Problem Solving Session

- The remainder of today's class will comprise the ***problem solving session*** (***PSS***).
- Your instructor will divide you into ***teams of 3 or 4 students***.
- Each team will ***work together*** to solve the following problems over the course of ***20-30 minutes***.
  - You may work on paper, a white board, or digitally as determined by your instructor.
  - You will submit your solution by pushing it to GitHub before the end of class.
- Your instructor will go over the solution before the end of class.
- If there is any time remaining, you will begin work on your homework assignment.

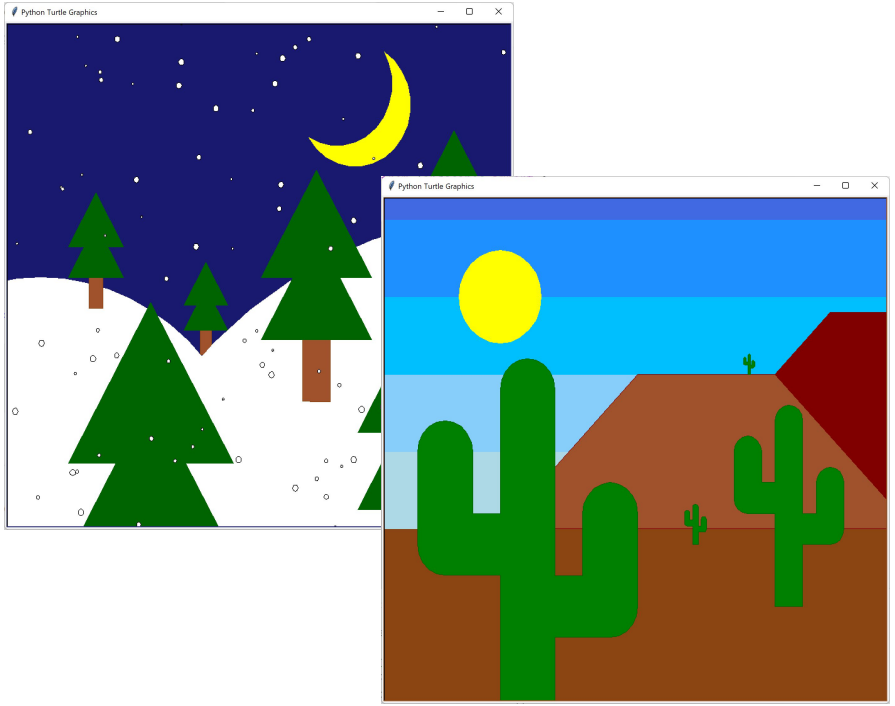
Class participation is a significant part of your grade (20%). This includes in class activities and the problem solving session.

Your graders will grade your participation by verifying that you pushed your solutions before the end of the class period each day.

Turtle Landscapes

Your next homework assignment will require you to use Python's Turtle Graphics to draw a landscape of your own design. The image should combine simple geometric shapes like circles, triangles, and rectangles to represent natural features like trees, rocks, and cacti.

Work together with your problem solving team to solve the following problems. They have been designed to give every student a solid head start on completing the homework, which will be due at the start of the next class. ***If you finish early, you should get started on implementing the homework in the time that you have left!***



Problem Solving 1

Remember, `turtle.circle(r)` draws a circle of radius `r` to the ***turtle's left***.

Working together with your problem solving team, write a function that uses the turtle to draw a filled in circle anywhere in the turtle window. Your function ***must*** use parameters to determine:

- The x,y coordinate of the starting point.
- The radius of the circle.
- The pen color.
- The fill color.

