CSE 101: Introduction to Computational and Algorithmic Thinking

Stony Brook University

Lab Assignment #8

Spring 2018

Assignment Due: March 30, 2018 by 11:59 pm

Assignment Objectives

This lab assignment will give you some practice with dictionaries and lists.

Getting Started

Visit Piazza and download the "bare bones" file `lab8.py` onto your computer. Open `lab8.py` in PyCharm and fill in the following information at the top:

1. your first and last name as they appear in Blackboard
2. your Net ID (e.g., jsmith)
3. your Stony Brook ID # (e.g., 111999999)
4. the course number (CSE 101)
5. the assignment name and number (Lab #8)

Submit your final `lab8.py` file to Blackboard by the due date and time. Late work will not be graded. Code that crashes and cannot be graded will earn no credit.

Preliminaries

In this lab you will be writing functions that manage rooms at a hotel. We will be using the following room types:

- Normal
- Double
- King
- Suite

Part I: Vacation Cost Calculator (20 points)

You are a travel agent and have been tasked with making reservations for a large group of people with a fixed budget staying at a resort hotel for a week. Each kind of room (Normal, Double, King, Suite) has a fixed price to rent for the week.

Complete the function `vacation_calculator` that takes arguments in the following order:

1. `room_prices`: A dictionary that maps a type of room (a string) to the price of the room (an integer).
2. `budget`: An integer value that represents the maximum amount of money the group will spend for the hotel stay.
3. `reservations`: A list of sub-lists wherein the first element of a sub-list contains the type of room being reserved, and the second element is the desired count of rooms of that type. For example, the sub-list `['King', 3]` would indicate that 3 King rooms are desired.

The function iterates over the `reservations` list, extracting the room type and room count of each reservation. Room types can appear multiple times in the list. Then, the function looks up the price of the desired room using the `room_prices` dictionary and computes what it would cost to reserve the desired number of rooms. You may assume that the room type in the reservation always has a corresponding price in `room_prices`. If there is money left in the budget to accept the reservation, the function (i) appends the sub-list to a *result* list and (ii) subtracts the cost of the reservation from the budget. If a certain reservation in `reservations` list does not fit within the remaining budget, the function skips over that reservation and moves on to the next one in the list.

At the end, the function returns two values in a tuple: the total spent on the vacation, followed by the list of reservations that were accepted by the resort.

Examples:

Function Call	**Return Values**
`vacation_calculator({'Normal': 70, 'Double': 105, 'King': 195, 'Suite': 249}, 543, [['Normal', 1], ['Suite', 2], ['Double', 2], ['Normal', 1], ['King', 3]])`	`(350, [['Normal', 1], ['Double', 2], ['Normal', 1]])`
`vacation_calculator({'Normal': 72, 'Double': 133, 'King': 192, 'Suite': 246}, 255, [['King', 3], ['Suite', 1], ['King', 1], ['Normal', 1], ['King', 2], ['Double', 3], ['Suite', 3]])`	`(246, [['Suite', 1]])`
`vacation_calculator({'Normal': 85, 'Double': 148, 'King': 190, 'Suite': 239}, 319, [['Suite', 1], ['Double', 1], ['Normal', 2], ['Double', 2], ['Double', 2], ['Suite', 2]])`	`(239, [['Suite', 1]])`
`vacation_calculator({'Normal': 99, 'Double': 104, 'King': 194, 'Suite': 275}, 516, [['Normal', 3], ['King', 2], ['Suite', 2], ['Normal', 3], ['Suite', 2], ['Double', 1]])`	`(401, [['Normal', 3], ['Double', 1]])`
`vacation_calculator({'Normal': 90, 'Double': 119, 'King': 158, 'Suite': 287}, 648, [['King', 2], ['Double', 1], ['Double', 3], ['King', 3], ['Double', 1]])`	`(554, [['King', 2], ['Double', 1], ['Double', 1]])`
`vacation_calculator({'Normal': 98, 'Double': 132, 'King': 169, 'Suite': 220}, 225, [['King', 1], ['King', 2], ['King', 2], ['King', 3], ['Double', 3], ['Double', 1]])`	`(169, [['King', 1]])`

`vacation_calculator({'Normal': 91, 'Double': 113, 'King': 181, 'Suite': 243}, 331, [['Suite', 3], ['Double', 3], ['Double', 2], ['Double', 3]])`	`(226, [['Double', 2]])`
`vacation_calculator({'Normal': 76, 'Double': 124, 'King': 173, 'Suite': 291}, 0, [['Double', 1], ['King', 2], ['Double', 2]])`	`(0, [])`
`vacation_calculator({'Normal': 78, 'Double': 148, 'King': 183, 'Suite': 213}, 541, [])`	`(0, [])`
`vacation_calculator({'Normal': 61, 'Double': 115, 'King': 196, 'Suite': 212}, 0, [])`	`(0, [])`

Part II: Let Me Inn (20 points)

You are the manager of the Let Me Inn. A travel agent has sent you a list of reservations that can be accommodated by the hotel. Your primary job is to send him back the total cost of the reservations. To make sure your hotel runs properly, you also will keep track of how much money you have made from each type of room (Normal, Double, King, Suite).

