work-tracker Documentation

Release 0.1.0

Sebastian Weigand

Sep 28, 2022

USER DOCUMENTATION:

CHAPTER

ONE

WORK-TRACKER

Simple tool to keep track of your work time and/or productivity

- Free software: Apache Software License 2.0
- Documentation: https://work-tracker.readthedocs.io.

1.1 Features

- TODO

1.2 Credits

This package was created with Cookiecutter and the audreyr/cookiecutter-pypackage project template.

CHAPTER

TWO

INSTALLATION

2.1 Stable release

To install work-tracker, run this command in your terminal:

```
$ pip install work_tracker
```

This is the preferred method to install work-tracker, as it will always install the most recent stable release.

If you don't have pip installed, this Python installation guide can guide you through the process.

2.2 From sources

The sources for work-tracker can be downloaded from the Github repo.

You can either clone the public repository:

```
$ git clone git://github.com/s-weigand/work-tracker
```

Or download the tarball:

```
$ curl  -OL https://github.com/s-weigand/work-tracker/tarball/master
```

Once you have a copy of the source, you can install it with:

```
$ python setup.py install
```

CHAPTER

THREE

USAGE

To use work-tracker in a project:

```
import work_tracker
```

CHAPTER

FOUR

CREDITS

4.1 Development Lead

- Sebastian Weigand <s.weigand.phy@gmail.com>

4.2 Contributors

None yet. Why not be the first?

CHAPTER

FIVE

HISTORY

5.1 0.1.0 (2018-12-09)

- First release on PyPI.

CHAPTER

SIX

INNER WORKINGS

6.1 Database Functionality

base_classes	Module containing the baseclass for data interactions.
calc_worktime	Module containing the Worktime calculator class.
helpfer_functions	Module containing helper functions.
update_work_db	Module containing Database interaction class.

6.1.1 base_classes

Module containing the baseclass for data interactions.

Classes

Summary

DbBaseClass	Baseclass for data interactions.

DbBaseClass

class DbBaseClass(*user_config_path='.user_config.ini'*)

Baseclass for data interactions.

Parameters

user_config_path (*str, optional*) – Path to the user specific config, which will overwrite default settings. by default ".user_config.ini"

Attributes Summary

default_config_path

Methods Summary

calc_file_hashes	Calculate hashvalues for files.
clean_db	Remove rows where the session work was less than 1min.
get_datetime_now	Helpermethod for mocking of datetime.datetime.now() in unittests.
get_pandas_now	Return datetime.now as pd.Timestamp.
get_remote_db	Download the db_file to db_path_online from the SFTP server.
load_config	Load the config files and sets all necessary properties.
load_db	Read in the db file if it exists or creates a new one.
merge_dbs	Merge local db with remote db.
push_remote_db	Push the db_file from db_path_offline to the SFTP server.

calc_file_hashes

DbBaseClass.**calc_file_hashes**() → DataFrame

Calculate hashvalues for files.

Returns
Dataframe with file hashes.

Return type
pd.DataFrame

clean_db

DbBaseClass.**clean_db**() → None

Remove rows where the session work was less than 1min.

get_datetime_now

DbBaseClass.**get_datetime_now**() → datetime

Helpermethod for mocking of datetime.datetime.now() in unittests.

Returns
datetime.now()

Return type
datetime.datetime

get_pandas_now

DbBaseClass.**get_pandas_now**() → Timestamp

Return datetime.now as pd.Timestamp.

Returns
current time as timestamp: pd.Timestamp

Return type
pd.Timestamp

get_remote_db

DbBaseClass.**get_remote_db**() → bool

Download the db_file to db_path_online from the SFTP server.

This uses the values specified at ["login"]["db_path"] in the config file.

Returns
Whether database retrieval succeeded or not.

Return type
bool

load_config

DbBaseClass.**load_config**() → ConfigParser

Load the config files and sets all necessary properties.

load_db

DbBaseClass.**load_db**(*db_path: str*) → DataFrame

Read in the db file if it exists or creates a new one.

Parameters
db_path (*str*) – path to the db_file on the SFTP server

Returns
Loaded database.

Return type
pd.Dataframe

merge_dbs

DbBaseClass.**merge_dbs**() → DataFrame

Merge local db with remote db.

The overlap (same start) is replaced with the max value of end.

Returns
Local db merged with remote db, with striped overlap.

