IC152: Assignment 9-10

OOP, Stacks and Queues

You must keep filenames as mentioned in the assignment since the assignments will be auto evaluated via a script. VERY IMPORTANT: Save all python files and outputs in a single folder. You must name python files as suggested in each question. Do not miss writing your roll number in the folder name of the zip file. Example folder name: b23000_assignment10.zip.

Problem 1: Pandas

Save the code in this question for all parts in file named problem1.py

The Cars93 dataset contains data from 93 cars that were on sale in the USA in 1993. The dataset has 93 rows listing various car models along with their features and specifications.

The columns of this dataset---there are 27 of them---comprise various attributes, e.g., Manufacturer, Type,

Price, mileage, horsepower etc. Refer to the Appendix (at the end of this assignment) for more information.

.....

.....

Getting started with pandas:
https://pandas.pydata.org/docs/getting_started/index.html
API reference:
https://pandas.pydata.org/docs/reference/index.html
# Reading the data from a csv file and storing it in a pandas dataframe
import pandas as pd
df = pd.read_csv("data/Cars93.csv")

.....

.....

Open data/Cars93.csv and observe its contents.

a. List all the models mentioned in the Cars93 data set in alphabetical order. Save them in ‘problem1a.csv’ with 2 columns: column 1 should have row numbers like input file, and column 2 should have sorted model names.
   Hints (print each, then choose the appropriate ones):
   df.sort_values(['col1', 'col2'])
   df.sort_values('col1', ascending=False)
   df['col1'].sort_values()

df['col1'].unique()
df.to_csv('problem1a.csv')
First four rows of 'problem1a.csv' is given in the table below, and remaining rows should follow similar pattern (i.e. unique elements in sorted order with new row numbers):-

	Model Sorted
1	100
2	190E
3	240

b. Print all details of the costliest car of each of the 'Types'.
Hints (print each, then choose the appropriate ones):
df.groupby('col_i')
df.groupby('Type').get_group('small')
df[df['col_i'] == df['col_i'].max()]

c. Write a function that asks you to enter the name of a manufacturer and print all the models produced by that manufacturer. Take the first command line argument of 'problem1.py' as the name of the manufacturer.

Hint: df[df[‘col1’]== ‘string1’]

d. Write the total count of cars per manufacturer in ‘problem1d.csv’ in the same format as that of problem1a.csv but with an extra column with heading ‘count’.
   Hints:
   df.groupby('col1').size()
   df.reset_index(name='col2')

Problem 2: Object Oriented Programming (OOP), Stacks and Queues

a. Create a class IC152 which captures the name, roll no and marks obtained by a student for IC152.
   - The Code takes as inputs: the name, roll no and marks of the student.
   - Before you create a object, make sure that the object is of proper form by checking the following things
     - Check the roll no to be of the form B23***
     - Check the marks to be in the range of 1 to 100.
   - Your code should take the input and if the input is valid, output “pass” (without quotes) if the student

has secured at least 33 marks and output “fail” if the student has secured less than 33 marks.
- If the input is invalid, output “Invalid Input”.
- Save the code for this question in file named problem2a.py

b. Create a class with a name “stack”, with stack_list (name of a list) and element as attributes. Write three methods under the stack class for push(), pop(), and peek() operations. Stack is a limited access data structure that follows LIFO (last in first out) rule.
   - The push() method should push its input at the end/top of stack_list.
   - The pop() method should remove the last/topmost element of stack_list, update it in the element attribute and return it as output.
   - The peek() method should just update the last/topmost element of stack_list in the element attribute and return it as output.
   - IMPORTANT: You are NOT allowed to use any of the list methods. You can use loops and indexing.
   - Test your code by giving different examples.
   - Save the code for this question in file named problem2b.py

c. Create a class with a name queue, with queue_list (name of a list) and element as attributes. Write three methods under the queue class for push(), pop(), and peek()

operations. The queue is a limited access data structure that follows FIFO (first in first out) rule.

