Time-Aware Tourist Itinerary Planner: Documentation

Overview

This project implements a time-constrained itinerary planning algorithm for tourists. It smartly selects places to visit in a day based on:

- Opening and closing hours of each location
- Hourly footfall (crowd density)
- Travel time between places
- Time budget (max total hours available to the tourist)

The goal is to create a plan that maximizes time utilization while avoiding crowded or closed destinations.

Key Concepts

Place

A data class that holds metadata for a location:

- name: string
- opening_time, closing_time: integers (in 24-hour format)
- footfall: dict[day][hour] → crowd level
- distance_time: dict[place] → {distance, time}

Footfall

The expected number of people at each location at a specific hour. Lower footfall is preferable for a better tourist experience.

Time Budget

Maximum total hours (including wait and travel time) that the tourist is allowed to spend.

Code Breakdown

Place Class

Stores data and contains helper methods:

- __post_init__() ensures midnight closure is treated as 24:00.
- get_busy_percentage(day, hour): Returns crowd level.
- is_open_at(hour): Checks if place is open at a given time.

visit_cost()

Calculates cost of visiting a place at a specific hour:

- Penalizes early arrivals (wait time)
- Returns inf if the place is closed
- Otherwise returns footfall as the cost

select_starting_place()

Finds the best starting location based on opening hours, footfall, and proximity from the user's current location.

calculate_travel_time_minutes()

Returns rounded travel time between two places in minutes.

plan_next_place()

Returns the list of next reachable, unvisited places.

log_itinerary_step()

Adds a visit record to the itinerary including time, crowd level, and reason for leaving.

to_time_str()

Formats float hours into HH:MM format for readability.

suggest_itinerary()

The core engine:

- Starts from the closest open place
- Iteratively chooses the next least busy location
- Adds stay and travel time to total_hours_spent
- Accounts for wait time if the place isn't yet open
- Skips locations that are either too busy or closed

- Stops when time budget is exhausted
- Tracks skipped places with reasons

Output Structure

Itinerary

A list of dicts for each place visited:

- place
- arrival_time
- footfall_at_arrival
- stay_duration
- leave_time
- reason_for_leaving

Skipped Places

A dictionary:

```
{
  "Phoenix mall": "Insufficient time within time budget",
  "KLING": "Closed at time of arrival"
}
```

Example Usage

```
itinerary, skipped = suggest_itinerary(
  places, "sunday", 11,
  user_stay_durations,
  current_place_distances,
  max_total_hours=6
)
```

Limitations & Future Work

- No support for multiple days or re-visits
- Assumes static travel time (not time-dependent traffic)
- Places not listing each other in distance_time are unreachable

Possible Enhancements

- Priority scoring for must-visit locations
- Export to CSV/JSON
- Map visualization with Folium
- Time-dependent travel durations