SI@UCF Introduction to Programming in Python/PyGame Test #2 Solutions
7/22/2015

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

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

```
n = int(input("How many rows do you want?\n"))
for i in range(n):                  # 3 pts
    for j in range(1,2*i+2,2):      # 5 pts
        print(j,end=" ")            # 3 pts
    print()                         # 1 pt

# Note: There are many ways to solve this, so trace each student
# response and map points accordingly.
```

2) (12 pts) Write a segment of code that reads in 1000 items from the user and add each of these items in the order they were entered into a list called numbers. (Note: You may use an empty prompt, "", for the user.)

```
for i in range(SIZE):                # 4 pts
    items.append(int(input("")))     # 4 pts read
                                     # 4 pts append
```

3) (15 pts) A tower sequence is defined by two numbers: a base and the number of terms. The first term of the sequence is the base. Each subsequent term of the sequence is the previous term squared. For example, the tower sequence with base 2 of 4 terms is 2, 4, 16, 256. Complete the function below so that it returns the sum of a tower sequence with a base of b with n terms:

```
def sumTower(b, n):

    total = 0               # 2 pts
    for i in range(n):      # 3 pts
        total += b          # 4 pts
        b *= b              # 4 pts
    return total            # 2 pts
```

4) (10 pts) What is the output of the following Python program?

```
def main():

    a = 3
    b = 5
    for i in range(5):
        c = 2*b - a
        print(a,b)
        b = c
        a = c - a
main()
```

3 5
4 7
6 10
8 14
12 20

Grading - 1 pt for each number, all or nothing.

5) (15 pts) Complete the segment of code below so that it will draw a green snowman shaped figure of three circles with radii 100, 60 and 20, respectively, from bottom to top.

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

pygame.init()
DISPLAYSURF = pygame.display.set_mode((1000, 600))
pygame.display.set_caption("Drawing!")
black = pygame.Color(0,0,0)
green = pygame.Color(0,255,0)

while True:

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

    DISPLAYSURF.fill(black)

    pygame.draw.ellipse(DISPLAYSURF,green, (400, 400, 200, 200))
    pygame.draw.ellipse(DISPLAYSURF,green, (440, 280, 120, 120))
    pygame.draw.ellipse(DISPLAYSURF,green, (480, 240, 40, 40))
    pygame.display.update()

    # Grading - 5 pts for each draw, 0 pts for update
    # 1 pt general syntax for each call, 4 pts for rect coords
```

6) (20 pts) Complete the program below so that instead of a red ball bouncing off each wall of the screen, the ball that hits the left end of the screen reappears at the right end, and vice versa, and a ball that hits the top of the screen scrolls down to the bottom, and vice versa. The direction of movement of the ball shouldn't change at all after the ball hits a boundary and moves as is described above.

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

pygame.init()
DISPLAYSURF = pygame.display.set_mode((1000, 600))
pygame.display.set_caption("Testing!")

black = pygame.Color(0,0,0)
red = pygame.Color(255,0,0)
clock = pygame.time.Clock()

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
    y += dy

    if x < 0:
        x += 1000
    elif x > 1000:
        x -= 1000
    elif y < 0:
        y += 600
    elif y > 600:
        y -= 600

    # Grading - 5 pts for each case. 2 pts if, 3 pts adjustment.




    DISPLAYSURF.fill(black)
    pygame.draw.ellipse(DISPLAYSURF, red, (x, y, 2*r, 2*r), 0)
    pygame.display.update()
    clock.tick(30)
```

7) (15 pts) Consider a program that takes in information about where people are currently located and answers queries about individuals whereabouts. The program will be menu driven, where the user can either select to supply information about the current whereabouts of an individual, or the user can make a query about an individual's whereabouts. We assume that the information is entered in chronological order and that the current whereabouts of an individual is simply the last city she was reported as being in. Complete the program below. No error checking is necessary. Assume all information entered is "valid."

```
def menu():

    print("Please select your choice."))
    print("1. Enter info about an individual's whereabouts.")
    print("2. Ask about an individual's whereabouts.")
    print("3. Quit.")
    choice = int(input(""))

    while choice < 1 or choice > 3:
        print("Sorry that choice is invalid. Try again.")
        print("1.Enter info about an individual's whereabouts.")
        print("2. Ask about an individual's whereabouts.")
        print("3. Quit.")
        choice = int(input(""))

def main():
    choice = menu()

    places = {}                                          # 2 pts
    while choice != 3:

        if choice == 1:

            person = input("Who is the person?\n")       # 3 pts
            location = input("Where is he/she now?\n")   # 3 pts
            places[person] = location                    # 4 pts

        elif choice == 2:

          person = input("Who are you looking for?\n")
          if person in places:
              print(person," is currently at ",places[person],".",sep="")
          else:
              print("Sorry, I don't know where",person,"is located.")

          # Grading - 3 pts for places[person]
          # rest is unnecessary.

        choice = menu()
main()
```

8) (1 pt) What shape is the Oval Office? **Oval (give to all)**