What is the state of the turtle when your function is finished drawing the circle?

```
def circle(x, y, radius, pencolor, fillcolor):
    turtle.pencolor(pencolor)
    turtle.fillcolor(fillcolor)
    turtle.up()
    turtle.goto(x, y)
    turtle.begin_fill()
    turtle.down()
    turtle.circle(radius)
    turtle.end_fill()
    turtle.up()

def main():
    turtle_state()
    turtle.bgcolor("Beige")
    circle(100, 100, 10, "Red", "Navy")
    input("Press enter to exit: ")
    turtle_state()

main()
```

```
def equilateral_triangle(x, y, size, angle, pencolor, fillcolor):
    turtle.pencolor(pencolor)
    turtle.fillcolor(fillcolor)
    turtle.up()
    turtle.goto(x, y)
    turtle.begin_fill()
    turtle.down()
    turtle.forward(size)
    turtle.left(angle)
    turtle.forward(size)
    turtle.left(angle)
    turtle.forward(size)
    turtle.left(angle)
    turtle.end_fill()
    turtle.penup()

def main():
    turtle_state()
    turtle.bgcolor("Beige")
    equilateral_triangle(-100, -100, 300, 120, "Navy", "Red")
    input("Press enter to exit: ")
    turtle_state()

main()
```

Problem Solving 2

Write a function that uses the turtle to draw a filled in *equilateral triangle* anywhere in the turtle window. Your function ***must*** use parameters to determine:

- The x,y coordinate of the bottom left corner.
- The size of the base of the triangle.
- The pen color.
- The fill color.

The turtle must end in the bottom left corner of the triangle with its original heading.

Hint: it may be a good idea to draw a picture of an equilateral triangle *first*; label the (x,y) coordinate of the bottom left, interior and exterior angles, etc.

Problem Solving 3

Write a function that uses the turtle to draw a filled in *rectangle* anywhere in the turtle window. Your function ***must*** use parameters to determine:

- The x,y coordinate of the bottom left corner.
- The width and height of the rectangle.
- The pen color.
- The fill color.

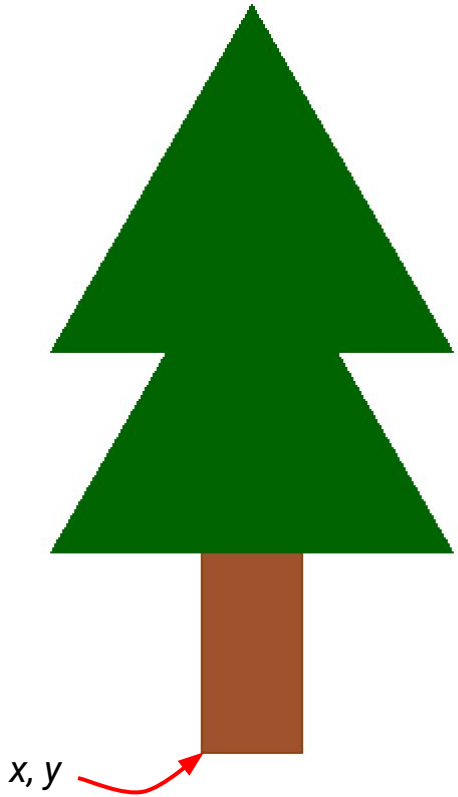
The turtle must end in the bottom left corner of the rectangle with its original heading.

```
def rectangle(x, y, width1, width2, height, pencolor, fillcolor):
    turtle.pencolor(pencolor)
    turtle.fillcolor(fillcolor)
    turtle.up()
    turtle.goto(x, y)
    turtle.begin_fill()
    turtle.down()
    turtle.forward(width1)
    turtle.left(height)
    turtle.forward(width2)
    turtle.left(height)
    turtle.forward(width1)
    turtle.left(height)
    turtle.forward(width2)
    turtle.left(height)
    turtle.end_fill()
    turtle.up()

def main():
    turtle_state()
    turtle.bgcolor("Beige")
    rectangle(-200, 200, 400, 200, 90, "Red", "Navy")
    input("Press enter to exit: ")
    turtle_state()

main()
```