Return type
pd.Dataframe

push_remote_db

DbBaseClass.**push_remote_db**() → bool

Push the db_file from db_path_offline to the SFTP server.

This uses the values specified at ["login"]["db_path"] in the config file.

Returns
Whether database upload succeeded or not.

Return type
bool

Methods Documentation

calc_file_hashes() → DataFrame

Calculate hashvalues for files.

Returns
Dataframe with file hashes.

Return type
pd.DataFrame

clean_db() → None

Remove rows where the session work was less than 1min.

get_datetime_now() → datetime

Helpermethod for mocking of datetime.datetime.now() in unittests.

Returns
datetime.now()

Return type
datetime.datetime

get_pandas_now() → Timestamp

Return datetime.now as pd.Timestamp.

Returns
current time as timestamp: pd.Timestamp

Return type
pd.Timestamp

get_remote_db() → bool

Download the db_file to db_path_online from the SFTP server.

This uses the values specified at ["login"]["db_path"] in the config file.

Returns
Whether database retrieval succeeded or not.

Return type
bool

load_config() → ConfigParser

Load the config files and sets all necessary properties.

load_db(*db_path: str*) → DataFrame

Read in the db file if it exists or creates a new one.

Parameters
db_path (*str*) – path to the db_file on the SFTP server

Returns
Loaded database.

Return type
pd.Dataframe

merge_dbs() → DataFrame

Merge local db with remote db.

The overlap (same start) is replaced with the max value of end.

Returns
Local db merged with remote db, with striped overlap.

Return type
pd.Dataframe

push_remote_db() → bool

Push the db_file from db_path_offline to the SFTP server.

This uses the values specified at ["login"]["db_path"] in the config file.

Returns
Whether database upload succeeded or not.

Return type
bool

6.1.2 calc_worktime

Module containing the Worktime calculator class.

Classes

Summary

WorktimeCalculator	Class to calculate the worktime.

WorktimeCalculator

class WorktimeCalculator(*user_config_path='.user_config.ini'*)

Class to calculate the worktime.

Parameters
user_config_path (*str, optional*) – Path to the user specific config, which will overwrite default settings. by default ".user_config.ini"

Attributes Summary

default_config_path

Methods Summary

add_time_columns	Add Year, Month, Week and Day columns to an existing Dataframe.
calc_file_hashes	Calculate hashvalues for files.
clean_db	Remove rows where the session work was less than 1min.
generate_contract_worktime_df	Generate a Dataframe with 'worktime' column.
get_daily_worktime	Calculate the mean daily work time based on the contract details.
get_datetime_now	Helpermethod for mocking of datetime.datetime.now() in unittests.
get_holiday_df	Generate a Dataframe containing all needed holidays.
get_manual_df_with_workime	Read the manual_db file.
get_pandas_now	Return datetime.now as pd.Timestamp.
get_plot_df	Return a Dataframe prepared for plotting.
get_remote_db	Download the db_file to db_path_online from the SFTP server.
get_total_df	Calculate total Dataframe.
init_holidays	Initialize the holidays with the custom holidays from the config.
load_config	Load the config files and sets all necessary properties.
load_db	Load Database remote or locally.
merge_dbs	Merge local db with remote db.
push_remote_db	Push the db_file from db_path_offline to the SFTP server.
split_date_overlap_session	Split sessions which contain midnight to to sessions.

add_time_columns

classmethod WorktimeCalculator.**add_time_columns**(*df: DataFrame*, *date_time_column_name: str = 'start'*) → DataFrame

Add Year, Month, Week and Day columns to an existing Dataframe.

Parameters

- **df** (*pd.DataFrame*) – Dataframe the columns should be added to.
- **date_time_column** (*str*) – Name of the column containing the used date

Returns

Dataframe with added Year, Month, Week and Day columns.

Return type

pd.DataFrame

calc_file_hashes

WorktimeCalculator.**calc_file_hashes**() → DataFrame

Calculate hashvalues for files.

Returns
Dataframe with file hashes.

Return type
pd.DataFrame

clean_db

WorktimeCalculator.**clean_db**() → None

Remove rows where the session work was less than 1min.

generate_contract_worktime_df

WorktimeCalculator.**generate_contract_worktime_df**() → DataFrame

Generate a Dataframe with 'worktime' column.

Generate a template like DataFrame containing the mean daily work time for each day counting as workday based on the contract details. This Template will be used later on to generate the manual_df and holiday_df.

