Justin Chang Pg 1

Lab 7

Mar 17, 2020

Goal: Practice extracting data from files. Build a program that reads a bill of materials and generates an order that orders all materials you need at the cheapest price.

Part 1: Parse Bill of Materials (BOM)

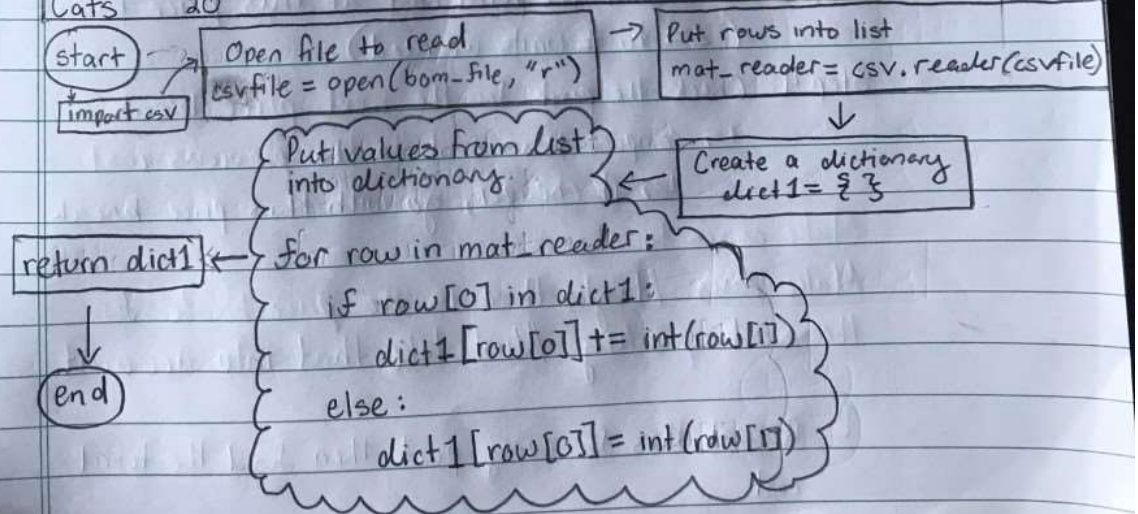
Problem: Function: parse_bom(bom_file), reads a file and return a dict.
In the dictionary:
Keys: Part number (string)
Values: Minimum number of units. (integers)

Test cases:

Sample Table:		>>> parse_bom("sample_table.csv")
Blue	12	{"Blue": 12, "Red": 20, "Green": 3}
Red	16	
Green	3	
Red	4	

Sample Table 2:		>>> parse_bom("sample_table_2.csv")
Dogs	3	{"Dogs": 3, "Cats": 26, "Bears": 5}
Cats	6	
Bears	5	
Cats	20	

Flow



Justin Chang Pg 2 Mar 17, 20

Problem

Part 2: Generate Order

Function: generate_order(bom, part_cost_filename)

Bom: a dictionary similar to the return dict of parse_bom

parts_~~file~~cost_filename: a string that contains the filename of a csv file that contains the pricing info for each part number.

Return:

1. a float containing total cost of order (2 decimal)
2. a dictionary containing order quantities for each part.
   Keys: Part numbers (strings)
   Values: Quantity to order (integers)

Method:

Step 1: Find the row of the desired part number
↳ can use a for loop

Step 2: Find the closest price break equal to or less than the number of units we need.
↳ use a for loop for the columns of that row.

Step 3: Determine the total cost at this price break (<= to # units we need)

Step 4: Check the cost for the next price break
↳ compare this cost to the previous cost
↳ if cheaper, go with this price break

Make cases. Maybe there is no price break for the next quantity. Must take that into account

↑ Repeat this for every part, then find the total cost.

est cases.

Flow:

Justin Chang Pg 3

Mar 17, 2026

Test cases.

Input:
bom = {"mat 1": 21, "mat2": 300, "mat3": 12}
pricing_file = "parts db1. csv"
cost, order = generate_order(bom, pricing_file)
print(cost)
print(order)

Output: 153.08 (Float)
{"mat1": 25, "mat3": 12, "mat2": 300}

Flow:

Plan

1. Import csv
2. Create variables: (Potential variables)
   - part_cost (calculates cost of specific part)
   - total_cost quantity.
   - dict? (But can probably just modify bom and return)
   - desired_row, desired_col
3. Make the table into a list using csv.reader(csvfile)
   - Loop through BOM
   - Find the row of desired part (Loop thru list) (loop down rows)
   - Find column of desired part (price break) (loop across list[0])
   - Calculate cost at this price break (row, col)
   - Calculate cost at next price break (row, col+1)
   - Compare.
   - Add to total cost the lower cost
   - add to dictionary the quantity
   - Go to next item in BOM