

CSCI3130 Grader

Generated by Doxygen 1.8.14

Contents

1	README	1
2	Namespace Index	3
2.1	Namespace List	3
3	Hierarchical Index	5
3.1	Class Hierarchy	5
4	Class Index	7
4.1	Class List	7
5	File Index	9
5.1	File List	9
6	Namespace Documentation	11
6.1	create_dates_diag Namespace Reference	11
6.2	dates_window Namespace Reference	11
6.3	db_init Namespace Reference	11
6.3.1	Function Documentation	12
6.3.1.1	commit_gen_report()	12
6.3.1.2	export_pdf()	13
6.3.1.3	gen_filenotfound_resp()	14
6.3.1.4	gen_report()	15
6.3.1.5	generate_final_grades()	16

6.3.1.6	get_all_grades_by_lid()	16
6.3.1.7	get_due_date_by_labid()	17
6.3.1.8	get_empty_grades_by_lid()	18
6.3.1.9	get_full_path()	18
6.3.1.10	get_grades_by_lab_and_att()	19
6.3.1.11	get_ids_in_class_by_year_semester()	19
6.3.1.12	get_import_dates_by_labid()	20
6.3.1.13	get_lab_filename()	21
6.3.1.14	get_lab_id()	21
6.3.1.15	get_lab_max_value()	22
6.3.1.16	get_lab_names()	23
6.3.1.17	get_labid_in_schedule()	24
6.3.1.18	get_max_grade_for_lab()	24
6.3.1.19	get_pipeline_ids()	25
6.3.1.20	get_pipids_in_class_by_year_semester()	26
6.3.1.21	get_prev_resp()	27
6.3.1.22	get_resp_and_grade()	27
6.3.1.23	grades_db_create()	28
6.3.1.24	import_previous_grades_into_db()	30
6.3.1.25	init_new_lab()	32
6.3.1.26	insert_students()	33
6.3.1.27	load_student_list_into_grades_db()	34
6.3.1.28	reconstruct_grades_and_comments()	35
6.3.1.29	register_lab_in_semester()	35
6.3.1.30	register_students_in_class()	36
6.3.1.31	save_a_grade_to_db()	37
6.3.1.32	save_grade_and_report()	38
6.3.1.33	settings_db_create()	38

6.3.1.34	settings_db_read_settings()	39
6.3.1.35	sync_files()	40
6.3.1.36	update_lab_submissions_paths()	41
6.3.1.37	update_settings()	42
6.3.2	Variable Documentation	43
6.3.2.1	SETTINGS_DB_NAME	43
6.4	generate Namespace Reference	43
6.4.1	Function Documentation	43
6.4.1.1	convert_to_pdf()	44
6.4.1.2	create_html_pdf_report2()	45
6.4.1.3	create_html_pdf_zero_report()	46
6.4.1.4	create_not_submitted()	50
6.4.1.5	generate_answers3()	51
6.4.1.6	time_to_str_with_tz()	53
6.5	main Namespace Reference	53
6.5.1	Function Documentation	54
6.5.1.1	get_grading_period()	54
6.5.1.2	read_settings()	55
6.5.2	Variable Documentation	56
6.5.2.1	app	56
6.5.2.2	MAIN_FILE_NAME	56
6.5.2.3	MAIN_FILE_NAME_OVERRIDE	56
6.5.2.4	MainWindow	57
6.5.2.5	styleData	57
6.5.2.6	ui	57
6.6	main_window Namespace Reference	57
6.7	manage_labs Namespace Reference	58
6.8	qt_class_improvements Namespace Reference	58
6.9	settings Namespace Reference	58
6.10	simple_dialog Namespace Reference	58

7	Class Documentation	59
7.1	qt_class_improvements.BetterLineEdit Class Reference	59
7.1.1	Detailed Description	60
7.1.2	Constructor & Destructor Documentation	60
7.1.2.1	__init__()	60
7.1.3	Member Function Documentation	60
7.1.3.1	eventFilter()	61
7.1.4	Member Data Documentation	61
7.1.4.1	dclicked	61
7.2	qt_class_improvements.BetterPlainTextEdit Class Reference	61
7.2.1	Detailed Description	62
7.2.2	Constructor & Destructor Documentation	62
7.2.2.1	__init__()	62
7.2.3	Member Function Documentation	63
7.2.3.1	eventFilter()	63
7.2.4	Member Data Documentation	63
7.2.4.1	focus_lost	63
7.3	main.CircFile.circ_type Class Reference	63
7.3.1	Detailed Description	64
7.3.2	Constructor & Destructor Documentation	64
7.3.2.1	__init__()	64
7.3.3	Member Data Documentation	64
7.3.3.1	input_pins	64
7.3.3.2	name	64
7.3.3.3	output_pins	65
7.4	main.CircFile Class Reference	65
7.4.1	Detailed Description	65
7.4.2	Constructor & Destructor Documentation	65

7.4.2.1	<code>__init__()</code>	66
7.4.3	Member Function Documentation	66
7.4.3.1	<code>get_parsed_pins()</code>	66
7.4.3.2	<code>get_parsed_pins2()</code>	67
7.4.4	Member Data Documentation	68
7.4.4.1	<code>filename</code>	68
7.4.4.2	<code>final_grade</code>	69
7.4.4.3	<code>subtract</code>	69
7.5	main.Grader Class Reference	69
7.5.1	Detailed Description	70
7.5.2	Constructor & Destructor Documentation	71
7.5.2.1	<code>__init__()</code>	71
7.5.3	Member Function Documentation	71
7.5.3.1	<code>add_to_common_answers()</code>	71
7.5.3.2	<code>check_circ_exist()</code>	72
7.5.3.3	<code>check_file()</code>	72
7.5.3.4	<code>check_files()</code>	73
7.5.3.5	<code>check_pins_facing()</code>	74
7.5.3.6	<code>check_wrong()</code>	75
7.5.3.7	<code>generate_response()</code>	75
7.5.3.8	<code>get_parsed_pins()</code>	76
7.5.3.9	<code>get_stud_circ_ind()</code>	77
7.5.3.10	<code>get_stud_id()</code>	77
7.5.3.11	<code>log_update()</code>	78
7.5.3.12	<code>next_circ()</code>	78
7.5.3.13	<code>open_dir()</code>	79
7.5.3.14	<code>precheck_PLDs()</code>	81
7.5.3.15	<code>prev_circ()</code>	83

7.5.3.16	read_prev_resp()	84
7.5.3.17	read_prev_resp2()	84
7.5.3.18	read_resp()	85
7.5.3.19	read_resp2()	86
7.5.3.20	save_all()	87
7.5.3.21	save_all2()	87
7.5.3.22	save_grade()	88
7.5.3.23	save_responce()	89
7.5.4	Member Data Documentation	89
7.5.4.1	all_my_circuits	89
7.5.4.2	attempt	90
7.5.4.3	circ_file_name	90
7.5.4.4	circ_obj_ref	90
7.5.4.5	cur_idx	90
7.5.4.6	file_list	90
7.5.4.7	final_grade	90
7.5.4.8	global_log	91
7.5.4.9	grader	91
7.5.4.10	input_correct	91
7.5.4.11	input_suggestion	91
7.5.4.12	lab_id	91
7.5.4.13	lab_max_grade	91
7.5.4.14	lab_num	92
7.5.4.15	lab_paths	92
7.5.4.16	lab_type	92
7.5.4.17	lid	92
7.5.4.18	logisim_pid	92
7.5.4.19	output_correct	92

7.5.4.20	previous_responses	93
7.5.4.21	resp_len	93
7.5.4.22	resp_text	93
7.5.4.23	semester	93
7.5.4.24	stud_id	93
7.5.4.25	stud_ids	93
7.5.4.26	submitted	94
7.5.4.27	subtract	94
7.5.4.28	time	94
7.5.4.29	time_cur	94
7.5.4.30	time_cur_qt	94
7.5.4.31	time_from	94
7.5.4.32	time_from_qt	95
7.5.4.33	time_to	95
7.5.4.34	time_to_qt	95
7.5.4.35	timestamps	95
7.5.4.36	to_date	95
7.5.4.37	tot_elem	95
7.5.4.38	user_comment	96
7.5.4.39	what_to_grade	96
7.5.4.40	working_dir	96
7.6	main.CircFile.PinType Class Reference	96
7.6.1	Detailed Description	96
7.6.2	Constructor & Destructor Documentation	97
7.6.2.1	__init__()	97
7.6.3	Member Data Documentation	97
7.6.3.1	facing	97
7.6.3.2	name	97

7.6.3.3	type	97
7.7	main.SimpleDialog Class Reference	98
7.7.1	Detailed Description	99
7.7.2	Member Function Documentation	99
7.7.2.1	setupUi()	99
7.8	create_dates_diag.Ui_Create_dates_dialog Class Reference	99
7.8.1	Detailed Description	100
7.8.2	Member Function Documentation	101
7.8.2.1	retranslateUi()	101
7.8.2.2	setupUi()	101
7.8.3	Member Data Documentation	102
7.8.3.1	buttonBox	102
7.8.3.2	first_label	103
7.8.3.3	first_subm_date_time	103
7.8.3.4	horizontalLayout	103
7.8.3.5	horizontalLayout_2	103
7.8.3.6	horizontalLayout_3	103
7.8.3.7	horizontalLayout_4	103
7.8.3.8	horizontalLayout_5	104
7.8.3.9	init_label	104
7.8.3.10	init_subm_date_time	104
7.8.3.11	lab_path	104
7.8.3.12	second_label	104
7.8.3.13	second_subm_date_time	104
7.8.3.14	third_label	105
7.8.3.15	third_subm_date_time	105
7.8.3.16	verticalLayout	105
7.9	main.Ui_Create_dates_dialog1 Class Reference	105

7.9.1	Detailed Description	106
7.9.2	Member Function Documentation	106
7.9.2.1	bind_functions()	107
7.9.2.2	date_select()	108
7.9.2.3	open_file_diag()	108
7.9.2.4	setupUi()	109
7.10	main.Ui_Create_settings_dialog Class Reference	111
7.10.1	Detailed Description	112
7.10.2	Member Function Documentation	112
7.10.2.1	bind_functions()	112
7.10.2.2	create_or_update_settings_db()	113
7.10.2.3	import_students()	115
7.10.2.4	open_simple_dialog()	116
7.10.2.5	read_settings_data()	117
7.10.2.6	set_apply_rested_active()	118
7.10.2.7	set_default_user_input_with_paths()	118
7.10.2.8	setupUi()	119
7.10.2.9	update_user_input_with_paths()	120
7.10.3	Member Data Documentation	121
7.10.3.1	simple_diag	122
7.11	dates_window.Ui_dates_window Class Reference	122
7.11.1	Detailed Description	123
7.11.2	Member Function Documentation	123
7.11.2.1	retranslateUi()	123
7.11.2.2	setupUi()	123
7.11.3	Member Data Documentation	124
7.11.3.1	buttonBox	124
7.11.3.2	calendarWidget	124

7.12	simple_dialog.Ui_Dialog Class Reference	124
7.12.1	Detailed Description	125
7.12.2	Member Function Documentation	125
7.12.2.1	retranslateUi()	125
7.12.2.2	setupUi()	126
7.12.3	Member Data Documentation	126
7.12.3.1	buttonBox_simple_dial	126
7.12.3.2	label_main_question	126
7.12.3.3	verticalLayout	127
7.13	main_window.Ui_mainWindow Class Reference	127
7.13.1	Detailed Description	129
7.13.2	Member Function Documentation	129
7.13.2.1	retranslateUi()	129
7.13.2.2	setupUi()	130
7.13.3	Member Data Documentation	135
7.13.3.1	but_begin	135
7.13.3.2	but_create_report	135
7.13.3.3	but_file_open	135
7.13.3.4	but_next	135
7.13.3.5	but_prev	136
7.13.3.6	but_regrade	136
7.13.3.7	but_reset	136
7.13.3.8	but_save_all	136
7.13.3.9	but_save_response	136
7.13.3.10	centralwidget	136
7.13.3.11	check_autosave	137
7.13.3.12	checkB_input_pin_status	137
7.13.3.13	checkB_output_pin_status	137

7.13.3.14 checkB_wrong	137
7.13.3.15 dateTimeEdit_from	137
7.13.3.16 dateTimeEdit_submitted	137
7.13.3.17 dateTimeEdit_to	138
7.13.3.18 filename_lineEdit	138
7.13.3.19 horizontalLayout	138
7.13.3.20 horizontalLayout_10	138
7.13.3.21 horizontalLayout_11	138
7.13.3.22 horizontalLayout_12	138
7.13.3.23 horizontalLayout_2	139
7.13.3.24 horizontalLayout_3	139
7.13.3.25 horizontalLayout_4	139
7.13.3.26 horizontalLayout_5	139
7.13.3.27 horizontalLayout_6	139
7.13.3.28 horizontalLayout_7	139
7.13.3.29 horizontalLayout_8	140
7.13.3.30 horizontalLayout_9	140
7.13.3.31 input_attempt	140
7.13.3.32 input_current_id	140
7.13.3.33 input_file_location	140
7.13.3.34 input_final_grade	140
7.13.3.35 input_log_browser	141
7.13.3.36 input_max_pos_grade	141
7.13.3.37 input_message_to_all	141
7.13.3.38 input_prev_response	141
7.13.3.39 input_response_browser	141
7.13.3.40 input_response_browser_user	141
7.13.3.41 input_subtract	142

7.13.3.42 label_attempt	142
7.13.3.43 label_current_id	142
7.13.3.44 label_final	142
7.13.3.45 label_from	142
7.13.3.46 label_max_pos	142
7.13.3.47 label_submitted	143
7.13.3.48 label_subtr	143
7.13.3.49 label_to	143
7.13.3.50 log_tab	143
7.13.3.51 manage_labs_but	143
7.13.3.52 popular_answers	143
7.13.3.53 progressBar	144
7.13.3.54 response_tab	144
7.13.3.55 set_style_checkbox	144
7.13.3.56 settings_but	144
7.13.3.57 splitter	144
7.13.3.58 tab_message_to_all	144
7.13.3.59 tab_prev_resp	145
7.13.3.60 tabs_for_log_and_resp	145
7.13.3.61 verticalLayout	145
7.13.3.62 verticalLayout_2	145
7.13.3.63 verticalLayout_3	145
7.13.3.64 verticalLayout_4	145
7.13.3.65 verticalLayout_5	146
7.13.3.66 verticalLayout_6	146
7.13.3.67 verticalLayout_7	146
7.13.3.68 verticalLayout_8	146
7.13.3.69 verticalLayout_9	146

7.14	manage_labs.Ui_manage_labs Class Reference	147
7.14.1	Detailed Description	148
7.14.2	Member Function Documentation	148
7.14.2.1	retranslateUi()	148
7.14.2.2	setupUi()	148
7.14.3	Member Data Documentation	149
7.14.3.1	create_due_dates_but	149
7.14.3.2	export_but	149
7.14.3.3	horizontalLayout	150
7.14.3.4	import_but	150
7.14.3.5	labs_select_comboBox	150
7.14.3.6	status_bar	150
7.14.3.7	sync_but	150
7.14.3.8	verticalLayout	150
7.15	main.Ui_manage_labs1 Class Reference	151
7.15.1	Detailed Description	152
7.15.2	Member Function Documentation	152
7.15.2.1	bind_functions()	152
7.15.2.2	check_for_due_dates()	153
7.15.2.3	due_date_creator()	154
7.15.2.4	export_pdfs()	154
7.15.2.5	import_lab()	155
7.15.2.6	open_dates_dialog()	158
7.15.2.7	scan_for_labs()	159
7.15.2.8	set_local_vars()	159
7.15.2.9	setupUi()	160
7.15.2.10	sync_files()	161
7.15.2.11	update_status_bar()	162

7.15.3	Member Data Documentation	163
7.15.3.1	cal_window	163
7.15.3.2	main_lab_path	163
7.15.3.3	pdf_files_len	163
7.15.3.4	selected_lab_name	164
7.15.3.5	selected_path	164
7.15.3.6	srv_sync_path	164
7.15.3.7	zip_files_len	164
7.16	settings.Ui_Settings Class Reference	165
7.16.1	Detailed Description	166
7.16.2	Member Function Documentation	166
7.16.2.1	retranslateUi()	167
7.16.2.2	setupUi()	167
7.16.3	Member Data Documentation	170
7.16.3.1	buttonBox	170
7.16.3.2	formLayout	170
7.16.3.3	formLayout_2	171
7.16.3.4	gridLayout	171
7.16.3.5	groupBox_db	171
7.16.3.6	groupBox_local	171
7.16.3.7	groupBox_user	171
7.16.3.8	import_stuents_btn	171
7.16.3.9	input_grader_name	172
7.16.3.10	input_grades_db	172
7.16.3.11	input_local_stor	172
7.16.3.12	input_logisim_path	172
7.16.3.13	input_rem_stor	172
7.16.3.14	input_settings_db	172

7.16.3.15 label_grad_year	173
7.16.3.16 label_grader_name	173
7.16.3.17 label_grades_db	173
7.16.3.18 label_local_stor	173
7.16.3.19 label_logisim_path	173
7.16.3.20 label_rem_stor	173
7.16.3.21 label_semester	174
7.16.3.22 label_settings_db	174
7.16.3.23 label_style	174
7.16.3.24 label_sync_comm	174
7.16.3.25 semester_comboBox	174
7.16.3.26 spin_year	174
7.16.3.27 style_checkBox	175
7.16.3.28 sync_command	175
7.16.3.29 verticalLayout	175
7.17 main.UiMainWindow1 Class Reference	175
7.17.1 Detailed Description	177
7.17.2 Constructor & Destructor Documentation	177
7.17.2.1 __init__()	177
7.17.3 Member Function Documentation	177
7.17.3.1 bind_functions()	178
7.17.3.2 change_win_style()	180
7.17.3.3 check_file()	181
7.17.3.4 check_wrong()	181
7.17.3.5 disable_fields()	182
7.17.3.6 dummy_d_1()	182
7.17.3.7 enable_fields()	183
7.17.3.8 generate_reports()	183

7.17.3.9	kill_logisim()	184
7.17.3.10	load_dir()	185
7.17.3.11	memorize_user_comment()	186
7.17.3.12	my_open_file()	187
7.17.3.13	next_circ()	188
7.17.3.14	open_file_diag()	189
7.17.3.15	open_manage_labs_diag()	190
7.17.3.16	open_settings_dialog()	190
7.17.3.17	prev_circ()	191
7.17.3.18	regrade()	192
7.17.3.19	reset_grade_resp()	193
7.17.3.20	run_logisim()	194
7.17.3.21	save_all()	195
7.17.3.22	save_grade()	196
7.17.3.23	save_response()	196
7.17.3.24	setupUi()	197
7.17.3.25	show_stat()	198
7.17.3.26	sync_params_to_settings()	199
7.17.3.27	track_final_grade()	200
7.17.3.28	update_popular_answers()	200
7.17.3.29	update_user_comment_from_popular_answers()	201
7.17.4	Member Data Documentation	201
7.17.4.1	cal_window	202
7.17.4.2	class_id_to_id	202
7.17.4.3	current_tz	202
7.17.4.4	grader_name	202
7.17.4.5	grader_ref	202
7.17.4.6	logisim_path	202
7.17.4.7	manage_labs_window	203
7.17.4.8	settings_window	203
7.17.4.9	working_dir	203

8	File Documentation	205
8.1	create_dates_diag.py File Reference	205
8.2	dates_window.py File Reference	205
8.3	db_init.py File Reference	205
8.4	generate.py File Reference	207
8.5	main.py File Reference	207
8.6	main_window.py File Reference	208
8.7	manage_labs.py File Reference	208
8.8	qt_class_improvements.py File Reference	208
8.9	README.md File Reference	209
8.10	settings.py File Reference	209
8.11	simple_dialog.py File Reference	209

Chapter 1

README

Chapter 2

Namespace Index

2.1 Namespace List

Here is a list of all namespaces with brief descriptions:

create_dates_diag	11
dates_window	11
db_init	11
generate	43
main	53
main_window	57
manage_labs	58
qt_class_improvements	58
settings	58
simple_dialog	58

Chapter 3

Hierarchical Index

3.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

main.CircFile.circ_type	63
main.CircFile	65
main.Grader	69
object	
create_dates_diag.Ui_Create_dates_dialog	99
main.Ui_Create_dates_dialog1	105
dates_window.Ui_dates_window	122
main_window.Ui_mainWindow	127
main.UiMainWindow1	175
manage_labs.Ui_manage_labs	147
main.Ui_manage_labs1	151
settings.Ui_Settings	165
main.Ui_Create_settings_dialog	111
simple_dialog.Ui_Dialog	124
main.SimpleDialog	98
main.CircFile.PinType	96
QLineEdit	
qt_class_improvements.BetterLineEdit	59
QPlainTextEdit	
qt_class_improvements.BetterPlainTextEdit	61

Chapter 4

Class Index

4.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

qt_class_improvements.BetterLineEdit	59
qt_class_improvements.BetterPlainTextEdit	61
main.CircFile.circ_type	63
main.CircFile	65
main.Grader	69
main.CircFile.PinType	96
main.SimpleDialog	
Wrapper class for very simple Ok Cancel dialog	98
create_dates_diag.Ui_Create_dates_dialog	99
main.Ui_Create_dates_dialog1	105
main.Ui_Create_settings_dialog	
Creates window that provides user with convenient way of changing settings that are stored in sqlite3	
db	111
dates_window.Ui_dates_window	122
simple_dialog.Ui_Dialog	124
main_window.Ui_mainWindow	127
manage_labs.Ui_manage_labs	147
main.Ui_manage_labs1	151
settings.Ui_Settings	165
main.UiMainWindow1	175

Chapter 5

File Index

5.1 File List

Here is a list of all files with brief descriptions:

create_dates_diag.py	205
dates_window.py	205
db_init.py	205
generate.py	207
main.py	207
main_window.py	208
manage_labs.py	208
qt_class_improvements.py	208
settings.py	209
simple_dialog.py	209

Chapter 6

Namespace Documentation

6.1 create_dates_diag Namespace Reference

Classes

- class [Ui_Create_dates_dialog](#)

6.2 dates_window Namespace Reference

Classes

- class [Ui_dates_window](#)

6.3 db_init Namespace Reference

Functions

- def [settings_db_create](#) (db_name=[SETTINGS_DB_NAME](#), force=False)
- def [settings_db_read_settings](#) (db_name=[SETTINGS_DB_NAME](#))
- def [update_settings](#) (paths, local, db_name=[SETTINGS_DB_NAME](#))
- def [grades_db_create](#) (db_name, force=False)
- def [load_student_list_into_grades_db](#) (db_name, year, semester, filename='students_list3.txt')
- def [insert_students](#) (ids, fname, lname, db_name='./grades.sqlite3')
- def [register_students_in_class](#) (pipeline_ids, year, semester, db_name='./grades.sqlite3')
- def [get_pipeline_ids](#) (db_name='./grades.sqlite3')
- def [get_ids_in_class_by_year_semester](#) (year, semester, db_name='./grades.sqlite3')
- def [import_previous_grades_into_db](#) (year, semester, db_name='./grades.sqlite3', filename='./grades.xls')
- def [gen_filenotfound_resp](#) (lab_id, stud_path, corr_file, grader, att=None, next_date=None, db_name='./grades.sqlite3')
- def [get_resp_and_grade](#) (grade_id, db_name='./grades.sqlite3')

- def [get_prev_resp](#) (grade_id, class_id, lab_id, db_name='./grades.sqlite3')
- def [save_a_grade_to_db](#) (grade_id, grade, grader_comment, extra_comment, grader_name, graded=True, pass_fail=True, db_name='./grades.sqlite3')
- def [init_new_lab](#) (stud_id, lab_name, att, submitted, lab_path, db_name='./grades.sqlite3')
- def [get_lab_names](#) (db_name='./grades.sqlite3')
- def [update_lab_submissions_paths](#) (db_name, repository_root, year, semester)
- def [get_empty_grades_by_lid](#) (lab_id, att, db_name='./grades.sqlite3')
- def [get_all_grades_by_lid](#) (lab_id, att, db_name='./grades.sqlite3')
- def [reconstruct_grades_and_comments](#) (db_name='./grades.sqlite3')
- def [generate_final_grades](#) (db_name, year, semester)
- def [get_max_grade_for_lab](#) (lid, year, semester, db_name='./grades.sqlite3')
- def [get_grades_by_lab_and_att](#) (lid, att, db_name='./grades.sqlite3')
- def [get_lab_filename](#) (lab_id, db_name='./grades.sqlite3')
- def [get_lab_max_value](#) (lab_id, db_name='./grades.sqlite3')
- def [get_full_path](#) (paths, local)
- def [sync_files](#) (self=None)
- def [export_pdf](#) (self=None)
- def [save_grade_and_report](#) (grade_id, grade, report, user_comment, grader, db_name='./grades.sqlite3')
- def [commit_gen_report](#) (grade_id, db_name='./grades.sqlite3')
- def [get_lab_id](#) (ltype, lab_num)
- def [register_lab_in_semester](#) (ltype, lab_num, year, semester, due_dates, db_name='./grades.sqlite3')
- def [get_labid_in_schedule](#) (lid, year, semester, db_name='./grades.sqlite3')
- def [get_due_date_by_labid](#) (lid_sem, att=None, db_name='./grades.sqlite3')
- def [get_import_dates_by_labid](#) (lid_sem, att=None, db_name='./grades.sqlite3')
- def [gen_report](#) (lid_sem, att=None, db_name='./grades.sqlite3')
- def [get_pipids_in_class_by_year_semester](#) (year, semester, db_name='./grades.sqlite3')