Returns
contract_worktime_df – Dataframe containing all workdays and their mean daily working time, since the 1st contract started until now

Return type
pandas.DataFrame

get_daily_worktime

WorktimeCalculator.**get_daily_worktime**(*frequenzy: str*, *worktime: Union[int, float]*, *weekmask: str*) → float

Calculate the mean daily work time based on the contract details.

Parameters
- **frequenzy** (*str*) – Timeperiod in which the the user is supposed to to work worktime*1h Supported values are: monthly, weekly
- **worktime** (*int, float*) – Time in hours that the user is supposed to work in frequenzy
- **weekmask** (*str*) – Listing of the abbreviated weekdays the user is supposed to work i.e.: "Mon Tue Wed Thu Fri Sat"

Returns
daily_worktime

Return type
float

get_datetime_now

WorktimeCalculator.**get_datetime_now**() → datetime

Helpermethod for mocking of datetime.datetime.now() in unittests.

Returns
datetime.now()

Return type
datetime.datetime

get_holiday_df

WorktimeCalculator.**get_holiday_df**() → DataFrame

Generate a Dataframe containing all needed holidays.

They begin at the start of the first contract until now, as well as work time based on the contract parameters for those days.

Returns
holiday_df – Dataframe containing all holidays from the start of the first contract until now, as well as work time based on the contract parameters for those days.

Return type
pandas.DataFrame

get_manual_df_with_workime

WorktimeCalculator.**get_manual_df_with_workime**() → DataFrame

Read the manual_db file.

That file contains information like sick time | vacation, specified by manual_db in the config. Then expands the given time periods to date_ranges on a daily base, drops the none workdays and adds the daily work time based on the contract parameters for that day.

Returns
Dataframe containing the to a daily base expanded entry's of manual_df.

Return type
pandas.DataFrame

get_pandas_now

WorktimeCalculator.**get_pandas_now**() → Timestamp

Return datetime.now as pd.Timestamp.

Returns
current time as timestamp: pd.Timestamp

Return type
pd.Timestamp

get_plot_df

WorktimeCalculator.**get_plot_df**(*rule='D'*, *date_time_column='start'*) → DataFrame

Return a Dataframe prepared for plotting.

Dataframe with a DateTimeIndex, columns named by occupation and containing the worked time of that occupation for the given samplingrate

Parameters
rule (*str*) – Resampling rule see pandas.DataFrame.resample

Returns
Dataframe with a DateDimeIndex, columns named by occupation and containing the worktime of that occupation for the samplingrate

Return type
pandas.DataFrame

get_remote_db

WorktimeCalculator.**get_remote_db**() → bool

Download the db_file to db_path_online from the SFTP server.

This uses the values specified at ["login"]["db_path"] in the config file.

Returns
Whether database retrieval succeeded or not.

Return type
bool

get_total_df

WorktimeCalculator.**get_total_df**() → DataFrame

Calculate total Dataframe.

Returns
Dataframe with 'worktime' and time columns

Return type
pd.DataFrame

See also:

add_time_columns

init_holidays

WorktimeCalculator.**init_holidays**() → Union[HolidayBase, Dict]

Initialize the holidays with the custom holidays from the config.

Returns the class instance from holiday, which matches the country and province given in the config and updates it with the special holidays, also given in the config

Returns
HolidayBase object for the given country and province or an empty dict.

Return type
Union[HolidayBase, Dict]

load_config

WorktimeCalculator.**load_config**()

Load the config files and sets all necessary properties.

load_db

WorktimeCalculator.**load_db**() → DataFrame

Load Database remote or locally.

Tries to load the database directly from the server if possible, else it loads the local database or throws an exception that isn't possible either.

Returns
db – Database with the actually worked time

Return type
pandas.DataFrame

merge_dbs

WorktimeCalculator.**merge_dbs**() → DataFrame

Merge local db with remote db.

The overlap (same start) is replaced with the max value of end.

Returns
Local db merged with remote db, with striped overlap.

Return type
pd.Dataframe

push_remote_db

WorktimeCalculator.**push_remote_db**() → bool

Push the db_file from db_path_offline to the SFTP server.

This uses the values specified at ["login"]["db_path"] in the config file.

Returns
Whether database upload succeeded or not.

Return type
bool

split_date_overlap_session

WorktimeCalculator.**split_date_overlap_session**() → None

Split sessions which contain midnight to to sessions.

