Part -1 used this for reference :

```
author = 'Alan Doonan'

import random
import time


class Building:
# defines class building
    number_of_floors = 0
# sets number_of_floors variable to 0
    customer_list = []
# creates an empty array for customer_list
    elevator = 0
# sets elevator variable to 0

    def __init__(self, floors, customers):
# initialize Building
        self.number_of_floors = floors
# assigns floors entered to number_of_floors
        for customerID in range(1, customers + 1):
# assigns number of customers entered to customer_list in order
            new = Customer(customerID,self.number_of_floors)
# creates an instance called new of Customer class for number of customers entered in
input
            self.customer_list.append(new)
# appends new instance of customer to customer_list
        self.customer_list.sort(key = lambda x: x.current_floor)
# sorts customer_list by current_floor customer is on
# prints
        self.elevator = Elevator(floors,self.customer_list)
# creates instance of elevator with inputted floors and assigns customer_list to
register_list
# prints
        self.run()
# runs run method below

    def run(self):
# method to operate the elevator
        print('++++++++++++++++++++++++++ELEVATOR IS NOW STARTING+++++++++++++++')
# prints
        print('There are %d customers in the building' % (len(self.customer_list)))
# prints
        number_of_customers = len(self.customer_list)
# assigns current number of customers to number_of_customers variable
        self.output()
# runs output method below

    def output(self):
        for customer in self.customer_list:
#prints lists of customers in building and their details
            print("Customer",customer.customerID,"is on
floor",customer.current_floor,"and wants to go to",customer.destination_floor)

        #ELEVATOR MOVING UP LOOP
        while (self.elevator.current_floor < self.elevator.number_of_floors):
            self.elevator.current_floor +=1
            print('ELEVATOR MOVING UP')
            print(len(self.customer_list),'Customers in lift.')
            print('++++++++++++++++++++++++++++++++++++++++++++++++++')
            print('FLOOR',self.elevator.current_floor)
```

```
            for customer in self.customer_list:
# Loop for each instance of Custumer in customer_list
                if (self.elevator.current_floor == customer.current_floor) &
customer.customer_direction == 1:
                    customer.in_elevator = True
                    print('Customer',customer.customerID,'has entered the lift')
                if (self.elevator.current_floor == customer.destination_floor) &
(customer.in_elevator == True) & customer.customer_direction ==1:
                    customer.in_elevator = False
                    self.customer_list.remove(customer)
                    print(customer.customerID,'has reached their destination')


        #ELEVATOR MOVING DOWN LOOP
        while (self.elevator.current_floor <= self.number_of_floors) &
(self.elevator.current_floor > 1):
            self.elevator.current_floor -= 1
            print(len(self.customer_list),'Customers in lift.')
            print('ELEVATOR MOVING DOWN')
            print('++++++++++++++++++++++++++++++++++++++++++++++++++')
            print('FLOOR',self.elevator.current_floor)

            for customer in self.customer_list:
                if (customer.in_elevator == True):
                    customer.current_floor = self.elevator.current_floor
                if (self.elevator.current_floor == customer.destination_floor) &
(customer.in_elevator == True) & (customer.customer_direction == -1):
                    customer.in_elevator = False
                    self.customer_list.remove(customer)
                    print('Customer',customer.customerID,'has reached their destination')

        print('There are',len(self.customer_list),'trapped in the elevator')
#prints
        print('There are',len(Elevator.register_list),'people left on the register')
        print('Elevator run is done!!!')
#prints

        print('CUSTOMERS STUCK IN LIFT ARE BELOW')
        for stuck in self.customer_list:
            print('Cust. ID:',stuck.customerID,'Dest.
Floor:',stuck.destination_floor,'Curr. Floor:',stuck.current_floor,'In
Elevator',stuck.in_elevator,'Direction',stuck.customer_direction)


class Elevator:
    number_of_floors = 0
# the number of floors
    register_list = []
# the list of customers in the elevator
    current_floor = 0
# the current floor of the elevator
    up = 1
# moves the elevator up
    down = -1
# moves the elevator down

    def __init__(self, number_of_floors, register_list):
        self.number_of_floors = number_of_floors
        self.register_list = register_list

    def move(self):
# method to move the elevator by 1 floor
        pass;
```

```
    def register_customer(self, customers):
# customer goes into elevator
        for reg in customers:
            self.register_list.append(reg)



    def cancel_customer(self, customers):
# customer goes out of the elevator
        pass;


class Customer:
    current_floor = 0
# the current floor of the elevator
    destination_floor = 0
# the destination floor of the elevator
    customerID = 0
# the customers ID
    in_elevator = False
# denotes whether customer is in the elevator
    finished = False
# denotes whether customer has reached the destination floor
    customer_direction = 0

    def __init__(self, customerID, floors):
# initilize Customer class
        self.customerID = customerID
# assigns self.customerID to customerID
        self.current_floor = random.randint(1, floors)
# assigns self.current_floor to random int between 1 and floors entered
        self.destination_floor = random.randint(1, floors)
# assigns seslf.destination_floor to random int between 1 and floors entered
        while self.destination_floor == self.current_floor:
            self.destination_floor = random.randint(1, floors)
        if self.current_floor < self.destination_floor:
            self.customer_direction = 1
        else:
            self.customer_direction = -1

def header():
# elevator animation at beginning of program
    print("                          ELEVATOR OPENING                          ")
    time.sleep(.2)
    print("++++++++++++++++++++++++++++++||++++++++++++++++++++++++++++++++++")
    time.sleep(.2)
    print("+++++++++++++++++++++++++|       |++++++++++++++++++++++++++++++++")
    time.sleep(.2)
    print("+++++++++++++++|                                 |+++++++++++++++++++")
    time.sleep(.2)
    print("++++++|                                                    |+++++++++")
    time.sleep(.2)
    print("                                                                  ")
    time.sleep(.2)
print("                          ELEVATOR CLOSING                          ")
    time.sleep(.2)
    print("++++++|                                                    |+++++++++")
    time.sleep(.2)
    print("+++++++++++++++|                                 |+++++++++++++++++++")
    time.sleep(.2)
    print("+++++++++++++++++++++++++|       |++++++++++++++++++++++++++++++++")
    time.sleep(.2)
```

```
    print("++++++++++++++++++++++++++++||++++++++++++++++++++++++++++++++++")


def main():
# main method
    try:
# try/except for user input menu
        floors = int(input('Enter the number of floors: '))
# enter floors and assign to floors
        customers = int(input('Enter number of customers: '))
# enter customers and assign to customers
        building = Building(floors, customers)  # instance of building created with
inputs of floors and customers      # create instance of Building class (building)
    except ValueError:
        print('YOU DIDNT ENTER A NUMBER. START AGAIN.')
        main()


if __name__ == "__main__":
    # header()
    main()
```