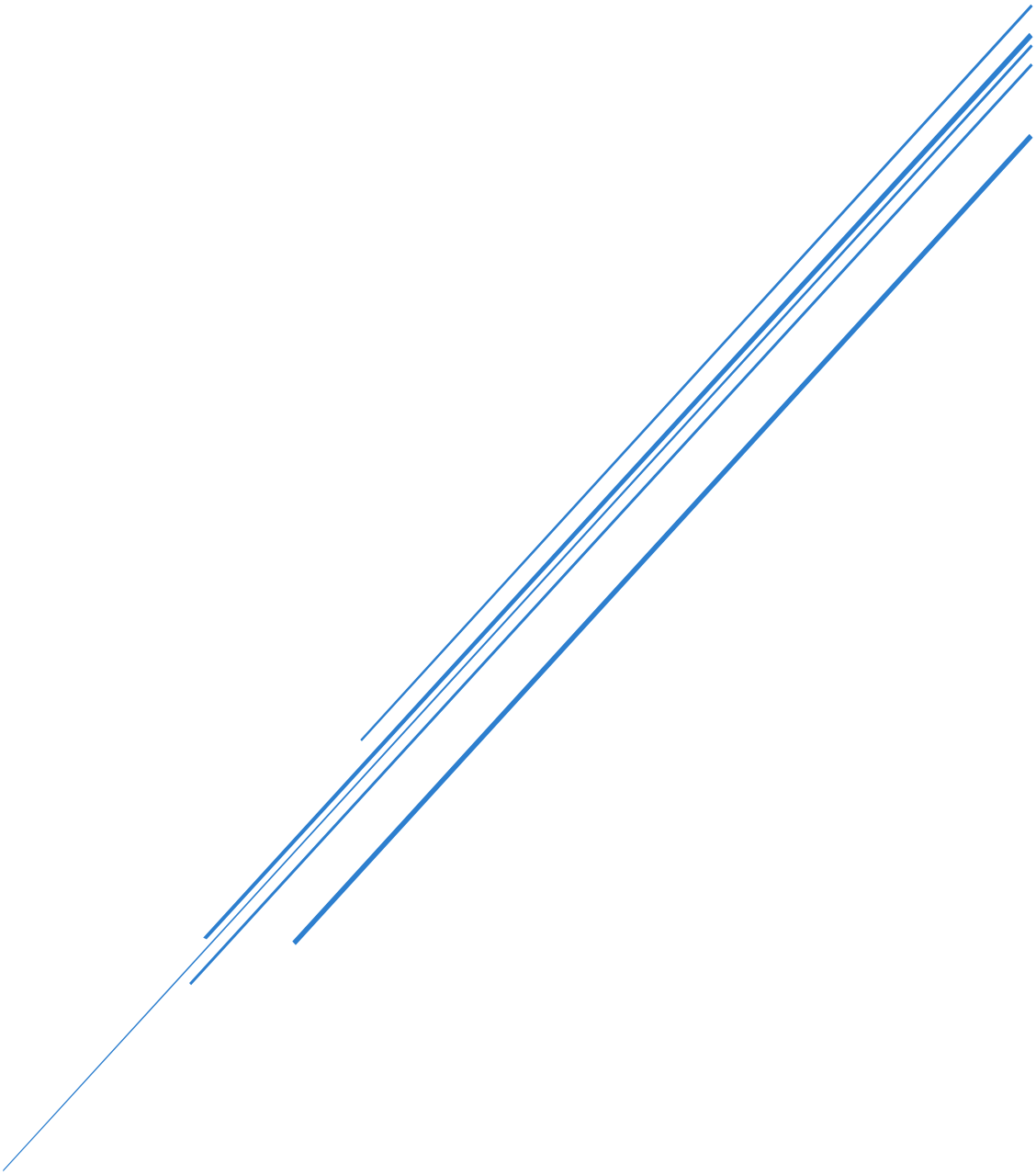
HABIT TRACKER APPLICATION

Documentation version 1.0



James Antony Das

OOPF

Contents

Class HabitPeriods(Enum)

Custom Enum for tracking periodicities in Habit tracker Application.

Attributes	
	HabitPeriods.DAILY = 'daily' Daily periodicity **HabitPeriods.WEEKLY = 'weekly'** Weekly periodicity **HabitPeriods.MONTHLY = 'monthly'** Monthly periodicity **HabitPeriods.YEARLY = 'yearly'** Yearly periodicity

Class HabitError(Exception)

Custom Exception for the Habit Tracker Application.