The first lasting until midnight and the second starting at midnight.

```
df before:
start   end
1.1.1970 21:00:00   2.1.1970 02:00:00
```

(continues on next page)

(continued from previous page)

```
df after:
start   end
1.1.1970 21:00:00   2.1.1970 00:00:00
2.1.1970 00:00:00   2.1.1970 02:00:00
```

Methods Documentation

classmethod add_time_columns(*df: DataFrame*, *date_time_column_name: str = 'start'*) → DataFrame

Add Year, Month, Week and Day columns to an existing Dataframe.

Parameters

- **df** (*pd.DataFrame*) – Dataframe the columns should be added to.
- **date_time_column** (*str*) – Name of the column containing the used date

Returns

Dataframe with added Year, Month, Week and Day columns.

Return type

pd.DataFrame

calc_file_hashes() → DataFrame

Calculate hashvalues for files.

Returns

Dataframe with file hashes.

Return type

pd.DataFrame

clean_db() → None

Remove rows where the session work was less than 1min.

generate_contract_worktime_df() → DataFrame

Generate a Dataframe with 'worktime' column.

Generate a template like DataFrame containing the mean daily work time for each day counting as workday based on the contract details. This Template will be used later on to generate the manual_df and holiday_df.

Returns

contract_worktime_df – Dataframe containing all workdays and their mean daily working time, since the 1st contract started until now

Return type

pandas.DataFrame

get_daily_worktime(*frequenzy: str*, *worktime: Union[int, float]*, *weekmask: str*) → float

Calculate the mean daily work time based on the contract details.

Parameters

- **frequenzy** (*str*) – Timeperiod in which the the user is supposed to to work worktime*1h Supported values are: monthly, weekly
- **worktime** (*int, float*) – Time in hours that the user is supposed to work in frequenzy
- **weekmask** (*str*) – Listing of the abbreviated weekdays the user is supposed to work i.e.: "Mon Tue Wed Thu Fri Sat"

Returns

daily_worktime

Return type

float

`get_datetime_now()` → datetime

Helpermethod for mocking of datetime.datetime.now() in unittests.

Returns
datetime.now()

Return type
datetime.datetime

`get_holiday_df()` → DataFrame

Generate a Dataframe containing all needed holidays.

They begin at the start of the first contract until now, as well as work time based on the contract parameters for those days.

Returns
holiday_df – Dataframe containing all holidays from the start of the first contract until now, as well as work time based on the contract parameters for those days.

Return type
pandas.DataFrame

`get_manual_df_with_workime()` → DataFrame

Read the manual_db file.

That file contains information like sick time | vacation, specified by manual_db in the config. Then expands the given time periods to date_ranges on a daily base, drops the none workdays and adds the daily work time based on the contract parameters for that day.

Returns
Dataframe containing the to a daily base expanded entry's of manual_df.

Return type
pandas.DataFrame

`get_pandas_now()` → Timestamp

Return datetime.now as pd.Timestamp.

Returns
current time as timestamp: pd.Timestamp

Return type
pd.Timestamp

`get_plot_df(rule='D', date_time_column='start')` → DataFrame

Return a Dataframe prepared for plotting.

Dataframe with a DateTimeIndex, columns named by occupation and containing the worked time of that occupation for the given samplingrate

Parameters
rule (*str*) – Resampling rule see pandas.DataFrame.resample

Returns
Dataframe with a DateDimeIndex, columns named by occupation and containing the worktime of that occupation for the samplingrate

Return type
pandas.DataFrame

`get_remote_db()` → bool

Download the db_file to db_path_online from the SFTP server.

This uses the values specified at ["login"]["db_path"] in the config file.

Returns
Whether database retrieval succeeded or not.

Return type
bool

get_total_df() → DataFrame

Calculate total Dataframe.

Returns
Dataframe with 'worktime' and time columns

Return type
pd.DataFrame

See also:

add_time_columns

init_holidays() → Union[HolidayBase, Dict]

Initialize the holidays with the custom holidays from the config.

Returns the class instance from holiday, which matches the country and province given in the config and updates it with the special holidays, also given in the config

Returns
HolidayBase object for the given country and province or an empty dict.

Return type
Union[HolidayBase, Dict]

load_config()

Load the config files and sets all necessary properties.

load_db() → DataFrame

Load Database remote or locally.

