A picture containing text, clipart

Description automatically generated



**PYTHON PROGRAMMING**

(355)

**REGIONAL 2024**

**APPLICATION KNOWLEDGE:**

RegionalPython (470 points)

***TOTAL POINTS (470 points)***

**Test Time: 90 minutes**

**Output**

**STEP 1 Item Quantality Entry**

A picture containing text

Description automatically generatedProgram prompts user how many pizza types they want to create, after entry the program confirms the number.

**STEP 2 Three Letter Code Entry**

Text

Description automatically generatedProgram prompts user to create a three character menu code.

**STEP 3 Pizza Name Entry**

A picture containing diagram

Description automatically generatedProgram gives feedback from STEP 2 and asks user to enter in name of each pizza item. In this example there is only 1 pizza item.

**STEP 4 Storage Quantity**

Program asks user to enter quantity to keep in the freezer.

**NOTE:** if the user choose to make more than one pizza item; the program will repeat data entry sections of STEPS 2-4 until all data for the number of pizza items is created. In STEP 5 the program will print out all data.

Text

Description automatically generated**STEP 5 Output of Data Entry 1 Item**

Program gives feedback with all information entered by the user. A sample for a single item entry is provided with the image on the right.

Text

Description automatically generated with medium confidence

**STEP 5 Output of Data Entry 2 Items**

Program gives feedback with all information entered by the user. A sample for two item entries is provided with the image on the left.

**STEP 1 Data Value Error**

The program detects if non whole numbers are entered.

**STEP 1 Range Violation:** The program detects if user enters values beyond 1-10 items.

A picture containing text

Description automatically generated**STEP 2 Range Violation:** user cannot enter in more than three characters for symbol.

**STEP 5 Data Value Error:** the program detects if non whole numbers are entered.

A picture containing chart

Description automatically generated

Your application will be graded on the following criteria:

**Solution and Project**

The project is present on the flash drive \_\_\_\_ 20 pts

The requiredfile is in one folder; the folder name is your contestant ID \_\_\_\_ 10 pts

**Program Execution**

Code copied to USB drive and the program runs from USB \_\_\_\_ 30 pts

*If the program does not execute, then the remaining items in this section receive a score of zero.*

STEP 1: Quantity of simulated pizza items question: accepts a positive whole number

entry, with feedback statement showing how much was entered \_\_\_\_ 10 pts

STEP 1: Quantity of simulated pizza items question: does not accepts out of

range entry gives feedback; forces user to reenter amount \_\_\_\_ 20 pts

STEP 1: Quantity of simulated pizza items question: does not accepts letters

gives feedback; forces user to reenter amount \_\_\_\_ 30 pts

STEP 2: Three letter menu code entry: accepts any three letter, number or

symbol and with feedback statement showing what was entered \_\_\_\_ 20 pts

STEP 2: Three letter menu code entry: does not accepts out of range entry

gives feedback; forces user to reenter amount \_\_\_\_ 20 pts

STEP 3: Pizza name entry: accepts any entry, and shows newly created pizza

name in the total inventory count question \_\_\_\_ 10 pts

STEP 4: Total inventory count: accepts a positive whole number \_\_\_\_ 10 pts

STEP 4: Total inventory count: does not accepts letters; forces user to

reenter amount \_\_\_\_ 30 pts

STEP 5: All pizza objects print with exact output as provided (no partial credit) \_\_\_\_ 30 pts

**Subtotal\_\_\_\_\_\_/240**

**Source Code Review**

*NOTE: you must place the comment flag in front of the comment in your code in order to get credit. The comment flag will precede the explanation. For example, if the flag is SC1, your comment must read as “#SC1…” in front of the part of the code being reviewed. Code must work to get credit.*

SC1: **Pizzas** constructor uses the three required parameters \_\_\_\_ 10 pts

SC2: **Pizzas** creates a random price between 10 and 20 (inclusive) \_\_\_\_ 10 pts

SC3: **Pizzas** method ***getValue*** returns correct calculated value of pizza items \_\_\_\_ 10 pts

SC4: **Pizzas** method ***toString*** returns string with correct object attributes \_\_\_\_ 10 pts

SC5: **Pizzas** price and total value formatted to US currency (no partial credit) \_\_\_\_ 20 pts