Class Habit

```
class Habit(
        name: str,
        description: str = '',
        periodicity: HabitPeriods = HabitPeriods.DAILY,
        creation_date: str = '',
        history: str = ''
)
```

Each user defined habit is represented by an instance of the Habit class.

Parameters	**name: str** Name of the Habit Object **description: str** Additional Description of the Habit Object **periodicity: HabitPeriods** Accepts HabitPeriods as periodicity (Default is daily) **creation_date: str** The date of creation in "%Y-%m-%d" format (Default is current date) **history: str** A string of dates in "%Y-%m-%d,%Y-%m-%d" format :returns: The function returns nothing
Attributes	**name**: str **description**: str **periodicity**: HabitPeriods **creation_date**: datetime.datetime **habit_history**: list

Methods

habit_streak() method

```
def habit_streak(self):
```

Calculates the habit Streak of the Habit Object

Parameters	The function accepts no parameters
Returns	The current habit streak and longest habit streak.
Raises	**HabitError**: Raises exceptions when there is an irregularity in history

habit_Streak() method

```
def habit_streak(self):
```

Calculates Longest habit breaks and the total number of habit breaks in the given Habit object.

Parameters	The function accepts no parameters
Returns	total number of check offs, total duration of the habit till today

habit_stats() method

```
def habit_stats(self):
```

Calculates the total number of check off entries, and total duration

Parameters	The function accepts no parameters
Returns	total number of check offs, total duration of the habit till today.
Raises	Habit Error when irregularities found in the habit history

Is_checked_off() method

```
def is_checked_off(self):
```

Checks to see if the given Habit is checked off for today.

Parameters	The function accepts no parameters
Returns	True if the habit is checked off for today, False otherwise.

check_off() method

```
def check_off (self):
```

Checks off the given Habit Object for today (for the current duration)

Parameters	The function accepts no parameters
Returns	True if habit is successfully checked off, False otherwise.

uncheck_habit() method

def uncheck_habit (self):

Unchecks off the given Habit object for the current duration.

Parameters	The function accepts no parameters
Returns	True if habit is successfully unchecked off, False otherwise.

get_history() method.

def get_history (self):

Exports history data of the object as a string

Parameters	The function accepts no parameters
Returns	Returns Habit history data as a string in "%Y-%m-%d,%Y-%m-%d" format

set_history() method

def set_history (self):

Imports history data from a string

Parameters	**history: str** History data as a string in "%Y-%m-%d,%Y-%m-%d" format
Returns	returns True if string is successfully imported, False otherwise.

get_creation_date() method

def get_creation_date (self):

Exports creation date as a string

Parameters	The function accepts no parameters
Returns	Creation date as string in "%Y-%m-%d" format

Class HabitManager

class HabitManager(db_name: str)

HabitManager manages a collection of Habit objects. The HabitManager class is linked to the HabittrackerCLI, and responsible for loading and saving the data to the database.

Parameters	db_name: str Name of the database file
Attributes	habits: dict Collection of Habit objects db_name: str Name of the database file

Methods

add_habit_obj() method

def add_habit_obj(self, habit: src.Habit.Habit) -> bool:

Adds the habit object to the Habit manager

Parameters	Habit habit: the habit object to be added.
Returns	True if successfully added, false otherwise:

load_habits_from_db() method

```
def load_habits_from_db(self) -> None:
```

Loads Habit data from the database files to the habit classes

Parameters	The function accepts no parameters
Returns	The function returns nothing

Is_habit_exist() method

```
def is_habit_exist(self, name: str) -> bool:
```

Checks whether if the given habit name exists

Parameters	str name: habit name
Returns	True if habit name exists, false otherwise

add_habit() method.

```
def add_habit(
        self,
        name: str,
        desc: str = '',
        periodicity: src.Habit.HabitPeriods = <HabitPeriods.DAILY: 'daily'>,
        history: str = ''
) -> bool:
```

Adds habit name to the habit manager

Parameters	**str name:** **str desc:** **HabitPeriods periodicity:** **str history:**
Attributes	True if successfully added, False otherwise.

delete_habit() method

```
def delete_habit(self, name: str) -> bool:
```

Deletes the habit from the application

Parameters	str name: habit name to be deleted
Returns	True if successfully deleted, False otherwise

edit_habit_name() method.

```
def edit_habit_name(self, old_name: str, new_name: str) -> bool:
```

Changes the description of the habit

Parameters	name: name of the habit desc: new description of the habit
Attributes	True if changed successfully, False otherwise.

edit_habit_periodicity() method

```
def edit_habit_periodicity(self, name, periodicity):
```

Changes the periodicity of the habit. Changing the periodicity will reset the history of the habit.

