lt(degree)

t.forward(distance)

**move to the right**

t.right(degree) / t.rt(degree)

t.forward(distance)

**end the window**

t.bye()

**or**

t.exitonclick()

**Question**: how do we draw a square??

t.forward(100)

t.left(90)

t.fd(100)

t.left(90)

t.fd(100)

t.left(90)

t.fd(100)

t.left(90)

t.exitonclick()

**OR**

t.forward(100)

for i in range(3):

t.left(90)

t.fd(100)

t.left(90)

t.exitonclick()

**Question2: how do we draw a circle?**

t.circle(10)

**Question3: draw 3 squares that each is 20 degrees more turn to the left**

t.left(20)

t.forward(50)

t.left(90)

t.forward(50)

t.left(90)

t.forward(50)

t.left(90)

t.forward(50)

t.left(90)

t.left(20)

t.forward(50)

t.left(90)

t.forward(50)

t.left(90)

t.forward(50)

t.left(90)

t.forward(50)

t.left(90)

t.left(20)

t.forward(50)

t.left(90)

t.forward(50)

t.left(90)

t.forward(50)

t.left(90)

t.forward(50)

t.left(90)

**Set a window**

screen = t.Screen()

**set the background color**

screen.bgcolor(‘lightgreen’)

**set screen title**

screen.title(‘Rick’s Program’)

**Exit properly**

Screen.exitonclick()

**Adding motions into the screen**

rick = t.Turtle() # naming a turtle

rick.color('red') # set up the turtle color

rick.shape('turtle') # set the shape of the turtle

rick.pencolor(‘blue’)

rick.pensize(20)

rick.forward(100)

rick.left(90)

rick.fd(100)

rick.left(90)

rick.fd(100)

rick.left(90)

rick.fd(100)

rick.left(90)

**lift up the pen**

rick.penup()

**putdown the pen**

rick.pendown()

**Two turtles in a same screen**

rick2 = t.Turtle()

rick2.shape(‘triangle’)

rick2.penup()

rick2.left(45)

rick2.forward(sqrt((50\*\*2)\*2))

rick2.circle(sqrt((50\*\*2)\*2)))

**what if the shape is too small or large?**

rick2.shapesize(0.5,0.5,0) ##width, length, outline

**have to use with**

rick2.resizemode(‘user’)

**want to start the turtle at a certain position?**

rick2.setpos(-100,0)

## if don’t want to show the line, use penup() & pendown()

rick2.penup()

rick2.setpos(-100,0)

rick2.pendown()

rick2.forward(100)

**don’t want to see the turtle?**

rick2.hideturtle() or rick2.ht()

**show turtle**

rick2.showturtle() or rick2.st()

turtle.listen() ## tell program to listen to the keyboard

turtle.onkey(**function**, “Left”)

**GAME**(part1):

##Set up screen

screen = t.Screen()

screen.bgcolor('lightgreen')

##set up border

border = t.Turtle()

border.shape('turtle')

border.color('black','red')

border.penup()

border.setpos(-300,-300)

border.pendown()

border.pensize(3)

for side in range(4):

border.forward(600)

border.lt(90)

border.hideturtle()

#set up player

player = t.Turtle()

player.color('blue')

player.shape('triangle')

player.penup()

#Create multiple goals

# maxGoals = 6

# goals = []

# for count in range(maxGoals):

# goals.append(t.Turtle())

# goals[count].color('red')

# goals[count].shape('circle')

# goals[count].penup()

# goals[count].speed(0)

# goals[count].setpos(random.randint(-300,300),random.randint(-300,300))

##Goal

goal = t.Turtle()

goal.color('red')

goal.shape('circle')

goal.penup()

goal.speed(0)

goal.setpos(-100,100)

#make the turtle run fastest

player.speed(0)

speed = 1

##define functions

def turnleft():

player.lt(30)

def turnright():

player.rt(30)

def speed5():

global speed

speed = 5

def speed1():

global speed

speed = 1

#optional

# def iscollision(p1,p2):

# d = np.sqrt((p1.xcor() - p2.xcor())\*\*2 + (p1.ycor() - p2.ycor())\*\*2)

# if d<20:

# return True

# else:

# return False

#set keyboard bindings

t.listen()

t.onkey(turnleft,'Left')

t.onkey(turnright, 'Right')

t.onkeypress(speed5, 'Up')

t.onkeyrelease(speed1, 'Up')

while True:

player.forward(speed)

##Boundary checking

if player.xcor()>300 or player.xcor()<-300:

player.right(180)

if player.ycor()>300 or player.ycor()<-300:

player.right(180)

d = np.sqrt((player.xcor() - goal.xcor())\*\*2 + (player.ycor() - goal.ycor())\*\*2)

if d<20:

goal.setpos(random.randint(-300,300),random.randint(-300,300))

##move the goal around

goal.forward(3)

if goal.xcor()>290 or goal.xcor()<-290:

goal.right(180)

if goal.ycor()>290 or goal.ycor()<-290:

goal.right(180)