lpm

Release 0.0.7

Jay Mody, Jessica Lim, Maanav Dalal

Apr 03, 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
```

Usage:

Start the program with: `lpm`

Use `lpm -h` for additional options.

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.
reset	Resets the settings for lpm.
settings	Allows user to modify lpm settings through their text editor.
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.reset

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

This method can update both the config and stats files, based on user choice.

lpm.commandline.settings

lpm.commandline.**settings**()
Allows user to modify lpm settings through their text editor.

This method will open the config file using the default text editor, thus replacing the command line window.

lpm.commandline.start

lpm.commandline.**start**(*languages=None*)
Starts the lpm typing interface.

Parameters **languages** (*list[str]*) – Whitelist of programming languages to load code snippets for.

lpm.commandline.stats

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

This method outputs text 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 = {'COLORS': {'author': [39, 235], 'background': [235, 235], 'c**
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.

Examples

from Config import Config

Config.BACKGROUND_COLOR

Methods

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

Attributes

COLORS	xterm256 colors for interface components.
CONFIG_PATH	Path to configuration file.
DEFAULT_LANGS	Code snippet programming languages to load lpm with by default.
INIT	Flag that stores if the config has been loaded.
MAX_COLS	Max cols allowed for a code snippet.
MAX_LINES	Max lines allowed for a code snippet.
SNIPPETS_PATH	Path to snippets file.
SNIPPET_PATH	
STATS_PATH	Path to stats file.

COLORS = {'author': [39, 235], 'background': [235, 235], 'correct': [243, 235], 'in
xterm256 colors for interface components.

CONFIG_PATH = '/Users/jay/.lpmconfig.json'
Path to configuration file.

DEFAULT_LANGS = ['python', 'java', 'javascript']

Code snippet programming languages to load lpm with by default.

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

MAX_COLS = 80
Max cols allowed for a code snippet.

MAX_LINES = 20
Max lines allowed for a code snippet.

SNIPPETS_PATH = '/Users/jay/.lpmsnippets.pickle'
Path to snippets file.

STATS_PATH = '/Users/jay/.lpmstats.pickle'
Path to stats file.

static load()
Loads the configuration file from CONFIG_PATH.

This method extracts information from the config file, located at 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*, *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 interaction during the browsing state.
done	Handles interaction during done state.
finished_snippet	Finished Snippet
get_state	Get the state of the game.
reset_snippet	Resets stats and browses
run	Main loop logic for typing game.
start_snippet	Start snippet
typing	Handles interaction during the typing (gameplay) state.

browsing(*key*)
Handles interaction during the browsing state.

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

done(*key*)
Handles interaction during done state.

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

finished_snippet()
Finished Snippet

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

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

reset_snippet()
Resets stats and browses

run()
Main loop logic for typing game.

start_snippet()
Start snippet

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

This handling of interaction includes special characters (Enter, backspace, escape) as well as normal keystrokes (letters, numbers, special characters).

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

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

clear	Clear the terminal.
deinit	Deinitializes curses.
get_key	Gets the most recently pressed key.
render_score	Render the score.
render_snippet	Renders the typing interface with the most up to date information.
render_update	Render an update to a currently active typing game.
resize	Resizes game interface based on current user terminal size.
setup_colors	Setups up terminal color from Config.COLORS.

Attributes

KEY_BACKSPACE
KEY_ENTER
KEY_ESCAPE
KEY_LEFT
KEY_RESIZE
KEY_RIGHT

columns	Returns number of terminal columns.
lines	Returns number of terminal lines.

clear()
Clear the terminal.

property columns
Returns number of terminal columns.

deinit()
Deinitializes curses.

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

property lines
Returns number of terminal lines.

render_score(*game*)
Render the score.

Parameters **game** (Game) – Game object.

render_snippet(*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.

render_update(*game*, *action*)
Render an update to a currently active typing game.

Parameters

- **game** (Game) – Game object.
- **action** (*str*) – Current action being taken, one of "back", "enter", "correct", or "incorrect", or None

resize(*game*)
Resizes game interface based on current user terminal size.

setup_colors()
Setups up terminal color from Config.COLORS.

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** (*list[str]*) – Text lines in the code snippet.
- **url** (*str*) – A link to the source of the code snippet (ie full url to github source file with lines permalinked)
- **author** (*str*) – The author of the code snippet (ie pallets/flask).
- **language** (*str*) – The programming language in which the code snippet is written.

Methods

from_url	Create Snippet object from github permalink.

Attributes

ext_to_lang

classmethod from_url(*snippet_id*, *url*)
Create Snippet object from github permalink.

A url must be of form: https://github.com/<USERNAME>/<REPO>/blob/<COMMIT_HASH>/path/to/file.ext#L<START>-L<END>

For example: https://github.com/jaymody/linkipedia/blob/09f3ca27e1ad858a6a010d2ef3d0768cbb9dda36/src/main/java/com/linkipedia/Graph.java#L9-L31

This will extract the code from the given url and create a Snippet object from it. The code will be assigned to lines, the url to url, and the author to <USERNAME>/<REPO>. Language will be infered from the extension of the file, so the file must have an extension.

Parameters

- **snippet_id** (*int*) – Unique ID for the code snippet.
- **url** (*str*) – Github permalink.

Returns Snippet object created using github permalink.

Return type *Snippet*

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

current_snippet	Get current entry
from_urls	Creates snippets object from github permalinks.
load	Loads snippets from specified filename.
next_snippet	Returns the next entry in the list of code snippets.
prev_snippet	Returns the previous entry in the list of code snippets.
save	Saves current statistics to the specified pickle file.
shuffle	Shuffle the list of snippets.

current_snippet()
Get current entry

classmethod from_urls(*urls*)
Creates snippets object from github permalinks.

See Snippet.from_url() for more information.

Parameters **urls** (*list[str]*) – List of github permalink urls.

Returns Snippets object with snippets from urls.

Return type *Snippets*

static load(*filename*, *languages=['python', 'java', 'javascript']*)
Loads snippets from specified filename.

Parameters

- **filename** (*str*) – A direct path to the filename to load snippets from.
- **languages** (*list[str]*) – List of string of languages to load snippets of.

Returns Returns Snippets object loaded from filename.

Return type *Snippets*

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

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

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

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

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=None*, *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

start	Set start_time to the current time (ie the code snippet attempt has started).
stop	Set end_time to the current time (ie the code snippet attempt is done).

Attributes

TIME_STR_FMT

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.

start()
Set start_time to the current time (ie the code snippet attempt has started).

stop()
Set end_time to the current time (ie the code snippet attempt is done).

property wpm
Words per minute.

lpm.stats.Stats

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

Data object for user statistics.

Parameters **stats** (*list[datetime]*) – A history of user snippet statistics in chronological order.

Methods

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

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

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

Returns Stats object loaded from pickle.

Return type *Stats*

save(*filename*)
Saves current statistics to the specified pickle 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 object to store to the history.

PYTHON MODULE INDEX

l

INDEX

S

T

U

W