Laboratorio di Programmazione

Compito d'esame per l'appello del 24/02/2022

Una *time series* (univariata) è una sequenza di coppie di valori ordinate temporalmente, dove il primo elemento della coppia corrisponde al tempo, mentre il secondo è il valore di una qualche quantità atta a descrivere un certo fenomeno.
Il file data.csv contiene la time series del numero totale mensile in migliaia di passeggeri su linee aeree internazionali, da Gennaio 1949 a Dicembre 1960. Il primo elemento di ogni riga rappresenta la data ed è in formato `Anno-Mese`. Il dato si presenta così:

```
date,passengers
1949-01,112
1949-02,118
1949-03,132
...
```

ovvero, messa sotto forma di tabella per comodità:

date	passengers
1949-01	112
1949-02	118
1949-03	132
...	...

Vogliamo leggere questo tipo di dati e capire se dati due anni consecutivi c'è stata una variazione nel numero di passeggeri tra coppie di mesi *quasi* uguale, da un'anno all'altro.
In altre parole, vogliamo calcolare la variazione tra ogni coppia di mesi, e paragonarla con la variazione della stessa coppia di mesi dell'anno seguente, valutando se questa variazione è uguale, con una tolleranza di ±2.
Per esempio, prendiamo in considerazione i primi due anni (1949 e 1950). Possiamo creare una lista di 12 elementi per ogni anno, dove l'elemento all'indice `i` corrisponde al numero di passeggeri per il mese `i+1` in quell'anno (all'indice 0 avremo il valore per il mese 1, Gennaio):

1949: [112,118,132,…]
1950: [115,126,141,…]

Per rispondere alla nostra domanda, iniziamo con il calcolare la differenza per ogni coppia di mesi. Consideriamo solo le coppie ordinate, ovvero Gennaio e Marzo non le consideriamo una coppia. Per esempio, per l'anno 1949 e 1950:

1949: [118-112, 132-118,…] = [6, 14, …]
1950: [126-115, 141-126,…] = [11, 15, …]

Dove il numero "6" è il risultato della differenza tra Gennaio e Febbraio, "14" il risultato della differenza tra Febbraio e Marzo, e così via.
Poi, confrontiamo i valori ottenuti per ogni coppia di mesi tra i due anni. In questo esempio, dobbiamo confrontare se 6 e11 sono simili con una tolleranza di ±2 (in questo caso no), e procedere con la coppia di mesi seguenti, controllando se 14 e 15 sono simili (ed in questo caso si).

Informazioni sullo svolgimento

Create la classe `CSVTimeSeriesFile`, modificando o estendendo la classe `CSVFile` vista a lezione (oppure scrivendola da zero). La classe deve essere istanziata sul nome del file tramite

la variabile `name` e deve avere un metodo `get_data()` che torni una lista di liste, dove il primo elemento delle liste annidate è la data ed il secondo il numero di passeggeri.
Questa classe si dovrà quindi poter usare così:

```
time_series_file = CSVTimeSeriesFile(name='data.csv')
time_series = time_series_file.get_data()
```

...ed il contenuto della variabile `time_series` tornato dal metodo `get_data()` dovrà essere così strutturato (come lista di liste):

```
[
  [“1949-01”, 112],
  [“1949-02”, 118],
  [“1949-03”, 132],
  ...
]
```

Per rilevare dove c’è stata una variazione nel numero mensile di passeggeri quasi uguale tra coppie di mesi, dovete creare una funzione a sé stante (cioè definita NON nella classe `CSVTimeSeriesFile` ma direttamente nel corpo principale del programma), di nome `detect_similar_monthly_variations`, che avrà come input primario la time series e che verrà usata così:

```
detect_similar_monthly_variations(time_series, years)
```

