lpm

Release 0.0.3

Jay Mody, Jessica Lim, Maanav Dalal

Mar 17, 2021

CONTENTS

Lines Per Minute (lpm) is a command-line typing practice tool made for programmers. Inspired by github.com/cslarsen/wpm.

Installation:

```
pip install lpm
```

CHAPTER ONE

MODULE INTERFACE SPECIFICATION

1.1 lpm

Lines Per Minute.

`lpm.commandline`	Module that specifies the lpm command-line interface.
`lpm.config`	Module that handles lpm configuration.
`lpm.game`	Module that contains main logic for lpm typing game.
`lpm.screen`	Module for command-line IO.
`lpm.snippets`	Module that specifies data structures, namely Snippet and Snippets.
`lpm.stats`	Module for tracking and computing lpm statistics.

1.1.1 lpm.commandline

Module that specifies the lpm command-line interface.

Use *lpm -h* for help.

Functions

`cli`	Main entry point for lpm CLI.
`helpmenu`	Displays the help menu.
`reset`	Resets the settings for lpm.
`settings`	Allows user to modify lpm settings.
`start`	Starts the lpm typing interface.
`stats`	Displays the users statistics to the command-line.

lpm.commandline.cli

`lpm.commandline.cli()`
Main entry point for lpm CLI.

lpm.commandline.helpmenu

`lpm.commandline.helpmenu()`
Displays the help menu.

lpm.commandline.reset

`lpm.commandline.reset()`
Resets the settings for lpm.

lpm.commandline.settings

`lpm.commandline.settings()`
Allows user to modify lpm settings.

lpm.commandline.start

`lpm.commandline.start()`
Starts the lpm typing interface.

lpm.commandline.stats

`lpm.commandline.stats()`
Displays the users statistics to the command-line.

1.1.2 lpm.config

Module that handles lpm configuration.

This module handles app configurations that can be modified by the user. The configuration is loaded from CONFIG_PATH, which the user may edit via: *lpm –settings*

Module attributes

DEFAULT_CONFIG	Stores the default configuration for lpm.

lpm.config.DEFAULT_CONFIG

`lpm.config.`**`DEFAULT_CONFIG = {}`**
Stores the default configuration for lpm.

Classes

Config	App configuration loaded from CONFIG_PATH.

lpm.config.Config

`class` `lpm.config.`**`Config`**
Bases: `object`

App configuration loaded from CONFIG_PATH.

Methods

load	Loads the configuration file from CONFIG_PATH.
reset	Resets the configuration file to DEFAULT_CONFIG.

Attributes

COLOR	Highlight color, used for stats header color.
COLOR_BACKGROUND	Background color.
COLOR_CORRECT	Color of snippet text that was correctly typed.
COLOR_INCORRECT	Color of snippet text that was incorrectly typed.
COLOR_INFO	Color of snippet information text (author, title, etc…).
COLOR_STATS	Color of stats text.
COLOR_TEXT	Color of snippet text.
CONFIG_PATH	Path to configuration file.
INIT	Flag that stores if the config has been loaded.
MAX_CHARS	Max number of characters allowed per line in a snippet.
MAX_LINES	Max lines allowed per snippet.

`COLOR = None`
Highlight color, used for stats header color.

`COLOR_BACKGROUND = None`
Background color.

`COLOR_CORRECT = None`
Color of snippet text that was correctly typed.

`COLOR_INCORRECT = None`
Color of snippet text that was incorrectly typed.

COLOR_INFO = None
Color of snippet information text (author, title, etc…).

COLOR_STATS = None
Color of stats text.

COLOR_TEXT = None
Color of snippet text.

CONFIG_PATH = None
Path to configuration file.

INIT = True
Flag that stores if the config has been loaded.

MAX_CHARS = None
Max number of characters allowed per line in a snippet.

MAX_LINES = None
Max lines allowed per snippet.

static load()
Loads the configuration file from CONFIG_PATH.

static reset()
Resets the configuration file to DEFAULT_CONFIG.

1.1.3 lpm.game

Module that contains main logic for lpm typing game.

Classes

`Game`	Game object that runs the lpm typing game.

lpm.game.Game

class lpm.game.**Game**(*snippets*, *screen*, *stats*)
Bases: object

Game object that runs the lpm typing game.

Parameters

- **snippets** (Snippets) – Snippets object containing database of code snippets.
- **screen** (Screen) – Screen object that handles command-line IO.
- **stats** (Stats) – Stats object that tracks user statistics.

Methods

browsing	Handles interactio during the browsing state.
done	Handles interaction during done state.
get_state	Get the state of the game.
run	Main loop logic for typing game.
typing	Handles interaction during the typing (gameplay) state.

browsing()
Handles interactio during the browsing state.

done()
Handles interaction during done state.

get_state(*key*)
Get the state of the game.

This should return one of the following values: 0 if the user is in browse mode 1 if the user is currently typing (ie attempting a code snippet) 2 if the user had completed a code snippet (similar to browse mode) 3 if the user is resizing the window -1 if the user is attempting to exit the game

Parameters **key** (*str or int*) – Most recent key pressed by the user.

Returns Current state of the game.

Return type int

run()
Main loop logic for typing game.

typing()
Handles interaction during the typing (gameplay) state.

1.1.4 lpm.screen

Module for command-line IO.

Classes

Screen	Screen object used for command-line IO.

lpm.screen.Screen

class lpm.screen.**Screen**

Bases: `object`

Screen object used for command-line IO.

Methods

get_key	Gets the most recently pressed key.
render	Renders the typing interface with the most up to date information.
resize	Resizes game interface based on current user terminal size.

get_key()

Gets the most recently pressed key.