Tries to load the database directly from the server if possible, else it loads the local database or throws an exception that isn't possible either.

Returns
db – Database with the actually worked time

Return type
pandas.DataFrame

merge_dbs() → DataFrame

Merge local db with remote db.

The overlap (same start) is replaced with the max value of end.

Returns
Local db merged with remote db, with striped overlap.

Return type
pd.Dataframe

push_remote_db() → bool

Push the db_file from db_path_offline to the SFTP server.

This uses the values specified at ["login"]["db_path"] in the config file.

Returns
Whether database upload succeeded or not.

Return type
bool

split_date_overlap_session() → None

Split sessions which contain midnight to to sessions.

The first lasting until midnight and the second starting at midnight.

```
df before:
start   end
1.1.1970 21:00:00   2.1.1970 02:00:00

df after:
start   end
1.1.1970 21:00:00   2.1.1970 00:00:00
2.1.1970 00:00:00   2.1.1970 02:00:00
```

6.1.3 helpfer_functions

Module containing helper functions.

Functions

Summary

debug_printer	Print variable names and their values.
get_abs_path	Helperfunction to get the absolute path in respect to the main file ("work_tracker.pyw").
get_midnight_datetime	Helperfunction to get the date at exactly midnight for a given datetime object.
hash_file	Calculate the md5 hash value of the file at file_path.
seconds_to_hm	Helperfunction to convert Number of seconds to hour:minute format.
str_datetime	Convert a string in Datetime64[ns] format to a datetime object.

debug_printer

debug_printer(*arg*)

Print variable names and their values.

Convenience function to print variables in a matter that they are easily seen and their name as well as their value is printed.

Parameters
arg (*anything*) –

get_abs_path

get_abs_path(*rel_path*)

Helperfunction to get the absolute path in respect to the main file ("work_tracker.pyw").

Parameters
rel_path (*str*) – relative path to a file

Returns
absolute path – absolute path evaluated from the relative path in respect to the path of the main file("work_tracker.pyw")

Return type
str

get_midnight_datetime

get_midnight_datetime(*datetime_obj: datetime*)

Helperfunction to get the date at exactly midnight for a given datetime object.

Parameters
datetime.datetime – datetime object containing a date

Returns
String representing the seconds which where passed as %h:%M

Return type
datetime.datetime

hash_file

hash_file(*file_path: str*) → Optional[str]

Calculate the md5 hash value of the file at file_path.

Parameters
file_path (*str*) – Path to the file that should be hashed.

Returns
MD5 hex hash value of the file at file_path.

Return type
str

seconds_to_hm

seconds_to_hm(*seconds: int*) → str

Helperfunction to convert Number of seconds to hour:minute format.

Parameters
seconds (*int*) – Amount of seconds that should be converted to an %h:%M string

Returns
String representing of seconds, in the form of %h:%M

Return type
str

str_datetime

str_datetime(*time_str: str*)

Convert a string in Datetime64[ns] format to a datetime object.

Parameters
time_str (*str*) – string representing the datetime

Returns
datetime which was represented by time_str

Return type
datetime

6.1.4 update_work_db

Module containing Database interaction class.

Classes

Summary

DbInteraction	Class for data interactions.

DbInteraction

class DbInteraction(*user_config_path='.user_config.ini'*)

Class for data interactions.

Parameters
user_config_path (*str, optional*) – Path to the user specific config, which will overwrite default settings. by default ".user_config.ini"

Attributes Summary

default_config_path

Methods Summary

calc_file_hashes	Calculate hashvalues for files.
change_occupation	Change value of self.occupation.
clean_db	Remove rows where the session work was less than 1min.
get_datetime_now	Helpermethod for mocking of datetime.datetime.now() in unittests.
get_pandas_now	Return datetime.now as pd.Timestamp.
get_remote_db	Download the db_file to db_path_online from the SFTP server.
get_session_time	Calculate the current session time in hours and minutes.
get_start_time	Return todays startime in hours and minutes.
get_today	Return datetime object for today at midnight.
load_config	Load the config files and sets all necessary properties.
load_db	Read in the db file if it exists or creates a new one.
merge_dbs	Merge local db with remote db.
push_remote_db	Push the db_file from db_path_offline to the SFTP server.
start_session	Start a session and update database.
update_db_locale	Update local database.
update_now_and_tomorrow	Update the instance variables yesterday, today and tomorrow.

calc_file_hashes

DbInteraction.**calc_file_hashes**() → DataFrame

Calculate hashvalues for files.