Complete the function `hotel_manager` which takes arguments in the following order:

1. `room_prices`: A dictionary that maps a type of room (a string) to the price of the room (an integer).
2. `room_counts`: A dictionary that maps a type of room (a string) to the number of available rooms of that type (an integer).
3. `reservations`: A list of sub-lists wherein the first element of a sub-list contains the type of room being reserved, and the second element is the desired count of rooms of that type. For example, the sub-list `['King', 3]` would indicate that 3 King rooms are desired.

The function iterates over the `reservations` list, extracting the room type and room count of each reservation. Room types can appear multiple times in the list. Then, the function looks up the price of the desired room using the `room_prices` dictionary and determines what it would cost to reserve the desired number of rooms. You may assume that the room type in the reservation always has a corresponding price in `room_prices` and a corresponding count in `room_counts`.

If there are enough available rooms to accept the reservation, the function (1) updates the total income generated by the reservation, (2) updates `room_counts` to reflect that fewer rooms are now available, and (3) updates a dictionary that tracks how much income was generated by each type of room. As an example, suppose that a King room costs $200 per night and 3 have such rooms been successfully reserved. Suppose also that we have created a dictionary called `room_incomes` to store the income generated by each room type. In this example, the function would add `600` both to `room_incomes['King']` and to the total income generated. It would also subtract 3 from `room_counts['King']`.

If a certain reservation in `reservations` list cannot be accommodated due to a shortage of rooms, the function skips over that reservation and moves on to the next one in the list.

At the end, the function returns two values in a tuple: the total income generated by the accepted reservations, followed by the updated `room_incomes` dictionary.

Examples:

Function Call	Return Values
hotel_manager({'Normal': 70, 'Double': 105, 'King': 195, 'Suite': 249}, {'Normal': 13, 'Double': 9, 'King': 5, 'Suite': 3}, [['Suite', 1], ['King', 3], ['Double', 6], ['King', 5], ['King', 3], ['King', 5]])	(1464, {'Suite': 249, 'King': 585, 'Double': 630})
hotel_manager({'Normal': 79, 'Double': 101, 'King': 190, 'Suite': 270}, {'Normal': 12, 'Double': 7, 'King': 7, 'Suite': 3}, [['King', 4], ['Double', 7], ['Suite', 2]])	(2007, {'King': 760, 'Double': 707, 'Suite': 540})
hotel_manager({'Normal': 60, 'Double': 128, 'King': 178, 'Suite': 226}, {'Normal': 11, 'Double': 8, 'King': 5, 'Suite': 4}, [['Suite', 1], ['King', 5], ['Suite', 2], ['Normal', 9]])	(2108, {'Suite': 678, 'King': 890, 'Normal': 540})
hotel_manager({'Normal': 78, 'Double': 106, 'King': 193, 'Suite': 239}, {'Normal': 12, 'Double': 10, 'King': 6, 'Suite': 3}, [['Suite', 3], ['Double', 5], ['King', 3], ['Suite', 2], ['King', 4], ['Double', 5], ['Double', 7]])	(2356, {'Suite': 717, 'Double': 1060, 'King': 579})
hotel_manager({'Normal': 97, 'Double': 144, 'King': 185, 'Suite': 242}, {'Normal': 11, 'Double': 7, 'King': 7, 'Suite': 4}, [['Normal', 10], ['King', 3], ['King', 4], ['King', 4]])	(2265, {'Normal': 970, 'King': 1295})
hotel_manager({'Normal': 73, 'Double': 145, 'King': 160, 'Suite': 295}, {'Normal': 11, 'Double': 8, 'King': 7, 'Suite': 3}, [['King', 5], ['Double', 5], ['Suite', 1], ['Double', 6], ['Double', 7], ['Suite', 3]])	(1820, {'King': 800, 'Double': 725, 'Suite': 295})
hotel_manager({'Normal': 73, 'Double': 145, 'King': 162, 'Suite': 215}, {'Normal': 11, 'Double': 7, 'King': 6, 'Suite': 4}, [['Normal', 11], ['Normal', 11], ['Suite', 1], ['Suite', 1], ['Double', 7], ['Double', 5], ['Suite', 0]])	(2248, {'Normal': 803, 'Suite': 430, 'Double': 1015})
hotel_manager({'Normal': 62, 'Double': 100, 'King': 164, 'Suite': 252}, {'Normal': 0, 'Double': 0, 'King': 0, 'Suite': 0}, [])	(0, {})
hotel_manager({'Normal': 60, 'Double': 124, 'King': 155, 'Suite': 296}, {'Normal': 12, 'Double': 8, 'King': 5, 'Suite': 4}, [])	(0, {})
hotel_manager({'Normal': 60, 'Double': 107, 'King': 155, 'Suite': 299}, {'Normal': 0, 'Double': 0, 'King': 0, 'Suite': 0}, [])	(0, {})

How to Submit Your Work for Grading

To submit your `.py` file for grading:

1. Login to Blackboard and locate the course account for CSE 101.
2. Click on "Assignments" in the left-hand menu and find the link for this assignment.
3. Click on the link for this assignment.
4. Click the "Browse My Computer" button and locate the `.py` file you wish to submit. Submit only that one `.py` file.
5. Click the "Submit" button to submit your work for grading.

Oops, I messed up and I need to resubmit a file!

No worries! Just follow the above directions again. We will grade only your last submission.