Parameters	name: name of the habit periodicity: new periodicity of the habit
Attributes	True if successfully changed, False otherwise.

Edit_habit_reset() method.

```
def edit_habit_reset(self, name) -> bool:
```

Resets the history of the habit

Parameters	str name: name of the habit
Attributes	returns True if the reset is successful. False otherwise.

Check_off_habits() method

```
def check_off_habits(self, habit_list: list) -> bool:
```

Checks off the list of habits for today

Parameters	list habit_list: list of habit names to be checked off
Attributes	True if successfully changed.

Uncheck_habit() method

def uncheck_habit(self, name: str):

Unchecks the given habit for today

Parameters	**str name: name of the habit**
Attributes	True if the habit is successfully unchecked. False otherwise.

data_as_dict() method

def data_as_dict(self):

Returns the habit data as a dictionary of habit objects. For usage in Analytics module.

Parameters	The function accepts no parameters
Attributes	dictionary of habit objects

Class HabitTrackerCLI

class HabitTrackerCLI(habit_manager: HabitManager)

HabitTrackerCLI class is responsible for rendering the command line interface for the application.

Parameters	habit_manager: HabitManager
Attributes	habit_manager: HabitManager

Methods

main_loop() method.

def main_loop(self):

Main menu of the application

Parameters	The function accepts no parameter
Returns	The function returns nothing

Database Module

Methods

Initialiaze_database() method.

def initialize_database(db_name: str):

Initializes the database by creating if tables doesn't exist

Parameters	str db_name: Name of the database
Returns	Returns True if successfully initialized.

Add_habit_to_db() method

def add_habit_to_db(db_name: str, habit: src.Habit.Habit):

Adds habit to the database

Parameters	str db_name: Name of the database file Habit habit: Habit object to be added to the database
Returns	True if successfully added, False otherwise

is_habit_exists() method.

def is_habit_exists(db_name, name: str) -> bool:

Checks if the given habit name is present in the database

Parameters	str db_name: Name of the database file str name: Name of the Habit to check
Returns	True if the habit present in the database, False otherwise

delete_habit() method.

def delete_habit(db_name: str, name: str) -> None:

Delete the given habit from the database

Parameters	**str db_name: Name of the database file** **str name: Name of the habit to delete**
Returns	Returns True if the habit is successfully deleted, False otherwise

load_habit_data() method.

def load_habit_data(db_name: str) -> list:

Loads the data from the given database

Parameters	str db_name: Name of the database file
Returns	list of sql row objects

load_habit_history() method.

def load_habit_history(db_name: str) -> list:

Loads habit history from the given database

Parameters	**str db_name: The name of the given database file**
Returns	a list of sql row objects

Update_habit_Data() method.

def update_habit_data(db_name, habit: src.Habit.Habit):

Updates the habit data from the given object

Parameters	**str db_name:** Name of the given database file **Habit habit:** Habit object to be updated
Returns	True if successfully updated else false

Update_habit_history() method.

def update_habit_history(db_name, habit: src.Habit.Habit):

Updates the habit history data from the given Habit object

Parameters	str db_name: Name of the given database file Habit habit: Habit object to be updated
Returns	True if successfully updated, else returns false

Analytics Module

Methods

Analytics_time_duration() method

def analytics_time_duration(recent_date: datetime, older_date: datetime, periodicity = HabitPeriods.DAILY) -> int:

Calculates the distance between two dates, based on the periodicity. (daily, weekly, etc.)

Parameters	recent_date: the recent date, from today older_date: the older date, from today periodicity: the periodicity of the habit
Returns	the distance between the dates

Analyze_streak() method

def analyze_streak(h_dates, older, periodicity) -> tuple[int, int]:

Recursive function for calculating the streaks.

Parameters	h_dates: list of dates to calculate from. older: the older date from the previous iteration. periodicity: periodicity of the habit
Returns	returns current streak, and longest streak.

analytics_habit_streak() method

def analytics_habit_streak(h_dates: list, periodicity) -> tuple[int, int]:

Function for wrapping analyze_streak function, finalizes the results.

Parameters	h_dates: list of dates from habit object periodicity: habit periodicity.
Returns	returns current streak, and longest streak.

analyze_breaks() method

def analyze_breaks(h_dates, older, periodicity) -> tuple:

Recursive function for calculates longest break, and total number of breaks.