Variables

- string [SETTINGS_DB_NAME](#) = 'settings.sqlite3'

6.3.1 Function Documentation

6.3.1.1 [commit_gen_report\(\)](#)

```
def db_init.commit_gen_report (
    grade_id,
    db_name = './grades.sqlite3' )
```

Definition at line 749 of file db_init.py.

```
749 def commit\_gen\_report (grade_id, db_name='./grades.sqlite3'):
750     if not os.path.isfile(db_name):
751         raise Exception("DB not found")
752     with lite.connect(db_name) as con:
753         cur = con.cursor()
754         cur.execute("UPDATE grades SET report_generated=strftime('%s','now') WHERE id=?", (grade_id,))
755         con.commit()
756
757
758
```

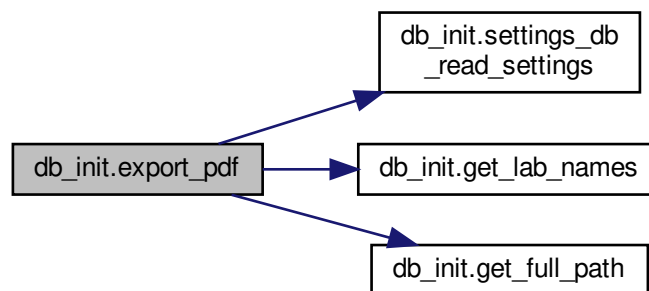

6.3.1.2 export_pdf()

```
def db_init.export_pdf (
    self = None )
```

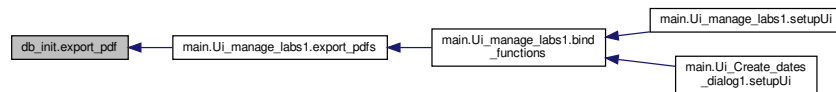
Definition at line 709 of file db_init.py.

```
709 def export_pdf(self=None):
710     import subprocess
711     import os
712
713     paths, local = settings_db_read_settings()
714     lab_ids, lab_types, lab_nums = get_lab_names()
715     lab_names = []
716     for i in range(len(lab_types)):
717         lab_names.append(lab_types[i] + '_Lab_' + str(lab_nums[i]))
718
719     full_path = get_full_path(paths, local) + "/"
720     for lab_name in lab_names:
721         nums_to_sync = '_'
722         i = 1
723         while os.path.isdir(full_path + lab_name + '_' + str(i) + '/Answers'):
724             nums_to_sync += str(i) + ','
725             i += 1
726         if i == 1:
727             continue
728         nums_to_sync = nums_to_sync[0:-1] + '}'
729         # for case when we have only one directory to sync
730         if len(nums_to_sync) == 4:
731             nums_to_sync = '_1'
732         if len(nums_to_sync) > 1:
733             command = local[4] + ' ' + full_path + lab_name + nums_to_sync + '/Answers/*.pdf ' +
os.path.expanduser(paths[2]) + lab_name + '/'
734             process = subprocess.Popen(os.path.expandvars(command), stdout=subprocess.PIPE, shell=True)
735             process.communicate()
736             # print(output)
737             # print(error)
738
739
```

Here is the call graph for this function:



Here is the caller graph for this function:



6.3.1.3 gen_filenotfound_resp()

```

def db_init.gen_filenotfound_resp (
    lab_id,
    stud_path,
    corr_file,
    grader,
    att = None,
    next_date = None,
    db_name = './grades.sqlite3' )
  
```

Definition at line 408 of file db_init.py.

```

408 def gen_filenotfound_resp(lab_id, stud_path, corr_file, grader, att=None,
    next_date=None, db_name='./grades.sqlite3'):
409     resp_text = 'file with name "{}" was not found.</br>'.format(corr_file)
410     file_found = os.listdir(stud_path)
411     potential_files = list()
412     for file in file_found:
413         if file not in ['grade.txt', 'penalty.txt', 'responce.txt', 'tech_info.txt', ]:
414             potential_files.append(file)
415     if potential_files:
416         resp_text += '\nNext files|folders were found:</br>\n'
417     for file in potential_files:
418         if os.path.isdir(os.path.join(stud_path, file)):
419             resp_text += file + ' - directory.</br>\n'
420         else:
421             resp_text += file + ' - regular file.</br>\n'
422
423     if att and att < 4 and next_date:
424         resp_text += 'Please submit your file by next due date ({}).</br>\n'.format(next_date)
425
426     if not os.path.isfile(db_name):
427         raise Exception("DB not found")
428     with lite.connect(db_name) as con:
429         cur = con.cursor()
430         cur.execute("UPDATE grades SET graded=strftime('%s','now'), pass_fail=FALSE, grader_comment=?,
    grader=? WHERE id=?", (resp_text, grader, lab_id))
431         con.commit()
432
433
  
```

Here is the caller graph for this function:



6.3.1.4 gen_report()

```

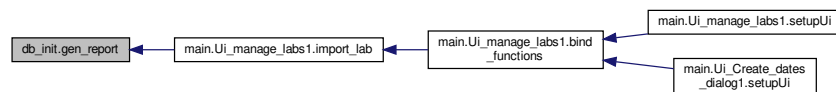
def db_init.gen_report (
    lid_sem,
    att = None,
    db_name = './grades.sqlite3' )
  
```

Definition at line 810 of file db_init.py.

```

810 def gen_report(lid_sem, att=None, db_name='./grades.sqlite3'):
811     if not os.path.isfile(db_name):
812         raise Exception("DB not found")
813     with lite.connect(db_name) as con:
814         cur = con.cursor()
815         cur.execute("UPDATE lab_schedule SET imported_{}=strftime('%s','now') WHERE id=?".format(att), (
            lid_sem,))
816         con.commit()
817
818
  
```

Here is the caller graph for this function:



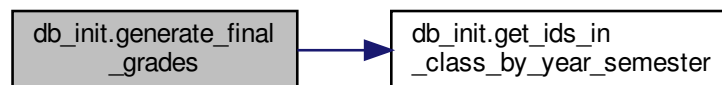
6.3.1.5 generate_final_grades()

```
def db_init.generate_final_grades (
    db_name,
    year,
    semester )
```

Definition at line 595 of file db_init.py.

```
595 def generate_final_grades(db_name, year, semester):
596     ids = get_ids_in_class_by_year_semester(year, semester, db_name)
597     with lite.connect(db_name) as con:
598         cur = con.cursor()
599
600         labs = list()
601         for sid in ids.values(): # using JOIN here will add too much extra data
602             result = cur.execute('SELECT lab, MAX(grade * (select percent from penalties where
603                 id=GRADES.attempt)/100) '
604                 'FROM GRADES WHERE class_id=? and attempt > 0 group by lab order by lab', (str(
605                 sid),))
606             labs.append(result.fetchall() )
607
608         stud_info = list()
609         for sid in ids.keys():
610             result = cur.execute('SELECT first_name, second_name FROM students WHERE pipeline_id=?', (str(
611             sid),))
612             stud_info.append(result.fetchall() )
613
614         df_stud_info = pd.DataFrame(dict(zip(ids.keys(), stud_info)))
615         df_grades = pd.DataFrame(dict(zip(ids.keys(), labs)))
616         # id_list = list(ids.keys())
617         # a = id_list[list(ids.values()).index(class_id)]
```

Here is the call graph for this function:



6.3.1.6 get_all_grades_by_lid()

```
def db_init.get_all_grades_by_lid (
    lab_id,
    att,
    db_name = './grades.sqlite3' )
```

Definition at line 540 of file db_init.py.

```

540 def get_all_grades_by_lid(lab_id, att, db_name='./grades.sqlite3'):
541     with lite.connect(db_name) as con:
542         cur = con.cursor()
543         result = cur.execute("SELECT submitted, class_id, id, lab_path FROM grades WHERE lab=? AND
attempt=? ", (lab_id, att))
544         try:
545             subm, class_id, lab_id, lab_path = zip(*result.fetchall())
546         except Exception as e:
547             print(e)
548             return None, None
549     return subm, class_id, lab_id, lab_path
550
551
552

```

6.3.1.7 get_due_date_by_labid()

```

def db_init.get_due_date_by_labid (
    lid_sem,
    att = None,
    db_name = './grades.sqlite3' )

```

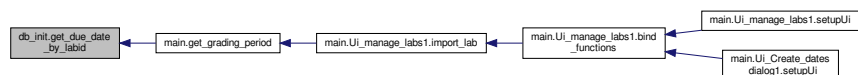
Definition at line 787 of file db_init.py.

```

787 def get_due_date_by_labid(lid_sem, att=None, db_name='./grades.sqlite3'):
788     with lite.connect(db_name) as con:
789         cur = con.cursor()
790         if att:
791             result = cur.execute('SELECT due_date_{0} FROM lab_schedule WHERE id={1}'.format(int(att)), (
lid_sem,))
792         else:
793             result = cur.execute('SELECT due_date_1, due_date_2, due_date_3, due_date_4 FROM lab_schedule
WHERE id=?', (lid_sem,))
794             return result.fetchone()
795     return None
796
797

```

Here is the caller graph for this function:



6.3.1.8 get_empty_grades_by_lid()

```
def db_init.get_empty_grades_by_lid (
    lab_id,
    att,
    db_name = './grades.sqlite3' )
```

Definition at line 527 of file db_init.py.

```
527 def get_empty_grades_by_lid(lab_id, att, db_name='./grades.sqlite3'):
528     with lite.connect(db_name) as con:
529         cur = con.cursor()
530         result = cur.execute("SELECT submitted, class_id, id, lab_path FROM grades WHERE lab=? AND
attempt=? AND graded is NULL", (lab_id, att))
531         try:
532             subm, class_id, lab_id, lab_path = zip(*result.fetchall())
533         except Exception as e:
534             # print(e)
535             return None, None, None, None
536
537     return subm, class_id, lab_id, lab_path
538
539
```

6.3.1.9 get_full_path()

```
def db_init.get_full_path (
    paths,
    local )
```

Definition at line 672 of file db_init.py.

```
672 def get_full_path(paths, local):
673     import os
674     return os.path.expanduser(paths[1]) + str(local[1]) + "_" + str(local[2])
675
676
```

Here is the caller graph for this function:



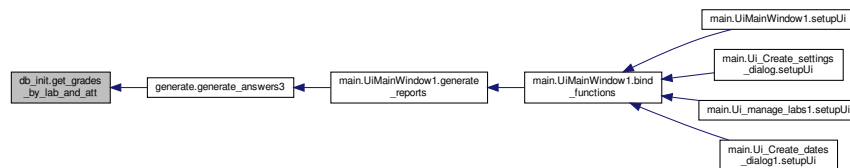
6.3.1.10 get_grades_by_lab_and_att()

```
def db_init.get_grades_by_lab_and_att (
    lid,
    att,
    db_name = './grades.sqlite3' )
```

Definition at line 635 of file db_init.py.

```
635 def get_grades_by_lab_and_att(lid, att, db_name='./grades.sqlite3'):
636     with lite.connect(db_name, detect_types=lite.PARSE_COLNAMES) as con:
637         cur = con.cursor()
638         result = cur.execute('select a.due_date_{0} as due_date, a.imported_{0} as import_date, '
639                             'b.type, b.num, b.max_grade, '
640                             'c.id as grade_id, c.submitted, c.graded, c.grade, c.pass_fail,
641                             c.grader_comment, c.extra_comment, c.grader, c.lab_path, '
642                             'd.pipeline_id, e.first_name, e.second_name, f.percent, c.grade*f.percent/100
643                             as final_grade '
644                             'from lab_schedule a '
645                             'join lab_names b on a.lab_id=b.id '
646                             'join grades c on c.lab=a.id '
647                             'join class d on d.id=c.class_id '
648                             'join students e on e.pipeline_id=d.pipeline_id '
649                             'join penalties f on f.id=c.attempt '
650                             'where c.attempt={0} AND a.id=? ORDER BY d.pipeline_id'.format(int(att)), (lid
651 ,))
652         info_tup = result.fetchall()
653         info_desc = result.description
654         return info_tup, info_desc
```

Here is the caller graph for this function:



6.3.1.11 get_ids_in_class_by_year_semester()

```
def db_init.get_ids_in_class_by_year_semester (
    year,
    semester,
    db_name = './grades.sqlite3' )
```

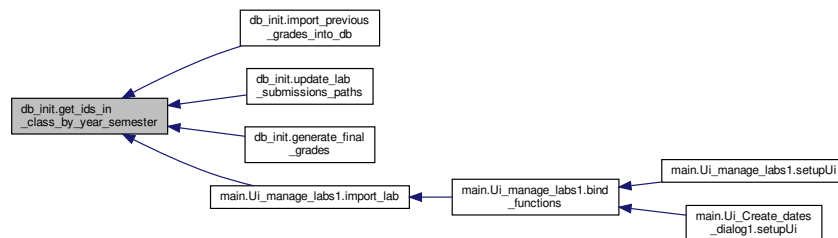
Definition at line 309 of file db_init.py.

```

309 def get_ids_in_class_by_year_semester(year, semester,
    db_name='./grades.sqlite3'):
310     with lite.connect(db_name) as con:
311         cur = con.cursor()
312         result = cur.execute("SELECT pipeline_id, id FROM class\
313                               WHERE year=" + str(year) + " and semester=" + str(semester))
314         try:
315             res = result.fetchall()
316             pip_to_id = dict(res)
317             to_id_to_pip = dict([(res_id[1], res_id[0]) for res_id in res])
318         except Exception as e:
319             print(e)
320             return None
321     return pip_to_id, to_id_to_pip
322
323
324 # Takes xls file with grades from previous semester(s) and loads all grades into DB.
325 # In case students are not found in the DB and xls file contains ids - loads them too
326 # :param year: year when grades were assigned
327 # :param semester: semester when grades were assigned
328 # :param db_name: specific name of the grades DB
329 # :param filename: xls file to load
330 # :return: nothing
331 #
332

```

Here is the caller graph for this function:



6.3.1.12 get_import_dates_by_labid()

```

def db_init.get_import_dates_by_labid (
    lid_sem,
    att = None,
    db_name = './grades.sqlite3' )

```

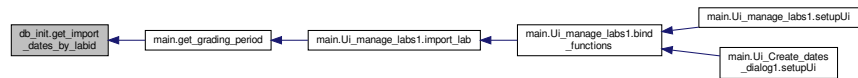
Definition at line 798 of file `db_init.py`.

```

798 def get_import_dates_by_labid(lid_sem, att=None, db_name='./grades.sqlite3'):
799     with lite.connect(db_name) as con:
800         cur = con.cursor()
801         if att:
802             result = cur.execute('SELECT imported_{0} FROM lab_schedule WHERE id=?'.format(int(att)), (
803                 lid_sem,))
804             else:
805                 result = cur.execute('SELECT imported_1, imported_2, imported_3, imported_4 FROM lab_schedule
806                                     WHERE id=?', (lid_sem,))
807                 return result.fetchone()
808             return None
809 # save_grade_and_report(self.grade_ids[self.cur_idx], self.final_grade, self.user_comment, self.grader)

```


Here is the caller graph for this function:



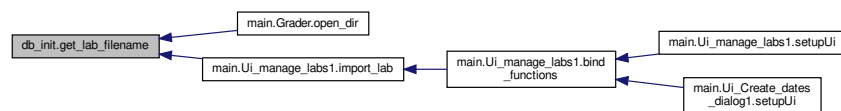
6.3.1.13 get_lab_filename()

```
def db_init.get_lab_filename (
    lab_id,
    db_name = './grades.sqlite3' )
```

Definition at line 654 of file db_init.py.

```
654 def get_lab_filename(lab_id, db_name='./grades.sqlite3'):
655     with lite.connect(db_name) as con:
656         cur = con.cursor()
657
658         result = cur.execute('SELECT mandatory_files FROM lab_names WHERE id=? ', (str(lab_id),))
659         return result.fetchall() [0]
660     return None
661
662
```

Here is the caller graph for this function:



6.3.1.14 get_lab_id()

```
def db_init.get_lab_id (
    ltype,
    lab_num )
```

Definition at line 759 of file db_init.py.

```

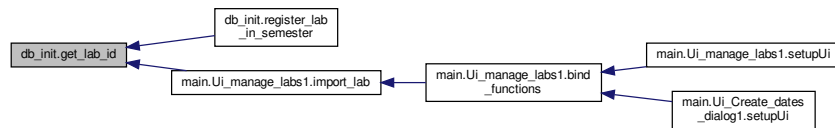
759 def get_lab_id(ltype, lab_num):
760     lab_ids, lab_types, lab_nums = get_lab_names()
761     for i, lid in enumerate(lab_ids):
762         if lab_types[i] == ltype and lab_num == lab_nums[i]:
763             return lid
764     return None
765
766

```

Here is the call graph for this function:



Here is the caller graph for this function:



6.3.1.15 get_lab_max_value()

```

def db_init.get_lab_max_value (
    lab_id,
    db_name = './grades.sqlite3' )

```

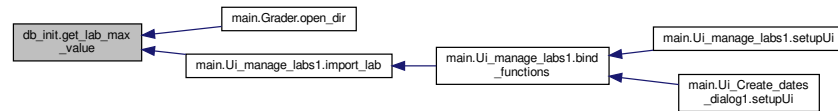
Definition at line 663 of file db_init.py.

```

663 def get_lab_max_value(lab_id, db_name='./grades.sqlite3'):
664     with lite.connect(db_name) as con:
665         cur = con.cursor()
666
667         result = cur.execute('SELECT max_grade FROM lab_names WHERE id=? ', (str(lab_id),))
668         return int(result.fetchone()[0])
669     return None
670
671

```

Here is the caller graph for this function:



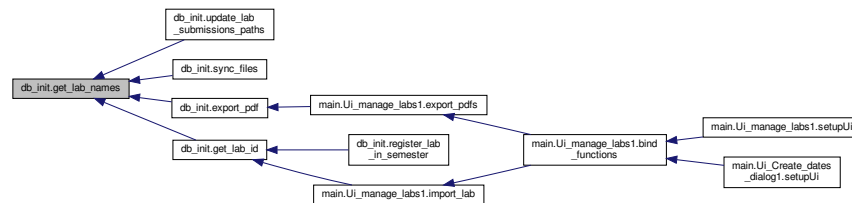
6.3.1.16 get_lab_names()

```
def db_init.get_lab_names (
    db_name = './grades.sqlite3' )
```

Definition at line 487 of file db_init.py.

```
487 def get_lab_names(db_name='./grades.sqlite3'):
488     with lite.connect(db_name) as con:
489         cur = con.cursor()
490         result = cur.execute("SELECT id, type, num FROM lab_names")
491         try:
492             lab_id, lab_type, lab_num = zip(*result.fetchall())
493         except Exception as e:
494             print(e)
495             return None, None, None
496     return lab_id, lab_type, lab_num
497
498
```

Here is the caller graph for this function:



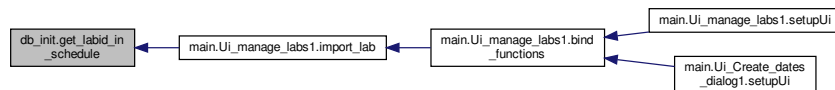
6.3.1.17 get_labid_in_schedule()

```
def db_init.get_labid_in_schedule (
    lid,
    year,
    semester,
    db_name = './grades.sqlite3' )
```

Definition at line 779 of file db_init.py.

```
779 def get_labid_in_schedule(lid, year, semester, db_name='./grades.sqlite3'):
780     with lite.connect(db_name) as con:
781         cur = con.cursor()
782         result = cur.execute('SELECT id FROM lab_schedule WHERE lab_id=? AND year=? AND semester=?', (lid,
783 year, semester))
784         fetched_red = result.fetchone()
785         return int(fetched_red[0]) if fetched_red is not None else None
786
```

Here is the caller graph for this function:



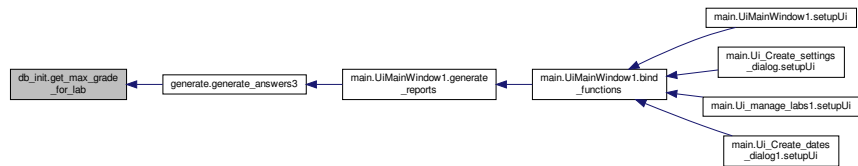
6.3.1.18 get_max_grade_for_lab()

```
def db_init.get_max_grade_for_lab (
    lid,
    year,
    semester,
    db_name = './grades.sqlite3' )
```

Definition at line 617 of file db_init.py.

```
617 def get_max_grade_for_lab(lid, year, semester, db_name='./grades.sqlite3'):
618     with lite.connect(db_name) as con:
619         cur = con.cursor()
620         result = cur.execute('SELECT e.pipeline_id as pipid, IFNULL(MAX(k.final_grade), 0) as grade '
621                                'FROM class e '
622                                'LEFT OUTER JOIN '
623                                ' (SELECT d.pipeline_id, c.grade*f.percent/100 AS final_grade '
624                                '   FROM grades c '
625                                '   JOIN class d ON d.id = c.class_id '
626                                '   JOIN penalties f ON f.id = c.attempt '
627                                '   WHERE c.lab = ? ) k '
628                                'ON e.pipeline_id = k.pipeline_id '
629                                'WHERE year=? AND semester=? '
630                                'GROUP BY e.pipeline_id '
631                                'ORDER BY e.pipeline_id ', (lid, int(year), int(semester)))
632         return result.fetchall()
633
634
```

Here is the caller graph for this function:



6.3.1.19 get_pipeline_ids()

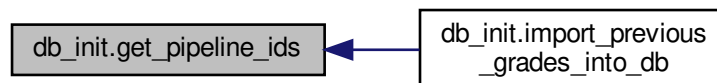
```
def db_init.get_pipeline_ids (
    db_name = './grades.sqlite3' )
```

Definition at line 290 of file db_init.py.

```

290 def get_pipeline_ids(db_name='./grades.sqlite3'):
291     with lite.connect(db_name) as con:
292         cur = con.cursor()
293         result = cur.execute("SELECT pipeline_id FROM students")
294         try:
295             resut = (ids[0] for ids in result.fetchall())
296         except Exception as e:
297             print(e)
298             return None
299     return result
300
301
302 # :param year:
303 # :param semester:
304 # :param db_name:
305 # :return:
306 #
307
308
```

Here is the caller graph for this function:



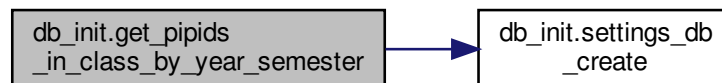
6.3.1.20 get_pipids_in_class_by_year_semester()

```
def db_init.get_pipids_in_class_by_year_semester (
    year,
    semester,
    db_name = './grades.sqlite3' )
```

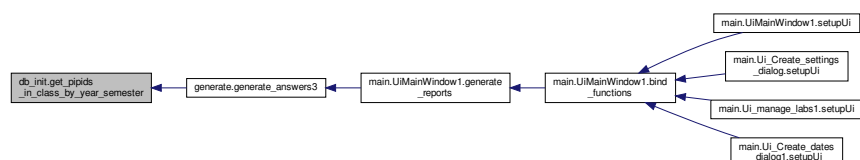
Definition at line 819 of file db_init.py.

```
819 def get_pipids_in_class_by_year_semester(year, semester,
      db_name='./grades.sqlite3'):
820     if not os.path.isfile(db_name):
821         raise Exception("DB not found")
822     with lite.connect(db_name) as con:
823         cur = con.cursor()
824         result = cur.execute('SELECT pipeline_id FROM class WHERE year=? AND semester=?', (year, semester))
825         all_ids = result.fetchall()
826         return [elem[0] for elem in all_ids]
827
828
829
```

Here is the call graph for this function:



Here is the caller graph for this function:



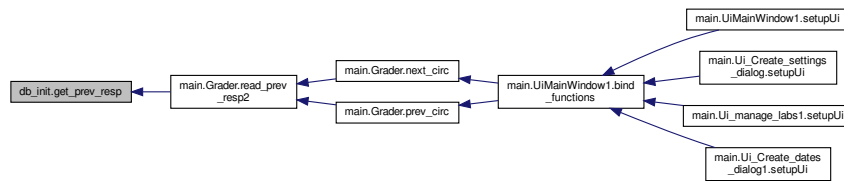
6.3.1.21 get_prev_resp()

```
def db_init.get_prev_resp (
    grade_id,
    class_id,
    lab_id,
    db_name = './grades.sqlite3' )
```

Definition at line 443 of file db_init.py.

```
443 def get_prev_resp(grade_id, class_id, lab_id, db_name='./grades.sqlite3'):
444     with lite.connect(db_name) as con:
445         cur = con.cursor()
446         result = cur.execute("SELECT grader_comment, extra_comment FROM grades WHERE class_id=? AND lab=?
AND id<?", (class_id, lab_id, grade_id))
447         res = result.fetchall()
448         if len(res) == 0:
449             return ''
450         else:
451             gresp, uresp = zip(*res)
452             return '\n'.join('{{ {} :{}\n}}'.format(gresp[i], uresp[i]) for i in range(len(gresp)))
453
454
```

Here is the caller graph for this function:



