PROGRAMMING PRINCIPLES - 02 JAN 25

Lo scopo della prova è **creare una classe**.

La classe, dovrà chiamarsi *Virgilio*
e dovrà essere definita all'interno del file *virgilio.py*.
La classe dovrà avere un attributo di istanza chiamato *directory,*
inizializzato nel metodo costruttore.
L'attributo potrà esser valorizzato, in fase di creazione oggetto,
col percorso assoluto della **cartella** del vostro pc,
nella quale risiedono i 34 Canti.

Il cuore della prova sono i **metodi** della classe.
La prova è infatti suddivisa in 18 esercizi,
ciascuno riguardante la creazione di un metodo
o la modifica di un metodo esistente.

Ogni esercizio che completerete correttamente
vi conferirà un massimo di 3 punti.
Il punteggio massimo della prova è 50 punti.
Ma ho voluto darvi un aiutino:
se provate a fare 18x3 ottenete un po' più di 50;
in questo modo, anche errando qualche esercizio,
potrete comunque raggiungere il punteggio massimo.

Prima di elencarvi gli esercizi, alcuni suggerimenti e raccomandazioni:

- Assicuratevi di assegnare il **nome corretto** a ciascun **metodo** e ai relativi **parametri**;
- Assicuratevi che ogni metodo **restituisca** il **valore** corretto, col **datatype** corretto;
- Una volta creata la classe potrete testarla istanziandone un oggetto e chiamandone i metodi, sullo stesso file virgilio.py o su un file a parte.
  Tuttavia, i vostri test non incideranno sul punteggio,
  quel che conta è solo la classe e la corretta definizione e funzionamento dei metodi;
- Nessun metodo richiede stampe mediante funzione print,
  ma potete liberamente usarle per testare la classe;
- Per il voto di ciascun esercizio si terrà conto sia del corretto funzionamento del metodo che dell'applicazione delle linee guida di **formattazione** e **documentazione**.
  Assicuratevi perciò che il codice
  sia sufficientemente documentato e formattato (non serve la perfezione :)).
  Opzionalmente, aiutatevi con flake8, autopep8 e ruff,
  ma se quest'ultimo rileva errori complessi non perdeteci tempo.
- Per risolvere gli esercizi vi saranno sufficienti i comandi riportati nelle **slides**.
  E' comunque consentito fare ricerche online
  di comandi e informazioni che vi agevolino nella risoluzione.
  Tuttavia, vi sconsiglio caldamente di copiare porzioni di codice
  o generarle mediante Intelligenza Artificiale,
  poiché verrebbero individuate dai nostri strumenti di rilevamento AI.
- Se create per bene il metodo *read_canto_lines*,
  potrete utilizzarlo all'interno degli altri metodi, facilitandovi il lavoro.
  Idealmente, l'operazione di apertura e lettura file,
  dovrebbe avvenire soltanto all'interno di tale metodo.
- In generale, spesso e volentieri potrete riutilizzare,
  all'interno dei metodi che implementate,
  altri metodi che avete già scritto in precedenza.
  Riutilizzare correttamente il codice
  vi potrebbe fruttare qualche punto in più,
  e sicuramente vi agevolerà nella risoluzione degli esercizi.

ESERCIZI:

1. Crea un metodo chiamato *read_canto_lines* che ha il parametro *canto_number*.
   Il metodo legge uno dei 34 file di testo rappresentanti i Canti.
   Il numero del canto da leggere è rappresentato da *canto_number*.

   Il metodo inizialmente calcola il percorso assoluto del file da leggere (*file_path*).
   Potete calcolarlo con un comando a vostra scelta
   Vi suggerisco però di utilizzare il seguente, che genera il percorso unendo la cartella e il nome del file, in modo coerente col sistema operativo su cui vi trovate:
   file_path = os.path.join(self.directory, f"Canto_{canto_number}.txt")
   (se lo usate, ricordate di importare il modulo *os*!)

   Il metodo, una volta ottenuto il percorso, lo usa per aprire in lettura il file.
   Quindi, **restituisce una lista contenente tutte le righe del file**
   (c'è un metodo di lettura apposito!).

2. Crea un metodo chiamato *count_verses* che ha il parametro *canto_number*.
   Il metodo restituisce il numero di versi (righe) del Canto specificato da *canto_number*.

3. Crea un metodo chiamato *count_tercets* che ha il parametro *canto_number.*
   Il metodo restituisce il numero (**int**) di terzine del Canto scelto.
   Per terzina si intende un gruppo di tre versi.
   Se un canto ha un numero di versi non divisibile per 3,
   i versi in eccesso non vanno conteggiati (si approssima per difetto).

4. Crea un metodo chiamato *count_word*
   che ha il parametro *canto_number* ed il parametro *word*.
   Il metodo restituisce il numero di volte che la parola *word* si presenta nel Canto scelto.
   Non importa se si tratta di una parola completa o solo di una lettera o carattere.
   Il metodo è case sensitive.

