SI@UCF Introduction to Programming in Python/PyGame Test #2 Solutions
6/29/2016

1) (12 pts) Write a segment of code which prints an "backwards odd counting triangle" of n rows. The i^{th} row of an odd counting triangle contains the first positive i integers, in descending order. For example, the odd counting triangle of 4 rows is

```
1
3 1
5 3 1
7 5 3 1
```

```
n = int(input("How many rows do you want?\n"))
```

```
    for i in range(n):                  # 3 pts
        start = 2*i + 1                 # 2 pts
        for j in range(i+1):            # 2 pts
            print(start,end=" ")        # 2 pt
            start -= 2                  # 2 pts
        print()                         # 1 pt

# 3 pts outer loop num iter
# 2 pts inner loop num iter
# 4 pts controlling values that get printed
# 3 pts particulars of printing
```

2) (12 pts) Assume that a list numbers has been created and filled with some positive integers. Write a segment of code that calculates the minimum value in the list, the maximum value in the list and the sum of the numbers in the list and prints out these three values.

```
    total = 0
    low = numbers[0]
    high = numbers[0]

    for i in range(len(numbers)):
        total += numbers [i]
        if numbers [i] < low:
            low = numbers [i]
        if numbers [i] > high:
            high = numbers [i]

    print("min is",low,"max is",high,"sum is",total)

# Grading 3 pts for going through the numbers
#         1 pt total, 2 pts update total
#         2 pts for min
#         2 pts for max
#         2 pts printing
```

3) (15 pts) A binary number is represented using 0s and 1s. The value of a binary number is simply the sum of the values of each of its bits set to 1. The value of a bit that is k locations from the rightmost place in the number is 2^k. For example, the binary number 110101 has the value $2^5+2^4+2^2+2^0 = 53$ in base 10. Write a function that takes in an integer in binary and returns its equivalent value in base 10. For example, base10Value(110101) should return 53. (Note: think of using integer division and mod repeatedly to access each bit of the input value. You will access these values in reverse order. While you do this, you can store another variable that starts at 1, doubling each time, so that each appropriate bit value is obtained.)

```
def base10Value(n):

    bit = 1                     # 1 pt
    res = 0                     # 1 pt
    while n > 0:                # 2 pts
        if n%2 == 1:            # 3 pts
            res += bit          # 3 pts
        n //= 10                # 2 pts
        bit *= 2                # 2 pts
    return res                  # 1 pt

# 3 pts for maintaining the bit value somehow
# 7 pts for updating the result
# 2 pts var init
# 2 pts looping mechanism
# 1 pt return
```

4) (10 pts) Write a segment of code that creates an empty dictionary that maps people to the college they attended. Add the information that "Hillary" attended "Wellesley" and that "Donald" attended "UPenn". Finally, ask the user to enter their name and college and add this to the dictionary.

```
colleges = {}                                   # 2 pts
colleges["Hillary"] = "Wellesley"               # 2 pts
colleges["Donald"] = "UPenn"                    # 2 pts
name = input("What is your name?\n")            # 1 pt
college = input("Where did you go to college?\n") # 1 pt
colleges[name] = college                        # 2 pts
```

5) (15 pts) Complete the segment of code below so that the image will move diagonally up and to the left 1 pixel when the Q key on the keyboard is pressed, and diagonally up and to the right 1 pixel when the E key is pressed. Your program should continue to move the image as long as the key is being pressed.

```
import pygame, sys
from pygame.locals import  *

pygame.init()
displaysurf = pygame.display.set_mode((1000, 1000))
pygame.display.set_caption("Diagonal")

img = pygame.image.load("raindrop.png")
x = 100
y = 100


while True:

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


    list = pygame.key.get_pressed()   # 3 pts
    if list[K_q] == True:             # 2 pts
        x = x-1                         # 2 pts
        y = y-1                         # 2 pts
    if list[K_e] == True:               # 2 pts
        x = x+1                         # 2 pts
        y = y-1                         # 2 pts



    displaysurf.fill(pygame.Color(255, 255, 255))
    displaysurf.blit(img, (x, y))
    pygame.display.update()
```

6) (15 pts) Complete the program below so that a green triangle and a brown rectangle are drawn in the image of a tree. Your trunk should be centered, 20 pixels wide and 50 pixels tall. The triangle that forms the leaves should be 100 pixels wide and 75 pixels tall.

```
import pygame, sys
from pygame.locals import  *

pygame.init()
displaysurf = pygame.display.set_mode((1000, 1000))
pygame.display.set_caption("Teleport")

green = pygame.Color(0, 255, 0)
brown = pygame.Color(139, 69, 19)

pts = [(50, 100), (150, 100), (100, 25)]  # 6 pts
                                          # Mult ans possible
while True:

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

    # 5 pts - 2 pts coordinates, 3 pts rest
    pygame.draw.rect(displaysurf, brown, (90, 100, 20, 50), 0)
    pygame.draw.polygon(displaysurf, green, pts, 0) # 4 pts

    pygame.display.update()
```

7) (20 pts) Complete the program below so that the a ball that hits the right edge of the screen will teleport to the left side of the screen, and a ball that hits the left edge of the screen will teleport to the right side of the screen. The ball should bounce off of the top and bottom edges of the screen.

```
import pygame, sys
from pygame.locals import  *

pygame.init()
displaysurf = pygame.display.set_mode((1000, 700))
pygame.display.set_caption("Teleport")
red = pygame.Color(255, 0, 0)
black = pygame.Color(0, 0, 0)
x=10
y=10
r=10
dx=2
dy=2

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

    x+=dx                 # 2 pts
    y+=dy                 # 2 pts

    if x<0:               # 2 pts
        x=1000            # 2 pts
    if x>1000:            # 2 pts
        x=0               # 2 pts
    if y<0:               # 2 pts
        dy = -dy          # 2 pts
    if y>700:             # 2 pts
        dy=-dy            # 2 pts


    displaysurf.fill(black)
    pygame.draw.ellipse(displaysurf, red, (x, y, 2*r, 2*r), 0)
    pygame.display.update()
```

8) (1 pt) In how many dimensions do 3D printers print objects? **3 (Give to All)**

Python List Methods

```
list.append(x) - adds x to the end of list

How to create an empty list:
items = []
```

PyGame Drawing

```
# Draws a rectangle with the designated color on the Surface
# inscribed in the rectangle Rect.
pygame.draw.rect(Surface, color, Rect)

# Draws a polygon designated by the list pts, which must be
# a list of ordered pairs, listing the points of the polygon
# in order (with the last pt connecting back to the first)
pygame.draw.polygon(Surface, color, pts)


# Returns a list (that can be indexed by key) of booleans,
# each of which represents whether or not a particular key was
# pressed down.
pygame.key.get_pressed()
```

Note: A rectangle is specified as the following list (x, y, width, height), where (x,y) are the pixel coordinates of the top left corner of the rectangle and width and height are the pixel width and height, respectively, of the bounding rectangle.

Key constants

```
K_e - key for lowercase e
K_q - key for lowercase q
```