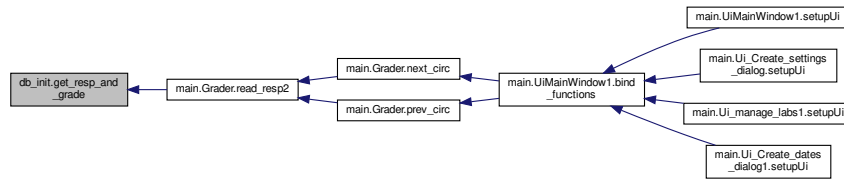
6.3.1.22 get_resp_and_grade()

```
def db_init.get_resp_and_grade (
    grade_id,
    db_name = './grades.sqlite3' )
```

Definition at line 434 of file db_init.py.

```
434 def get_resp_and_grade(grade_id, db_name='./grades.sqlite3'):
435     with lite.connect(db_name) as con:
436         cur = con.cursor()
437         result = cur.execute("SELECT grade, grader_comment, extra_comment, graded FROM grades WHERE id=?",
(grade_id,))
438         grade, resp, uresp, graded = result.fetchone()
439
440         return grade, resp, uresp, graded
441
442
```

Here is the caller graph for this function:



6.3.1.23 grades_db_create()

```
def db_init.grades_db_create (
    db_name,
    force = False )
```

Definition at line 94 of file db_init.py.

```

94 def grades_db_create(db_name, force=False):
95     # from pathlib import Path
96     print("I am going to create a grades DB with next name: ", db_name)
97     db_name = str(db_name)
98     if not os.path.isfile(db_name) or force:
99         # compute some vars before the connection
100         lab_names = list()
101         for i in range(1, 13):
102             lab_names.append(('CLA' + str(i), 'Closed', i, 10))
103         for i in range(1, 9):
104             lab_names.append(('OLA' + str(i), 'Open', i, 20))
105         lab_names.append(('OLA9', 'Open', 9, 100))
106
107     with lite.connect(db_name) as con:
108         cur = con.cursor()
109         # TODO: force should remove 'IF NOT EXISTS' and add 'DROP TABLE' to ensure new table creation
110         # WISH: add TRY blocks for each CREATE and spawn new info window in case of error
111         print('Creating students...')
112         cur.execute("""CREATE TABLE students (
113             pipeline_id    TEXT        NOT NULL
114                             PRIMARY KEY,
115             first_name     TEXT        NOT NULL,
116             second_name    TEXT        NOT NULL,
117             comment         TEXT,
118             cheating_ratio  INTEGER DEFAULT (0)    );""")
119         con.commit()
120         print('Done.')
121         print('Creating semesters...')
122         cur.execute("""CREATE TABLE semesters (
123             semester CHAR (1) NOT NULL PRIMARY KEY,
124             name      VARCHAR    );""")
125         con.commit()
126         print('Done.')
127         print('Creating class...')
128         cur.execute("""CREATE TABLE class (
129             id            INTEGER PRIMARY KEY AUTOINCREMENT,
130             pipeline_id   TEXT    REFERENCES students (pipeline_id),
131             year           INTEGER,
132             semester      INTEGER REFERENCES semesters (semester),
133             cheating_ratio INTEGER DEFAULT (0),
134             UNIQUE (
135                 pipeline_id,
136                 year,
```



```

137         semester)    );""")
138     con.commit()
139     print('Done.')
140     print('Creating labs...')
141     cur.execute("""CREATE TABLE lab_names (
142         id            INT      NOT NULL PRIMARY KEY,
143         type          TEXT     NOT NULL,
144         num           INTEGER NOT NULL,
145         max_grade     INTEGER NOT NULL,
146         name          VARCHAR,
147         description   VARCHAR,
148         grader_comment VARCHAR,
149         mandatory_files VARCHAR );""")
150     con.commit()
151     print('Done.')
152     print('Creating grades...')
153     cur.execute("""CREATE TABLE grades (
154         id            INTEGER PRIMARY KEY AUTOINCREMENT,
155         class_id      NOT NULL
156                     REFERENCES class (id) ON UPDATE CASCADE,
157         lab           NOT NULL
158                     REFERENCES lab_names (id) ON UPDATE CASCADE,
159         attempt       INT      DEFAULT (0),
160         submitted     INTEGER,
161         graded        INTEGER,
162         grade         INTEGER NOT NULL
163                     DEFAULT (0),
164         pass_fail     BOOLEAN NOT NULL
165                     DEFAULT (0),
166         grader_comment TEXT,
167         extra_comment TEXT,
168         report_generated BOOLEAN,
169         report_time   INTEGER,
170         lab_path       VARCHAR,
171         UNIQUE (
172             class_id,
173             lab,
174             attempt,
175             pass_fail) ON CONFLICT REPLACE );""")
176     con.commit()
177     print('Done.')
178
179     print('Creating lab schedule...')
180     cur.execute("""CREATE TABLE lab_schedule (
181         id            INTEGER PRIMARY KEY AUTOINCREMENT,
182         lab_id        REFERENCES lab_names (id),
183         year          INTEGER NOT NULL,
184         semester      INTEGER REFERENCES semesters (semester)
185                     NOT NULL,
186         due_date_1    INTEGER,
187         due_date_2    INTEGER,
188         due_date_3    INTEGER,
189         due_date_4    INTEGER,
190         imported_1    INTEGER,
191         imported_2    INTEGER,
192         imported_3    INTEGER,
193         imported_4    INTEGER,
194         posted_1      INTEGER,
195         posted_2      INTEGER,
196         posted_3      INTEGER,
197         posted_4      INTEGER
198     );""")
199     con.commit()
200     print('Done.')
201
202
203
204     print('Filling semesters...')
205     cur.executemany('INSERT OR REPLACE INTO semesters\
206         (semester, name) VALUES (?, ?)', [(1, 'SPRING'), (2, 'SUMMER'), (3, 'FALL')])
207     con.commit()
208     print('Done.')
209     print('Filling labs...')
210     cur.executemany('INSERT OR REPLACE INTO lab_names\
211         (id, type, num, max_grade) VALUES (?, ?, ?, ?)', lab_names)
212     con.commit()
213     print('Done.')
214     print('Vacuuming...')
215
216     cur.execute('VACUUM;')
217     con.commit()

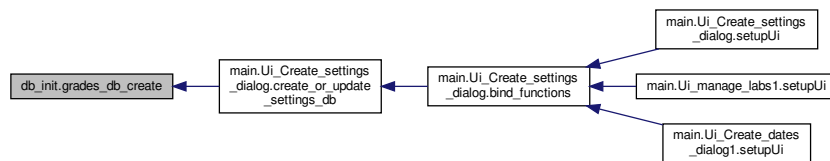
```

```

218
219         print('Done.')
220         print('Creation of GRADES DB finished.')
221
222         return True
223
224
225 # Imports list of students from file in format: `id % lname, fname` into Grades DB.
226 # Should be called before first grading.
227 # :param db_name: db that contains grades and student info
228 # :param year: grading (current) year
229 # :param semester: grading (current) semester
230 # :param filename: file that contains student list
231 # :return: nothing
232 #
233

```

Here is the caller graph for this function:



6.3.1.24 import_previous_grades_into_db()

```

def db_init.import_previous_grades_into_db (
    year,
    semester,
    db_name = './grades.sqlite3',
    filename = './grades.xls' )

```

Definition at line 333 of file `db_init.py`.

```

333 def import_previous_grades_into_db(year, semester,
    db_name='./grades.sqlite3', filename='./grades.xls'):
334     if not os.path.isfile(db_name):
335         raise Exception("DB not found")
336
337     df1 = pd.read_excel(filename)
338
339     try:
340         cls = df1.filter(like='CL')
341     except Exception as e:
342         print(e)
343         cls = None # no CLA's found
344
345     try:
346         ols = df1.filter(like='OL')
347     except Exception as e:
348         print(e)
349         ols = None # no OLAs found
350
351     try:
352         ids = df1.filter(like='sername').values.ravel().tolist()

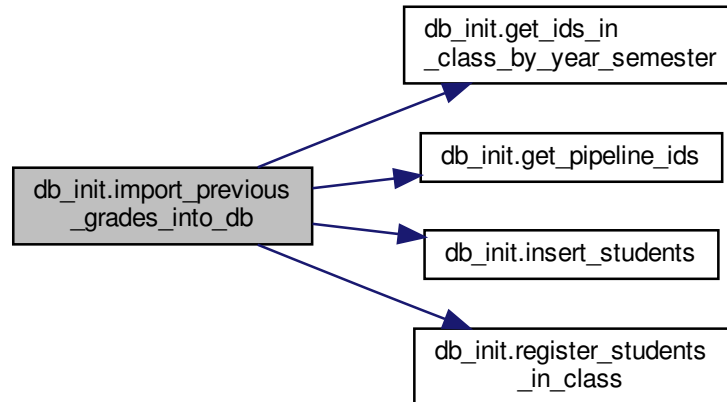
```

```

353         ids_len = len(ids)
354     except Exception as e:
355         print('Was not able to parse user ids, check xls file you are trying to import: ', e)
356         raise e # may be improved in the future - strange case
357     try:
358         names = df1.filter(like='Name').values.ravel().tolist()
359     except Exception as e: # either does not exist or has different name
360         print(e)
361         names = None
362
363     class_dict = get_ids_in_class_by_year_semester(year, semester, db_name
364 )
365
366     if (not class_dict and not names) or (class_dict and len(class_dict) < ids_len and not names):
367         raise Exception('Did not find ids in table CLASS and did not find names in xls file')
368     elif names and (not class_dict or (class_dict and len(class_dict) < ids_len)):
369         print('Did not find existing students, but found names in xls\nAdding new students...\n')
370         existing_ids = get_pipeline_ids(db_name)
371         need_to_update_students = False
372         # otherwise just add ids to the class list
373         if existing_ids:
374             for sid in ids:
375                 if sid not in existing_ids:
376                     need_to_update_students = True
377         else:
378             need_to_update_students = True
379
380         if need_to_update_students:
381             fname, lname = zip(*(name.split(' ', ' ') for name in names))
382             fname = (name.strip() for name in fname)
383             lname = (name.strip() for name in lname)
384             insert_students(ids, fname, lname, db_name)
385             register_students_in_class(ids, year, semester, db_name)
386
387         class_ids = [class_dict[sid] for sid in ids]
388         if ols is None and cls is None or len(class_ids) == 0:
389             raise Exception('No grades to load')
390
391         grades_tuples = list()
392         if ols is not None:
393             for lab_name in ols:
394                 grades = (str(grade) for grade in ols[lab_name].values)
395                 grades_tuples += list(zip(class_ids, [lab_name] * ids_len, [-1] * ids_len, grades, ['TRUE'] *
396 ids_len))
397
398         if cls is not None:
399             for lab_name in cls:
400                 grades = (str(grade) for grade in cls[lab_name].values)
401                 grades_tuples += list(zip(class_ids, [lab_name] * ids_len, [-1] * ids_len, grades, ['TRUE'] *
402 ids_len))
403
404         with lite.connect(db_name) as con:
405             cur = con.cursor()
406             cur.executemany('INSERT OR REPLACE INTO grades\
407                 (class_id, lab, attempt, grade, pass_fail) VALUES (?, ?, ?, ?, ?)', grades_tuples)
408             con.commit()

```

Here is the call graph for this function:



6.3.1.25 `init_new_lab()`

```

def db_init.init_new_lab (
    stud_id,
    lab_name,
    att,
    submitted,
    lab_path,
    db_name = './grades.sqlite3' )

```

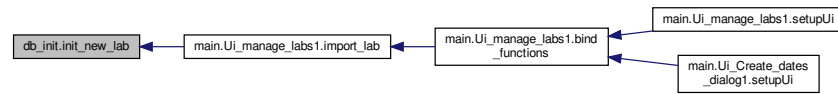
Definition at line 473 of file `db_init.py`.

```

473 def init_new_lab(stud_id, lab_name, att, submitted, lab_path, db_name='./grades.sqlite3'):
474     if not os.path.isfile(db_name):
475         raise Exception("DB not found")
476     with lite.connect(db_name) as con:
477         cur = con.cursor()
478         cur.execute('INSERT INTO grades (class_id, lab, attempt, submitted, lab_path) VALUES (?, ?, ?, ?,
?)', (stud_id, lab_name, att, submitted, lab_path))
479         con.commit()
480
481
482 # :param db_name:
483 # :return:
484 #
485
486

```

Here is the caller graph for this function:



6.3.1.26 insert_students()

```

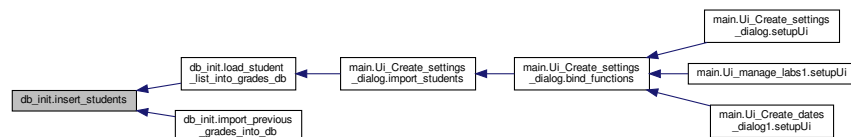
def db_init.insert_students (
    ids,
    fname,
    lname,
    db_name = './grades.sqlite3' )
  
```

Definition at line 257 of file db_init.py.

```

257 def insert_students(ids, fname, lname, db_name='./grades.sqlite3'):
258     names_tuple = list(zip(ids, fname, lname, [0] * len(ids)))
259     with lite.connect(db_name) as con:
260         cur = con.cursor()
261         cur.executemany('INSERT OR REPLACE INTO STUDENTS \
262             (pipeline_id, first_name, second_name, cheating_ratio)'
263             ' VALUES (?, ?, ?, ?)', names_tuple)
264         con.commit()
265
266
267 # :param pipeline_ids:
268 # :param year:
269 # :param semester:
270 # :param db_name:
271 # :return:
272 #
273
274
  
```

Here is the caller graph for this function:



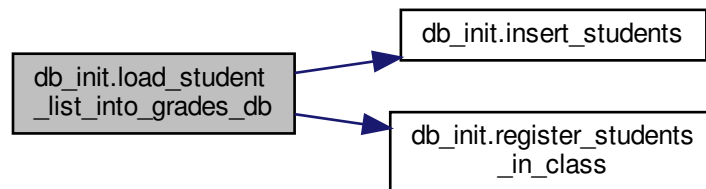
6.3.1.27 load_student_list_into_grades_db()

```
def db_init.load_student_list_into_grades_db (
    db_name,
    year,
    semester,
    filename = 'students_list3.txt' )
```

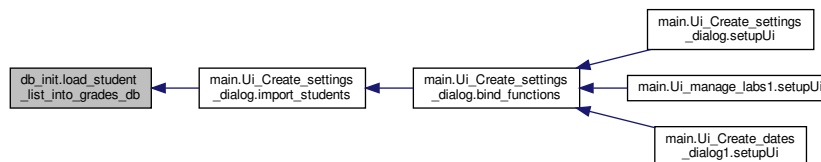
Definition at line 234 of file db_init.py.

```
234 def load_student_list_into_grades_db(db_name, year, semester,
    filename='students_list3.txt'):
235
236     with open(filename, 'r') as sl:         ids, names = zip(*(line.strip().split('%') for line in sl))
237         ids = list(sid.strip() for sid in ids)
238         names = (name.strip() for name in names) # for case when file contains extra whitespaces
239         lname, fname = zip(*(namer.split(',') for namer in names))
240         lname = (name.strip() for name in lname)
241         fname = (name.strip() for name in fname)
242
243     if os.path.isfile(db_name):
244         insert_students(ids, fname, lname, db_name)
245         register_students_in_class(ids, year, semester, db_name)
246
247
248 # Takes students' info from the parameters and insert them into grades DB
249 # :param ids: pipeline ids
250 # :param fname: first name
251 # :param lname: last name
252 # :param db_name: specific name for grades DB
253 # :return: nothing
254 #
255
256
```

Here is the call graph for this function:



Here is the caller graph for this function:



6.3.1.28 reconstruct_grades_and_comments()

```
def db_init.reconstruct_grades_and_comments (
    db_name = './grades.sqlite3' )
```

Definition at line 553 of file db_init.py.

```
553 def reconstruct_grades_and_comments(db_name='./grades.sqlite3'):
554     lab_id, lab_path = get_empty_grades(db_name)
555     updated_grades = list()
556     for l_iter in range(len(lab_path)):
557         lpath = lab_path[l_iter]
558         submission_t = int(lpath.split('-')[-1])
559         try:
560             with open(lpath+'/grade.txt', 'r') as gfile:
561                 cur_grade = int(gfile.readline().strip())
562         except Exception as e:
563             print("Error during grade file reading :", e)
564             cur_grade = 0
565         try:
566             cur_t_graded = int(os.path.getmtime(lpath + '/grade.txt'))
567         except Exception as e:
568             print("Error during grade file statistics retrieval: ", e)
569             cur_t_graded = None
570
571         pass_fail = 'TRUE' if cur_grade else 'FALSE'
572         try:
573             with open(lpath+'/responce.txt', 'r') as rfile:
574                 cur_resp = rfile.readlines()
575                 if type(cur_resp) == list:
576                     cur_resp = ' '.join(cur_resp)
577         except Exception as e:
578             print("Error during grade file reading", e)
579             cur_resp = 'NULL'
580         updated_grades.append((submission_t, cur_grade, cur_t_graded, pass_fail, cur_resp, lab_id[l_iter]))
581
582
583     with lite.connect(db_name) as con:
584         cur = con.cursor()
585         cur.executemany('UPDATE grades SET submitted=?, grade=?, graded=?, pass_fail=?, grader_comment=? '
586                         'WHERE id=?', updated_grades)
587         con.commit()
588
589     with lite.connect(db_name) as con:
590         cur = con.cursor()
591         cur.execute('VACUUM;')
592         con.commit()
593
594
```

6.3.1.29 register_lab_in_semester()

```
def db_init.register_lab_in_semester (
    ltype,
    lab_num,
    year,
    semester,
    due_dates,
    db_name = './grades.sqlite3' )
```

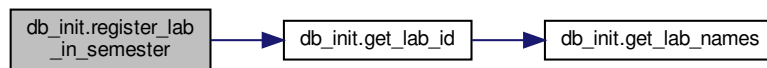
Definition at line 767 of file db_init.py.

```

767 def register_lab_in_semester(ltype, lab_num, year, semester, due_dates,
    db_name='./grades.sqlite3'):
768     lid = get_lab_id(ltype, int(lab_num))
769     # TODO: add a check so you do not insert lab twice
770     if lid is None:
771         raise Exception('No such lab')
772     if not os.path.isfile(db_name):
773         raise Exception("DB not found")
774     with lite.connect(db_name) as con:
775         cur = con.cursor()
776         cur.execute('INSERT OR REPLACE INTO lab_schedule (lab_id, year, semester, due_date_1, due_date_2,
    due_date_3, due_date_4) VALUES (?, ?, ?, ?, ?, ?, ?)', (lid, year, semester, due_dates[0], due_dates[1],
    due_dates[2], due_dates[3]))
777         con.commit()
778

```

Here is the call graph for this function:



6.3.1.30 register_students_in_class()

```

def db_init.register_students_in_class (
    pipeline_ids,
    year,
    semester,
    db_name = './grades.sqlite3' )

```

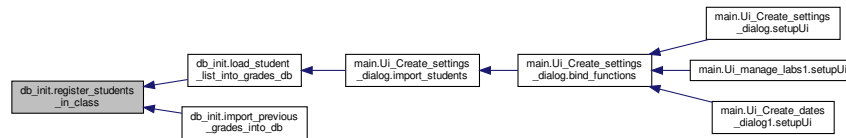
Definition at line 275 of file `db_init.py`.

```

275 def register_students_in_class(pipeline_ids, year, semester,
    db_name='./grades.sqlite3'):
276     len_id = len(pipeline_ids)
277     names_tupple = list(zip(pipeline_ids, [year] * len_id, [semester] * len_id, [0] * len_id))
278     with lite.connect(db_name) as con:
279         cur = con.cursor()
280         cur.executemany('INSERT OR REPLACE INTO class\
281             (pipeline_id, year, semester, cheating_ratio) VALUES (?, ?, ?, ?)', names_tupple)
282         con.commit()
283
284
285 # :param db_name:
286 # :return:
287 #
288
289

```


Here is the caller graph for this function:



6.3.1.31 save_a_grade_to_db()

```

def db_init.save_a_grade_to_db (
    grade_id,
    grade,
    grader_comment,
    extra_comment,
    grader_name,
    graded = True,
    pass_fail = True,
    db_name = './grades.sqlite3' )

```

Definition at line 455 of file db_init.py.

```

455 def save_a_grade_to_db(grade_id, grade, grader_comment, extra_comment, grader_name,
    graded=True, pass_fail=True, db_name='./grades.sqlite3'):
456     pass
457
458
459 # def get_submissions_to_grade(lab_id, att, db_name='./grades.sqlite3'):
460 #     if not os.path.isfile(db_name):
461 #         raise Exception("DB not found")
462 #     with lite.connect(db_name) as con:
463 #         cur = con.cursor()
464 #         result = cur.execute("SELECT id, FROM grades where lab=lab_id attempt=att and graded is NULL")
465 #         try:
466 #             lab_id, lab_type, lab_num = zip(*result.fetchall())
467 #         except Exception as e:
468 #             print(e)
469 #             return None, None, None
470 #     return lab_id, lab_type, lab_num
471
472

```

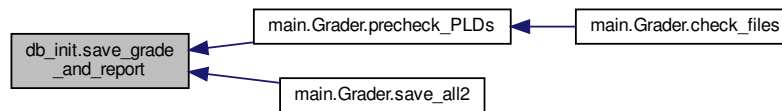
6.3.1.32 save_grade_and_report()

```
def db_init.save_grade_and_report (
    grade_id,
    grade,
    report,
    user_comment,
    grader,
    db_name = './grades.sqlite3' )
```

Definition at line 740 of file db_init.py.

```
740 def save_grade_and_report(grade_id, grade, report, user_comment, grader,
    db_name='./grades.sqlite3'):
741     if not os.path.isfile(db_name):
742         raise Exception("DB not found")
743     with lite.connect(db_name) as con:
744         cur = con.cursor()
745         cur.execute("UPDATE grades SET graded=strftime('%s','now'), pass_fail=TRUE, grade=?,
    grader_comment=?, extra_comment=?, grader=? WHERE id=?", (grade, report, user_comment, grader, grade_id))
746         con.commit()
747
748
```

Here is the caller graph for this function:



6.3.1.33 settings_db_create()

```
def db_init.settings_db_create (
    db_name = SETTINGS_DB_NAME,
    force = False )
```

Definition at line 18 of file db_init.py.