- The push() method should push its input at the end/top of queue_list.
- The pop() method should remove the first/lowest element of queue_list, update it in the element attribute and return it as output.
- The peek() method should just update the first/lowest element of queue_list in the element attribute and return it as output.
- IMPORTANT: You are NOT allowed to use any of the list methods. You can use loops and indexing.
- Test your code by giving different examples.
- Save the code for this question in file named problem2c.py

Problem 3: Extra Question on Application of Stacks

- Given a string of single-digit integers, binary operands and brackets/parentheses, find out if the brackets/ parentheses are balanced.
- In a stack, one can add and remove elements only from the top. They are referred to as push and pop operations. Hence, while performing a pop, the element which is retrieved is the one which was pushed the last. In other words, last-in-first-out.
- A stack can be used for finding out if the parentheses are balanced in the input string, as follows. Each time an opening parenthesis is encountered in the input string,

push it on the stack. Each time a closing parenthesis is encountered, pop an element from the stack and check if the opening and closing parentheses are of matching type. If not, then you know that the string is unbalanced. Integers and binary operands may be ignored. If at the end, there are extra opening/closing parenthesis, then again, you know that the string is unbalanced. Return 1 if the string is balanced and 0 otherwise.

- E.g.
  1 ∗ (2 + 3 ∗ [4 − 5]) should return 1.
  1 ∗ (2 + 3 ∗ [4 − 5}) should return 0.
  1 ∗ (2 − 3 should return 0.
- Modify the program created in q1b to achieve the above task. Save your code in problem2.py.

Create the folder having your python files and inputs/outputs, with name having your roll number followed by “_assignment10” (don’t use inverted commas in folder name), compress the folder with .zip extension and submit it on moodle.

Appendix

Source:

https://stat.ethz.ch/R-manual/R-devel/library/MASS/html/Cars93.html (retrieved on 31 Dec'22)

Details:

Cars93 {MASS} R Documentation

Data from 93 Cars on Sale in the USA in 1993

Description

The Cars93 data frame has 93 rows and 27 columns.

Usage

Cars93

Format

This data frame contains the following columns:

Manufacturer

Manufacturer.

Model

Model.

Type

Type: a factor with levels "Small", "Sporty", "Compact", "Midsize", "Large" and "Van".

Min.Price

Minimum Price (in $1,000): price for a basic version.

Price

Midrange Price (in $1,000): average of Min.Price and Max.Price.

Max.Price

Maximum Price (in $1,000): price for “a premium version”.

MPG.city

City MPG (miles per US gallon by EPA rating).

MPG.highway

Highway MPG.

AirBags

Air Bags standard. Factor: none, driver only, or driver & passenger.

DriveTrain

Drive train type: rear wheel, front wheel or 4WD; (factor).

Cylinders

Number of cylinders (missing for Mazda RX-7, which has a rotary engine).

EngineSize

Engine size (litres).

Horsepower

Horsepower (maximum).

RPM

RPM (revs per minute at maximum horsepower).

Rev.per.mile

Engine revolutions per mile (in highest gear).

Man.trans.avail

Is a manual transmission version available? (yes or no, Factor).

Fuel.tank.capacity

Fuel tank capacity (US gallons).

Passengers

Passenger capacity (persons)

Length

Length (inches).

Wheelbase

Wheelbase (inches).

Width

Width (inches).

Turn.circle

U-turn space (feet).

Rear.seat.room

Rear seat room (inches) (missing for 2-seater vehicles).

Luggage.room

Luggage capacity (cubic feet) (missing for vans).

Weight

Weight (pounds).

Origin

Of non-USA or USA company origins? (factor).

Make

Combination of Manufacturer and Model (character).

Details

Cars were selected at random from among 1993 passenger car models that were listed in both the *Consumer Reports* issue and the *PACE Buying Guide*. Pickup trucks and Sport/Utility vehicles were eliminated due to incomplete information in the *Consumer Reports* source. Duplicate models (e.g., Dodge Shadow and Plymouth Sundance) were listed at most once.

Further description can be found in Lock (1993).

Source

Lock, R. H. (1993) 1993 New Car Data. *Journal of Statistics Education* **1**(1). doi:10.1080/10691898.1993.11910459

References

Venables, W. N. and Ripley, B. D. (2002) *Modern Applied Statistics with S-PLUS.* Fourth Edition. Springer.