Day5- FundamentalPROGRAMMING FUNDAMENTALS USING PYTHON

Assignment 34

Write a python function, **find\_pairs\_of\_numbers()** which accepts a list of positive integers with no repetitions and returns count of pairs of numbers in the list that adds up to n. The function should return 0, if no such pairs are found in the list.  
  
Also write the pytest test cases to test the program.

|  |  |
| --- | --- |
| **Sample Input** | **Expected Output** |
| [1, 2, 7, 4, 5, 6, 0, 3], 6 | 3 |
| [3, 4, 1, 8, 5, 9, 0, 6], 9 | 4 |

#PF-Assgn-34

def find\_pairs\_of\_numbers(num\_list,n):

#Remove pass and write your logic here

count=0

for x in num\_list:

if num\_list.index(x) == 0:

index= num\_list.index(x)

else:

index=num\_list.index(x)

for y in range(index,len(num\_list)):

if x+num\_list[y]==n:

count+=1

return count

num\_list=[1, 2, 4, 5, 6]

n=6

print(find\_pairs\_of\_numbers(num\_list,n))

Assignment 35:

A teacher is in the process of generating few reports based on the marks scored by the students of her class in a project based assessment.

Assume that the marks of her 10 students are available in a tuple. The marks are out of 25.

Write a python program to implement the following functions:

1. **find\_more\_than\_average()**: Find and return the percentage of students who have scored more than the average mark of the class.
2. **generate\_frequency()**: Find how many students have scored the same marks. For example, how many have scored 0, how many have scored 1, how many have scored 3….how many have scored 25. The result should be populated in a list and returned.
3. **sort\_marks()**: Sort the marks in the increasing order from 0 to 25. The sorted values should be populated in a list and returned.

#PF-Assgn-35

#Global variable

list\_of\_marks=(12,18,25,24,2,5,18,20,20,21)

def find\_more\_than\_average():

#Remove pass and write your logic here

length=len(list\_of\_marks)

avg = sum(list\_of\_marks)/length

count = 0

for i in range(length):

if list\_of\_marks[i] > avg:

count += 1

return ((count/length)\*100)

def sort\_marks():

#Remove pass and write your logic here

return sorted( list\_of\_marks)

def generate\_frequency():

#Remove pass and write your logic here

list1=[]

for i in range(0,26):

list1+=[list\_of\_marks.count(i)]

return list1

print(find\_more\_than\_average())

print(generate\_frequency())

print(sort\_marks())

Assignment 36:

Write a python function, **create\_largest\_number()**, which accepts a list of numbers and returns the largest number possible by concatenating the list of numbers.  
  
**Note**: Assume that all the numbers are two digit numbers.  
  
Also write the pytest test cases to test the program.

|  |  |
| --- | --- |
| **Sample Input** | **Expected Output** |
| 23,34,55 | 553423 |

#PF-Assgn-36

def create\_largest\_number(number\_list):

#remove pass and write your logic here

number\_list.sort()

res = ""

for num in number\_list:

res = str(num) + res

return (int(res))

number\_list=[23,45,67]

largest\_number=create\_largest\_number(number\_list)

print(largest\_number)

Assignment 37:

A teacher is conducting a camp for a group of five children. Based on their performance and behavior during the camp, the teacher rewards them with chocolates.  
Write a Python function to

1. Find the total number of chocolates received by all the children put together.  
   Assume that each child is identified by an id and it is stored in a tuple and the number of chocolates given to each child is stored in a list.
2. The teacher also rewards a child with few extra chocolates for his/her best conduct during the camp.
   * If the number of extra chocolates is less than 1, an error message "Extra chocolates is less than 1", should be displayed.
   * If the given child Id is invalid, an error message "Child id is invalid" should be displayed. Otherwise, the extra chocolates provided for the child must be added to his/her existing number of chocolates and display the list containing the total number of chocolates received by each child.