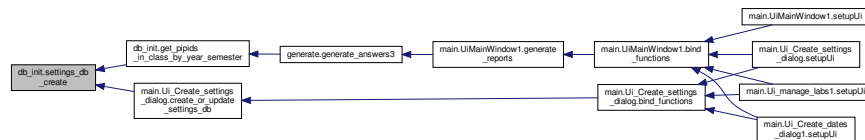
```
18 def settings_db_create(db_name=SETTINGS_DB_NAME, force=False):
19     if not force and os.path.isfile(db_name):
20         user_choice = input('Do you really want to drop database ? Type "yes" to continue\n ')
21         if not user_choice.isalpha() or not user_choice.lower() == 'yes':
22             return False
23
24     # DB creation logic goes here
25     with lite.connect(db_name) as con:
26         cur = con.cursor()
27         cur.execute('DROP TABLE IF EXISTS PATHS')
```

```

28         cur.execute("CREATE TABLE PATHS "
29                     "( LOGISIM_HOME VARCHAR NOT NULL,\
30                      GRADING_PATH VARCHAR NOT NULL,\
31                      IMPORT_PATH VARCHAR,\
32                      GRADES_DB VARCHAR); ")
33     cur.execute("CREATE TABLE LOCAL (\
34                 GRADER_NAME VARCHAR,\
35                 YEAR          INT,\
36                 SEMESTER      CHAR (1),\
37                 USE_STYLE     BOOLEAN,\
38                 SYNC_COMMAND  VARCHAR); ")
39     con.commit()
40     return True
41
42
43 # Reads settings from the DB with specified name in 'db_name'
44 # :param db_name: name of DB to query
45 # :return: paths - list of paths to various locations and local - info about grader, grading year, etc.
46 #
47

```

Here is the caller graph for this function:



6.3.1.34 settings_db_read_settings()

```

def db_init.settings_db_read_settings (
    db_name = SETTINGS_DB_NAME )

```

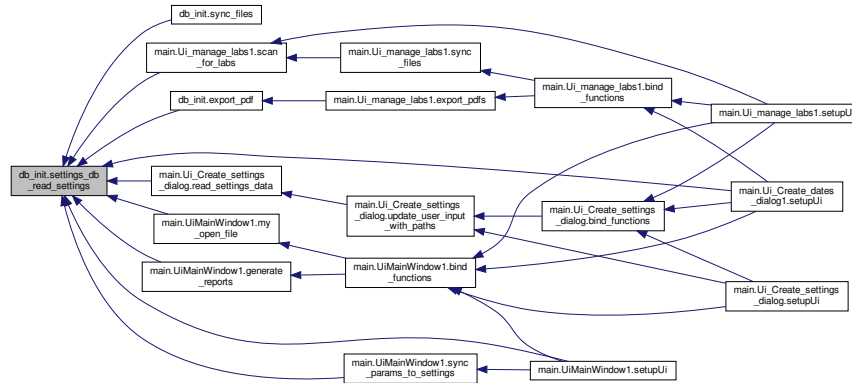
Definition at line 48 of file db_init.py.

```

48 def settings_db_read_settings (db_name=SETTINGS_DB_NAME) :
49     paths = local = None
50     if os.path.isfile(db_name):
51         with lite.connect(db_name) as con:
52             cur = con.cursor()
53             result = cur.execute("SELECT LOGISIM_HOME, GRADING_PATH, IMPORT_PATH, GRADES_DB\
54                                 FROM PATHS")
55             paths = result.fetchone()
56             result = cur.execute("SELECT GRADER_NAME, YEAR, SEMESTER, USE_STYLE, SYNC_COMMAND\
57                                 FROM LOCAL")
58             local = result.fetchone()
59
60     return paths, local
61
62
63 # Procedure that loads parameters specified in paths and local into settings DB
64 # :param paths: list of paths to various locations
65 # :param local: local - info about grader, grading year, etc.
66 # :param db_name: name of DB to query to update
67 # :return: nothing
68 #
69

```

Here is the caller graph for this function:



6.3.1.35 sync_files()

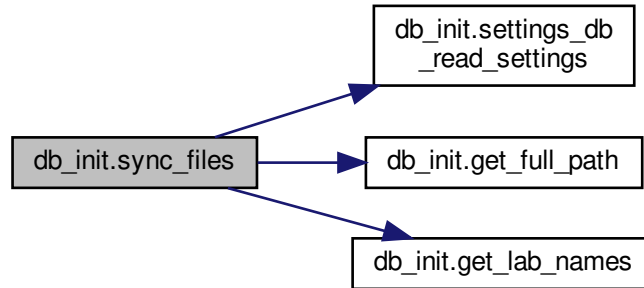
```
def db_init.sync_files (
    self = None )
```

Definition at line 677 of file db_init.py.

```

677 def sync_files(self=None):
678     import subprocess
679     import os
680
681     paths, local = settings_db_read_settings()
682     full_path = get_full_path(paths, local) + "/server_sync/"
683     lab_ids, lab_types, lab_nums = get_lab_names()
684     lab_names = []
685     for i in range(len(lab_types)):
686         lab_names.append(lab_types[i] + '_Lab_' + str(lab_nums[i]))
687
688     if not os.path.isdir(full_path):
689         os.makedirs(full_path)
690         for lab_name in lab_names:
691             os.makedirs(full_path + lab_name)
692
693     proc_arr = []
694     for lab_name in lab_names:
695         command = local[4] + ' ' + os.path.expanduser(paths[2] + lab_name) + '/*.zip' + ' ' + full_path +
lab_name + '/'
696         try:
697             proc_arr.append(subprocess.Popen(os.path.expandvars(command), stdout=subprocess.PIPE, shell=
True))
698             proc_arr[-1].communicate()
699         except Exception as e:
700             print('Error in rsync: ', e)
701             # output, error = process.communicate()
702             # print(output)
703             # print(error)
704
705     for proc_elem in proc_arr:
706         proc_elem.wait()
707
708
```

Here is the call graph for this function:



6.3.1.36 update_lab_submissions_paths()

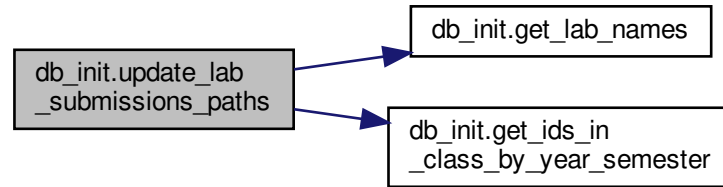
```
def db_init.update_lab_submissions_paths (
    db_name,
    repository_root,
    year,
    semester )
```

Definition at line 499 of file `db_init.py`.

```

499 def update_lab_submissions_paths(db_name, repository_root, year, semester):
500     import fnmatch
501     import glob
502     # import_previous_grades_into_db(year, semester, db_name, repository_root+'grades.xlsx')
503     lab_id, lab_type, lab_num = get_lab_names()
504     if lab_id is None or lab_type is None or lab_num is None:
505         raise Exception("Error during lab type/num import: ")
506     class_dict = get_ids_in_class_by_year_semester(year, semester, db_name
507 )
508     total_labs = len(lab_type)
509     all_dirs = list()
510     for lab_iter in range(total_labs):
511         for attempt in range(1, 5): # class rule - 4 attempts
512             full_lab_name = repository_root + lab_type[lab_iter] + '_Lab_' + str(lab_num[lab_iter]) + '_' +
513 str(attempt) + '/'
514             print('Processing ', full_lab_name)
515             for stud_id in class_dict.keys():
516                 found_dir = glob.glob(full_lab_name+stud_id+'*')
517                 if found_dir:
518                     # since it is initial pass, we do not set pass/fail. It will be set later with grade
519                     and comment.
520                     all_dirs.append((class_dict[stud_id], lab_id[lab_iter], attempt, 'FALSE', found_dir[-1])
521 ))
522     with lite.connect(db_name) as con:
523         cur = con.cursor()
524         cur.executemany('INSERT OR REPLACE INTO grades (class_id, lab, attempt, pass_fail, lab_path)'
525             ' VALUES (?, ?, ?, ?, ?)', all_dirs)
526         con.commit()
```

Here is the call graph for this function:



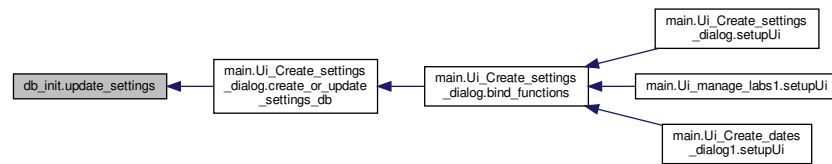
6.3.1.37 update_settings()

```
def db_init.update_settings (
    paths,
    local,
    db_name = SETTINGS_DB_NAME )
```

Definition at line 70 of file `db_init.py`.

```
70 def update_settings(paths, local, db_name=SETTINGS_DB_NAME):
71     if os.path.isfile(db_name):
72         with lite.connect(db_name) as con:
73             cur = con.cursor()
74             cur.execute('DELETE FROM PATHS;')
75             cur.execute('INSERT OR REPLACE INTO PATHS (LOGISIM_HOME, GRADING_PATH, IMPORT_PATH, GRADES_DB)'
76                 ' VALUES (?, ?, ?, ?);', paths)
77             cur.execute('DELETE FROM LOCAL;')
78             cur.execute('INSERT OR REPLACE INTO LOCAL (GRADER_NAME, YEAR, SEMESTER, USE_STYLE,
79 SYNC_COMMAND)'
80                 'VALUES (?, ?, ?, ?, ?);', local)
81             con.commit()
82         with lite.connect(db_name) as con:
83             cur = con.cursor()
84             cur.execute('VACUUM;')
85             con.commit()
86
87
88 # Will create database that contains all information about grades
89 # :param db_name: path and name of the database
90 # :param force: flag to overwrite existing db
91 # :return: Unknown
92 #
93
```

Here is the caller graph for this function:



6.3.2 Variable Documentation

6.3.2.1 SETTINGS_DB_NAME

```
string db_init.SETTINGS_DB_NAME = 'settings.sqlite3'
```

Definition at line 8 of file `db_init.py`.

6.4 generate Namespace Reference

Functions

- def [convert_to_pdf](#) (html_file, func_type)
- def [create_html_pdf_report2](#) (lab_dict)
 - Creates nice html report for submitted labs and converts it to pdf format.*
- def [create_html_pdf_zero_report](#) (filename, stud_name, top_part, bot_part)
- def [create_not_submitted](#) (stud_id, lab_type, lab_num, dir_name)
- def [generate_answers3](#) (lid, att, year, semester, db_name='./grades.sqlite3')
- def [time_to_str_with_tz](#) (in_time)

6.4.1 Function Documentation

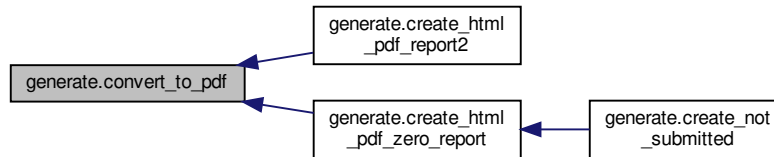
6.4.1.1 `convert_to_pdf()`

```
def generate.convert_to_pdf (
    html_file,
    func_type )
```

Definition at line 19 of file generate.py.

```
19 def convert_to_pdf(html_file, func_type):
20     if func_type == "wkhtmltopdf": # old way
21         from subprocess import call
22         call(["wkhtmltopdf", "-q", html_file, html_file[:-4] + 'pdf'])
23     elif func_type == "pdftkit": # best margins
24         import pdftkit
25         options = {
26             'page-size': 'A4',
27             'margin-top': '0.0in',
28             'margin-right': '0.0in',
29             'margin-bottom': '0.0in',
30             'margin-left': '0.0in',
31         }
32         pdftkit.from_url(html_file, html_file[:-4] + 'pdf', options=options)
33     elif func_type == 'weasyprint': # potentially the fastest
34         # if string is passed as param, but has margins problem
35         from weasyprint import HTML
36         with open(html_file, 'r') as html_in_file:
37             cont = html_in_file.readlines()
38             str_file = ''.join(cont)
39             pdf = HTML(string=str_file)
40             pdf.write_pdf(html_file[:-4] + 'pdf')
41
42
43 # def create_html_pdf_report(joined_path, stud_name, cur_dir, grade, max_points, penalty,
44 #                             final_score, top_part, bot_part, generated_time):
45 #
46 #     """
47 #     Creates nice html report for submitted labs and converts it to pdf format.
48 #     TODO: use latex instead of ugly html.
49 #     :param joined_path: working directory
50 #     :param stud_name: full student name(first, last)
51 #     :param cur_dir: directory with all labs(usually same as joined_path)
52 #     :param grade: what grade to assign.
53 #     :param max_points: max possible grade for this lab.
54 #     :param penalty: usually for resubmission, like 90%, 70%...
55 #     :param final_score: final grade = grade * penalty
56 #     :param top_part: predefined top part of html document
57 #     :param bot_part: predefined bottom part of html document
58 #     :param generated_time: some extra statistics for curious students.
59 #     :return: nothing, pdf is generated instead.
60 #     """
61 #     with open(joined_path + '-returned.html', 'w') as stud_report:
62 #         stud_report.writelines(top_part)
63 #         stud_report.write('<p>Grading directory : ' + cur_dir + ' </br>')
64 #
65 #         with open(joined_path + '/tech_info.txt', 'r') as tech_file:
66 #             stud_report.writelines(tech_file.readlines())
67 #
68 #         stud_report.write('</p><p><i>Dear ' + stud_name + ', ' +
69 #
70 #         with open(joined_path + '/responce.txt', 'r') as resp_file:
71 #             stud_report.writelines(resp_file.readlines())
72 #
73 #         stud_report.write("</i></p>\n"
74 #
75 #             "<p>According to the comment above, next grade was assigned: "
76 #             "%d of %d <br/>\n \
77 #             Your final grade is %d*%.1f=<b>%d</b> of %d <br/>\n"
78 #             % (grade, max_points, grade, penalty, final_score, max_points))
79 #         stud_report.write('This report was generated {} </p>'.format(generated_time))
80 #         # TODO add current date/time
81 #         stud_report.writelines(bot_part)
82 #
83 #     convert_to_pdf(joined_path + '-returned.html', "pdftkit")
84 #     os.remove(joined_path + '-returned.html')
```


Here is the caller graph for this function:



6.4.1.2 create_html_pdf_report2()

```
def generate.create_html_pdf_report2 (
    lab_dict )
```

Creates nice html report for submitted labs and converts it to pdf format.

:return: nothing, pdf is generated instead.

Definition at line 90 of file generate.py.

```

90 def create_html_pdf_report2(lab_dict):
91     with open('./answer.top', 'r') as partial_html:
92         top_part = partial_html.readlines()
93
94     with open('./answer.bottom', 'r') as partial_html:
95         bot_part = partial_html.readlines()
96
97     with open(lab_dict['lab_path'] + '-returned.html', 'w') as stud_report:
98         stud_report.writelines(top_part)
99
100         stud_report.write('<p>Grading directory : {} </br>'.format(lab_dict['lab_path'].split('/')[1]))
101         stud_report.write('Due date was {} <br/>'.format(time_to_str_with_tz(lab_dict['
due_date'])))
102         stud_report.write('File was submitted at {} <br/>'.format(
time_to_str_with_tz(lab_dict['submitted'])))
103         stud_report.write('I imported your file at {} <br/>'.format(
time_to_str_with_tz(lab_dict['import_date'])))
104         if lab_dict['graded'] is not None:
105             stud_report.write('I graded your lab at {} <br/>'.format(
time_to_str_with_tz(lab_dict['graded'])))
106         else:
107             stud_report.write('I did not grad your lab or grade timestamp was not set.<br/>')
108             stud_report.write('Lab type : \{}\' and it\'s number is \{}\' <br/>'.format(lab_dict['type'],
lab_dict['num']))
109         stud_report.write('</p><p><i>Dear {} {}, '.format(lab_dict['first_name'], lab_dict['second_name']))
110         if lab_dict['grader_comment'] is None or len(lab_dict['grader_comment']) < 2:
111             stud_report.write('There were no comments.')
112         else:
113             stud_report.write(lab_dict['grader_comment'])
114         if lab_dict['extra_comment'] is not None and len(lab_dict['extra_comment']) > 0:
115             stud_report.write('<br/>\nExtra comment: {}'.format(lab_dict['extra_comment']))
116
117         stud_report.write("</i></p>\n"
118             "<p>According to the comment above, next grade was assigned: {} of {} <br/>\n"
119             " Your final grade is computed as {}*{:.1f}=<b>{}</b> of {} <br/>\n"
120             "{}.format(lab_dict['final_grade'], lab_dict['max_grade'], lab_dict['grade'],

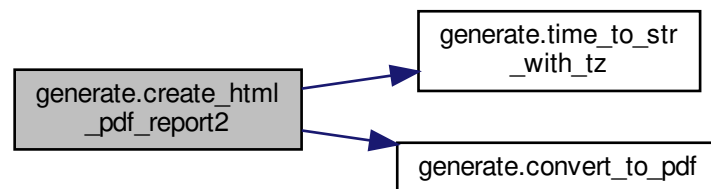
```

```

lab_dict['percent']/100, lab_dict['final_grade'], lab_dict['max_grade']))
121     if lab_dict['grade'] == 0:
122         stud_report.write('<br/>Don\'t forget to resubmit it by {} <br/><br/>\n'.format(
time_to_str_with_tz(lab_dict['due_date'] + 604800))) # one extra week
123         stud_report.write('This report was generated {} </p>\n'.format(QDateTime.currentDateTime().toString
(Qt.DefaultLocaleLongDate)))
124
125         stud_report.writelines(bot_part)
126
127         convert_to_pdf(lab_dict['lab_path'] + '-returned.html', "pdftk")
128         os.remove(lab_dict['lab_path'] + '-returned.html')
129
130
131 #     Creates nice html report for nonsubmitted labs and converts it to pdf format.
132 #     :param filename: filename with correct naming(zeroes instead of timestamp)
133 #     :param stud_name: full student name(first, last)
134 #     :param top_part: predefined top part of html document
135 #     :param bot_part: predefined bottom part of html document
136 #     :return:
137 #
138

```

Here is the call graph for this function:



6.4.1.3 create_html_pdf_zero_report()

```

def generate.create_html_pdf_zero_report (
    filename,
    stud_name,
    top_part,
    bot_part )

```

Definition at line 139 of file generate.py.

```

139 def create_html_pdf_zero_report(filename, stud_name, top_part, bot_part):
140     with open(filename, 'w') as zeroes_file:
141         zeroes_file.writelines(top_part)
142         zeroes_file.write(stud_name + ' : You did not submit your lab. :(</p>\n')
143         zeroes_file.write("<p>According to comments above, next grade was assigned : 0 </p>")
144         zeroes_file.write("<p>Please submit your file before the next due date.")
145         zeroes_file.writelines(bot_part)
146         convert_to_pdf(filename, "pdftk")
147         os.remove(filename)

```

```

148
149
150 # def generate_answers(resubmit_num, dir_name, lab_type, lab_num, year, semester, grader_name):
151 #     """
152 #     general function that figures out max points, filenames, etc
153 #     and calls generate function with appropriate parameters
154 #     :param resubmit_num: resubmission attempt
155 #     :param dir_name: working dir
156 #     :param lab_type: open or closed lab
157 #     :param lab_num: just lab identifier
158 #     :param year: used wit semester to identify correct class list
159 #     :param semester: used wit year to identify correct class list
160 #     :param grader_name: name that will be displayed in the report
161 #     :return:
162 #     """
163 #     students = {}
164 #     # select
165 #
166 #     ids = get_pipids_in_class_by_year_semester(year, semester, 'grades.sqlite3')
167 #     with lite.connect('grades.sqlite3') as con:
168 #         cur = con.cursor()
169 #
170 #         for sid in ids.keys():
171 #             result = cur.execute('SELECT first_name, second_name FROM students WHERE pipeline_id=?',
172 # (str(sid),))
173 #             students[sid] = " ".join(result.fetchall()[0])
174 #
175 #     if not students:
176 #         with open('students_list1.txt', 'r') as stud_list_file:
177 #             temp_arr = stud_list_file.readlines()
178 #             for line in temp_arr:
179 #                 sid, name = line.split('%')
180 #                 students[sid.strip()] = name.strip()
181 #             del temp_arr
182 #
183 #     if lab_type == 'Closed':
184 #         max_points = 10
185 #         type_for_name = 'CL'
186 #     elif lab_type == 'Open':
187 #         max_points = 20
188 #         type_for_name = 'OL'
189 #     else:
190 #         raise Exception('Unknown lab type')
191 #
192 #     if resubmit_num == 1:
193 #         penalty = 1.0
194 #     elif resubmit_num == 2:
195 #         penalty = 0.9
196 #     elif resubmit_num == 3:
197 #         penalty = 0.7
198 #     elif resubmit_num == 4:
199 #         penalty = 0.5
200 #     else:
201 #         penalty = 0.0
202 #
203 #     generated_time = QDateTime.currentDateTime().toString(Qt.DefaultLocaleLongDate)
204 #
205 #     print('This is ', type_for_name, ' lab, so max points is ', max_points)
206 #
207 #     try:
208 #         shutil.rmtree(dir_name + 'Answers', ignore_errors=True)
209 #         os.remove(dir_name + "grades.csv")
210 #         os.remove(dir_name + "grades_for_" + type_for_name + "lab_num.csv")
211 #     except Exception as e:
212 #         print('Exception during dir preparation : ', e)
213 #
214 #     dirs = os.walk(dir_name).__next__()[1]
215 #
216 #     with open('./answer.top', 'r') as partial_html:
217 #         top_part = partial_html.readlines()
218 #
219 #     with open('./answer.bottom', 'r') as partial_html:
220 #         bot_part = partial_html.readlines()
221 #
222 #     grades = list()
223 #     for cur_dir in dirs:
224 #         student_id = cur_dir.split('-')[0]
225 #         joined_path = os.path.join(dir_name, cur_dir)
226 #         with open(joined_path + '/grade.txt', 'r') as grade_file:
227 #             grade = grade_file.readlines()

```

```

228 #
229 #     grade = int(grade[0].strip())
230 #     final_score = grade * penalty
231 #     grades.append((student_id, final_score))
232 #     create_html_pdf_report(joined_path, students[student_id], cur_dir, grade,
233 #                             max_points, penalty, final_score, top_part, bot_part, generated_time)
234 #
235 #     submitted = [x.split('-')[0] for x in dirs]
236 #
237 #     zeroes = list()
238 #     for student in students:
239 #         if student not in submitted:
240 #             grades.append((student, 0))
241 #             zeroes.append(student)
242 #
243 #     if resubmit_num == 1:
244 #         for student_id in zeroes:
245 #             filename = '%s/%s-%s%d-0000000000-returned' % \
246 #                 (dir_name, student_id, type_for_name, lab_num)
247 #             create_html_pdf_zero_report(filename+'.html', students[student_id], top_part, bot_part)
248 #
249 #     with open(dir_name + '/' + 'grades.csv', 'w') as grades_file:
250 #         for grade in sorted(grades):
251 #             grades_file.write("%s, %f\n" % grade)
252 #
253 #     os.mkdir(dir_name + '/Answers')
254 #     files = os.walk(dir_name).__next__()[2]
255 #     for file in files:
256 #         if file[-3:] == 'pdf':
257 #             shutil.move(dir_name + '/' + file, dir_name + '/Answers/' + file)
258 #
259 #     print('Done')
260 #
261 #
262 # def generate_answers2(resubmit_num, dir_name, lab_type, lab_num, year, semester, grader_name):
263 #     """
264 #     general function that figures out max points, filenames, etc
265 #     and calls generate function with appropriate parameters
266 #     :param resubmit_num: resubmission attempt
267 #     :param dir_name: working dir
268 #     :param lab_type: open or closed lab
269 #     :param lab_num: just lab identifier
270 #     :param year: used wit semester to identify correct class list
271 #     :param semester: used wit year to identify correct class list
272 #     :param grader_name: name that will be displayed in the report
273 #     :return:
274 #     """
275 #     students = {}
276 #     # select
277 #     import sqlite3 as lite
278 #     from db_init import get_ids_in_class_by_year_semester
279 #     ids = get_ids_in_class_by_year_semester(year, semester, 'grades.sqlite3')
280 #     with lite.connect('grades.sqlite3') as con:
281 #         cur = con.cursor()
282 #
283 #         for sid in ids.keys():
284 #             result = cur.execute('SELECT first_name, second_name FROM students WHERE pipeline_id=?',
285 # (str(sid),))
286 #             students[sid] = " ".join(result.fetchall()[0])
287 #
288 #     if not students:
289 #         with open('students_list1.txt', 'r') as stud_list_file:
290 #             temp_arr = stud_list_file.readlines()
291 #             for line in temp_arr:
292 #                 sid, name = line.split('%')
293 #                 students[sid.strip()] = name.strip()
294 #             del temp_arr
295 #
296 #     if lab_type == 'Closed':
297 #         max_points = 10
298 #         type_for_name = 'CL'
299 #     elif lab_type == 'Open':
300 #         max_points = 20
301 #         type_for_name = 'OL'
302 #     else:
303 #         raise Exception('Unknown lab type')
304 #
305 #     if resubmit_num == 1:
306 #         penalty = 1.0
307 #     elif resubmit_num == 2:

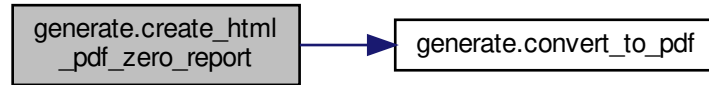
```

```

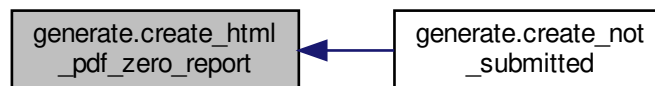
308 #         penalty = 0.9
309 #     elif resubmit_num == 3:
310 #         penalty = 0.7
311 #     elif resubmit_num == 4:
312 #         penalty = 0.5
313 #     else:
314 #         penalty = 0.0
315 #
316 #     generated_time = QDateTime.currentDateTime().toString(Qt.DefaultLocaleLongDate)
317 #
318 #     print('This is ', type_for_name, ' lab, so max points is ', max_points)
319 #
320 #     try:
321 #         shutil.rmtree(dir_name + 'Answers', ignore_errors=True)
322 #         os.remove(dir_name + "grades.csv")
323 #         os.remove(dir_name + "grades_for_" + type_for_name + "lab_num.csv")
324 #     except Exception as e:
325 #         print('Exception during dir preparation : ', e)
326 #
327 #     dirs = os.walk(dir_name).__next__()[1]
328 #
329 #     with open('./answer.top', 'r') as partial_html:
330 #         top_part = partial_html.readlines()
331 #
332 #     with open('./answer.bottom', 'r') as partial_html:
333 #         bot_part = partial_html.readlines()
334 #
335 #     grades = list()
336 #     for cur_dir in dirs:
337 #         student_id = cur_dir.split('-')[0]
338 #         joined_path = os.path.join(dir_name, cur_dir)
339 #         with open(joined_path + '/grade.txt', 'r') as grade_file:
340 #             grade = grade_file.readlines()
341 #
342 #             grade = int(grade[0].strip())
343 #             final_score = grade * penalty
344 #             grades.append((student_id, final_score))
345 #             create_html_pdf_report(joined_path, students[student_id], cur_dir, grade,
346 #                                   max_points, penalty, final_score, top_part, bot_part, generated_time)
347 #
348 #     submitted = [x.split('-')[0] for x in dirs]
349 #
350 #     zeroes = list()
351 #     for student in students:
352 #         if student not in submitted:
353 #             grades.append((student, 0))
354 #             zeroes.append(student)
355 #
356 #     if resubmit_num == 1:
357 #         for student_id in zeroes:
358 #             filename = '%s/%s-%s%d-0000000000-returned' % \
359 #                 (dir_name, student_id, type_for_name, lab_num)
360 #             create_html_pdf_zero_report(filename+'.html', students[student_id], top_part, bot_part)
361 #
362 #     with open(dir_name + '/' + 'grades.csv', 'w') as grades_file:
363 #         for grade in sorted(grades):
364 #             grades_file.write("%s, %f\n" % grade)
365 #
366 #     os.mkdir(dir_name + '/Answers')
367 #     files = os.walk(dir_name).__next__()[2]
368 #     for file in files:
369 #         if file[-3:] == 'pdf':
370 #             shutil.move(dir_name + '/' + file, dir_name + '/Answers/' + file)
371 #
372 #     print('Done')
373 #
374 #
375 # def create_a_report(lab_dict):
376 #
377 #     create_html_pdf_report2(lab_dict)
378 #
379 #

```

Here is the call graph for this function:



Here is the caller graph for this function:



6.4.1.4 create_not_submitted()

```

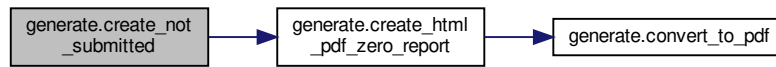
def generate.create_not_submitted (
    stud_id,
    lab_type,
    lab_num,
    dir_name )
  
```

Definition at line 380 of file generate.py.

```

380 def create_not_submitted(stud_id, lab_type, lab_num, dir_name):
381     with open('./answer.top', 'r') as partial_html:
382         top_part = partial_html.readlines()
383
384     with open('./answer.bottom', 'r') as partial_html:
385         bot_part = partial_html.readlines()
386     filename = '%s/%s-%s%d-0000000000-returned' % \
387         (dir_name, stud_id, lab_type, lab_num)
388     create_html_pdf_zero_report(filename + '.html', stud_id, top_part, bot_part)
389
390
  
```

Here is the call graph for this function:



6.4.1.5 generate_answers3()

```

def generate.generate_answers3 (
    lid,
    att,
    year,
    semester,
    db_name = './grades.sqlite3' )

```

Definition at line 391 of file generate.py.

```

391 def generate_answers3(lid, att, year, semester, db_name='./grades.sqlite3'):
392     all_ids = get_pipids_in_class_by_year_semester(year, semester)
393     info_tup, info_desc = get_grades_by_lab_and_att(lid, att)
394     col_names = [elem[0] for elem in info_desc]
395     main_list = list()
396     for tup in info_tup:
397         a = dict()
398         for i, elem in enumerate(tup):
399             a[col_names[i]] = elem
400         main_list.append(a)
401     graded_students = [elem['pipeline_id'] for elem in main_list]
402     grades = [elem['final_grade'] for elem in main_list]
403     grade_dict = dict(zip(graded_students, grades))
404     lab_type = main_list[0]['type']
405     lab_num = main_list[0]['num']
406     dir_name = main_list[0]['lab_path']
407     dir_name = dir_name[:dir_name.rfind('/')]
408     correctd_lab_type = 'CL' if lab_type == 'Closed' else 'OL'
409
410     # for elem in main_list:
411     #     create_a_report(elem)
412     #
413     # for elem in main_list:
414     #     commit_gen_report(elem['grade_id'])
415
416     not_subm_ids = [stud_id for stud_id in all_ids if stud_id not in graded_students]
417
418     if len(main_list) + len(not_subm_ids) == 0:
419         return
420
421     ans_dir = os.path.join(dir_name, 'Answers')
422     if os.path.exists(ans_dir):
423         shutil.rmtree(ans_dir, ignore_errors=True)
424     gr_file = os.path.join(dir_name, 'grades.csv')
425     if os.path.exists(gr_file):
426         os.remove(gr_file)
427     gr_long_file = os.path.join(dir_name, "grades_for_{}lab_num.csv".format(correctd_lab_type))
428     if os.path.exists(gr_long_file):
429         os.remove(gr_long_file)
430     files_to_rem = (os.path.join(dir_name, file) for file in (el for el in os.walk(dir_name).__next__() [2]
431                     if el[-3:] in ['pdf', 'html']))

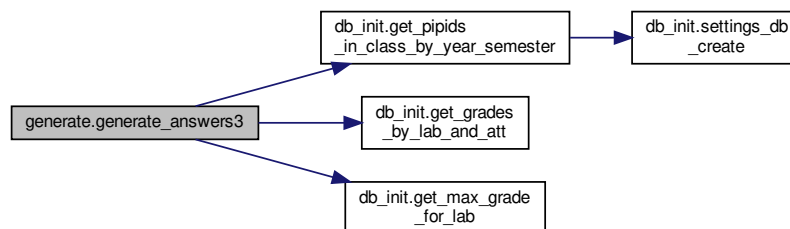
```

```

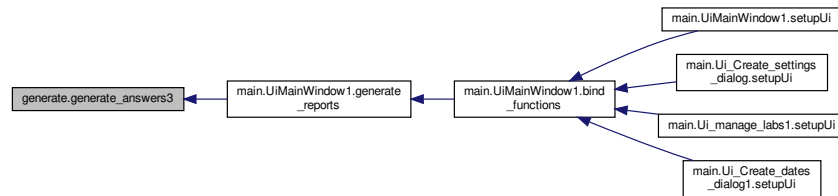
431
432     with mp.Pool() as pool:
433         pool.map(os.remove, files_to_remove)
434         r1 = pool.map_async(create_html_pdf_report2, main_list)
435         r2 = pool.map_async(commit_gen_report, (elem['grade_id'] for elem in main_list))
436         if att == 1:
437             pool.starmap(create_not_submitted, ((stud_id, correctd_lab_type, lab_num, dir_name) for stud_id
in not_subm_ids))
438             r1.wait()
439             r2.wait()
440
441     with open(os.path.join(dir_name, '{}_lab{}_grades.csv'.format(lab_num, lab_type)), 'w') as grades_file
:
442         grades_file.write("{} Lab {}, {} Lab {}\n".format(lab_num, lab_type))
443         for stud_id in all_ids:
444             if stud_id not in not_subm_ids:
445                 grades_file.write("{}: {}, {}:\n".format(stud_id, int(grade_dict[stud_id])))
446             else:
447                 grades_file.write("{}: {}, {}:\n".format(stud_id, 0))
448
449
450     best_grade_list = get_max_grade_for_lab(lid, year, semester)
451     with open(os.path.join(dir_name, '{}_lab{}_grades_best_so_far.csv'.format(lab_num, lab_type)), 'w') as
grades_file:
452         grades_file.write("{} Lab {}, {} Lab {}\n".format(lab_num, lab_type))
453         for stud_tup in best_grade_list:
454             grades_file.write('{} {}, {}{}\n'.format(stud_tup[0], stud_tup[1]))
455
456     # for elem in main_list:
457     #     create_html_pdf_report2(elem)
458     # for elem in main_list:
459     #     commit_gen_report(elem['grade_id'])
460
461     # if att == 1: # we do not form report for second attempt since most people are happy with previous
grade
462     # for stud_id in not_subm_ids:
463     #     create_not_submitted(stud_id, correctd_lab_type, lab_num, dir_name)
464
465     os.mkdir(os.path.join(dir_name, 'Answers'))
466     files = os.walk(dir_name).__next__()[2]
467     for file in files:
468         if file[-3:] == 'pdf':
469             shutil.move(os.path.join(dir_name, file), os.path.join(dir_name, 'Answers/{}'.format(file)))
470
471     print('Done')
472
473

```

Here is the call graph for this function:



Here is the caller graph for this function:



6.4.1.6 time_to_str_with_tz()

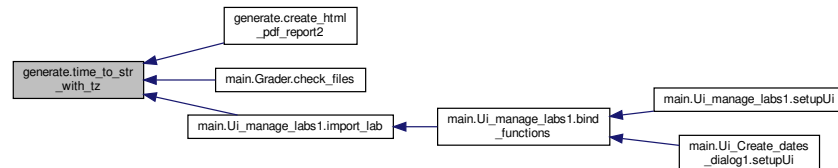
```
def generate.time_to_str_with_tz (
    in_time )
```

Definition at line 474 of file generate.py.

```

474 def time_to_str_with_tz(in_time):
475     return datetime.datetime.fromtimestamp(in_time).replace(tzinfo=tz.tzutc()).astimezone(tz.tzlocal()).strftime(
        '%m-%d-%Y %H:%M')
476 # if __name__ == '__main__':
477 #     generate_answers(3, 'Open_Lab_3_3', 'Open', 3)
478 
```

Here is the caller graph for this function:



6.5 main Namespace Reference

Classes

- class [CircFile](#)
- class [Grader](#)
- class [SimpleDialog](#)

Wrapper class for very simple OK/Cancel dialog.

- class [Ui_Create_dates_dialog1](#)
- class [Ui_Create_settings_dialog](#)

Creates window that provides user with convenient way of changing settings that are stored in sqlite3 db.

- class [Ui_manage_labs1](#)
- class [UiMainWindow1](#)

Functions

- def [read_settings](#) (db_name='settings.sqlite3')
- def [get_grading_period](#) (lid, cur_only=False)

Variables

- string [MAIN_FILE_NAME](#) = "
- string [MAIN_FILE_NAME_OVERRIDE](#) = "
- string [styleData](#)
- [app](#) = QtWidgets.QApplication(sys.argv)
- [MainWindow](#) = QtWidgets.QMainWindow()
- [ui](#) = UiMainWindow1()

6.5.1 Function Documentation

6.5.1.1 [get_grading_period\(\)](#)

```
def main.get_grading_period (
    lid,
    cur_only = False )
```

Definition at line 1874 of file main.py.

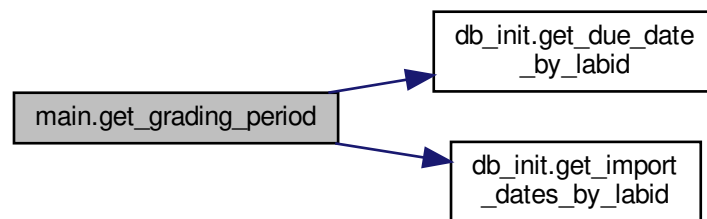
```
1874 def get_grading_period(lid, cur_only=False):
1875     # should compute correct grading period and return the due date in Unix timestamp format
1876     import time
1877     # due_timestamps = [int(f.split('_')[2]) for f in due_files]
1878
1879     current_timestamp = int(time.time())
1880     due_timestamps1 = get_due_date_by_labid(lid)
1881     import_timestamps1 = get_import_dates_by_labid(lid)
1882     cur_check = len(due_timestamps1)
1883     for i, ts in enumerate(import_timestamps1):
1884         if ts is None:
1885             cur_check = i
1886             break
1887     i = 0
1888     if cur_check:
1889         while i < len(due_timestamps1) and import_timestamps1[i] is not None and due_timestamps1[i] <
current_timestamp and due_timestamps1[i] < import_timestamps1[cur_check-1]:
1890             i += 1
1891
1892     if cur_only: # neede for CLA2-2
1893         i = max(0, i-1)
1894
1895     if i == 0:
1896         from_time = 0
1897         to_time = due_timestamps1[i]
1898     elif i > len(due_timestamps1)-1:
1899         from_time = due_timestamps1[i-1]
1900         to_time = int(time.time())
1901     else:
1902         from_time = due_timestamps1[i - 1]
1903         to_time = due_timestamps1[i]
1904
1905     cur_check_num = i+1
```

```

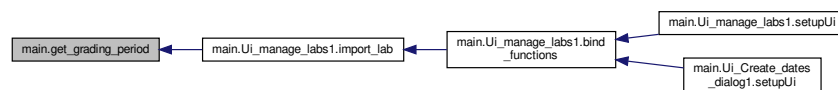
1906     # cur_check += 1
1907
1908
1909     #
1910     # check_files = [int(f.split('_')[2]) for f in os.listdir(dir) if 'check_' in f]
1911     # if len(check_files) > 0:
1912     #     if len(check_files) >= 4:
1913     #         cur_check_num = 0
1914     #         from_time = 0
1915     #         to_time = 0
1916     #     else:
1917     #         cur_check_num = len(check_files) + 1           # 1 + 1
1918     #         from_time = due_timestamps[cur_check_num - 2] # 0 => after first due date
1919     #         to_time = due_timestamps[cur_check_num - 1]   # 1 => before second due date
1920     # else:
1921     #     from_time = 0
1922     #     to_time = due_timestamps[0]
1923     #     cur_check_num = 1
1924
1925     return cur_check_num, from_time, to_time, current_timestamp
1926
1927

```

Here is the call graph for this function:



Here is the caller graph for this function:



6.5.1.2 read_settings()

```

def main.read_settings (
    db_name = 'settings.sqlite3' )

```

Definition at line 63 of file main.py.

```

63 def read_settings(db_name = 'settings.sqlite3' ):
64     import os.path
65
66     if os.path.exists(db_name):
67         with lite.connect(db_name) as con:
68             cur = con.cursor()
69             try:
70                 cur.execute('SELECT * FROM PATHS')
71                 result = cur.fetchone()
72                 for row in result:
73                     print(row)
74                     logisim_path = result[0][0]
75                     working_dir = result[0][1]
76                     # since import is not implemented - ignore import path: import_path = result[0][2]
77                     return logisim_path, working_dir
78             except Exception as e:
79                 print('Was not able to get results from settings DB: ', e)
80     return None
81
82

```

6.5.2 Variable Documentation

6.5.2.1 app

```
main.app = QtWidgets.QApplication(sys.argv)
```

Definition at line 2012 of file main.py.

6.5.2.2 MAIN_FILE_NAME

```
string main.MAIN_FILE_NAME = ''
```

Definition at line 38 of file main.py.

6.5.2.3 MAIN_FILE_NAME_OVERRIDE

```
string main.MAIN_FILE_NAME_OVERRIDE = ''
```

Definition at line 39 of file main.py.

6.5.2.4 MainWindow

```
main.MainWindow = QtWidgets.QMainWindow()
```

Definition at line 2013 of file main.py.

6.5.2.5 styleData

```
string main.styleData
```

Initial value:

```
1 = """
2 /* https://stackoverflow.com/questions/22332106/python-qtgui-qprogressbar-color */
3 QProgressBar
4 {
5     border: 1px solid grey;
6     border-radius: 5px;
7     text-align: center;
8     font-weight: bold;
9 }
10 QProgressBar::chunk
11 {
12     background-color: #d7801a;
13     width: 2.15px;
14     margin: 0.5px;
15 }
16 """
```

Definition at line 41 of file main.py.

6.5.2.6 ui

```
main.ui = UiMainWindow1()
```

Definition at line 2014 of file main.py.

6.6 main_window Namespace Reference

Classes

- class [Ui_mainWindow](#)

6.7 `manage_labs` Namespace Reference

Classes

- class [Ui_manage_labs](#)

6.8 `qt_class_improvements` Namespace Reference

Classes

- class [BetterLineEdit](#)
- class [BetterPlainTextEdit](#)

6.9 `settings` Namespace Reference

Classes

- class [Ui_Settings](#)

6.10 `simple_dialog` Namespace Reference

Classes

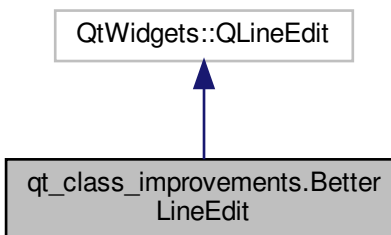
- class [Ui_Dialog](#)

Chapter 7

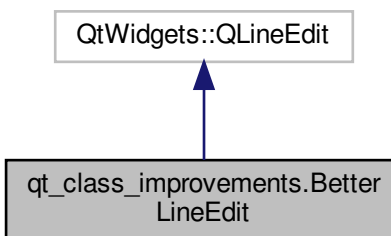
Class Documentation

7.1 qt_class_improvements.BetterLineEdit Class Reference

Inheritance diagram for qt_class_improvements.BetterLineEdit:



Collaboration diagram for qt_class_improvements.BetterLineEdit:



Public Member Functions

- `def __init__ (self, args, kwargs)`
- `def eventFilter (self, obj, event)`
typical way to add event handler

Static Public Attributes

- `dclicked = QtCore.pyqtSignal()`

7.1.1 Detailed Description

Definition at line 11 of file `qt_class_improvements.py`.

7.1.2 Constructor & Destructor Documentation

7.1.2.1 `__init__()`

```
def qt_class_improvements.BetterLineEdit.__init__ (
    self,
    args,
    kwargs )
```

Definition at line 14 of file `qt_class_improvements.py`.

```
14     def __init__(self, *args, **kwargs):
15         QtWidgets.QLineEdit.__init__(self, *args, **kwargs)
16
17         self.installEventFilter(self)
18
```

7.1.3 Member Function Documentation

7.1.3.1 eventFilter()

```
def qt_class_improvements.BetterLineEdit.eventFilter (
    self,
    obj,
    event )
```

typical way to add event handler

Definition at line 20 of file qt_class_improvements.py.

```
20     def eventFilter(self, obj, event):
21         if event.type() == QtCore.QEvent.MouseButtonDblClick:
22             self.dclicked.emit()
23         return False
24
25
26 #     Overloaded QPlainTextEdit to track focus out.
27 #     Needed to implement autosaving of user answer.
28 #
29
```

7.1.4 Member Data Documentation

7.1.4.1 dclicked

```
qt_class_improvements.BetterLineEdit.dclicked = QtCore.pyqtSignal() [static]
```

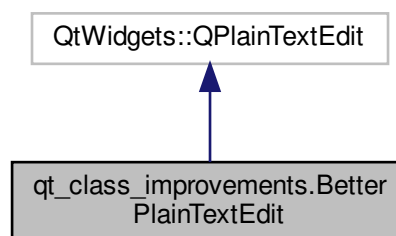
Definition at line 12 of file qt_class_improvements.py.

The documentation for this class was generated from the following file:

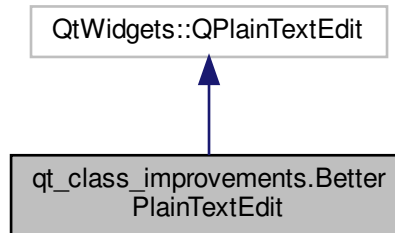
- [qt_class_improvements.py](#)

7.2 qt_class_improvements.BetterPlainTextEdit Class Reference

Inheritance diagram for qt_class_improvements.BetterPlainTextEdit:



Collaboration diagram for qt_class_improvements.BetterPlainTextEdit:



Public Member Functions

- `def __init__(self, args, kwargs)`
- `def eventFilter(self, obj, event)`
typical way to add event handler

Static Public Attributes

- `focus_lost = QtCore.pyqtSignal()`

7.2.1 Detailed Description

Definition at line 30 of file qt_class_improvements.py.

7.2.2 Constructor & Destructor Documentation

7.2.2.1 `__init__()`

```
def qt_class_improvements.BetterPlainTextEdit.__init__ (
    self,
    args,
    kwargs )
```

Definition at line 33 of file qt_class_improvements.py.

```
33     def __init__(self, *args, **kwargs):
34         QtWidgets.QPlainTextEdit.__init__(self, *args, **kwargs)
35
36         self.installEventFilter(self)
37
```

7.2.3 Member Function Documentation

7.2.3.1 eventFilter()

```
def qt_class_improvements.BetterPlainTextEdit.eventFilter (
    self,
    obj,
    event )
```

typical way to add event handler

Definition at line 39 of file qt_class_improvements.py.

```
39     def eventFilter(self, obj, event):
40         if event.type() == QtCore.QEvent.FocusOut:
41             self.focus_lost.emit()
42         return False
43
```

7.2.4 Member Data Documentation

7.2.4.1 focus_lost

```
qt_class_improvements.BetterPlainTextEdit.focus_lost = QtCore.pyqtSignal() [static]
```

Definition at line 31 of file qt_class_improvements.py.

The documentation for this class was generated from the following file:

- [qt_class_improvements.py](#)

7.3 main.CircFile.circ_type Class Reference

Public Member Functions

- [def __init__\(self, name\)](#)

Public Attributes

- [name](#)
- [input_pins](#)
- [output_pins](#)

7.3.1 Detailed Description

Definition at line 85 of file main.py.

7.3.2 Constructor & Destructor Documentation

7.3.2.1 `__init__()`

```
def main.CircFile.circ_type.__init__ (
    self,
    name )
```

Definition at line 86 of file main.py.

```
86         def __init__(self, name):
87             self.name = name
88             self.input_pins = list()
89             self.output_pins = list()
90
91     class PinType:
```

7.3.3 Member Data Documentation

7.3.3.1 `input_pins`

`main.CircFile.circ_type.input_pins`

Definition at line 88 of file main.py.

7.3.3.2 `name`

`main.CircFile.circ_type.name`

Definition at line 87 of file main.py.

7.3.3.3 output_pins

`main.CircFile.circ_type.output_pins`

Definition at line 89 of file main.py.

The documentation for this class was generated from the following file:

- [main.py](#)

7.4 main.CircFile Class Reference

Classes

- class [circ_type](#)
- class [PinType](#)

Public Member Functions

- def [__init__](#) (self, [filename](#))
- def [get_parsed_pins](#) (self)
 : *return:*
- def [get_parsed_pins2](#) (self, [what_to_grade](#))

Public Attributes

- [filename](#)
- [subtract](#)
- [final_grade](#)

7.4.1 Detailed Description

Definition at line 83 of file main.py.

7.4.2 Constructor & Destructor Documentation

7.4.2.1 `__init__()`

```
def main.CircFile.__init__ (
    self,
    filename )
```

Definition at line 97 of file main.py.

```
97     def __init__(self, filename):
98         self.filename = filename
99         self.subtract = 0
100        self.final_grade = 10
101        self.__all_circuits = list()
102
```

7.4.3 Member Function Documentation

7.4.3.1 `get_parsed_pins()`

```
def main.CircFile.get_parsed_pins (
    self )
```

:return:

Definition at line 124 of file main.py.

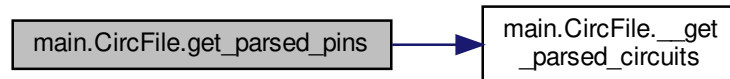
```
124     def get_parsed_pins(self):
125         self.__get_parsed_circuits()
126         arr = self.__all_circuits
127         all_pins = list()
128         for elem in arr:
129             pins = list()
130             for child in elem.findall('comp'):
131                 if child.get('name') == 'Pin':
132                     pins.append(child)
133                     # print(child.tag, child.attrib)
134             all_pins.append(pins)
135
136         clean_data = list()
137         if all_pins:
138             for pins in all_pins: # Although this looks like an error - it is not,
139                 # there is only one iteration. This code will be extended later
140                 # as I had in my older scripts to grade all PLDs.
141                 clean_data = list()
142                 for pin in pins:
143                     name = '0'
144                     io_type = '0'
145                     facing = ''
146                     for elem in list(pin):
147                         if elem.get('name') in ['output', 'input', 'tristate']:
148                             io_type = elem.get('name')
149                         elif elem.get('name') == 'label':
150                             name = elem.get('val')
151                         elif elem.get('name') == 'facing':
152                             facing = elem.get('val')
153                     clean_data.append(self.PinType(name, io_type, facing))
154         else:
155             raise Exception('Error in pin parsing(all_pins)')
156
157         output_pins = list()
```

```

158         input_pins = list()
159         other_pins = list()
160
161         if clean_data:
162             for pin in clean_data:
163                 if pin.type == 'output':
164                     output_pins.append(pin)
165                 elif pin.type == 'input' or pin.type == 'tristate':
166                     input_pins.append(pin)
167                 else:
168                     other_pins.append(pin)
169         else:
170             raise Exception('Error in pin parsing(clean data)')
171
172         return input_pins, output_pins, other_pins
173
174

```

Here is the call graph for this function:



Here is the caller graph for this function:



7.4.3.2 get_parsed_pins2()

```

def main.CircFile.get_parsed_pins2 (
    self,
    what_to_grade )

```

Definition at line 175 of file main.py.

```

175     def get_parsed_pins2(self, what_to_grade):
176
177         tree = ET.parse(self.filename)
178         root = tree.getroot()
179         arr=list()
180         for child in root:
181             # print(child.tag)
182             if child.tag == 'circuit':
183                 arr.append(child)
184             # if child.attrib["name"] == what_to_grade:
185             #     a = child
186             #     b =1
187
188         all_circs = list()
189         good_arr = list()
190         for node in arr:
191             if node.get('name').upper() in what_to_grade:
192                 good_arr.append(node)
193                 circ_instance = self.circ_type(node.get('name'))
194                 all_circs.append(circ_instance)
195                 # print(list(node)[0].items()[0][1])
196
197         all_pins = list()
198         for elem in good_arr:
199             pins = list()
200             for child in elem.findall('comp'):
201                 if child.get('name') == 'Pin':
202                     pins.append(child)
203                     # print(child.tag, child.attrib)
204             all_pins.append(pins)
205
206
207         clean_all_pins = list()
208         for pins in all_pins:
209             clean_data = list()
210             for pin in pins:
211                 name = '0'
212                 type = '0'
213                 for elem in list(pin):
214                     if elem.get('name') in ['output', 'input', 'tristate']:
215                         type = elem.get('name')
216                     elif elem.get('name') == 'label':
217                         name = elem.get('val')
218                 clean_data.append(self.PinType(name, type))
219             clean_all_pins.append(clean_data)
220         for i in range(len(clean_all_pins)):
221             for pin in clean_all_pins[i]:
222                 if pin.type == 'output':
223                     all_circs[i].output_pins.append(pin.name)
224                 else:
225                     all_circs[i].input_pins.append(pin.name)
226         return all_circs
227
228

```

7.4.4 Member Data Documentation

7.4.4.1 filename

main.CircFile.filename

Definition at line 98 of file main.py.

7.4.4.2 final_grade

`main.CircFile.final_grade`

Definition at line 100 of file main.py.

7.4.4.3 subtract

`main.CircFile.subtract`

Definition at line 99 of file main.py.

