Programmers documentation for BI-SKJ @ FIT CTU term work

Author: Tomas Kvasnicka Year: 2012/2013

__main__.py

- Main order of execution
- Order of execution: parse command line, check for required commands, create temporary files, parse configuration files, check parsed arguments, set each source file properties, set the whole animation properties, create animation frames, create animation
- Catches exceptions and exits with correct exit codes

skj_parser_cmdline.py

- Functions parsing command line arguments
- *parse_args():* use argparse modul to parse command line arguments, create help, return *dict()* containing parsed arguments
- *alter_args():* create *dict()* with default argument values, correct value of verbose argument if needed, check if config file exists and is a valid file

skj_parser_cnffile.py

- Functions parsing configuration file directives
- parse_directives(): strip out comments/white spaces, check if line contains a valid configuration directive, check if directive has a value, if the value should be a float check if it valid
- *add_directives_to_args():* check if option was configured using config file and wasn't configured using command line, add such options to *dict()* containing parsed arguments

skj_checker_common.py

- Functions checking user input syntax
- check_parsed_args(): deduplicate repeatable parametrs, download online files and store them to temp location, call routines to check input sytax in this order: check critical values syntax, check axis dimensions syntax, check effects syntax
- *check_command_exists():* using subprocess module check if string is a callable program, return *str()* with program name
- *check_time_format():* check if *str()* is in format as specified by -t and therefore can be converted to *timestruct,* return *timestruct* with time specified in *str()*
- *check_float_ok():* check if *str()* can be converted to valid float (ie. is not inf/nan), return *float()*

from that *str()*

- *check_file():* check if *str()* is a valid file name && is a regular file && is readable && has non-zero length && is unicode encoded, return *str()* with file name
- *check_effects_syntax():* parse -e <effects>, check it for errors, return *dict()*
- *check_x():* check -x/-X value for errors
- *check_y():* check -y/-Y value for errors
- *check_critical_value():* parse -c <crit_val>, check it for errors, return *dict()*

skj_subprocess_gnuplot.py

- *Functions setting frame properties and creating animation frames*
- *set_file_properties():* set correct filesystem path to file, count number of valid lines, check each date for syntax errors, check each data value for syntax errors, find minimum/maximum in date/data values, set date/data column, return *dict()*
- *draw_animation_frames():* configure basic gnuplot options, configure usage of effects with gnuplot, configure usage of critical values with gnuplot, submit user gnuplot options to gnuplot, set frames type based on animation type, use gnuplot to create frames
- *set_file_name():* on each call yield one name of frame output file
- *create_output_command():* use file name and temp directory name to create output command for gnuplot
- *configure_crit_values():* use critical values configured using -c to create appropriate gnuplot commands
- *get_record_extremes():* determine global maximum/minimum date/time values
- *configure_xy_basics():* set terminal type, x axis data type, set grid, unset key, set axis dimensions, set axis labels
- *create_frames_oneline():* read all lines from all files into memory, randomize lines, create gnuplot plot commands for “oneline” animation type
- *create_frames_multiplot():* read all lines from all files into memory, randomize lines, create gnuplot plot commands for “multiplot” animation type
- *set_scheme_lines():* on each call yield one gnuplot linetype with colors based on chosen scheme
- *configure_effects():* set terminal size, set backgroud color, set line colors and types

skj_animation.py

- Functions manipulating animation properties and creating final animation
- set_animation_properties(): set animation type, number of valid records, calculate speed/time/fps, create sequence of records added to every created frame

- *create_animation():* determine output directory name, try to create output directory, call ffmpeg with correct parameters
- *calculate_sfft():* calculate speed, calculate fps, calculate time, calculate number of frames
- *determine_anim_type():* based on each files date minimum/maximum and global date minimum/maximum determine if animation type should be “oneline” or “multiplot”
- *create_speed_seq():* create a *list()* of integers representing number of records to be added to frame on each frame creation, split speed into integer and fractional part, check if fractional part is non-zero, add two fractional parts, add fractional part to sequence each time it gets > 1
- *get_anim_records():* on each call number of records for one source file

skj_std.py

- Functions && variables used in all other modules providing most common and basic operations
- *create_error_msg():* return *str()* containing error message created based on function parametrs
- *create_info_msg():* return *str()* containing info message created based on function parametrs
- *create_debug_msg():* return *str()* containing debug message created based on function parametrs
- *print_msg_verbose():* print error/info/debug message based on verbosity level && parameters
- *download_url():* download file specified by url, try to save it to temporary directory
- *cleanup_temp_files():* clean temporary files
- *create_temp_files():* create directory hiearchy for temporary files
- *exit_with_failure():* print error message as specified by exception that caused the exit, clean temp files, exit with correct exit code
- *exit_with_success():* clean temp files, exit with correct exit code