Returns

Dataframe with file hashes.

Return type

pd.DataFrame

change_occupation

DbInteraction.**change_occupation**(*occupation: str*) → None

Change value of self.occupation.

Parameters

occupation (*str*) – Current occupation.

clean_db

DbInteraction.**clean_db**() → None

Remove rows where the session work was less than 1min.

get_datetime_now

DbInteraction.**get_datetime_now**() → datetime

Helpermethod for mocking of datetime.datetime.now() in unittests.

Returns

datetime.now()

Return type

datetime.datetime

get_pandas_now

DbInteraction.**get_pandas_now**() → Timestamp

Return datetime.now as pd.Timestamp.

Returns

current time as timestamp: pd.Timestamp

Return type

pd.Timestamp

get_remote_db

DbInteraction.**get_remote_db**() → bool

Download the db_file to db_path_online from the SFTP server.

This uses the values specified at ["login"]["db_path"] in the config file.

Returns

Whether database retrieval succeeded or not.

Return type

bool

get_session_time

DbInteraction.**get_session_time**()

Calculate the current session time in hours and minutes.

Returns
String representation in hours and minutes

Return type
str

get_start_time

DbInteraction.**get_start_time**()

Return todays startime in hours and minutes.

Returns
String representing start time as %h:%M

Return type
str

get_today

DbInteraction.**get_today**()

Return datetime object for today at midnight.

Returns
datetime.datetime.now() but with the hours, minutes, seconds, microseconds set to 0

Return type
datetime.datetime

load_config

DbInteraction.**load_config**() → ConfigParser

Load the config files and sets all necessary properties.

load_db

DbInteraction.**load_db**(*db_path: str*) → DataFrame

Read in the db file if it exists or creates a new one.

Parameters
db_path (*str*) – path to the db_file on the SFTP server

Returns
Loaded database.

Return type
pd.Dataframe

merge_dbs

DbInteraction.**merge_dbs**() → DataFrame

Merge local db with remote db.

The overlap (same start) is replaced with the max value of end.

Returns
Local db merged with remote db, with striped overlap.

Return type
pd.Dataframe

push_remote_db

DbInteraction.**push_remote_db**() → bool

Push the db_file from db_path_offline to the SFTP server.

This uses the values specified at ["login"]["db_path"] in the config file.

Returns
Whether database upload succeeded or not.

Return type
bool

start_session

DbInteraction.**start_session**() → None

Start a session and update database.

update_db_locale

DbInteraction.**update_db_locale**() → Tuple[str, str]

Update local database.

Returns
start_time and session_time

Return type
Tuple[str, str]

update_now_and_tomorrow

DbInteraction.**update_now_and_tomorrow**()

Update the instance variables yesterday, today and tomorrow.

In case the date has changed during the session. Preventing error in self.update_db_locale, due to a wrong date.

Methods Documentation

calc_file_hashes() → DataFrame

Calculate hashvalues for files.

Returns
Dataframe with file hashes.

Return type
pd.DataFrame

change_occupation(*occupation: str*) → None

Change value of self.occupation.

Parameters
occupation (*str*) – Current occupation.

clean_db() → None

Remove rows where the session work was less than 1min.

get_datetime_now() → datetime

Helpermethod for mocking of datetime.datetime.now() in unittests.

Returns
datetime.now()

Return type
datetime.datetime

get_pandas_now() → Timestamp

Return datetime.now as pd.Timestamp.

Returns
current time as timestamp: pd.Timestamp

Return type
pd.Timestamp

get_remote_db() → bool

Download the db_file to db_path_online from the SFTP server.

This uses the values specified at ["login"]["db_path"] in the config file.

Returns
Whether database retrieval succeeded or not.

Return type
bool

get_session_time()

Calculate the current session time in hours and minutes.

Returns
String representation in hours and minutes

Return type
str

get_start_time()

Return todays startime in hours and minutes.

Returns
String representing start time as %h:%M

Return type
str

get_today()

Return datetime object for today at midnight.

Returns
datetime.datetime.now() but with the hours, minutes, seconds, microseconds set to 0
Return type
datetime.datetime

`load_config()` → ConfigParser

Load the config files and sets all necessary properties.