Parameters	h_dates: list of dates. older: oldest date from the previous iteration. periodicity: periodicity of the habit
Returns	Returns longest break and total breaks

analytics_habit_breaks() method

def analytics_habit_breaks(h_dates: list, periodicity) -> tuple[int, int]:

Function acts as a wrapper to the analyze_breaks function. Finalizes the results.

Parameters	h_dates: list of dates from the habit object. periodicity: periodicity of the habit object.
Returns	longest breaks, total breaks.

list_currently_tracker_habits() method

def list_currently_tracked_habits(habit_data: list) -> list:

Lists the names of currently tracked habits

Parameters	list habit_data: list of all habit objects
Returns	list of names as string

List_currently_unchecked_habits() method

def list_currently_unchecked_habits(habit_data: list) -> list:

Lists the names of currently unchecked habits for today

Parameters	list habit_data: list of all habit objects
Returns	list of names as string

List_habit_with_periodicity() method

def list_habit_with_periodicity(habit_data: list, periodicity = HabitPeriods.DAILY):

Lists the names of habit with given periodicity

Parameters	list habit_data: list of all habit objects. HabitPeriods periodicity: periodicity to be selected.
Returns	list of names as string

Str_truncate() method

def str_truncate(name: str) -> str:

Truncate the given string for table

Parameters	**name: string to be truncated.**
Returns	resulting string

float_truncate() method

def float_truncate(number: float) -> str:

Truncate the given float for usage in tables

Parameters	number: number to be truncated
Returns	resulting number.

Habit_info() method

def habit_info(habit: Habit) -> str:

Add analytic information to the given list of habit name

Parameters	Habit habit: Habit object to be generated
Returns	string containing analytical information

List_habit_into() method

def list_habit_info(habit_data: dict, name_list: list) -> list:

Lists the analytics info of the given habit names

Parameters	Habit habit_data: list of all habit objects name_list: list of names of habit
Returns	list of analytics information

Habit_analytics() method

def habit_analytics(habit: Habit) -> list:

Creates a list of all analytical information of the given habit object, such as name, current streak, longest streak, longest breaks, total breaks, total checkoffs, total task duration, habit periodicity, habit creation date and habit description

Parameters	Habit habit: habit object
Returns	list of analytical information

List_habit_analytics() method

def list_habit_analytics(habit_data: dict, name_list: list):

Generates a list containing list of analytical information of given habit names.

Parameters	dict habit_data: dictionary of habit objects name_list: list containing the name of the habits
Returns	list of habit names with analytics.

Habit_analytics_table() method

def habit_analytics_table(habit: Habit) -> list:

Creates a list of all analytical information of the given habit object, such as name, current streak, longest streak, longest breaks, total breaks, total checkoffs, total task duration, habit periodicity, habit creation date and habit description.

Parameters	Habit habit: habit object
Returns	list of analytical information

List_habit_analytics_table() method

def list_habit_analytics_table(habit_data: dict, name_list: list):

Generates a list containing list of analytical information of given habit names.

Parameters	dict habit_data: dictionary of habit objects name_list: list containing the name of the habits
Returns	list of habit names with analytics.

Interface Module

Methods

ask_habit_name() method.

def ask_habit_name():

Parameters	
Attributes	

Ask_habit_dec() method

def ask_habit_desc():

Parameters	
Attributes	

Ask_habit_periodicity() method

def ask_habit_periodicity():

Parameters	
Attributes	

Edit_habit_desc() method

```
def edit_habit_desc(habit_name: str,
                    habit_manager: src.HabitManager.HabitManager):
```

Parameters	**str habit_name**: name of the habit to be edited **HabitManager habit_manager**: link form HabitTrackerCLI object
Attributes	

Edit_habit_periodicty() method

```
def edit_habit_periodicity(habit_name: str,
                           habit_manager: src.HabitManager.HabitManager):
```

Parameters	
Attributes	

Rename_habit() method.

def rename_habit(habit_name: str, habit_manager: src.HabitManager.HabitManager):

Parameters	
Attributes	

check_or_uncheck_habit() method

def check_or_uncheck_habit(habit_name: str, habit_manager: src.HabitManager.HabitManager):

Parameters	
Attributes	

Reset_habit() method

def reset_habit(habit_name: str, habit_manager: src.HabitManager.HabitManager):

Parameters	
Attributes	

delete_habit() method.

def delete_habit(habit_name: str, habit_manager: src.HabitManager.HabitManager):

Parameters	
Attributes	

Habit_report_one() method

def habit_report_one(habit_name: str, habit_manager: src.HabitManager.HabitManager):

Parameters	
Attributes	