The documentation for this class was generated from the following file:

- [main.py](#)

7.5 main.Grader Class Reference

Public Member Functions

- `def __init__ (self, working_directory, grader='Ivan')`
- `def open_dir (self)`
- `def check_files (self)`
- `def get_stud_circ_ind (self, student_circuits, circ_to_grade)`
- `def precheck_PLDs (self, stud_ind)`
- `def get_stud_id (self)`
- `def log_update (self, log_event)`
- `def get_parsed_pins (self)`
- `def check_pins_facing (self, pins, corr_facing)`
- `def check_file (self)`
- `def check_circ_exist (self)`
- `def read_resp (self)`
- `def read_resp2 (self)`
- `def read_prev_resp2 (self)`
- `def read_prev_resp (self)`
- `def next_circ (self)`
- `def prev_circ (self)`
- `def check_wrong (self)`
- `def save_grade (self)`
- `def save_responce (self)`
- `def save_all (self)`
- `def save_all2 (self)`
- `def generate_response (self)`
- `def add_to_common_answers (self, typed)`

Public Attributes

- [to_date](#)
- [attempt](#)
- [timestamps](#)
- [stud_ids](#)
- [stud_id](#)
- [submitted](#)
- [input_correct](#)
- [output_correct](#)
- [lab_max_grade](#)
- [subtract](#)
- [final_grade](#)
- [global_log](#)
- [previous_responses](#)
- [file_list](#)
- [resp_text](#)
- [user_comment](#)
- [cur_idx](#)
- [working_dir](#)
- [input_suggestion](#)
- [resp_len](#)
- [logisim_pid](#)
- [circ_file_name](#)
- [lab_type](#)
- [lab_num](#)
- [time](#)
- [circ_obj_ref](#)
- [tot_elem](#)
- [lab_id](#)
- [grader](#)
- [semester](#)
- [lid](#)
- [lab_paths](#)
- [time_from](#)
- [time_to](#)
- [time_cur](#)
- [time_from_qt](#)
- [time_to_qt](#)
- [time_cur_qt](#)
- [what_to_grade](#)
- [all_my_circuits](#)

7.5.1 Detailed Description

Definition at line 229 of file main.py.

7.5.2 Constructor & Destructor Documentation

7.5.2.1 __init__()

```
def main.Grader.__init__ (
    self,
    working_directory,
    grader = 'Ivan' )
```

Definition at line 230 of file main.py.

```
230     def __init__(self, working_directory, grader='Ivan'):
231         self.__from_date = 0
232         self.to_date = 0
233         self.attempt = 0
234         self.timestamps = list()
235         self.stud_ids = list()
236         self.stud_id = ''
237         self.submitted = 0
238         self.input_correct = False
239         self.output_correct = False
240         self.lab_max_grade = 0
241         self.subtract = 0
242         self.__wrong_clicked = False
243         self.final_grade = 0
244         self.__possible_answers_dict = {}
245         self.global_log = ''
246         self.previous_responses = ''
247         self.__message_to_all = ''
248         self.__graded_idlist = list()
249         self.file_list = list()
250         self.resp_text = 'I did not find any errors. Good job!\n'
251         self.user_comment = ''
252         self.cur_idx = 0
253         self.working_dir = working_directory
254         self.input_suggestion = set('',)
255         self.resp_len = 38
256         self.logisim_pid = -1
257         self.circ_file_name = MAIN_FILE_NAME
258         self.lab_type = ''
259         self.lab_num = 0
260         self.time = 0
261         self.circ_obj_ref = None
262         self.tot_elem = 0
263         self.lab_id = ''
264         self.grader = grader
265
```

7.5.3 Member Function Documentation

7.5.3.1 add_to_common_answers()

```
def main.Grader.add_to_common_answers (
    self,
    typed )
```

Definition at line 712 of file main.py.

```
712     def add_to_common_answers(self, typed):
713         self.input_suggestion.add(typed)
714
715
```

7.5.3.2 check_circ_exist()

```
def main.Grader.check_circ_exist (
    self )
```

Definition at line 519 of file main.py.

```
519     def check_circ_exist(self):
520         if not os.path.isfile(self.file_list[self.cur_idx] + '/' + self.circ_file_name):
521             self.resp_text = 'File was not found'
522             file_found = os.listdir(self.file_list[self.cur_idx])
523             potential_files = list()
524             for file in file_found:
525                 if file not in ['grade.txt', 'penalty.txt', 'responce.txt', 'tech_info.txt', ]:
526                     potential_files.append(file)
527             if potential_files:
528                 self.resp_text += '\nNext files|folders were found:\n'
529             for file in potential_files:
530                 if os.path.isdir(self.file_list[self.cur_idx] + '/' + file):
531                     self.resp_text += file + ' - directory.\n'
532                 else:
533                     self.resp_text += file + ' - regular file.\n'
534             self.resp_len = len(self.resp_text)
535             self.final_grade = 0
536             return False
537         return True
538
```

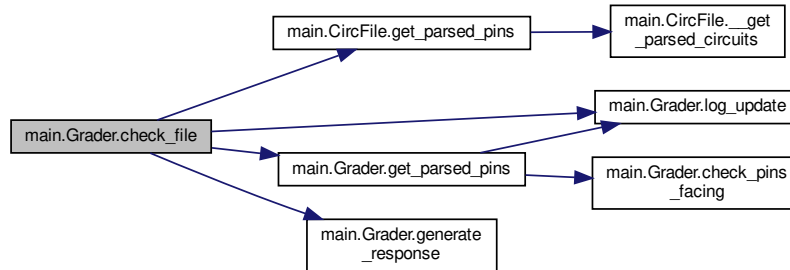
7.5.3.3 check_file()

```
def main.Grader.check_file (
    self )
```

Definition at line 498 of file main.py.

```
498     def check_file(self):
499         file = os.path.join(self.file_list[self.cur_idx], MAIN_FILE_NAME)
500
501         circ_obj = CircFile(file)
502         self.circ_obj_ref = circ_obj
503         self.subtract = 0
504         try:
505             self.get_parsed_pins()
506
507             self.log_update('Pins successfully parsed.')
508             self.final_grade = self.lab_max_grade - self.subtract
509             self.generate_response()
510         except Exception as e:
511             print(e)
512             self.log_update(sys.exc_info()[0])
513
```

Here is the call graph for this function:



7.5.3.4 check_files()

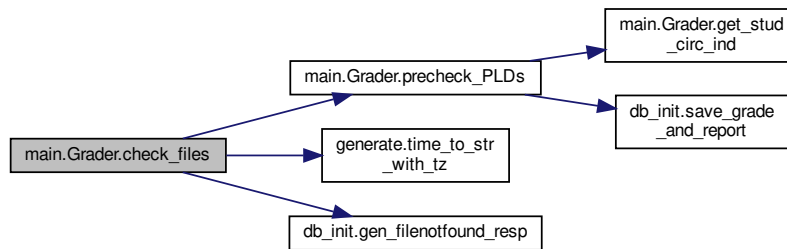
```
def main.Grader.check_files (
    self )
```

Definition at line 348 of file main.py.

```

348     def check_files(self):
349         paths_with_files_list = list()
350         good_ids = list()
351         good_sids = list()
352         good_tss = list()
353
354         for i, stud_path in enumerate(self.lab_paths):
355             cur_path = os.path.join(stud_path, self.circ_file_name)
356             if os.path.exists(cur_path):
357                 paths_with_files_list.append(stud_path)
358                 good_ids.append(self.grade_ids[i])
359                 good_sids.append(self.stud_ids[i])
360                 good_tss.append(self.timestamps[i])
361                 if self.lab_num > 8 and self.lab_type == 'Closed':
362                     self.precheck_PLDs(i)
363             else:
364                 if self.attempt > 1:
365                     next_date = time_to_str_with_tz(self.time_to + self.time_to - self.
time_from)
366                 else:
367                     next_date = time_to_str_with_tz(self.time_to + 604800) # 604800 -
one week in unix time, this line needs corrections for case when you skip a week
368                     gen_filenotfound_resp(self.grade_ids[i], stud_path, self.
circ_file_name, self.grader, self.attempt, next_date)
369                 # self.grade_ids = good_ids
370                 # self.stud_ids = good_sids
371                 # self.timestamps = good_tss
372             return good_tss, good_sids, good_ids, paths_with_files_list
373
```

Here is the call graph for this function:



7.5.3.5 check_pins_facing()

```

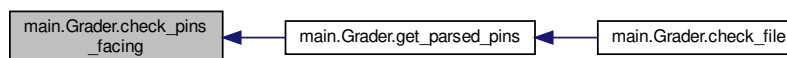
def main.Grader.check_pins_facing (
    self,
    pins,
    corr_facing )
  
```

Definition at line 487 of file main.py.

```

487  def check_pins_facing(self, pins, corr_facing):
488      for pin in pins:
489          if pin.facing != corr_facing and pin.facing != '':
490              return False
491      return True
492
  
```

Here is the caller graph for this function:



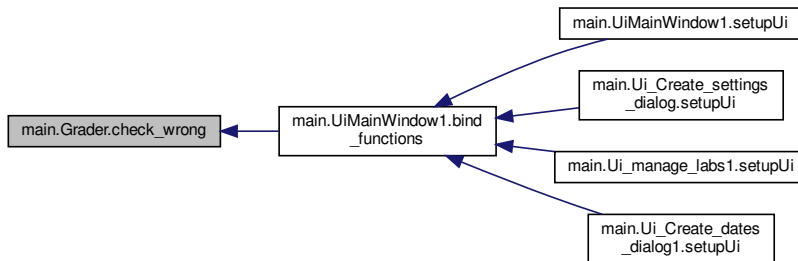
7.5.3.6 check_wrong()

```
def main.Grader.check_wrong (
    self )
```

Definition at line 643 of file main.py.

```
643     def check_wrong(self):
644         self.final_grade = 0
645         self.resp_text = 'your lab was marked as wrong. You should fix errors listed below and resubmit it.'
646         self.resp_len = len(self.resp_text)
647
```

Here is the caller graph for this function:



7.5.3.7 generate_response()

```
def main.Grader.generate_response (
    self )
```

Definition at line 693 of file main.py.

```
693     def generate_response(self):
694         self.resp_text = ''
695         self.user_comment = ''
696         if self.input_correct and self.output_correct:
697             self.resp_text = 'I did not find any errors. Good job!'
698         else:
699             if not self.input_correct:
700                 self.resp_text += 'Your input pins have wrong orientation.\n'
701             if not self.output_correct:
702                 self.resp_text += 'Your output pins have wrong orientation.\n'
703         self.resp_len = len(self.resp_text)
704
705
```

Here is the caller graph for this function:



7.5.3.8 get_parsed_pins()

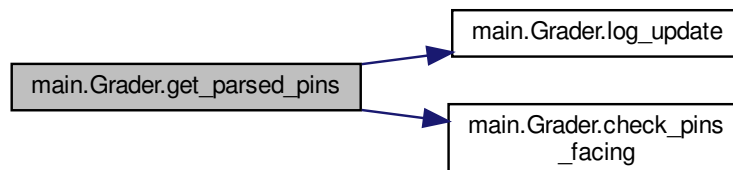
```
def main.Grader.get_parsed_pins (
    self )
```

Definition at line 459 of file main.py.

```

459     def get_parsed_pins(self):
460         try:
461             input_pins, output_pins, other_pins = self.circ_obj_ref.get_parsed_pins()
462             if other_pins:
463                 self.log_update('I was not able to recognize ' + str(len(other_pins)) + " pins.")
464             self.input_correct = True
465             self.output_correct = True
466             if not self.check_pins_facing(pins=input_pins, corr_facing='east'):
467                 self.subtract += 1
468             self.input_correct = False
469             if not self.check_pins_facing(pins=output_pins, corr_facing='west'):
470                 self.subtract += 1
471             self.output_correct = False
472         except Exception as e: # TODO check for FileNotFoundError and assign ZERO
473             print(e)
474             # self.log_update(sys.exc_info()[0])
475             # print(sys.exc_info()[0])
476             raise
477         # self.log_update('Done checking: ' + self.filename)
478
479
```

Here is the call graph for this function:



Here is the caller graph for this function:



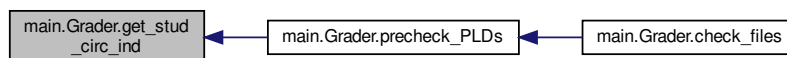
7.5.3.9 `get_stud_circ_ind()`

```
def main.Grader.get_stud_circ_ind (
    self,
    student_circuits,
    circ_to_grade )
```

Definition at line 374 of file main.py.

```
374     def get_stud_circ_ind(self, student_circuits, circ_to_grade):
375         for stud_circ in student_circuits:
376             if stud_circ.name.upper() == circ_to_grade.upper():
377                 return student_circuits.index(stud_circ)
378         for stud_circ in student_circuits:
379             print(stud_circ.name.upper())
380         return -1
381
```

Here is the caller graph for this function:



7.5.3.10 `get_stud_id()`

```
def main.Grader.get_stud_id (
    self )
```

Definition at line 443 of file main.py.

```
443     def get_stud_id(self):
444         return self.stud_id
445
```

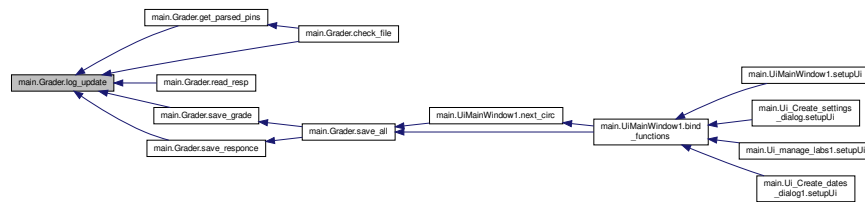
7.5.3.11 log_update()

```
def main.Grader.log_update (
    self,
    log_event )
```

Definition at line 452 of file main.py.

```
452     def log_update(self, log_event):
453         self.global_log += str(self.stud_id) + ': ' + str(log_event) + '\n'
454
```

Here is the caller graph for this function:



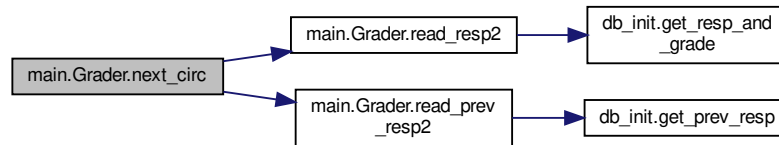
7.5.3.12 next_circ()

```
def main.Grader.next_circ (
    self )
```

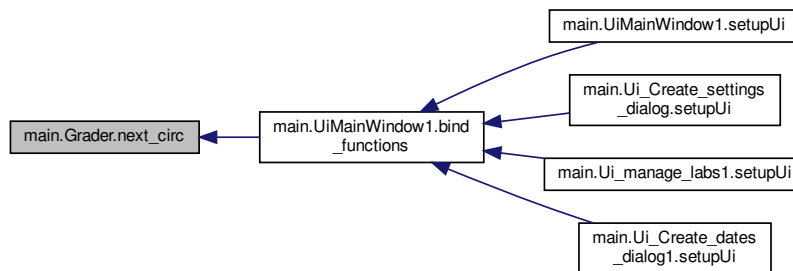
Definition at line 601 of file main.py.

```
601     def next_circ(self):
602         self.cur_idx += 1
603         # self.check_file(self.cur_idx)
604         self.user_comment = ''
605         graded = self.read_resp2()
606         # if graded:
607         self.read_prev_resp2()
608         # if self.check_circ_exist():
609         #     self.read_resp()
610         self.stud_id = self.stud_ids[self.cur_idx]
611         # try:
612         #     self.read_prev_resp()
613         # except Exception as e:
614         #     print('Error during attempt to read prev resp when opening next circuit: ', e)
615         #     # TODO add handler
616         return self.cur_idx
617
```

Here is the call graph for this function:



Here is the caller graph for this function:



7.5.3.13 open_dir()

```
def main.Grader.open_dir (
    self )
```

Definition at line 270 of file main.py.

```

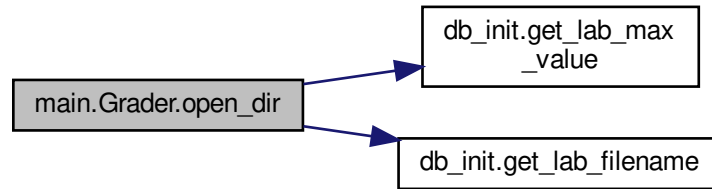
270     def open_dir(self):
271         # TODO check behaviour when directory is wrong.
272         # if len(self.working_directory) < 3:
273         #     wdir = './'
274         # else:
275         #     wdir = self.working_directory
276
277
278         root, dirs, files = os.walk(self.working_dir).__next__()
279         files.sort()
280         # check_file = files[0] # not used at this time
281         # if len(files) < 1:
282         #     raise Exception("No due files ? Extra files in working directory ?")
283         # due_file = files[1] # TODO: change this to a better design. - Already changed
284
285         self.lab_type = self.working_dir.split('/')[-2].split('_')[0]
286         self.lab_num = int(self.working_dir.split('/')[-2].split('_')[2])
  
```

```

287         self.attempt = int(self.working_dir.split('/')[-2].split('_')[3])
288
289         if self.lab_type == 'Closed':
290             self.lab_id = 'CLA{}'.format(self.lab_num)
291             # self.lab_max_grade = 10
292         else: # Open
293             # self.lab_max_grade = 20
294             self.lab_id = 'OLA{}'.format(self.lab_num)
295
296         self.lab_max_grade = get_lab_max_value(self.lab_id)
297
298         # self.time = int(due_file[6:])
299
300         # dirs.sort() # sort list of submitted labs
301         # if dirs[0] == 'Answers':
302         #     dirs.pop(0)
303
304         self.circ_file_name = get_lab_filename(self.lab_id)[0]
305         self.year, self.semester = self.working_dir.split('/')[-3].split('_')
306         self.lid = get_labid_in_schedule(get_lab_id(self.lab_type, self.
lab_num), self.year, self.semester)
307         self.timestamps, self.stud_ids, self.grade_ids, self.lab_paths =
get_empty_grades_by_lid(self.lid, self.attempt)
308
309         atime = get_grading_period(self.lid, cur_only=True)
310         self.time_from = atime[1]
311         self.time_to = atime[2]
312         self.time_cur = atime[3]
313
314         self.time_from_qt = QDateTime.fromSecsSinceEpoch(self.time_from)
315         self.time_to_qt = QDateTime.fromSecsSinceEpoch(self.time_to)
316         self.time_cur_qt = QDateTime.fromSecsSinceEpoch(self.time_cur)
317
318         if self.lab_num > 8 and self.lab_type == 'Closed':
319             if self.lab_num == 9:
320                 self.what_to_grade = ['PC_BUS', 'AR_LD', 'PC_LD', 'PC_INC', 'DR_LD', 'DR_BUS']
321             elif self.lab_num == 10:
322                 self.what_to_grade = ["R_LD", "R_BUS", "S_LD", "ACC_CLR", "ACC_LD", "ACC_BUS", "ALU_SEL"]
323             elif self.lab_num == 11:
324                 self.what_to_grade = ["Z_LD", "OUTR_LD", "RAM_RW", "RAM_EN", "IR_LD", "SC_CLR"]
325             circ = CircFile('/home/vanya/Documents/3130_labs/2018_2/PLDs.circ')
326             self.all_my_circuits = circ.get_parsed_pins2(self.what_to_grade)
327
328         if self.lab_paths is not None and len(self.lab_paths) > 0:
329             self.timestamps, self.stud_ids, self.grade_ids, self.lab_paths = self.check_files()
330
331         if self.lab_paths is None or len(self.lab_paths) == 0: # if there are no ungraded labs - display
all labs
332             self.timestamps, self.stud_ids, self.grade_ids, self.lab_paths =
get_all_grades_by_lid(self.lid, self.attempt)
333
334             # self.grades = [self.lab_max_grade]*len(self.grade_ids)
335             # self.stud_ids = dirs
336             # self.stud_ids = list()
337             # self.timestamps = list()
338             # # directory_list = list()
339             # for name in dirs:
340             #     self.file_list.append(os.path.join(root, name))
341             #     temp_arr = name.split('-')
342             #     self.stud_ids.append(temp_arr[0])
343             #     self.timestamps.append(int(temp_arr[2]))
344
345             # for file in self.file_list:
346             #     print(file)
347

```

Here is the call graph for this function:



7.5.3.14 precheck_PLDs()

```
def main.Grader.precheck_PLDs (
    self,
    stud_ind )
```

Definition at line 382 of file main.py.

```

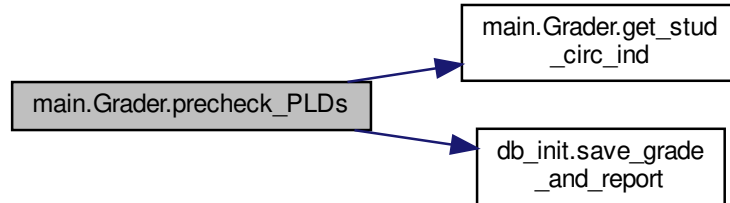
382     def precheck_PLDs(self, stud_ind):
383         file = os.path.join(self.lab_paths[stud_ind], self.circ_file_name)
384
385         student_circuits = CircFile(file).get_parsed_pins2(self.what_to_grade)
386         errors = 0
387
388         out_str = '<br> Next part was generated by automatic grader that I wrote several years ago.' \
389             'If you are not agree with something or suspect an error - please send me a
message.<br>With this grading approach you cat get nonzero grade ' \
390             'even if not everything was correct.<br>'
391         for circ_to_grade in self.what_to_grade:
392             for good_circ in self.all_my_circuits:
393                 if good_circ.name.upper() == circ_to_grade.upper():
394                     cur_ind = self.get_stud_circ_ind(student_circuits, circ_to_grade)
395                     out_str += '<br>'
396                     if cur_ind == -1:
397                         out_str += '<font color="red">{} NOT FOUND!<br> </font>'.format(circ_to_grade)
398                         errors += 1
399                     else:
400                         check_pins = student_circuits[cur_ind].input_pins
401                         for i in range(len(check_pins)):
402                             if check_pins[i][0].lower() != 'c':
403                                 # print(check_pins[i])
404                                 if len(check_pins[i][1:]) > 0:
405                                     try:
406                                         pos = None
407                                         for ch in check_pins[i]:
408                                             if not ch.isalpha():
409                                                 pos = check_pins[i].index(ch)
410                                                 break
411                                         num = int(check_pins[i][pos:])
412                                     except Exception as e:
413                                         print(e)
414                                         continue
415                                 check_pins[i] = check_pins[i][0:1] + str(num)
416         student_circuits_sorted = sorted(check_pins)
```

```

417         good_circ_sorted = sorted(good_circ.input_pins)
418         sm = difflib.SequenceMatcher(None, student_circuits_sorted, good_circ_sorted)
419         res_ratio = sm.ratio()
420
421         if res_ratio > 0.99:
422             out_str += '<font color="green"> {} : PERFECT MATCH!<br> </font>'.format(
circ_to_grade)
423         elif res_ratio > 0.15:
424             out_str += '{} :Great news : you match ratio is {:.1%} (>75%)<br>{} :
<b>FOUND</b></b> {} <br>{} : <b>EXPECTED</b> {} <br>' \
425                 .format(circ_to_grade, res_ratio, circ_to_grade, ' '.join(
student_circuits_sorted), circ_to_grade, ' '.join(good_circ_sorted))
426             errors += 1
427         else:
428             out_str += '<font color="red">{} Bad news : you match ratio is only', \
429                 '{} {:.1%} - this means that you have to significantly change your
circuit. <br> ' \
430                 'Please send me a message if you need some advice.<br> </font>'.
format(circ_to_grade, res_ratio)
431             errors += 1
432
433         final_grade = math.ceil(10 * (len(self.what_to_grade) - errors) / len(self.what_to_grade))
434         # out_str += '<br> Bad grade confidence: ' + conf + ' (this is for Ivan)<br>' + '<br> Next part
will be typed manually: <br>'
435         save_grade_and_report(self.grade_ids[stud_ind], final_grade, out_str, None,
self.grader)
436         return final_grade, out_str
437
438

```

Here is the call graph for this function:



Here is the caller graph for this function:



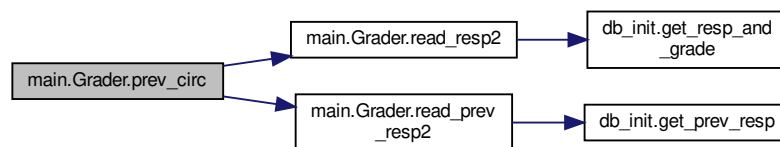
7.5.3.15 prev_circ()

```
def main.Grader.prev_circ (
    self )
```

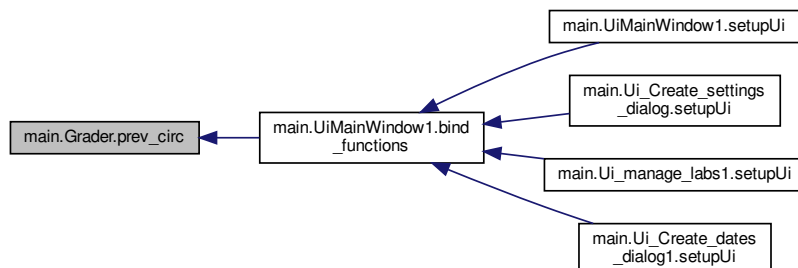
Definition at line 622 of file main.py.

```
622     def prev_circ(self):
623         self.cur_idx -= 1
624         # self.check_file(self.cur_idx)
625         self.user_comment = ''
626         graded = self.read_resp2()
627         if graded:
628             self.read_prev_resp2()
629         # if self.check_circ_exist():
630         #     self.read_resp()
631         self.stud_id = self.stud_ids[self.cur_idx]
632         # try:
633         #     self.read_prev_resp()
634         # except Exception as e:
635         #     print('Error during attempt to read prev resp when opening prev circuit: ', e)
636         #     # TODO add handler
637         return self.cur_idx
638
```