SC6: **DRIVER SECTION** uses ***try/except*** to catch entry error for pizza items \_\_\_\_ 40 pts

SC7: **DRIVER SECTION** code to check for rangeentry error for pizza items \_\_\_\_ 20 pts

SC8: **DRIVER SECTION** code to check for pizza symbol size \_\_\_\_ 20 pts

SC9: **DRIVER SECTION** code to enter in pizza name \_\_\_\_ 10 pts

SC10: **DRIVER SECTION** uses ***try/except*** to catch entry error inventory count \_\_\_\_ 40 pts

SC11: **DRIVER SECTION** construct pizza objects and place into data structure \_\_\_\_ 20 pts

SC12: **DRIVER SECTION** code to print all pizza objects from data structure \_\_\_\_ 20 pts

**Subtotal\_\_\_\_\_\_/230**

**Total Points: \_\_\_\_\_\_\_\_\_/470pts**

1 import random  
 2 import math  
 3   
 4 class Pizzas: #SC1  
 5 min = 10  
 6 max = 20  
 7 def \_\_init\_\_(self, MenuCode, MenuName, PizzaCount):  
 8 self.symbol = MenuCode  
 9 self.name = MenuName  
10 self.count = PizzaCount  
11 self.price = math.floor(random.randint(self.min, self.max)) #SC2 #STUDENT\_POINT

12   
13 def getValue(self): #SC3 #STUDENT\_POINT  
14 temp = math.floor(self.price \* self.count \* 100)  
15 temp /= 100  
16 return temp  
17   
18 def toString(self): #SC4 #SC5 #STUDENT\_POINT  
19 return "Menu Code: " + str(self.symbol) + "\nMenu Name: " + str(self.name) + "\nMenu Price: " + "$" + "{0:.2f}".format(self.price) + "\nTotal Pizza Count: " + str(self.count) + "\nTotal Inventory Cost: " + "$" + " {0:.2f}".format(self.getValue())+"\n"

20   
21 ########################  
22 ########################  
23 #DRIVER SECTION  
24   
25 quantity = 0  
26 pizzas = []  
27 controlOuter = True   
28 controlInner = True   
29   
30   
31 while controlOuter:  
32 while controlInner:  
33 try: #SC6  
34 quantity = input("How many simulated frozen pizza items do you want to create? You must enter between 1 and 10.\n")

35 quantity = int(quantity)  
36 break   
37 except ValueError: #SC6 #STUDENT\_POINT  
38 print("WARNING! You cannot enter in letters. Try again.\n")

39   
40 while controlInner: #SC7 #STUDENT\_POINT  
41 if (quantity > 0 and quantity <= 10):  
42 controlOuter = False  
43 break   
44 print("WARNING! Your entered too low or too high for the quantity. Try again please.\n")

45 break  
46   
47 print("You entered: " + str(quantity))  
48 print("\nYou will now create the menu item code for the inventory management system, \nname of the menu item and how much inventory for the pizza to keep in the freezer. " + "\nThe price will be randomly generated.\n")

49 i = 0  
50 n = 1  
51 while (i < quantity):  
52 while True: #SC8 #STUDENT\_POINT  
53 print("\nPlease enter a three letter/number menu code for pizza number "+ str(n) + ": ")

54 tempMenuCode = str(input())  
55 if len(tempMenuCode) == 3:  
56 print("\nYou entered: "+tempMenuCode)  
57 break  
58 else:  
59 print("\nYou entered an incorrect value:")  
60   
61 print("Please enter the pizza name:") #SC9   
62 tempMenuName = str(input())  
63   
64 while True:  
65 try: #SC10 #STUDENT\_POINT  
66 print("Please enter total inventory count of this " + tempMenuName + " pizza to keep in the freezer:")

67 tempPizzaCount = int(input())  
68 break  
69 except ValueError: #SC10   
70 print("WARNING! You cannot enter in letters. Try again.\n")  
71   
72 tempEntry = Pizzas(tempMenuCode, tempMenuName, tempPizzaCount) #STUDENT\_POINT

73 pizzas.append(tempEntry) #SC11 #STUDENT\_POINT  
74 i = i + 1  
75 n += 1  
76 print("\nListed below is your current pizza inventory:")  
77 for object in pizzas: #STUDENT\_POINT #SC12   
78 print(object.toString())  
79