Problem Solving 4



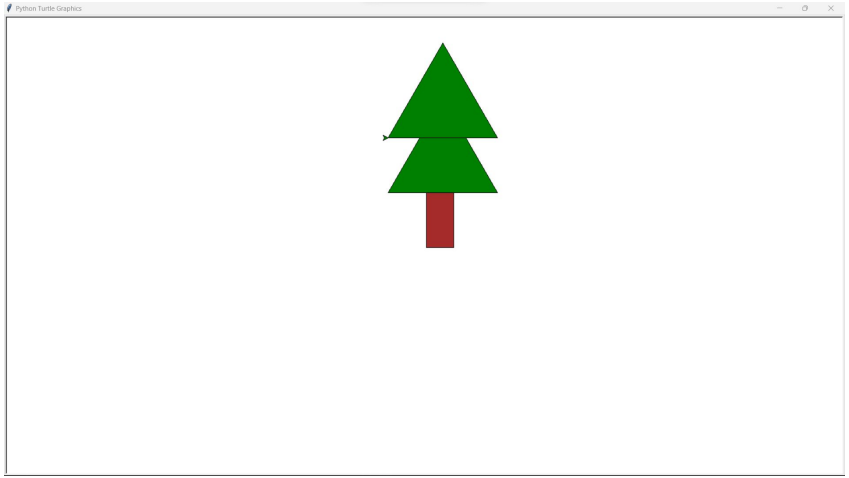
Using the functions that you wrote previously, work together with your team to write a function that draws a pine tree like the one depicted to the left. You must declare parameters:

- The x,y coordinates of the bottom left corner of the trunk.
- The "size" of the tree. What this means is up to you to decide, but your function should be able to draw trees of different sizes by changing this value.

If you are working digitally, write your function on the next slide.

Your Tree Drawing Function

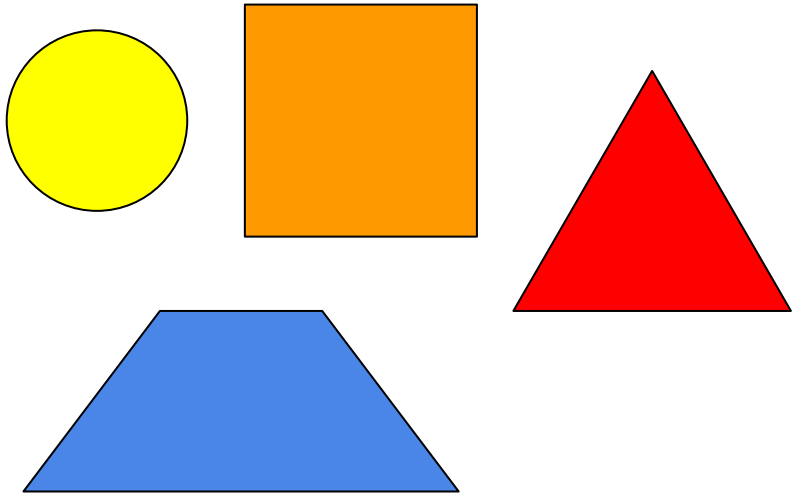
```
def tree():
    turtle.pencolor("Black")
    turtle.fillcolor("Brown")
    turtle.begin_fill()
    turtle.down()
    turtle.forward(50)
    turtle.left(90)
    turtle.forward(100)
    turtle.left(90)
    turtle.forward(50)
    turtle.left(90)
    turtle.forward(100)
    turtle.left(180)
    turtle.forward(100)
    turtle.end_fill()
    turtle.right(90)
    turtle.forward(25)
    equilateral_triangle(-70, 100, 200, 120, "Black", "Green")
    turtle.forward(100)
    turtle.left(90)
    turtle.forward(200)
```



