SI@UCF Computer Science Camp
Object Oriented Python and pyGame
Quiz #1 Solutions
Date: 6/14/2025

1) (10 pts) Complete the Python program below so that it reads in the number of seconds it took to complete a 5 mile run and prints out the time in minutes and seconds. You may assume that the user inputs an integer in between 1 and 3599, inclusive. (So less than one hour!)

```
totalsec = int(input("How many seconds for the run?\n"))

numMin = totalsec//60
numSec = totalsec%60

print("Your run took",min,"minutes and", sec,"seconds")
```

Grading: 5 pts for each statement, take off 2 pts if real number division is used, take off 3 pts total if the expressions are swapped.

2) (15 pts) An arithmetic sequence is a sequence of numbers where the difference between consecutive terms is the same. We can specify an arithmetic sequence by defining the first term, common difference and number of terms. For example, the sequence with the first term first = 5, common difference d = 4 and number of terms n = 6 is 5, 9, 13, 17, 21 and 25. The sum of these six terms is 90. Complete the program below so that it **uses a loop** to find the sum of the arithmetic sequence specified by the values entered by the user.

```
a = int(input("What is the first term of your arithmetic sequence?\n"))
d = int(input("What is the common difference of your arithmetic sequence?\n"))
n = int(input("How many terms I your arithmetic sequence?\n"))

total = 0

for i in range(n):                  // 3 pts
    total += a                      // 6 pts
    a += d                          // 6 pts

print("The sum of the terms in your arithmetic sequence is", total)
```

Grading: Many ways to solve this. Map points as you see fit.

3) (30 pts) In lecture, a small fraction class was started, but not finished. Below is the code written in class. For this question, you'll be asked to asked to add some methods to this class.

```
# Arup Guha
# 6/10/2025
# Start of a Fraction Class

import math

class Fraction:

    # Basic constructor.
    def __init__(self, num, den):

        # Get the common divisor.
        div = math.gcd(num, den)

        # Set up the fraction in lowest terms.
        self.num = num//div
        self.den = den//div

    # Add self to frac2 and return the result in a new Fraction.
    def add(self, frac2):

        newNum = self.num*frac2.den + frac2.num*self.den
        newDen = self.den*frac2.den
        return Fraction(newNum, newDen)

    # String representation of a fraction.
    def __str__(self):
        return str(self.num)+"/"+str(self.den)

# Quick test.
def test():

    # Create two fractions.
    f1 = Fraction(2,3)
    f2 = Fraction(5,6)

    # Add and print out!
    ans = f1.add(f2)
    print(f1,"+",f2,"=",ans)

# Run it.
test()
```

3a) (10 pts) Add a method to subtract two fractions. Note that if we allow for subtraction, we have to support negative fraction values. If we agree that the value of the denominator is always positive, then for all negative fractions the numerator will be a negative number. If for whatever reason an intermediate calculation makes the denominator negative, then internally, both the numerator and denominator must immediately negated so that the object is equivalent with a positive denominator. Complete the subtract method below, which should return a new Fraction object that is the result of subtracting frac2 from this fraction object (self).

```
# Subtract frac2 from self return the result in a new Fraction.
def subtract(self, frac2):

    newNum = self.num*frac2.den - frac2.num*self.den
    newDen = self.den*frac2.den
    return Fraction(newNum, newDen)
```

Grading: newNum – 4 pts, newDen – 3 pts, return – 3 pts

3b) (10 pts) Add a method to multiply two fractions. The method should return a new Fraction that is the product of self and frac2.

```
# Multiply self and frac2, return the result in a new Fraction.
def multiply(self, frac2):

    newNum = self.num*frac2.num
    newDen = self.den*frac2.den
    return Fraction(newNum, newDen)
```

Grading: newNum – 4 pts, newDen – 3 pts, return – 3 pts

3c) (10 pts) Add a method to raise fraction (self) to a positive exponent, exp. Your method should call the multiply method and it should return a new Fraction object storing the result.

```
# Add self to frac2 and return the result in a new Fraction.
def pow(self, exp):

    ans = Fraction(1,1)
    for i in range(exp):
        ans = ans.multiply(self)
    return ans
```

Grading: 3 pts – accumulator var (can be 1 or self), 2 pts loop, 4 pts multiply, 1 pt return

4) (15 pts) The flag of Italy is three rectangles from left to right in the following colors: green, white, red. For this question, complete the pyGame program below to draw the Italian flag. Please make the top left corner of the green rectangle (200, 100), the top left corner of the white rectangle (400, 100) and the top left corner of the red rectangle (600, 100). Make the height of each rectangle 400 pixels.

```
import pygame, sys
from pygame.locals import *
pygame.init()
DISPLAYSURF = pygame.display.set_mode((1000, 600))
pygame.display.set_caption("Italian Flag")
BLACK = pygame.Color(0,0,0)
RED = pygame.Color(255,0,0)
GREEN = pygame.Color(0,255,0)
WHITE = pygame.Color(255,255,255)

while True:
    for event in pygame.event.get():
        if event.type == QUIT:
            pygame.quit()
            sys.exit()

    DISPLAYSURF.fill(BLACK)
    # PUT YOUR CODE HERE, SHOULD BE THREE LINES.
    pygame.draw.rect(DISPLAYSURF, GREEN, (200, 100, 200, 400))
    pygame.draw.rect(DISPLAYSURF, WHITE, (400, 100, 200, 400))
    pygame.draw.rect(DISPLAYSURF, RED,   (600, 100, 200, 400))
    pygame.display.update()
```