..e dove `years` è una lista di due elementi corrispondenti ad i due anni consecutivi da valutare, ad esempio: `years = [1949,1950]`.
La funzione dovrà ritornare (tramite un `return`) una lista di 11 elementi (poiché con 12 mesi ho 11 coppie), dove ogni elemento sarà `True` se la variazione è *simile* tra quella coppia di mesi per i due anni consecutivi o `False` altrimenti, ovvero:

```
[
  False,                # la coppia gennaio-febbraio non ha variazione simile nei due anni
  True,                 # la coppia febbraio-marzo ha variazione simile nei due anni
  ...
]
```

Il file in cui scrivere il vostro codice deve chiamarsi **"esame.py"** e le eccezioni da alzare in caso di input non corretti o casi limite devono essere istanze di una specifica classe `ExamException`, che dovete definire nel codice come segue, senza modifica alcuna (copia-incollate le due righe):

```
class ExamException(Exception):
    pass
```

...e che poi userete come una normale eccezione, ad esempio:

```
raise ExamException('Errore, lista valori vuota')
```

Qualche informazione in più sulle specifiche e qualche e suggerimento:

- Dovete tenere in considerazione che possono esserci dei dati mancanti! Nelle misurazioni di un anno, può mancare il numero di passeggeri per uno o più mesi, ma assumiamo di avere almeno una misurazione all’anno.
- I dati nel file CSV sono solo un’esempio: potreste avere anche dati dell’anno scorso, per capirci.
- Come accennato, per un anno potrebbe mancare il numero di passeggeri per uno o più mesi. In questo caso, assumere che la differenza nel numero di passeggeri per questi mesi tra i due anni NON sia mai simile (quindi False).
- Attenzione che gli anni presi come input dalla funzione devono essere validi: per esempio se ci interessano gli anni 1950 e 1951, questi devono essere presenti nei dati, altrimenti va alzata un’eccezione.
- I valori che leggete dal file CSV sono da aspettarsi di tipo intero positivo. Un valore non numerico, oppure vuoto o nullo o negativo non deve essere accettato, ma tutto deve procedere comunque senza alzare eccezioni.

- La serie temporale nel file CSV è da considerare sempre ordinata, se per caso ci dovesse essere un timestamp fuori ordine va alzata un'eccezione (dentro la funzione `get_data()`) senza cercare di riordinare la serie. Stesso discorso se c'è un timestamp duplicato: si alza un'eccezione.
- Il file CSV può contenere letteralmente di tutto. Da linee incomplete a pezzi di testo che non c'entrano niente, e ogni errore *salvo quello di un timestamp fuori ordine o duplicato* va ignorato (ovvero, ignoro la riga contenente l'errore e vado a quella dopo). Nota: se riuscite a leggere due valori (data e temperatura) ma c'è un campo di troppo sulla stessa riga, questo non è da considerarsi un'errore e non bisogna ignorare quella riga.
- La classe `CSVTimeSeriesFile` controlla l'esistenza del file solo quando viene chiamato il metodo `get_data()` e, nel caso il file non esista o non sia leggibile, alza un'eccezione.

Informazioni sulla consegna

La consegna dell'esame deve avvenire tassativamente entro l'ora di inizio dell'appello orale, e può avvenire in due modi: allegando lo script esame.py, oppure indicando il commit hash da valutare su un repository GitHub, entrambi mandati via mail a stefano.russo@gmail.com, come descritto più in dettaglio sul repository del corso: https://github.com/sarusso/ProgrammingLab

Attenzione: se il file non si chiama "esame.py", se le eccezioni alzate in caso di errori non sono di tipo "`ExamException`" o se le classi ed i metodi non si chiamano come indicato da specifiche, l'esame non potrà essere valutato!

Nota benissimo: il vostro file deve includere SOLO una classe, una funzione ed una eccezione, non deve includere nessun codice "main" o tantomeno chiedere input all'utente.

Se non vi ricordate bene come si usa Git, potete fare riferimento al tutorial dei tutor che è disponibile qui: https://github.com/drpOpZ/proglab2021-tutors/blob/master/git_quickstart.md.