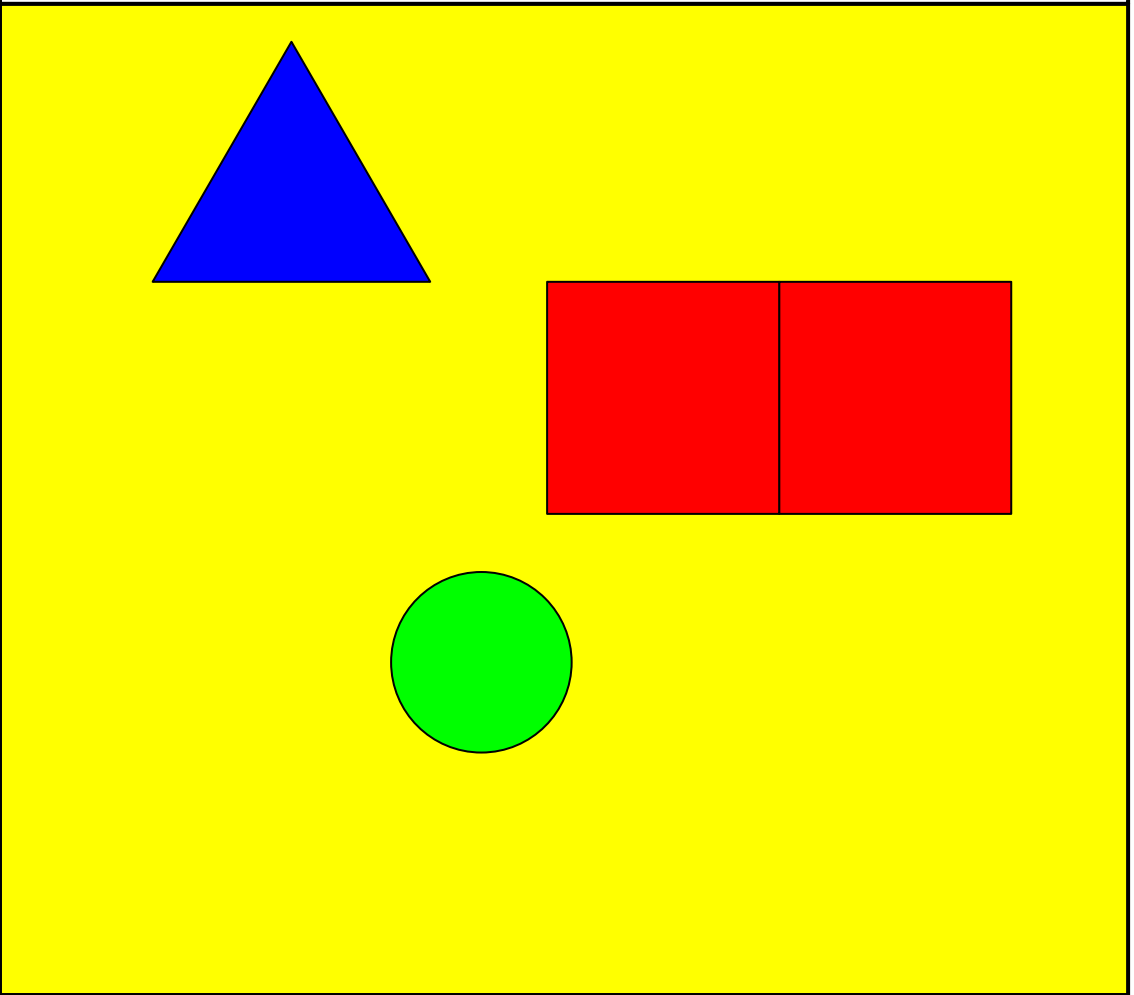
Problem Solving 5

Create a sketch of a landscape of your own design that uses any combination of the functions that you have written up to this point. Annotate the diagram with details including x,y coordinates, sizes, and colors.

If you are working digitally copy & paste, resize, and recolor the provided shapes. Create your landscape on the next slide.