`load_db(db_path: str)` → DataFrame

Read in the db file if it exists or creates a new one.
Parameters
db_path (*str*) – path to the db_file on the SFTP server
Returns
Loaded database.
Return type
pd.Dataframe

`merge_dbs()` → DataFrame

Merge local db with remote db.

The overlap (same start) is replaced with the max value of end.
Returns
Local db merged with remote db, with striped overlap.
Return type
pd.Dataframe

`push_remote_db()` → bool

Push the db_file from db_path_offline to the SFTP server.

This uses the values specified at ["login"]["db_path"] in the config file.
Returns
Whether database upload succeeded or not.
Return type
bool

`start_session()` → None

Start a session and update database.

`update_db_locale()` → Tuple[str, str]

Update local database.
Returns
start_time and session_time
Return type
Tuple[str, str]

`update_now_and_tomorrow()`

Update the instance variables yesterday, today and tomorrow.

In case the date has changed during the session. Preventing error in self.update_db_locale, due to a wrong date.

6.2 GUI Functionality

worktracker_main	Module containing the main GUI class.

6.2.1 work_tracker.UI_files.worktracker_main

Module containing the main GUI class.

CHAPTER

SEVEN

CONTRIBUTING

Contributions are welcome, and they are greatly appreciated! Every little bit helps, and credit will always be given.

You can contribute in many ways:

7.1 Types of Contributions

7.1.1 Report Bugs

Report bugs at https://github.com/s-weigand/work-tracker/issues.

If you are reporting a bug, please include:

- Your operating system name and version.
- Any details about your local setup that might be helpful in troubleshooting.
- Detailed steps to reproduce the bug.

7.1.2 Fix Bugs

Look through the GitHub issues for bugs. Anything tagged with "bug" and "help wanted" is open to whoever wants to implement it.

7.1.3 Implement Features

Look through the GitHub issues for features. Anything tagged with "enhancement" and "help wanted" is open to whoever wants to implement it.

7.1.4 Write Documentation

work-tracker could always use more documentation, whether as part of the official work-tracker docs, in docstrings, or even on the web in blog posts, articles, and such.

7.1.5 Submit Feedback

The best way to send feedback is to file an issue at https://github.com/s-weigand/work-tracker/issues.

If you are proposing a feature:

- Explain in detail how it would work.
- Keep the scope as narrow as possible, to make it easier to implement.
- Remember that this is a volunteer-driven project, and that contributions are welcome :)

7.2 Get Started!

Ready to contribute? Here's how to set up `work-tracker` for local development.

1. Fork the `work-tracker` repo on GitHub.
2. Clone your fork locally:

```
$ git clone git@github.com:your_name_here/work_tracker.git
```

3. Install your local copy into a virtualenv. Assuming you have virtualenvwrapper installed, this is how you set up your fork for local development:

```
$ mkvirtualenv work_tracker
$ cd work_tracker/
$ python -m pip install -c constraints.txt -r requirements_dev.txt
```

4. Install the pre-commit and pre-push hooks

```
$ pre-commit install && pre-commit install -t pre-push
```

5. Create a branch for local development:

```
$ git checkout -b name-of-your-bugfix-or-feature
```

   Now you can make your changes locally.

6. When you're done making changes, check that your changes pass flake8 and the tests, including testing other Python versions with tox:

```
$ tox
```

   To get flake8 and tox, just pip install them into your virtualenv.

7. Commit your changes and push your branch to GitHub:

```
$ git add .
$ git commit -m "Your detailed description of your changes."
$ git push origin name-of-your-bugfix-or-feature
```

8. Submit a pull request through the GitHub website.

7.3 Pull Request Guidelines

Before you submit a pull request, check that it meets these guidelines:

1. The pull request should include tests.
2. If the pull request adds functionality, the docs should be updated. Put your new functionality into a function with a docstring, and add the feature to the list in README.rst.
3. The pull request should work for Python 3.7, 3.8, 3.9 and 3.10. Check https://github.com/s-weigand/work-tracker/actions and make sure that the tests pass for all supported Python versions.

7.4 Tips

To run a subset of tests:

```
$ py.test tests.test_work_tracker
```

7.5 Deploying

A reminder for the maintainers on how to deploy. Make sure all your changes are committed (including an entry in HISTORY.rst). Then run:

```
$ bumpversion patch # possible: major / minor / patch
$ git push
$ git push --tags
```

Travis will then deploy to PyPI if tests pass.

CHAPTER

EIGHT

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

W

INDEX

P

S

U

W