Grading: 5 pts each – 1 pt func call, displaysurf and color, 4 pts for rectangle

5) (10 pts) In lecture, we added "levels" to the "New Dot Game". In order to do this, we created these 2 lists:

```
GOODDOTSPEED = [50,40,30,20,10,5]
BADDOTSPEED =  [35,30,25,15,8, 3]
```

What do the numbers in these two lists represent? For example, what do both 10 and 8 specifically mean with respect to the game? (Note: This question requires that you paid attention during class as the only way to answer it is based on this section of live coding.)

The number at index i in the first list represents how many frames to wait between producing good dots when the player is playing level i+1. For example, the 10 means that once every ten frames, when we are at level 5, a new good dot gets produced, and the 8 means that once every eight frames when we are at level 5, a bad dot gets produced.

Grading: Grader's discretion

6 (16 pts) In class we changed the Token class to store a circle with both the coordinates of the circle's center and its radius. For convenience the **most of** the class is included below. For this problem you'll add four methods: distToTop(self), distToLeft(self), distToBottom(self, SCREEN_H), and disttoRight(self, SCREEN_W). These four methods will return the distance of the Token (circle) to the top of the screen, left side of the screen, bottom of the screen and the right side of the screen, respectively. You may assume these methods are only called with the Token is strictly within the viewing screen. Each method should be one line long. Please fill them in at the very end.

```
import random
import math
import time
import pygame, sys
from pygame.locals import *

# Token Class we'll use for drawing objects in pyGame
class token:

    # Constructor.
    def __init__(self,myx,myy,mydx,mydy,myr,mycolor):
        self.x = myx
        self.y = myy
        self.dx = mydx
        self.dy = mydy
        self.r = myr
        self.color = mycolor

    # Call each frame.
    def move(self):
        self.x += self.dx
        self.y += self.dy

    # Executes updating dx as necessary for bouncing off the left wall.
    def bounceLeft(self):
        if self.x + self.dx < self.r:
            self.dx = -self.dx

    # Executes updating dx as necessary for bouncing off the right wall.
    def bounceRight(self, SCREEN_W):
        if self.x + self.dx > SCREEN_W-self.r:
            self.dx = -self.dx

    # Executes updating y as necessary for bouncing off the top wall.
    def bounceUp(self):
        if self.y + self.dy < self.r:
            self.dy = -self.dy

    # Executes updating y as necessary for bouncing off the bottom wall.
    def bounceDown(self, SCREEN_H):
        if self.y + self.dy > SCREEN_H-self.r:
            self.dy = -self.dy
```

```
# Adds addDX to the the change in x per frame, and adds addDY to
# the change in y per frame.
def changeVelocity(self,addDX,addDY):
    self.dx += addDX
    self.dy += addDY

# Draws this object on the display surface as a circle.
# Likely to be overridden most of the time.
def draw(self, DISPLAYSURF):
    pygame.draw.circle(DISPLAYSURF, self.color,(self.x,self.y),self.r, 0)

# Update for a single frame. Maybe this will typically be overridden.
def updateFrame(self, DISPLAYSURF):
    self.move()
    self.bounceLeft()
    self.bounceRight(DISPLAYSURF.get_width())
    self.bounceUp()
    self.bounceDown(DISPLAYSURF.get_height())
    self.draw(DISPLAYSURF)

# Returns true iff the two circles represented by self and other intersect.
def collide(self, other):
    center_dx = self.x - other.x
    center_dy = self.y - other.y
    dist_sq = center_dx*center_dx + center_dy*center_dy
    dist_sq_radii = (self.r + other.r)*(self.r + other.r)
    return dist_sq <= dist_sq_radii

# Returns true iff the position pos is within the circle.
def inCircle(self, pos):
    # Calculate the distance squared between pos and circle center.
    dist_sq = (pos[0]-self.x)*(pos[0]-self.x) +
              (pos[1]-self.y)*(pos[1]-self.y)
    return dist_sq <= self.r*self.r

 # FILL THIS IN!!!
 def distToTop(self):
     return self.y-self.r

 # FILL THIS IN!!!
 def distToLeft(self):
     return self.x-self.r

 # FILL THIS IN!!! SCREEN_H is height of the screen.
 def distToBottom(self, SCREEN_H):
     return SCREEN_H - (self.y+self.r)

 # FILL THIS IN!!! SCREEN_W is the width of the screen.
 def distToRight(self, SCREEN_W):
     return SCREEN_W - (self.x+self.r)
```

Grading: 4 pts for each part

7) (4 pts) June 14th (the day this exam was made) was World Blood Donor Day. What do Blood Donors donate? **Blood (Give to All)**