Esercitazione Python.1-4

I punteggi per gli esercizi pratici sono riportati accanto al titolo degli esercizi, per le domande a risposta multipla ogni risposta corretta vale 0.2 punti. Il voto minimo per passare l'esame è 6/10.
Importante: leggere bene il testo dell'esercizio!

Nome:

Cognome:

Corso:

Data:

<u>Tempo a disposizione 3 ore</u>

Classe AppointmentScheduler - PUNTI 2

Progetta una classe AppointmentScheduler per gestire un insieme di appuntamenti.

Attributi:

- `appointments: dict[str, dict]`: il dizionario `appointments` contiene tutti gli appuntamenti. In dettaglio, la chiave è un `app_id` (stringa) e il valore è un altro dizionario che per ogni appuntamento ha le seguenti coppie chiave, valore:
  - Chiave: `str = "data" e Valore: str = “Una data dell'appuntamento…”`
  - Chiave: `str = "programmato"`: e Valore: `bool = True` oppure `False`

Funzioni:

- `schedule_appointment(app_id: str, data: str) -> dict | str`
  Se `app_id` esiste già: restituisci "Errore: appuntamento esiste già.", altrimenti aggiungi il nuovo appuntamento (con `programmato=True`) e restituisci un dizionario con il solo appuntamento appena creato.
- `reschedule_appointment(app_id: str, nuova_data: str) -> dict | str`
  Se non esiste restituisci: "Errore: appuntamento non trovato.", altrimenti aggiorna la data e restituisci un dizionario con il solo appuntamento aggiornato.
- `cancel_appointment(app_id: str) -> dict | str`
  Se non esiste restituisci: "Errore: appuntamento non trovato.", altrimenti imposta `programmato=False` e restituisci un dizionario con il solo appuntamento aggiornato.
- `remove_appointment(app_id: str) -> dict | str`
  Se non esiste restituisci: "Errore: appuntamento non trovato.", altrimenti rimuovi e restituisci un dizionario con il solo appuntamento rimosso.
- `list_appointments() -> list[str]`
  Restituisce la lista di tutti gli `app_id`.
- `get_appointment(app_id: str) -> dict | str`
  Restituisce il `dict` dell'appuntamento o "Errore: appuntamento non trovato."

Sistema di Gestione di un Cinema - PUNTI 2

Implementa tre classi interagenti per gestire biglietti di film.

Classe Ticket:

Attributi

- `ticket_id: str`
- `movie: str`
- `seat: str`
- `is_booked: bool`

Metodi:

- `book() -> None`: se `"is_booked"` è `False`, lo imposta a `True`; altrimenti stampa "Il biglietto per '{self.movie}' posto '{self.seat}' è già prenotato."
- `cancel() -> None`: se `"is_booked"` è `True`, lo imposta a `False`; altrimenti stampa "Il biglietto per '{self.movie}' posto '{self.seat}' non risulta prenotato."

Classe Viewer:

Attributi

- `viewer_id: str`
- `name: str`
- `booked_tickets: list[Ticket]`

Metodi:

- `book_ticket(ticket: Ticket) -> None`: se `"ticket.is_booked"` è `False`, aggiunge `"ticket"` a `"booked_tickets"` e chiama `"ticket.book()"`; altrimenti stampa "Il biglietto per '{ticket.movie}' non è disponibile."
- `cancel_ticket(ticket: Ticket) -> None`: se `"ticket"` è in `"booked_tickets"`, lo rimuove e chiama `"ticket.cancel()"`; altrimenti stampa "Il biglietto per '{ticket.movie}' non è stato prenotato da questo spettatore."

Classe Cinema:

Attributi

- `tickets: dict[str, Ticket]`
- `viewers: dict[str, Viewer]`

Metodi:

- `add_ticket(ticket_id: str, movie: str, seat: str) -> None`: se `"ticket_id"` esiste: stampa "Il biglietto con ID '{ticket_id}' esiste già.", altrimenti aggiunge un nuovo `"Ticket"`.
- `register_viewer(viewer_id: str, name: str) -> None`: se `"viewer_id"` esiste: stampa "Lo spettatore con ID '{viewer_id}' è già registrato.", altrimenti aggiunge un nuovo `"Viewer"`.
- `book_ticket(viewer_id: str, ticket_id: str) -> None`: se entrambi esistono, invoca `"viewer.book_ticket(ticket)"`; altrimenti stampa "Spettatore o biglietto non trovato."
- `cancel_ticket(viewer_id: str, ticket_id: str) -> None`: se entrambi esistono, invoca `"viewer.cancel_ticket(ticket)"`; altrimenti stampa "Spettatore o biglietto non trovato."
- `list_available_tickets() -> list[str]`: restituisce la lista di `"ticket_id"` con `"is_booked == False"`.
- `list_viewer_bookings(viewer_id: str) -> list[str] | str`:
  - se lo spettatore esiste, restituisce lista di `\"ticket_id\"` prenotati;
  - Altrimenti restituisce "Errore: spettatore non trovato."

Filtra e Concatena Numeri - PUNTI 1

Scrivi una funzione con il seguente header:

`filter_and_concat(nums: list[int], min_val: int) -> str`che prenda una lista di interi e un valore minimo, e restituisci una stringa concatenata di tutti i numeri di `nums` che sono maggiori di `min_val`, separati da virgola (es. "5,7,9").

Calcola Deviazione Standard - PUNTI 1

Scrivi una funzione con il seguente header:

`calculate_std_dev(nums: list[float]) -> float` che, data una lista di numeri, ritorni la deviazione standard (radice quadrata della varianza). Se la lista è vuota, solleva un'eccezione `ValueError` con messaggio "lista vuota". Attenzione: non usare funzioni built-in di python o librerie.calculate_std_dev(nums: list[float]) -> float che, data una lista di numeri, ritorni la deviazione standard (radice quadrata della varianza). Se la lista è vuota, solleva un'eccezione ValueError con messaggio "lista vuota". Attenzione: non usare funzioni built-in di python o librerie.

Nota Bene: *Il calcolo della varianza misura la dispersione dei dati rispetto alla media. Si calcola come la media dei quadrati delle differenze tra ciascun valore all'interno della lista e la media dei valori della lista di numeri*

Esempio:

Se `nums = [1.0, 2.0, 3.0, 4.0, 5.0]`:

1. **Calcolo della media:**
   (1.0 + 2.0 + 3.0 + 4.0 + 5.0) / 5 = 15.0 / 5 = 3.0
2. **Calcolo della varianza:**
   ((1.0 - 3.0)^2 + (2.0 - 3.0)^2 + (3.0 - 3.0)^2 + (4.0 - 3.0)^2 + (5.0 - 3.0)^2) / 5
   = ((-2.0)^2 + (-1.0)^2 + (0.0)^2 + (1.0)^2 + (2.0)^2) / 5
   = (4.0 + 1.0 + 0.0 + 1.0 + 4.0) / 5
   = 10.0 / 5 = 2.0
3. **Calcolo della deviazione standard:**
   radice quadrata di 2.0 ≈ 1.41421356

Controllo Sicurezza - PUNTI 1

Scrivi una funzione che verifica se una combinazione di sensori (S1, S2, S3) attiva l'allarme. L'allarme si attiva solo se S1 è vero e (S2 o S3 è falso). La funzione deve ritornare "Allarme attivato" oppure "Nessun allarme" a seconda delle condizioni.

`check_security_alarm(s1: bool, s2: bool, s3: bool) -> str`