#PF-Assgn-37

#Global variables

child\_id=(10,20,30,40,50)

chocolates\_received=[12,5,3,4,6]

def calculate\_total\_chocolates():

#Remove pass and write your logic here to find and return the total number of chocolates

sum = 0

for choco in chocolates\_received:

sum = sum + choco

return sum

def reward\_child(child\_id\_rewarded,extra\_chocolates):

#Remove pass and write your logic here

if extra\_chocolates < 1:

print("Extra chocolates is less than 1")

elif child\_id\_rewarded not in child\_id:

print("Child id is invalid")

else:

length = len(child\_id)

for num in range(length-1):

if child\_id\_rewarded == child\_id[num]:

chocolates\_received[num] = chocolates\_received[num] + extra\_chocolates

print(chocolates\_received)

# Use the below given print statements to display the output

# Also, do not modify them for verification to work

#print("Extra chocolates is less than 1")

#print("Child id is invalid")

#print(chocolates\_received)

print(calculate\_total\_chocolates())

reward\_child(20,2)

Assignment 38:

Write a python function, **check\_double(number)** which accepts a whole number and returns True if it satisfies the given conditions.

1. The number and its double should have exactly the same number of digits.
2. Both the numbers should have the same digits ,but in different order.

Otherwise it should return False.

**Example**: If the number is 125874 and its double, 251748, contain exactly the same digits, but in a different order.

#PF-Assgn-38

def check\_double(number):

#Remove pass and write your logic here

iftrue=False

double = number\*2

dstr= str(double)

nstr= str(number)

for i in nstr:

if i in dstr:

iftrue = True

else:

iftrue = False

break

return iftrue

#Provide different values for number and test your program

print(check\_double(125874))

Assignment 39:

A vendor at a food court is in the process of automating his order management system.  
The vendor serves the following menu – Veg Roll, Noodles, Fried Rice and Soup and also maintains the quantity available for each item. The customer can order any combination of items. The customer is provided the item if the requested quantity of item is available with the vendor.

Write a python program which implements the following functions.  
**place\_order(\*item\_tuple)**: This function accepts the order placed by the customer. Consider it to be a variable length argument as each customer may have a different order.  
The function should check whether the items requested are present in the vendor’s menu and if so, it should check whether the requested quantity is available for each by invoking the check\_quantity\_available() method.

The function should display appropriate messages for each item in the order for the below scenarios:

1. When the requested item is not available in vendor’s menu, display <Item Name> is not available
2. When the quantity requested by the customer is not available, display <Item Name> stock is over
3. When the requested quantity of the item is available with the vendor, display <Item Name> is available

**check\_quantity\_available(index,quantity\_requested)**: This function should check whether the requested quantity of the specified item is available. If so, it should reduce the quantity requested from the quantity available for that item and return True. Otherwise, it should return False.  
Test your code by using the given sample inputs.  
Verify your code by using the **2nd** sample input(highlighted) given below:

|  |  |  |
| --- | --- | --- |
| **Sample Input** | | **Expected Output** |
| **Menu and quantity available** | **Items Ordered** |  |
| (Veg Roll, Noodles, Fried Rice , Soup) [2,200,250,3] | Veg Roll,2 Noodles,2 | Veg Roll is available Noodles is available |
| (Veg Roll, Noodles, Fried Rice , Soup) [2,200,3,0] | Fried Rice,2 Soup,1 |  |

#PF-Assgn-39

#This verification is based on string match.

#Global variables

menu=('Veg Roll','Noodles','Fried Rice','Soup')

quantity\_available=[2,200,3,0]

'''This method accepts the item followed by the quantity required by a customer in the format item1, quantity\_required, item2, quantity\_required etc.'''

def place\_order(\*item\_tuple):

#Remove pass and write your logic here

item\_tuple = list(item\_tuple)

while len(item\_tuple) != 0:

index = 0

available = False

for item in menu:

if item == item\_tuple[0]:

available = check\_quantity\_available(index,item\_tuple[1])

if available == True:

print(item\_tuple[0]," is available")

else:

print(item\_tuple[0]," stock is over")

break

index += 1

if len(menu) == index:

print(item\_tuple[0]," is not available")

del item\_tuple[:2]

#Populate the item name in the below given print statements

#Use it to display the output wherever applicable

#Also, do not modify the text in it for verification to work

#print("<item name>is not available")

#print("<item name>stock is over")

#print ("<item name>is available")

'''This method accepts the index position of the item requested by the customer in the quantity\_available list, and the requested quantity of the item.'''

def check\_quantity\_available(index,quantity\_requested):

#Remove pass and write your logic here to return an appropriate boolean value.

if quantity\_available[index] >= quantity\_requested:

quantity\_available[index] = quantity\_available[index] - quantity\_requested

return True

else:

return False

#Provide different values for items ordered and test your program

#place\_order("Veg Roll",2,"Noodles",2)

place\_order("Fried Rice",2,"Soup",1)