Here is the call graph for this function:



Here is the caller graph for this function:



7.5.3.16 read_prev_resp()

```
def main.Grader.read_prev_resp (
    self )
```

Definition at line 581 of file main.py.

```
581     def read_prev_resp(self):
582         if self.attempt > 1:
583             self.previous_responses = '' # TODO find same name in folder name
584             prev_att = int(self.working_dir[-2:-1])
585             for i in range(prev_att-1, 0, -1):
586                 prev_working_dir = self.working_dir[:-2] + str(i) + '/'
587                 for file in os.listdir(prev_working_dir):
588                     if file.__contains__(self.stud_id):
589                         # print(file)
590                         try:
591                             with open(prev_working_dir + file + '/responce.txt', 'r') as resp_file:
592                                 self.previous_responses += str(i) + 'th submission :\n\t' \
593                                                         + '\n'.join(resp_file.readlines())
594                         except Exception as e:
595                             print('Error in read prev responce: ', e)
596
```

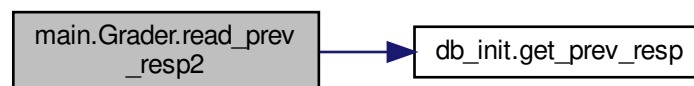
7.5.3.17 read_prev_resp2()

```
def main.Grader.read_prev_resp2 (
    self )
```

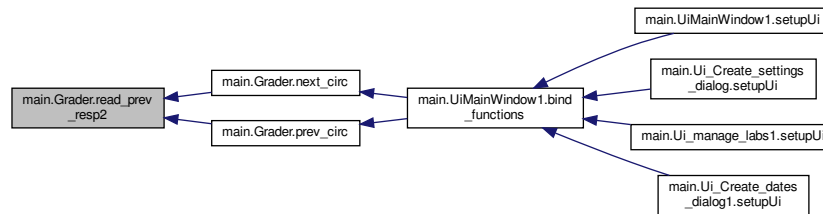
Definition at line 573 of file main.py.

```
573     def read_prev_resp2(self):
574         self.previous_responses = get_prev_resp(self.grade_ids[self.cur_idx], self.stud_ids[
575             self.cur_idx], self.lid)
```

Here is the call graph for this function:



Here is the caller graph for this function:



7.5.3.18 read_resp()

```
def main.Grader.read_resp (
    self )
```

Definition at line 544 of file main.py.

```

544     def read_resp(self):
545         self.submitted = self.timestamps[self.cur_idx]
546         try:
547             with open(os.path.join(self.file_list[self.cur_idx], 'responce.txt'), 'r') as resp_file:
548                 a = resp_file.readlines()
549                 self.resp_text = ''.join(a)
550                 self.resp_len = len(self.resp_text)
551         except Exception as e:
552             print(e)
553             self.log_update(sys.exc_info()[0])
554
555         try:
556             with open(os.path.join(self.file_list[self.cur_idx], 'grade.txt'), 'r') as grade_file:
557                 self.final_grade = int(grade_file.readline())
558         except Exception as e:
559             print(e)
560             self.log_update(sys.exc_info()[0])
561
562         # self.read_prev_resp()
563
```

Here is the call graph for this function:



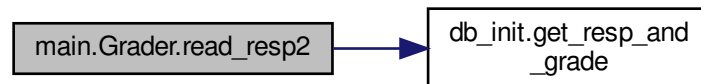
7.5.3.19 read_resp2()

```
def main.Grader.read_resp2 (
    self )
```

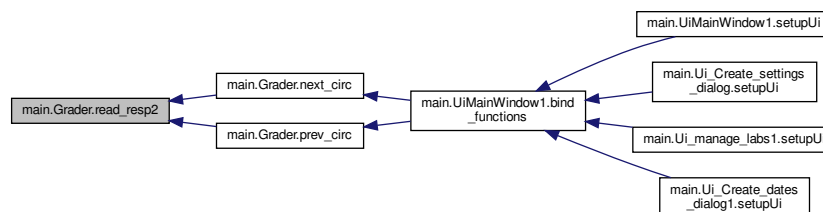
Definition at line 564 of file main.py.

```
564     def read_resp2(self):
565         self.final_grade, self.resp_text, self.user_comment, graded =
get_resp_and_grade(self.grade_ids[self.cur_idx])
566         if graded is None:
567             self.final_grade = self.lab_max_grade
568             self.resp_text = 'I did not find any errors. Good job!'
569             # self.resp_text = '' if self.resp_text is None else self.resp_text
570             self.resp_len = len(self.resp_text)
571             return graded
572
```

Here is the call graph for this function:



Here is the caller graph for this function:



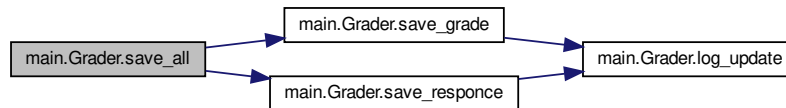
7.5.3.20 save_all()

```
def main.Grader.save_all (
    self )
```

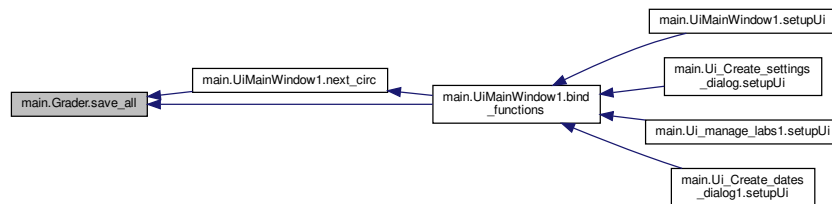
Definition at line 677 of file main.py.

```
677     def save_all(self):
678         self.save_grade()
679         self.save_responce()
680
681
```

Here is the call graph for this function:



Here is the caller graph for this function:



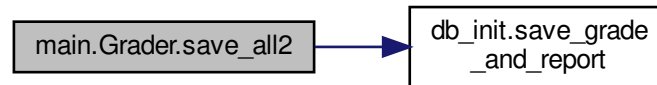
7.5.3.21 save_all2()

```
def main.Grader.save_all2 (
    self )
```

Definition at line 686 of file main.py.

```
686     def save_all2(self):
687         save_grade_and_report(self.grade_ids[self.cur_idx], self.final_grade, self.
        resp_text, self.user_comment, self.grader)
688
```

Here is the call graph for this function:



7.5.3.22 save_grade()

```
def main.Grader.save_grade (
    self )
```

Definition at line 652 of file main.py.

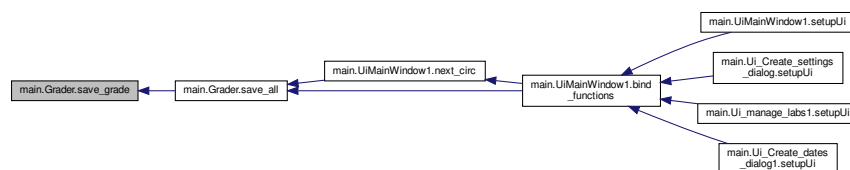
```

652     def save_grade(self):
653         file = os.path.join(self.lab_paths[self.cur_idx], 'grade.txt')
654         with open(file, 'w') as grade_file:
655             grade_file.write(str(self.final_grade))
656
657         self.log_update('Grade saved')
658 
```

Here is the call graph for this function:



Here is the caller graph for this function:



7.5.3.23 save_response()

```
def main.Grader.save_response (
    self )
```

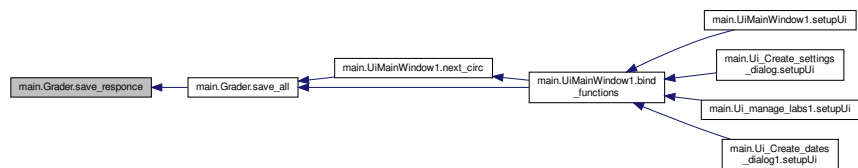
Definition at line 664 of file main.py.

```
664     def save_response(self):
665         file = os.path.join(self.lab_paths[self.cur_idx], 'response.txt')
666         with open(file, 'w') as resp_file:
667             resp_file.write(self.resp_text)
668             if self.user_comment:
669                 resp_file.write('\nAdditional comment: ' + self.user_comment + '\n')
670         self.log_update('Response saved')
671
```

Here is the call graph for this function:



Here is the caller graph for this function:



7.5.4 Member Data Documentation

7.5.4.1 all_my_circuits

```
main.Grader.all_my_circuits
```

Definition at line 326 of file main.py.

7.5.4.2 attempt

`main.Grader.attempt`

Definition at line 233 of file main.py.

7.5.4.3 circ_file_name

`main.Grader.circ_file_name`

Definition at line 257 of file main.py.

7.5.4.4 circ_obj_ref

`main.Grader.circ_obj_ref`

Definition at line 261 of file main.py.

7.5.4.5 cur_idx

`main.Grader.cur_idx`

Definition at line 252 of file main.py.

7.5.4.6 file_list

`main.Grader.file_list`

Definition at line 249 of file main.py.

7.5.4.7 final_grade

`main.Grader.final_grade`

Definition at line 243 of file main.py.

7.5.4.8 global_log

`main.Grader.global_log`

Definition at line 245 of file main.py.

7.5.4.9 grader

`main.Grader.grader`

Definition at line 264 of file main.py.

7.5.4.10 input_correct

`main.Grader.input_correct`

Definition at line 238 of file main.py.

7.5.4.11 input_suggestion

`main.Grader.input_suggestion`

Definition at line 254 of file main.py.

7.5.4.12 lab_id

`main.Grader.lab_id`

Definition at line 263 of file main.py.

7.5.4.13 lab_max_grade

`main.Grader.lab_max_grade`

Definition at line 240 of file main.py.

7.5.4.14 lab_num

`main.Grader.lab_num`

Definition at line 259 of file main.py.

7.5.4.15 lab_paths

`main.Grader.lab_paths`

Definition at line 307 of file main.py.

7.5.4.16 lab_type

`main.Grader.lab_type`

Definition at line 258 of file main.py.

7.5.4.17 lid

`main.Grader.lid`

Definition at line 306 of file main.py.

7.5.4.18 logisim_pid

`main.Grader.logisim_pid`

Definition at line 256 of file main.py.

7.5.4.19 output_correct

`main.Grader.output_correct`

Definition at line 239 of file main.py.

7.5.4.20 previous_responses

`main.Grader.previous_responses`

Definition at line 246 of file main.py.

7.5.4.21 resp_len

`main.Grader.resp_len`

Definition at line 255 of file main.py.

7.5.4.22 resp_text

`main.Grader.resp_text`

Definition at line 250 of file main.py.

7.5.4.23 semester

`main.Grader.semester`

Definition at line 305 of file main.py.

7.5.4.24 stud_id

`main.Grader.stud_id`

Definition at line 236 of file main.py.

7.5.4.25 stud_ids

`main.Grader.stud_ids`

Definition at line 235 of file main.py.

7.5.4.26 submitted

`main.Grader.submitted`

Definition at line 237 of file main.py.

7.5.4.27 subtract

`main.Grader.subtract`

Definition at line 241 of file main.py.

7.5.4.28 time

`main.Grader.time`

Definition at line 260 of file main.py.

7.5.4.29 time_cur

`main.Grader.time_cur`

Definition at line 312 of file main.py.

7.5.4.30 time_cur_qt

`main.Grader.time_cur_qt`

Definition at line 316 of file main.py.

7.5.4.31 time_from

`main.Grader.time_from`

Definition at line 310 of file main.py.

7.5.4.32 time_from_qt

`main.Grader.time_from_qt`

Definition at line 314 of file main.py.

7.5.4.33 time_to

`main.Grader.time_to`

Definition at line 311 of file main.py.

7.5.4.34 time_to_qt

`main.Grader.time_to_qt`

Definition at line 315 of file main.py.

7.5.4.35 timestamps

`main.Grader.timestamps`

Definition at line 234 of file main.py.

7.5.4.36 to_date

`main.Grader.to_date`

Definition at line 232 of file main.py.

7.5.4.37 tot_elem

`main.Grader.tot_elem`

Definition at line 262 of file main.py.

7.5.4.38 `user_comment`

`main.Grader.user_comment`

Definition at line 251 of file `main.py`.

7.5.4.39 `what_to_grade`

`main.Grader.what_to_grade`

Definition at line 320 of file `main.py`.

7.5.4.40 `working_dir`

`main.Grader.working_dir`

Definition at line 253 of file `main.py`.

The documentation for this class was generated from the following file:

- [main.py](#)

7.6 `main.CircFile.PinType` Class Reference

Public Member Functions

- `def __init__(self, name, iotype, facing=None)`

Public Attributes

- [name](#)
- [type](#)
- [facing](#)

7.6.1 Detailed Description

Definition at line 91 of file `main.py`.

7.6.2 Constructor & Destructor Documentation

7.6.2.1 `__init__()`

```
def main.CircFile.PinType.__init__ (
    self,
    name,
    iotype,
    facing = None )
```

Definition at line 92 of file main.py.

```
92     def __init__(self, name, iotype, facing=None):
93         self.name = name
94         self.type = iotype
95         self.facing = facing
96
97     def __init__(self, filename):
```

7.6.3 Member Data Documentation

7.6.3.1 `facing`

main.CircFile.PinType.facing

Definition at line 95 of file main.py.

7.6.3.2 `name`

main.CircFile.PinType.name

Definition at line 93 of file main.py.

7.6.3.3 `type`

main.CircFile.PinType.type

Definition at line 94 of file main.py.

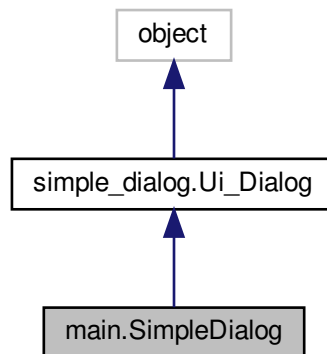
The documentation for this class was generated from the following file:

- [main.py](#)

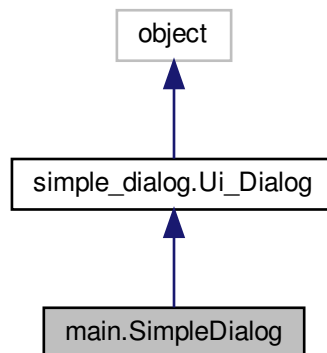
7.7 main.SimpleDialog Class Reference

Wrapper class for very simple Ok|Cancel dialog.

Inheritance diagram for main.SimpleDialog:



Collaboration diagram for main.SimpleDialog:



Public Member Functions

- def [setUpUi](#) (self, Dialog, phrase)

Additional Inherited Members

7.7.1 Detailed Description

Wrapper class for very simple Ok|Cancel dialog.

Definition at line 1564 of file main.py.

7.7.2 Member Function Documentation

7.7.2.1 setupUi()

```
def main.SimpleDialog.setupUi (
    self,
    Dialog,
    phrase )
```

Definition at line 1569 of file main.py.

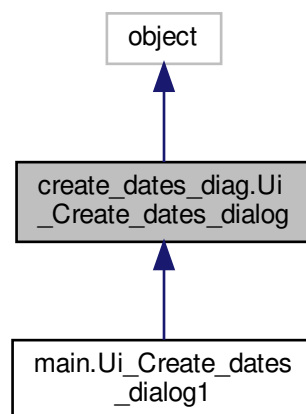
```
1569     def setupUi(self, Dialog, phrase):
1570         super().setupUi(Dialog)
1571         self.label_main_question.setText(phrase)
1572
1573
```

The documentation for this class was generated from the following file:

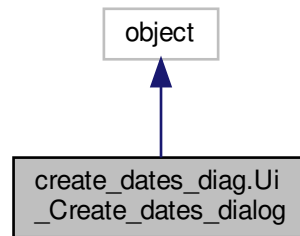
- [main.py](#)

7.8 create_dates_diag.Ui_Create_dates_dialog Class Reference

Inheritance diagram for create_dates_diag.Ui_Create_dates_dialog:



Collaboration diagram for create_dates_diag.Ui_Create_dates_dialog:



Public Member Functions

- def [setUpUi](#) (self, Create_dates_dialog)
- def [retranslateUi](#) (self, Create_dates_dialog)

Public Attributes

- [verticalLayout](#)
- [horizontalLayout_5](#)
- [lab_path](#)
- [horizontalLayout](#)
- [init_subm_date_time](#)
- [init_label](#)
- [horizontalLayout_2](#)
- [first_subm_date_time](#)
- [first_label](#)
- [horizontalLayout_3](#)
- [second_subm_date_time](#)
- [second_label](#)
- [horizontalLayout_4](#)
- [third_subm_date_time](#)
- [third_label](#)
- [buttonBox](#)

7.8.1 Detailed Description

Definition at line 11 of file `create_dates_diag.py`.

7.8.2 Member Function Documentation

7.8.2.1 retranslateUi()

```
def create_dates_diag.Ui_Create_dates_dialog.retranslateUi (
    self,
    Create_dates_dialog )
```

Definition at line 96 of file create_dates_diag.py.

```
96     def retranslateUi(self, Create_dates_dialog):
97
100         _translate = QtCore.QCoreApplication.translate
101         Create_dates_dialog.setWindowTitle(_translate("Create_dates_dialog", "Dialog"))
102         self.lab_path.setToolTip(_translate("Create_dates_dialog", "Tripple for file dialog"))
103         self.lab_path.setPlaceholderText(_translate("Create_dates_dialog", "DoubleClick to select path"))
104         self.init_label.setText(_translate("Create_dates_dialog", "Submission date"))
105         self.first_label.setText(_translate("Create_dates_dialog", "1st resubmission"))
106         self.second_label.setText(_translate("Create_dates_dialog", "2nd resubmission"))
107         self.third_label.setText(_translate("Create_dates_dialog", "3rd resubmission"))
108
```

7.8.2.2 setupUi()

```
def create_dates_diag.Ui_Create_dates_dialog.setupUi (
    self,
    Create_dates_dialog )
```

Definition at line 12 of file create_dates_diag.py.

```
12     def setupUi(self, Create_dates_dialog):
13         Create_dates_dialog.setObjectName("Create_dates_dialog")
14         Create_dates_dialog.resize(589, 250)
15         Create_dates_dialog.setMinimumSize(QtCore.QSize(500, 250))
16         Create_dates_dialog.setMaximumSize(QtCore.QSize(1000, 300))
17         icon = QtGui.QIcon()
18         icon.addPixmap(QtGui.QPixmap("os_linux_1.ico"), QtGui.QIcon.Normal, QtGui.QIcon.Off)
19         Create_dates_dialog.setWindowIcon(icon)
20         self.verticalLayout = QtWidgets.QVBoxLayout(Create_dates_dialog)
21         self.verticalLayout.setObjectName("verticalLayout")
22         self.horizontalLayout_5 = QtWidgets.QHBoxLayout()
23         self.horizontalLayout_5.setObjectName("horizontalLayout_5")
24         self.lab_path = BetterLineEdit(Create_dates_dialog)
25         self.lab_path.setFocusPolicy(QtCore.Qt.StrongFocus)
26         self.lab_path.setStatusTip("")
27         self.lab_path.setWhatsThis("")
28         self.lab_path.setAccessibleName("")
29         self.lab_path.setAccessibleDescription("")
30         self.lab_path.setInputMask("")
31         self.lab_path.setReadOnly(False)
32         self.lab_path.setCursorMoveStyle(QtCore.Qt.LogicalMoveStyle)
33         self.lab_path.setClearButtonEnabled(False)
34         self.lab_path.setObjectName("lab_path")
35         self.horizontalLayout_5.addWidget(self.lab_path)
36         self.verticalLayout.addLayout(self.horizontalLayout_5)
37         self.horizontalLayout = QtWidgets.QHBoxLayout()
38         self.horizontalLayout.setObjectName("horizontalLayout")
```

```

39     self.init_subm_date_time = QtWidgets.QDateTimeEdit(Create_dates_dialog)
40     self.init_subm_date_time.setMaximumSize(QtCore.QSize(150, 40))
41     self.init_subm_date_time.setCalendarPopup(True)
42     self.init_subm_date_time.setObjectName("init_subm_date_time")
43     self.horizontalLayout.addWidget(self.init_subm_date_time)
44     self.init_label = QtWidgets.QLabel(Create_dates_dialog)
45     self.init_label.setObjectName("init_label")
46     self.horizontalLayout.addWidget(self.init_label)
47     self.verticalLayout.addLayout(self.horizontalLayout)
48     self.horizontalLayout_2 = QtWidgets.QHBoxLayout()
49     self.horizontalLayout_2.setObjectName("horizontalLayout_2")
50     self.first_subm_date_time = QtWidgets.QDateTimeEdit(Create_dates_dialog)
51     self.first_subm_date_time.setMaximumSize(QtCore.QSize(150, 35))
52     self.first_subm_date_time.setCalendarPopup(True)
53     self.first_subm_date_time.setObjectName("first_subm_date_time")
54     self.horizontalLayout_2.addWidget(self.first_subm_date_time)
55     self.first_label = QtWidgets.QLabel(Create_dates_dialog)
56     self.first_label.setObjectName("first_label")
57     self.horizontalLayout_2.addWidget(self.first_label)
58     self.verticalLayout.addLayout(self.horizontalLayout_2)
59     self.horizontalLayout_3 = QtWidgets.QHBoxLayout()
60     self.horizontalLayout_3.setObjectName("horizontalLayout_3")
61     self.second_subm_date_time = QtWidgets.QDateTimeEdit(Create_dates_dialog)
62     self.second_subm_date_time.setMaximumSize(QtCore.QSize(150, 35))
63     self.second_subm_date_time.setCalendarPopup(True)
64     self.second_subm_date_time.setObjectName("second_subm_date_time")
65     self.horizontalLayout_3.addWidget(self.second_subm_date_time)
66     self.second_label = QtWidgets.QLabel(Create_dates_dialog)
67     self.second_label.setObjectName("second_label")
68     self.horizontalLayout_3.addWidget(self.second_label)
69     self.verticalLayout.addLayout(self.horizontalLayout_3)
70     self.horizontalLayout_4 = QtWidgets.QHBoxLayout()
71     self.horizontalLayout_4.setObjectName("horizontalLayout_4")
72     self.third_subm_date_time = QtWidgets.QDateTimeEdit(Create_dates_dialog)
73     self.third_subm_date_time.setMaximumSize(QtCore.QSize(150, 35))
74     self.third_subm_date_time.setCalendarPopup(True)
75     self.third_subm_date_time.setObjectName("third_subm_date_time")
76     self.horizontalLayout_4.addWidget(self.third_subm_date_time)
77     self.third_label = QtWidgets.QLabel(Create_dates_dialog)
78     self.third_label.setObjectName("third_label")
79     self.horizontalLayout_4.addWidget(self.third_label)
80     self.verticalLayout.addLayout(self.horizontalLayout_4)
81     self.buttonBox = QtWidgets.QDialogButtonBox(Create_dates_dialog)
82     self.buttonBox.setMaximumSize(QtCore.QSize(16777215, 40))
83     self.buttonBox.setOrientation(QtCore.Qt.Horizontal)
84     self.buttonBox.setStandardButtons(QtWidgets.QDialogButtonBox.Abort |
QtWidgets.QDialogButtonBox.SaveAll)
85     self.buttonBox.setObjectName("buttonBox")
86     self.verticalLayout.addWidget(self.buttonBox)
87
88     self.retranslateUi(Create_dates_dialog)
89     self.buttonBox.accepted.connect(Create_dates_dialog.accept)
90     self.buttonBox.rejected.connect(Create_dates_dialog.reject)
91     QtCore.QMetaObject.connectSlotsByName(Create_dates_dialog)
92     Create_dates_dialog.setTabOrder(self.init_subm_date_time, self.first_subm_date_time)
93     Create_dates_dialog.setTabOrder(self.first_subm_date_time, self.second_subm_date_time)
94     Create_dates_dialog.setTabOrder(self.second_subm_date_time, self.third_subm_date_time)
95

```

7.8.3 Member Data Documentation

7.8.3.1 buttonBox

create_dates_diag.Ui_Create_dates_dialog.buttonBox

Definition at line 81 of file create_dates_diag.py.

7.8.3.2 first_label

`create_dates_diag.Ui_Create_dates_dialog.first_label`

Definition at line 55 of file `create_dates_diag.py`.

7.8.3.3 first_subm_date_time

`create_dates_diag.Ui_Create_dates_dialog.first_subm_date_time`

Definition at line 50 of file `create_dates_diag.py`.

7.8.3.4 horizontalLayout

`create_dates_diag.Ui_Create_dates_dialog.horizontalLayout`

Definition at line 37 of file `create_dates_diag.py`.

7.8.3.5 horizontalLayout_2

`create_dates_diag.Ui_Create_dates_dialog.horizontalLayout_2`

Definition at line 48 of file `create_dates_diag.py`.

7.8.3.6 horizontalLayout_3

`create_dates_diag.Ui_Create_dates_dialog.horizontalLayout_3`

Definition at line 59 of file `create_dates_diag.py`.

7.8.3.7 horizontalLayout_4

`create_dates_diag.Ui_Create_dates_dialog.horizontalLayout_4`

Definition at line 70 of file `create_dates_diag.py`.

7.8.3.8 horizontalLayout_5

`create_dates_diag.Ui_Create_dates_dialog.horizontalLayout_5`

Definition at line 22 of file `create_dates_diag.py`.

7.8.3.9 init_label

`create_dates_diag.Ui_Create_dates_dialog.init_label`

Definition at line 44 of file `create_dates_diag.py`.

