VERIFICA INFORMATICA
Classe 4 C INF

29 Aprile 2024

1. GESTORE MAGAZZINO

Crea una classe GestoreMagazzino che gestisca un magazzino di prodotti. La classe dovrà avere i seguenti attributi:
Un dizionario “prodotti” che mappa i nomi dei prodotti ai rispettivi oggetti “Prodotto” (che descriverai in seguito)
Una variabile “costo_magazzinaggio” che indica il costo per immagazzinare ogni prodotto per un mese
La classe dovrà avere i seguenti metodi:
Un metodo “aggiungi_prodotto” che aggiunga un nuovo prodotto al magazzino
Un metodo “rimuovi_prodotto” che rimuova un prodotto dal magazzino
Un metodo “calcola_costi_magazzinaggio” che calcoli i costi di magazzinaggio per tutti i prodotti presenti nel magazzino
Crea inoltre una classe Prodotto che abbia gli attributi “nome”, “prezzo” e “scorta”.

2. ESERCIZIO

E’ un giorno importante oggi: la classe ha appena svolto un test di matematica. Ti viene fornita una lista di voti. Realizza una funzione che:

- calcola la media dei voti dell’intera classe arrotondandola alla terza cifra decimale.

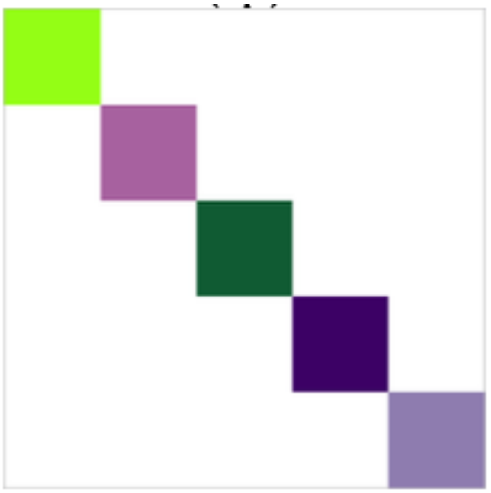
- crea un dizionario con le chiavi “h”, “a”, “l” per chiarire quanti voti alti (***high***), medi (***average***) e bassi (***low***) sono stati assegnati. I voti alti sono 9 e 10, i voti medi sono 7 e 8, i voti bassi sono quelli da 1 a 6.

- Ritorna la lista [media_classe, dictionary] se ci sono categorie di voti differenti oppure la lista [media_classe, dictionary, “La classe è stata brava”] se ci sono solo voti alti (high).

  Esempi:
  [10, 9, 9, 10, 9, 10, 9] ==> [9.429, {'h': 7, 'a': 0, 'l': 0}, 'La classe è stata brava']

  [5, 6, 4, 8, 9, 8, 9, 10, 10, 10] ==> [7.9, {'h': 5, 'a': 2, 'l': 3}]

3. ESERCIZIO



Chiedere all'utente un numero n.
Su un canvas 500x500 disegnare n quadrati tutti delle stesse dimensioni, posizionati lungo tutta la diagonale e ciascuno con un colore casuale
Determinare il lato di ogni quadrato in base alla dimensione del canvas

4. ESERCIZIO

Data una lista di numeri interi, scrivere una funzione che restituisca una nuova lista di lunghezza *num* che contiene gli ultimi *num* numeri pari della lista originaria (nello stesso ordine). La lista originaria non sarà vuota e conterrà almeno *num* numeri pari (non vi è dunque necessità di gestire eventuali possibili errori).

Esempio:
numPari([1, 2, 3, 4, 5, 6, 7, 8, 9], 3) => [4, 6, 8] numPari([-22, 5, 3, 11, 26, -6, -7, -8, -9, -8, 26], 2) => [-8, 26] numPari([6, -25, 3, 7, 5, 5, 7, -3, 23], 1) => [6]