Turtle Window

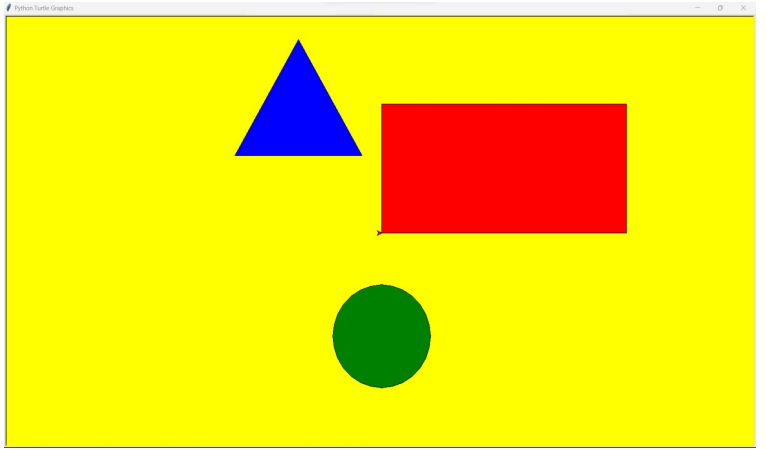


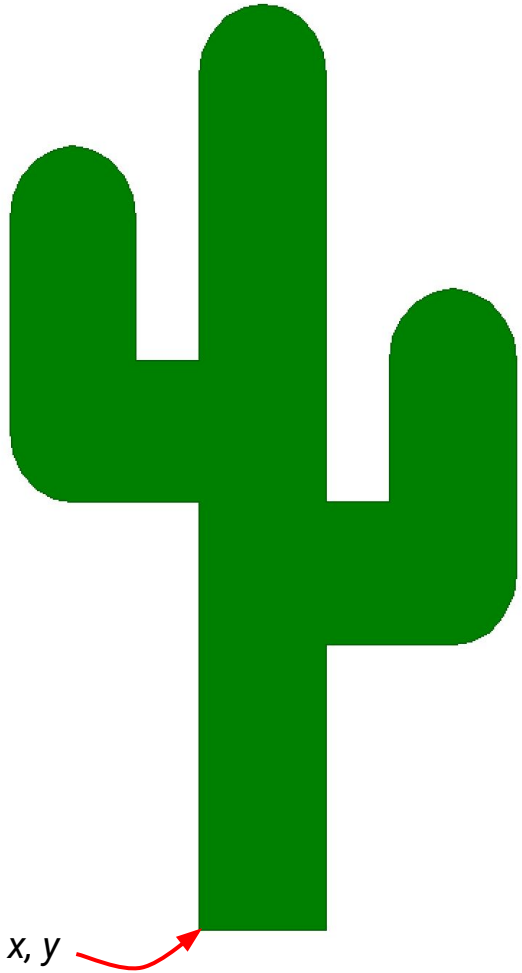
Programs/Code:

```
def main():
    turtle_state()
    turtle.bgcolor("Yellow")
    rectangle(0, 0, 500, 250, 90, "Black", "Red")
    equilateral_triangle(-300, 150, 260, 120, "Black", "Blue")
    circle(0, -300, 100, "Black", "Green")
    input("Press enter to exit: ")
    turtle_state()

main()
```

Screen:





Problem Solving 6

Not every element of your final drawing needs to be made up of basic geometric shapes. Working together with your team, write a function that can draw a cactus like the one depicted to the right. Your function must declare parameters for:

- The x,y coordinates of the bottom left corner of the trunk.
- The "size" of the cactus. What this means is up to you to decide, but your function should be able to draw cacti of different sizes by changing this value.

If you are working digitally, write you function on the next slide.

Your Cactus Drawing Function

```
Import turtle
Def cactus(x,y,size):
     turtle.penup()
     turtle.goto(x,y)
     turtle.pendown()
     turtle.color(“green”)
     turtle.begin_fill()
     turtle.forward(size)
     turtle.right(90)
     turtle.forward(size)
     turtle.right(90)
     turtle.forward(size)
     turtle.right(90)
     turtle.forward(size)
     turtle.end_fill()
     turtle.penup()
     turtle.goto(x,y+size)
     turtle.pendown()
     turtle.begin_fill()
     turtle.circle(size/2)
     turtle.end_fill()

Def main():
     cactus(0,0,100)
     cactus(100,100,100)
     cactus(100,200,100)

main()
```