SI@UCF Introduction to Programming in Python Test #2 Solutions
6/26/2019

1) (8 pts) An artistic design is created by taking a rectangular sheet of paper (height by width) and cutting out of it several (number) congruent rectangles of a smaller size (holeHeight by holeWidth). It is guaranteed that it's possible to cut out this many of the smaller sized rectangles. Complete the program below so that it prints out the total area of the original sheet of paper that is left after the holes are cut out.

```
height = int(input("What is the height of the paper?\n"))
width = int(input("What is the width of the paper?\n"))
number = int(input("How many holes are cut out of the paper?\n"))
holeHeight = int(input("What is the height of each hole?\n"))
holeWidth = int(input("What is the width of each hole?\n"))

print("The area of the paper left is",height*width-
number*holeHeight*holeWidth)
```

Grading: 2 pts print, 2 pts h*w, 1 pt subtract, 3 pts for n*hH*hW

2) (15 pts) Pizzarific prices their slices as follows: $4 for a plain cheese slice. The first five toppings are an extra 50 cents each. The next five toppings are 40 cents each. Each additional topping beyond ten is only 25 cents each. Write a complete python program that asks the user to enter the number of toppings they want on their slice and prints out the total cost of their slice of pizza.

```
toppings = int(input("How many toppings on your slice?\n"))
price = 4

if toppings <= 5:
    price = price + toppings*.50
elif toppings <= 10:
    price = price + 5*.50 + (toppings-5)*.40
else:
    price = price + 5*.50 + 5*.40 + (toppings-10)*.25

print("The cost of your slice is",price)
```

Grading: 2 pts prompt, 3 pts formula for toppings <= 5, 4 pts formula for toppings in between 6 and 10, 5 pts for formula for more than 10 toppings, 1 pt print

3) (10 pts) What is the output of the following segment of code in python? (Note: you should have 10 numbers separated by spaces all on one line.

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

5 2 2 8 8 -4 -4 20 20 -28

Grading: 1 pt per number no exceptions

4) (10 pts) Complete the python program below so that it takes the list `values` (assume this list is already populated with integers), creates two new lists, `even` and `odd`, and ***moves*** all of the even numbers in the list `values` into the list `even` and moves all of the odd numbers from the list `values` into the list `odd`.

```
values = []
# Code here that adds integers to list values.

# Write your solution below.
odd = []
even = []

for x in values:
    if x%2 == 0:
        even.append(x)
    else:
        odd.append(x)

values.clear()
```

Grading: 2 pts for creating each list, 3 pts for the for loop, 4 pts for placing each item in the appropriate list, 1 pt for clearing the values list.

5) (15 pts) Consider writing a pygame program that moves a sprite left and right on the screen. The sprite should wrap around the screen so that it is always visible to the player. The sprite is a file ("sprite.gif") that needs to be loaded into the program for use. Complete the framework below to achieve this task.

```
import pygame, sys
from pygame.locals import *
pygame.init()

width = 1000 # Dimensions of screen
height = 600
DISPLAYSURF = pygame.display.set_mode((width, height))

# Import image
sprite = pygame.image.load("sprite.gif") # 2 pts
# Scale image to 64x64
sprite = pygame.transform.scale(sprite, (64, 64)) # 3 pts

dx = 0
posX = 500
posY = 500

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

         # Get keyboard input
        if event.type == KEYDOWN: # 2 pts
            if event.key == K_a:
                dx = -10 # Move sprite left by 10, 1 pt
            if event.key == K_d:
                dx = 10 # Move sprite right by 10, 1 pt

        if event.type == KEYUP:
            dx = 0 # Stop sprite from moving, 1 pt

    posX = posX + dx # Update position of sprite, 1 pt

    # Wrap sprite around screen
    if posX >= width:                                 # 1 pt
        posX = 0                                      # 1 pt
    elif posX <= 0:                                   # 1 pt
        posX = width                                  # 1 pt

    DISPLAYSURF.fill(pygame.Color(0, 0, 0))
    DISPLAYSURF.blit(sprite, (posX, posY))
    pygame.display.update()
```

6) (20 pts) Write a function that takes in a single positive odd integer, n, greater than one and a character c, and prints out a bowtie of the desired size with the desired character. Here is a bowtie of size 5 made of the character '$':

```
$   $
$$ $$
$$$$$
$$ $$
$   $
```

Formally, the bowtie has n rows and is symmetric about column number (n+1)/2. To the left of this column (and including it), the design is a sideways pyramid, which you previously completed for a homework assignment. Alternatively, we can describe the left side of the design as having i left-justified characters on row number i, for i ranging from 1 to (n+1)/2. After that point, each subsequent row has one fewer character than the previous one. **Note: Due to the complexity of this problem, please use the multiplication operator to "multiply" a string several times.** Complete the function below to print out this design:

```
def bowtie(n,c):

    numCh = 1                           # 9 pts top portion
    numSp = n-2                         # 2 pts correct # lines
    for i in range((n-1)//2):           # 2 pts correct pattern
        print(c*numCh,end="")           # 3 pts correct copies c
        print(" "*numSp,end="")         # 2 pts correct spaces
        print(c*numCh,end="")
        print()

        numCh += 1
        numSp -= 2

    print(c*n)                          # 2 pts middle line

    numCh = (n-1)//2
    numSp = 1

    for i in range((n-1)//2):           # 9 pts bottom portion
                                        # 2 pts correct # lines
        print(c*numCh,end="")           # 2 pts correct pattern
        print(" "*numSp,end="")         # 3 pts correct copies c
        print(c*numCh,end="")           # 2 pts correct spaces

        print()

        numCh -= 1
        numSp += 2
```

7) (15 pts) For this question we define a valid n-permutation to be of any ordering of n integers which contains each of the integers from the set {0, 1, 2, …, n-1} exactly once. For example, [3, 0, 2, 1] is a valid 4-permutation. Complete the function below, which takes in a list, `vals`, and returns true if it's a valid permutation of length `len(vals)` and returns false otherwise. (Note: There are many ways to solve this problem. One of the easier ways, in terms of the length of the code, is utilizing the in operator discussed in class.)

```
def isPerm(vals):


    for i in range(len(vals)):          # 3 pts
        if not i in vals:               # 4 pts
            return False                # 5 pts

    return True                         # 3 pts
```

8) (5 pts) In class we discussed using a dictionary to store replacements for censored words as well as storing the number of votes each candidate in an election received. Give another example, different than these and different from a traditional dictionary, where a dictionary in Python would be an appropriate way to store information. (Note: I get to subjectively judge what's different.)

Many examples work. One example would be keeping track of everyone's Social Security Number. The key to the dictionary would be your name and the entry associated with each key would be that person's SSN. (Alternatively, this dictionary could be mapped the other way, keyed by SSN with the entries equal to the names.)

Grading: Generally be lax; make sure it's reasonable for the keys to be unique and be mapped to one item.

9) (2 pt) Topper's Craft Creamery sells $1 cones on campus on Wednesdays (today). What are you allowed to add to the cone if you pay 50 cents more for each one? **Toppings (Give to all)**