Laboratorio di Programmazione

Compito d’esame per l’appello del 14/7/2022

Una [*time series*](https://www.google.com/url?q=https://it.wikipedia.org/wiki/Serie_storica&sa=D&source=editors&ust=1658485611761199&usg=AOvVaw3iyEm-9D8A1bM-YxuOGSX2) (univariata) è una serie di coppie di punti dove il primo elemento della coppia è un istante di tempo, anche detto *timestamp,* ed il secondo è il valore di una qualche quantità relativa a quell’istante, come ad esempio la temperatura.

Il timestamp può essere rappresentato in vari formati, uno dei più comuni in informatica è l’*epoch*, che sta a rappresentare il numero di secondi passati dalla mezzanotte (00:00) del  primo Gennaio 1970 sulla timezone UTC (ovvero il meridiano fondamentale di Greenwich, senza cambi di ora legale). All’una di notte del primo Gennaio 1970 il timestamp epoch vale quindi “3600” secondi, alle due vale “7200” secondi, e così via.

Ad esempio, se facciamo finta di essere il 2 Febbraio 2021 alle 16:20 in Italia, ovvero alle 15:20 UTC, l’epoch vale “1612279200” secondi. Tra un minuto questo epoch sarà incrementato di 60 secondi (e varrà “1612279260”). E domani alla stessa ora sarà incrementato di 86400 secondi (60 secondi volte 60 minuti volte 24 ore) e varrà quindi “1612365600”. Questo sito fornisce un rapido modo di convertire una data e ora in epoch e viceversa: [epochconverter.com](https://www.google.com/url?q=https://www.epochconverter.com/&sa=D&source=editors&ust=1658485611762409&usg=AOvVaw1PrOHIDmG-eL06GArkzCib).

Il file [**data.csv**](https://www.google.com/url?q=https://raw.githubusercontent.com/sarusso/ProgrammingLab/main/data.csv&sa=D&source=editors&ust=1658485611762964&usg=AOvVaw2IHaxU5EoTaWpnh9STTgoO) contiene la time series della temperatura dell’interno di un’abitazione, registrata una volta all’ora per un mese (Marzo 2019), e si presenta così:

epoch,temperature

1551398400,21.50

1551402000,21.40

1551405600,21.30

...

ovvero, messa sotto forma di tabella per comodità:

|  |  |
| --- | --- |
| epoch | temperature |
| 1551398400 | 21.50 |
| 1551402000 | 21.40 |
| 1551405600 | 21.30 |
| ... | ... |

**Vogliamo leggere questo tipo di dati e calcolare, giorno per  giorno, la *differenza massima* tra le rilevazioni della temperatura (detta anche escursione termica).**Con questo si intende che se ad esempio in una giornata ho i valori di temperatura 25, 28, 29 e 27, la differenza massima è 4, ovvero la differenza tra il 25 ed il 29 (che sono minimo e massimo, rispettivamente). Per calcolarla si devono quindi processare tutte le misurazioni orarie appartenenti ad un dato giorno ed implementare tale logica, ripetendo il procedimento per ogni giorno della time series.

Informazioni sullo svolgimento

Non c’è un limite minimo o massimo a quanti dati ci sono nel file CSV. Potrebbe esserci una misurazione soltanto, una manciata di misurazioni tutte appartenenti alla stessa giornata, oppure misurazioni per decine di giornate o anche mesi e mesi di dati.

Un punto chiave dello svolgimento sta nel capire, dato il timestamp epoch delle misurazioni, se quest’ultime appartengono alla stessa giornata (non a ***quale*** giornata, ma solo se alla ***stessa*** giornata) e gestire correttamente il tutto.

Possono esserci dei dati mancanti, quindi attenzione a come gestite le aggregazioni giornaliere ed ai conti che fate. Tuttavia, se mancano dati, questi non mancano mai per più di una giornata (ovvero, ***per ogni giornata è presente almeno una misurazione di temperatura***).

Lavoriamo esclusivamente sulla timezone UTC (dove non si considera mai il passaggio all’ora legale, che invece nella realtà è parecchio fastidioso).

Alla luce di tutto questo, 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 l’epoch ed il secondo la temperatura.

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):

    [

      [1546819200, 22.78],

      [1546819800, 22.84],

      [1546820400, 22.85],

      ...

    ]

Per calcolare la differenza massima giornaliera, dovete invece creare una funzione a sé stante (cioè posizionata non nella classe CSVTimeSeriesFile ma direttamente nel corpo principale del programma), di nome compute\_daily\_max\_difference, che avrà come input la time series e che verrà usata così:

    compute\_daily\_max\_difference(time\_series)

..e che dovrà ritornare (tramite un return) in output una lista di elementi, dove ogni elemento rappresenta la differenza massima (escursione termica) di una giornata del dataset:

    [

      differenza\_massima\_giorno\_1,

      differenza\_massima\_giorno\_2,

      differenza\_massima\_giorno\_3,

      ...

    ]

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:

* Attenzione che la differenza massima di un solo elemento **non è definita**! In tal caso (cioè nel caso che ci sia una sola misurazione di temperatura per un dato giorno) va aggiunto alla lista dei risultati il valore “None” per quella giornata.
* I timestamp epoch che leggete da file sono di tipo intero. Se per caso dovessero esserci dei timestamp epoch floating point, vanno convertiti silenziosamente ad interi (tramite cast diretto con int(), non tramite arrotondamento) e tutto deve procedere comunque.
* I valori di temperatura che leggete dal file CSV sono da aspettarsi di tipo numerico (intero o floating point), un valore di temperatura non numerico, oppure vuoto o nullo  
  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 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 (epoch 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.
* Se leggete correttamente una serie temporale dal file CSV, questa (come già accennato) è assicurato che abbia almeno una misurazione di temperatura per giorno. In pratica, non dovete preoccuparvi di avere “buchi” di giornate intere nei dati.
* 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.
* Suggerimento: per trovare l’inizio di un giorno dato un timestamp epoch, potete usare l’operazione [modulo](https://www.google.com/url?q=https://it.wikipedia.org/wiki/Operazione_modulo&sa=D&source=editors&ust=1658485611776147&usg=AOvVaw0uoXsvcH6Bnf_kyGV0n9HI) che calcola il resto di una divisione, e sottrarlo al timestamp stesso, in questo modo: day\_start\_epoch = epoch - (epoch % 86400)
* Attenzione alle condizioni nel controllo dell’appartenenza di un timestamp epoch ad un determinato giorno: la mezzanotte appartiene sempre al giorno dopo, anche sull’orologio che avete sul polso / cellulare / comodino!

Informazioni sulla consegna

La consegna va effettuata il giorno dell’appello condividendo via email il file o il link al repository e l’hash da valutare.

Più info qui: [https://github.com/sarusso/ProgrammingLab#modalità-di-esame](https://www.google.com/url?q=https://github.com/sarusso/ProgrammingLab%23modalit%25C3%25A0-di-esame&sa=D&source=editors&ust=1658485611777438&usg=AOvVaw3Kxhvg191JXrGzvbwOBeUA)

*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!*