Returns Returns the integer value for a special key, otherwise str value.

Return type str or int

render(*game*)

Renders the typing interface with the most up to date information.

Parameters **game** (Game) – The game object is used to render the relevant snippet, statistics, and user state.

resize()

Resizes game interface based on current user terminal size.

1.1.5 lpm.snippets

Module that specifies data structures, namely Snippet and Snippets.

Classes

Snippet	Data for a single code snippet.
Snippets	Stores database of code snippets.

lpm.snippets.Snippet

class lpm.snippets.**Snippet**(*snippet_id*, *lines*, *url*, *author*, *language*)

Bases: `object`

Data for a single code snippet.

Parameters

- **snippet_id** (*int*) – Unique ID for each code snippet.
- **lines** (*int*) – Number of lines for the snippet.
- **url** (*str*) – A link to the source of the code snippet.

- **author** (*str*) – The author of the code snippet.
- **language** (*str*) – The programming language in which the code snippet is written.

Methods

from_dict	Build Snippet object from a dictionary.

classmethod from_dict(*d*)
Build Snippet object from a dictionary.

Parameters d (*dict*) – Dictionary containing snippet data.

lpm.snippets.Snippets

class lpm.snippets.**Snippets**(*snippets*)
Bases: object

Stores database of code snippets.

Parameters snippets (*list[Snippet]*) – A list of Snippet objects.

Methods

load	Loads snippets from specified filename
next_entry	Returns the next entry in the list of code snippets.
prev_entry	Returns the previous entry in the list of code snippets.
shuffle	Shuffle the list of snippets.

classmethod load(*filename*)
Loads snippets from specified filename

Parameters filename (*str*) – A direct path to the filename to load snippets from. snippets.json by default.

Returns Returns Snippets object loaded from filename.

Return type *Snippets*

next_entry()
Returns the next entry in the list of code snippets.

prev_entry()
Returns the previous entry in the list of code snippets.

shuffle()
Shuffle the list of snippets.

1.1.6 lpm.stats

Module for tracking and computing lpm statistics.

Functions

`accuracy`	Calculates user accuracy for a given section.
`characters_per_minute`	Calculates characters per minute.
`lines_per_minute`	Calculates lines per minute.
`words_per_minute`	Calculates words per minute based on average 5.6 characters per word.

lpm.stats.accuracy

`lpm.stats.`**`accuracy`**(*correct*, *wrong*)

Calculates user accuracy for a given section.

Parameters

- **`correct`** (*`int`*) – Number of characters correctly typed
- **`wrong`** (*`int`*) – Number of characters incorrectly typed

Returns The user's fractional accuracy for the given accuracy

Return type double

lpm.stats.characters_per_minute

`lpm.stats.`**`characters_per_minute`**(*num_chars*, *elapsed*)

Calculates characters per minute.

Parameters

- **`num_chars`** (*`int`*) – Number of characters typed during elapsed time.
- **`elapsed`** (*`double`*) – Number of seconds elapsed in user's typing.

Returns Number of characters per minute a user is typing.

Return type double

lpm.stats.lines_per_minute

`lpm.stats.`**`lines_per_minute`**(*num_lines*, *elapsed*)

Calculates lines per minute.

Parameters

- **`num_lines`** (*`int`*) – Number of lines typed during elapsed time.
- **`elapsed`** (*`double`*) – Number of seconds elapsed in user's typing.

Returns Number of lines per minute a user is typing.

Return type double

lpm.stats.words_per_minute

`lpm.stats.`**`words_per_minute`**`(`*num_chars*, *elapsed*`)`

Calculates words per minute based on average 5.6 characters per word.

Parameters

- **`num_chars`** (*`int`*) – Number of characters typed during elapsed time.
- **`elapsed`** (*`double`*) – Number of seconds elapsed in user's typing.

Returns Number of words per minute a user is typing.

Return type double

Classes

`Stat`	Statistics for a single snippet attempt.
`Stats`	Data object for user statistics.

lpm.stats.Stat

`class` `lpm.stats.`**`Stat`**`(`*start_time*, *end_time=None*`)`

Bases: `object`

Statistics for a single snippet attempt.

Parameters

- **`start_time`** (*`datetime`*) – Datetime object for when attempt was started.
- **`end_time`** (*`datetime, optional`*) – Datetime object for when attempt was completed.

Methods

—

Attributes

`acc`	Accuracy.
`cpm`	Characters per minute.
`elapsed`	Elapsed time in seconds since stat was started.
`lpm`	Lines per minute.
`wpm`	Words per minute.

`property acc`

Accuracy.

`property cpm`

Characters per minute.

`property elapsed`

Elapsed time in seconds since stat was started.

property lpm
Lines per minute.

property wpm
Words per minute.

lpm.stats.Stats

class `lpm.stats.`**`Stats`**(*stats*)
Bases: `object`

Data object for user statistics.

Parameters **stats** (*dict datetime.datime -> Stat*) – A history of user snippet statistics stored in a dictionary that maps a datetime to a Stat object.

Methods

load	Loads stats from the provided stats JSON file.
save	Saves current statistics to the specified JSON file.
update	Update the stats history with a new Stat entry.

classmethod load(*filename*)
Loads stats from the provided stats JSON file.

Parameters **filename** (*str*) – File path to load stats from.

save(*filename*)
Saves current statistics to the specified JSON file.

Parameters **filename** (*str*) – File path to save stats to.

update(*stat*)
Update the stats history with a new Stat entry.

Parameters **stat** (`Stat`) – Stat for the current

PYTHON MODULE INDEX

INDEX