   Suggerimento: per questo ed altri metodi di conteggio,
   potrebbe aiutarti fare prima un join di tutte le righe lette,
   per ottenere una stringa sulla quale effettuare poi il conteggio richiesto.

5. Crea un metodo chiamato *get_verse_with_word*
   che ha il parametro *canto_number* ed il parametro *word*.
   Il metodo restituisce il primo verso (partendo dall'inizio) del Canto scelto,
   in cui è presente la parola *word*.
   Il metodo è case sensitive.

6. Crea un metodo chiamato *get_verses_with_word*
   che ha il parametro *canto_number* ed il parametro *word*.
   Il metodo restituisce una **lista** di tutti i versi del Canto scelto
   in cui è presente la parola *word*.
   Il metodo è case sensitive.

7. Crea un metodo chiamato *get_longest_verse* che ha il parametro *canto_number*.
   Il metodo restituisce il verso più lungo del Canto scelto.
   Se ci sono più versi con la stessa lunghezza,
   restituisce il primo che trova partendo dall'inizio del Canto.

8. Crea un metodo chiamato *get_longest_canto*.
   Il metodo restituisce un **dizionario** con due coppie chiave-valore.
   La chiave *canto_number* ha come valore un numero intero,
   che rappresenta il Canto con più versi dell'intero Inferno.
   La chiave *canto_len* ha come valore un numero intero,
   che rappresenta quanti versi possiede tale Canto.

9. Crea un metodo chiamato *count_words* che ha il parametro *canto_number*
   ed il parametro *words*.
   Tale parametro è una **lista** di parole da conteggiare all'interno del Canto scelto.
   Il metodo deve infatti restituire un **dizionario** dove ogni coppia chiave-valore
   ha come chiave la parola
   e come valore il numero di volte che essa si presenta nel Canto.
   Il metodo è case sensitive.

10. Crea un metodo chiamato *get_hell_verses* che restituisce una **lista**
    contenente **tutti i versi** di tutti e 34 Canti i dell'Inferno,
    in ordine dal primo all'ultimo.

11. Crea un metodo chiamato *count_hell_verses* che restituisce
    un **int** rappresentante il numero totale di versi contenuti nell'Inferno.

12. Crea un metodo chiamato *get_hell_verse_mean_len* che restituisce
    un **float** rappresentante la lunghezza media dei versi dell'inferno.
    Il verso deve essere privato di spazi o caratteri affini
    presenti all'inizio e alla fine.
    Gli spazi fra le parole invece, non devono essere eliminati,
    ma conteggiati anch'essi.

13. Modifica il metodo *read_canto_lines* aggiungendo
    il parametro opzionale booleano *strip_lines*,
    avente *False* come valore di default.
    Nel caso in cui *strip_lines* venga valorizzato a *True,* il metodo
    prima di restituire la lista di righe, effettua uno *strip* di ciascuna di esse,
    al fine di rimuovere i caratteri a capo *\n*
    e qualsiasi altro spazio (o carattere affine) all'inizio e alla fine della riga.

14. Modifica il metodo *read_canto_lines* aggiungendo
    il parametro opzionale intero *num_lines*,
    avente *None* come valore di default.
    Nel caso in cui *num_lines* venga valorizzato,
    il metodo legge solo un numero di righe pari ad esso,
    partendo dall'inizio del Canto scelto.

15. Modifica il metodo *read_canto_lines* affinché
    verifichi prima di tutto se *canto_number* è un numero intero,
    in altre parole se è un'istanza della classe int (questo è un indizio :P).
    Se tale condizione **non è** rispettata,
    viene sollevata l'eccezione *TypeError*, col seguente messaggio:
    canto_number must be an integer

16. Modifica il metodo *read_canto_lines* affinché, verifichi se *canto_number* ha un valore incompatibile col numero dei Canti (che va da 1 a 34)
    In tal caso solleva un'eccezione personalizzata chiamata *CantoNotFoundError*, che ha come messaggio: *canto_number must be between 1 and 34*.
    Tale eccezione va definita **all'interno** della classe *Virgilio*.

17. Modifica il metodo *read_canto_lines* affinché, in fase di apertura file, gestisca qualsiasi eccezione che possa verificarsi.
    Se un'eccezione si verifica, il metodo deve restituire, invece che la solita lista, il messaggio seguente:
    *error while opening **file_path***
    dove *file_path* è la variabile definita al punto 1.

18. Modifica il metodo *count_words* affinché, oltre restituire il dizionario, lo serializzi in un file json, che dovrà essere chiamato *word_counts.json* e dovrà essere creato all'interno della *directory* contenente i Canti.

Completare il primo esercizio è fondamentale per poter risolvere i successivi.
Tutti gli altri esercizi sono opzionali, ma provare a risolverli in ordine potrebbe semplificarvi l'impresa