7.8.3.10 init_subm_date_time

`create_dates_diag.Ui_Create_dates_dialog.init_subm_date_time`

Definition at line 39 of file `create_dates_diag.py`.

7.8.3.11 lab_path

`create_dates_diag.Ui_Create_dates_dialog.lab_path`

Definition at line 24 of file `create_dates_diag.py`.

7.8.3.12 second_label

`create_dates_diag.Ui_Create_dates_dialog.second_label`

Definition at line 66 of file `create_dates_diag.py`.

7.8.3.13 second_subm_date_time

`create_dates_diag.Ui_Create_dates_dialog.second_subm_date_time`

Definition at line 61 of file `create_dates_diag.py`.

7.8.3.14 third_label

`create_dates_diag.Ui_Create_dates_dialog.third_label`

Definition at line 77 of file `create_dates_diag.py`.

7.8.3.15 third_subm_date_time

`create_dates_diag.Ui_Create_dates_dialog.third_subm_date_time`

Definition at line 72 of file `create_dates_diag.py`.

7.8.3.16 verticalLayout

`create_dates_diag.Ui_Create_dates_dialog.verticalLayout`

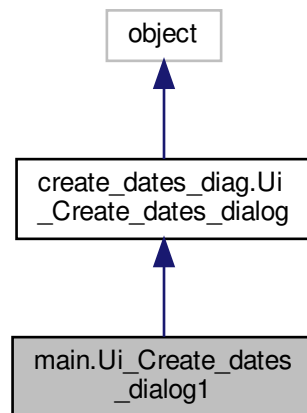
Definition at line 20 of file `create_dates_diag.py`.

The documentation for this class was generated from the following file:

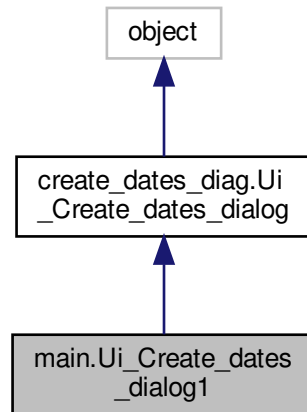
- [create_dates_diag.py](#)

7.9 main.Ui_Create_dates_dialog1 Class Reference

Inheritance diagram for `main.Ui_Create_dates_dialog1`:



Collaboration diagram for `main.Ui_Create_dates_dialog1`:



Public Member Functions

- def [bind_functions](#) (self)
- def [setupUi](#) (self, Create_dates_dialog, selected_lab="")
- def [date_select](#) (self)
- def [open_file_diag](#) (self)

Additional Inherited Members

7.9.1 Detailed Description

Definition at line 1928 of file main.py.

7.9.2 Member Function Documentation

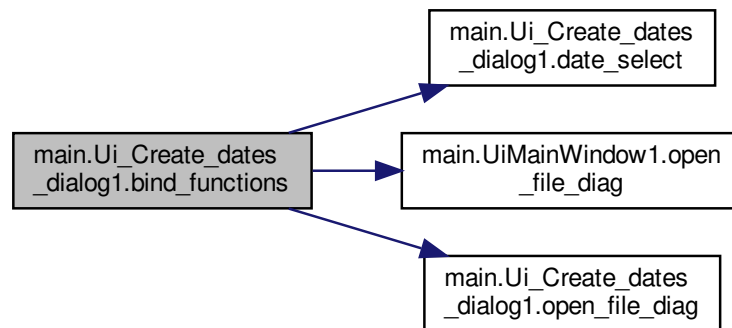
7.9.2.1 bind_functions()

```
def main.Ui_Create_dates_dialog1.bind_functions (
    self )
```

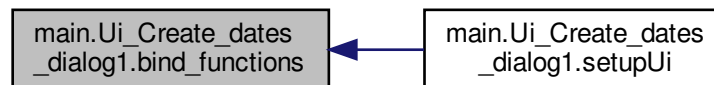
Definition at line 1934 of file main.py.

```
1934     def bind_functions(self):
1935         self.init_subm_date_time.dateTimeChanged.connect(self.date_select)
1936         # self.select_file_path.clicked.connect(self.open_file_diag)
1937         # self.lineEdit.left_clicked[int].connect(self.dummy_d)
1938         self.lab_path.dclicked.connect(self.open_file_diag)
1939
```

Here is the call graph for this function:



Here is the caller graph for this function:



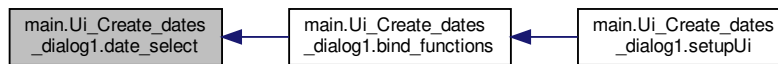
7.9.2.2 date_select()

```
def main.Ui_Create_dates_dialog1.date_select (
    self )
```

Definition at line 1987 of file main.py.

```
1987     def date_select(self):
1988         self.first_subm_date_time.setDateTime(self.init_subm_date_time.dateTime().addDays(7))
1989         self.second_subm_date_time.setDateTime(self.init_subm_date_time.dateTime().addDays(14))
1990         self.third_subm_date_time.setDateTime(self.init_subm_date_time.dateTime().addDays(21))
1991
```

Here is the caller graph for this function:



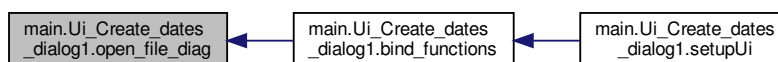
7.9.2.3 open_file_diag()

```
def main.Ui_Create_dates_dialog1.open_file_diag (
    self )
```

Definition at line 1996 of file main.py.

```
1996     def open_file_diag(self):
1997         obtained_dir = QFileDialog.getExistingDirectory(caption='Select where to create due files',
1998                                                         directory=self.lab_path.text())+'/'
1999         if len(obtained_dir) > 1:
2000             self.lab_path.setText(obtained_dir)
2001
2002
```

Here is the caller graph for this function:



7.9.2.4 setupUi()

```
def main.Ui_Create_dates_dialog1.setupUi (  
    self,  
    Create_dates_dialog,  
    selected_lab = '' )
```

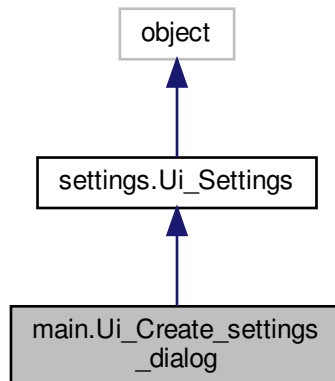
Definition at line 1971 of file main.py.

```
1971     def setupUi(self, Create_dates_dialog, selected_lab=''):  
1972         super().setupUi(Create_dates_dialog)  
1973         self.bind_functions()  
1974         self.init_subm_date_time.setDateTime(QDateTime.currentDateTime())  
1975         paths, local = settings_db_read_settings()  
1976         good_path = get_full_path(paths, local) + '/server_sync/'  
1977         try:  
1978             good_path += selected_lab + '/'  
1979         except Exception as e:  
1980             print('Exception when tried to append selected folder from Manage labs. ', e)  
1981             self.lab_path.setText(good_path)  
1982
```

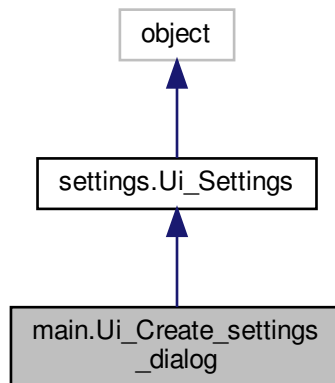

7.10 main.Ui_Create_settings_dialog Class Reference

Creates window that provides user with convenient way of changing settings that are stored in sqlite3 db.

Inheritance diagram for main.Ui_Create_settings_dialog:



Collaboration diagram for main.Ui_Create_settings_dialog:



Public Member Functions

- def [bind_functions](#) (self)

- def [setUpUi](#) (self, Settings)
- def [update_user_input_with_paths](#) (self)
Reads settings parameters from DB and sets appropriate fields with obtained values.
- def [set_default_user_input_with_paths](#) (self)
- def [read_settings_data](#) (self)
- def [create_or_update_settings_db](#) (self)
- def [import_students](#) (self)
- def [set_apply_restet_active](#) (self)
- def [open_simple_dialog](#) (self, phrase)

Public Attributes

- [simple_diag](#)

7.10.1 Detailed Description

Creates window that provides user with convenient way of changing settings that are stored in sqlite3 db.

Definition at line 1341 of file main.py.

7.10.2 Member Function Documentation

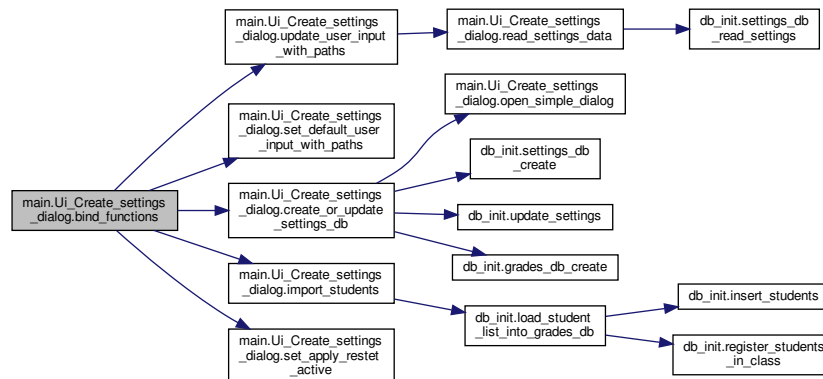
7.10.2.1 bind_functions()

```
def main.Ui_Create_settings_dialog.bind_functions (
    self )
```

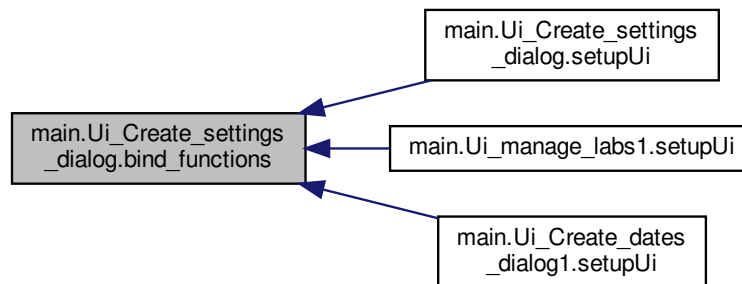
Definition at line 1347 of file main.py.

```
1347     def bind_functions(self):
1348         self.buttonBox.button(self.buttonBox.Reset).clicked.connect(self.update_user_input_with_paths)
1349         self.buttonBox.button(self.buttonBox.RestoreDefaults).clicked.connect(self.
set_default_user_input_with_paths)
1350         self.buttonBox.button(self.buttonBox.Apply).clicked.connect(self.create_or_update_settings_db)
1351         self.buttonBox.button(self.buttonBox.Ok).clicked.connect(self.create_or_update_settings_db)
1352
1353         self.import_stuents_btn.clicked.connect(self.import_students)
1354
1355         # TODO: make 'personal' events and update only fields that have been changed
1356         self.input_logisim_path.textChanged.connect(self.set_apply_restet_active)
1357         self.input_local_stor.textChanged.connect(self.set_apply_restet_active)
1358         self.input_rem_stor.textChanged.connect(self.set_apply_restet_active)
1359         self.input_grader_name.textChanged.connect(self.set_apply_restet_active)
1360         self.spin_year.valueChanged.connect(self.set_apply_restet_active)
1361         self.semester_comboBox.currentIndexChanged.connect(self.set_apply_restet_active)
1362         self.style_checkBox.stateChanged.connect(self.set_apply_restet_active)
1363         self.sync_command.textChanged.connect(self.set_apply_restet_active)
1364         self.input_grades_db.textChanged.connect(self.set_apply_restet_active)
1365
```

Here is the call graph for this function:



Here is the caller graph for this function:



7.10.2.2 create_or_update_settings_db()

```
def main.Ui_Create_settings_dialog.create_or_update_settings_db (
    self )
```

Definition at line 1439 of file main.py.

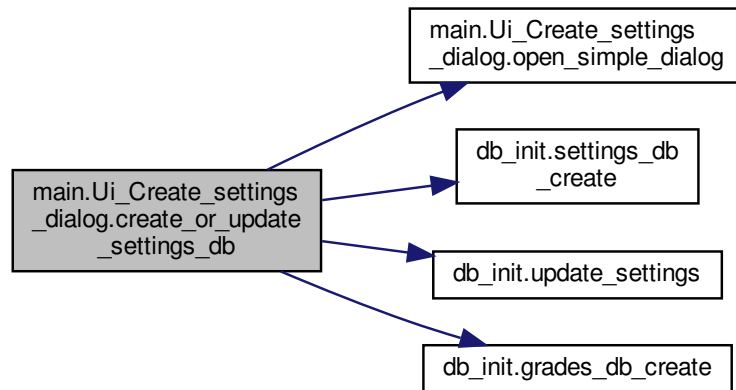
```
1439     def create_or_update_settings_db(self):
1440         from pathlib import Path
1441         settings_location = str(Path(os.path.expandvars(os.path.expanduser('./settings.sqlite3'))).absolute
    ())
1442         if not os.path.isfile(settings_location):
```

```

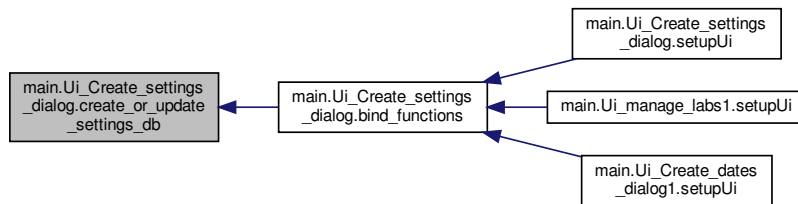
1443         if self.open_simple_dialog("Do you want to create settings database ?"):
1444             if not settings_db_create(force=True):
1445                 raise Exception('Was not able to create SETTINGS db.')
1446     if len(self.input_local_stor.text()) > 0:
1447         if self.input_local_stor.text()[-1] != '/':
1448             self.input_local_stor.setText(self.input_local_stor.text() + '/')
1449     if len(self.input_rem_stor.text()) > 0:
1450         if self.input_rem_stor.text()[-1] != '/':
1451             self.input_rem_stor.setText(self.input_rem_stor.text() + '/')
1452     if len(self.input_logisim_path.text()) > 0:
1453         if self.input_logisim_path.text()[-1] != '/':
1454             self.input_logisim_path.setText(self.input_logisim_path.text() + '/')
1455
1456     paths = (self.input_logisim_path.text(), self.input_local_stor.text(), self.input_rem_stor.text(),
1457              self.input_grades_db.text())
1458     if os.path.isfile(settings_location):
1459         local = (self.input_grader_name.text(), int(self.spin_year.text()),
1460                 self.semester_comboBox.currentIndex(), self.style_checkBox.checkState(), self.
1461                 sync_command.text())
1462     if len(self.input_local_stor.text()) > 0:
1463         local_stor = str(Path(os.path.expanduser(os.path.expandvars(self.input_local_stor.text()))
1464                               .absolute()))
1465         if local_stor[-1] != '/':
1466             local_stor += '/'
1467         if not os.path.isdir(local_stor):
1468             os.mkdir(local_stor)
1469         local_grading_path = local_stor + self.spin_year.text() + '_' + \
1470                               str(self.semester_comboBox.currentIndex())
1471         if not os.path.isdir(local_grading_path):
1472             os.mkdir(local_grading_path)
1473         update_settings(paths, local)
1474
1475     grades_location = str(Path(os.path.expandvars(os.path.expanduser(self.input_grades_db.text()))
1476                               .absolute()))
1477     if len(self.input_grades_db.text()) > 1 and not os.path.isfile(grades_location):
1478         if self.open_simple_dialog("Do you want to create GRADES database ?"):
1479             print('Before grades creation.')
1480             if not grades_db_create(grades_location, force=True):
1481                 raise Exception('Was not able to create GRADES db.')
1482
1483     if os.path.isfile(settings_location) and os.path.isfile(grades_location):
1484         self.buttonBox.button(self.buttonBox.Apply).setDisabled(True)
1485         self.buttonBox.button(self.buttonBox.Apply).repaint()
1486         self.buttonBox.button(self.buttonBox.Reset).setDisabled(True)
1487         self.buttonBox.button(self.buttonBox.Reset).repaint()
1488         if not self.groupBox_user.isEnabled():
1489             self.groupBox_user.setEnabled(True)
1490         if not self.input_logisim_path.isEnabled():
1491             self.input_logisim_path.setEnabled(True)
1492             self.label_logisim_path.setEnabled(True)
1493         if not self.input_local_stor.isEnabled():
1494             self.input_local_stor.setEnabled(True)
1495             self.label_local_stor.setEnabled(True)
1496         if not self.input_rem_stor.isEnabled():
1497             self.input_rem_stor.setEnabled(True)
1498             self.label_rem_stor.setEnabled(True)
1499         if not self.spin_year.isEnabled():
1500             self.spin_year.setEnabled(True)
1501         if not self.semester_comboBox.isEnabled():
1502             self.semester_comboBox.setEnabled(True)
1503         if not self.style_checkBox.isEnabled():
1504             self.style_checkBox.setEnabled(True)
1505         if not self.input_grader_name.isEnabled():
1506             self.input_grader_name.setEnabled(True)
1507         if not self.sync_command.isEnabled():
1508             self.sync_command.setEnabled(True)
1509
1510     # if len(self.input_local_stor.text()) > 1:
1511     #     full_path = Path(self.input_local_stor.text()).absolute()
1512     #     if not os.path.exists(full_path) or not os.path.isdir(full_path):
1513     #         os.makedirs(full_path)

```

Here is the call graph for this function:



Here is the caller graph for this function:



7.10.2.3 import_students()

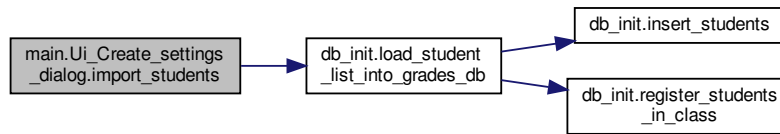
```
def main.Ui_Create_settings_dialog.import_students (
    self )
```

Definition at line 1518 of file main.py.

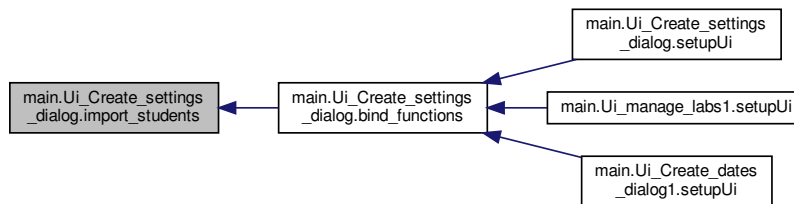
```

1518     def import_students(self):
1519         self.import_stuents_btn.setEnabled(False)
1520         stud_file = QFileDialog.getOpenFileName(caption="Select file with students' info", directory='.',
filter="Text files (*.txt)")
1521         if len(stud_file[0]) > 3:
1522             load_student_list_into_grades_db(self.input_grades_db.text(),
self.spin_year.value(), self.semester_comboBox.currentIndex(), filename=stud_file[0])
1523
1524
1525         self.import_stuents_btn.setEnabled(True)
1526
```

Here is the call graph for this function:



Here is the caller graph for this function:



7.10.2.4 open_simple_dialog()

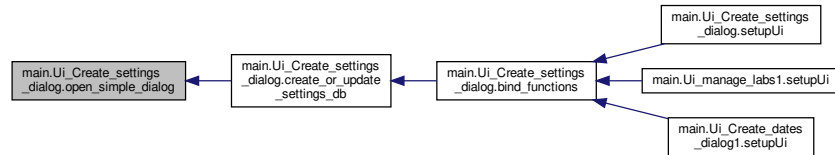
```
def main.Ui_Create_settings_dialog.open_simple_dialog (
    self,
    phrase )
```

Definition at line 1543 of file main.py.

```

1543     def open_simple_dialog(self, phrase):
1544         self.simple_diag = QtWidgets.QDialog()
1545         dui = SimpleDialog()
1546         dui.setupUi(self.simple_diag, phrase)
1547
1548         self.buttonBox.setDisabled(True)
1549         self.buttonBox.repaint()
1550
1551         self.simple_diag.setWindowTitle('Settings confirmation')
1552         self.simple_diag.show()
1553
1554         result = self.simple_diag.exec_()
1555
1556         self.buttonBox.setEnabled(True)
1557
1558         return result
1559
1560
```


Here is the caller graph for this function:



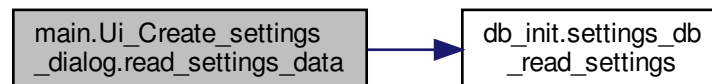
7.10.2.5 read_settings_data()

```
def main.Ui_Create_settings_dialog.read_settings_data (
    self )
```

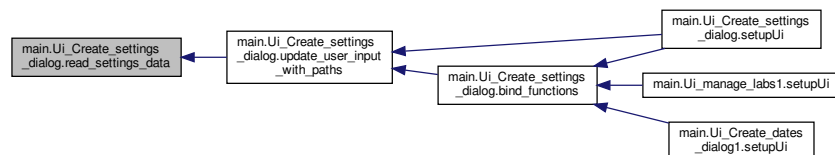
Definition at line 1428 of file main.py.

```
1428     def read_settings_data(self):
1429         self.buttonBox.button(self.buttonBox.Reset).setEnabled(True)
1430         return settings_db_read_settings()
1431
```

Here is the call graph for this function:



Here is the caller graph for this function:



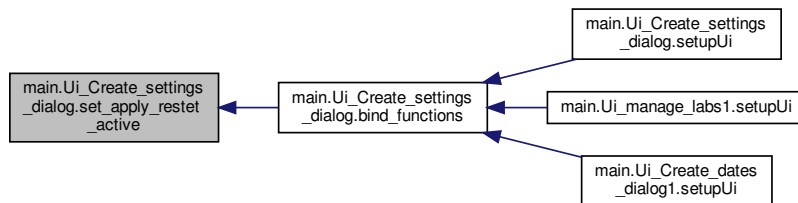
7.10.2.6 set_apply_restet_active()

```
def main.Ui_Create_settings_dialog.set_apply_restet_active (
    self )
```

Definition at line 1531 of file main.py.

```
1531     def set_apply_restet_active(self):
1532         self.buttonBox.button(self.buttonBox.Reset).setEnabled(True)
1533         self.buttonBox.button(self.buttonBox.Apply).setEnabled(True)
1534
```

Here is the caller graph for this function:



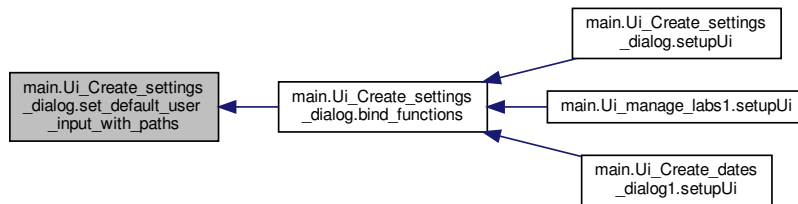
7.10.2.7 set_default_user_input_with_paths()

```
def main.Ui_Create_settings_dialog.set_default_user_input_with_paths (
    self )
```

Definition at line 1415 of file main.py.

```
1415     def set_default_user_input_with_paths(self):
1416         self.input_logisim_path.setText("~/Downloads/")
1417         self.input_local_stor.setText("~/Documents/3130_labs/")
1418         self.input_grades_db.setText("~/Documents/3130_labs/grades.sqlite3")
1419         self.input_rem_stor.setText("") # impossible to predict
1420         self.groupBox_user.setEnabled(True)
1421         self.buttonBox.button(self.buttonBox.Reset).setEnabled(True)
1422         self.buttonBox.button(self.buttonBox.Apply).setEnabled(True)
1423
```

Here is the caller graph for this function:



7.10.2.8 setupUi()

```
def main.Ui_Create_settings_dialog.setupUi (
    self,
    Settings )
```

Definition at line 1370 of file main.py.

```
1370     def setupUi(self, Settings):
1371         super().setupUi(Settings)
1372         self.buttonBox.button(self.buttonBox.Reset).setDisabled(True)
1373         self.buttonBox.button(self.buttonBox.Apply).setDisabled(True)
1374         self.bind_functions()
1375         self.update_user_input_with_paths()
1376
```



Reads settings parameters from DB and sets appropriate fields with obtained values.

dependa on a number of settings obtained from read_settings_data :return: Nothing

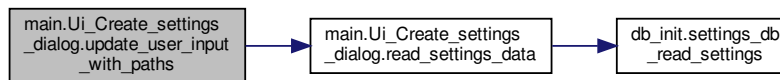
Generated by Doxygen

```

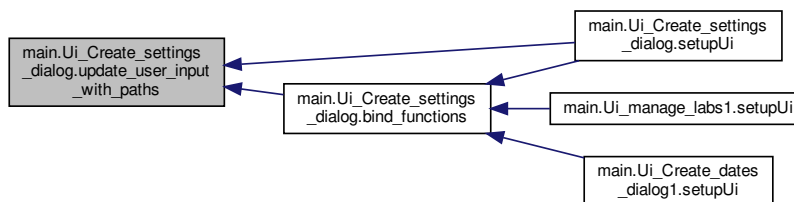
1382     def update_user_input_with_paths(self):
1383         paths, local = self.read_settings_data()
1384         if paths and len(paths) >= 4:
1385             self.input_logisim_path.setText(paths[0])
1386             self.input_local_stor.setText(paths[1])
1387             self.input_rem_stor.setText(paths[2])
1388             self.input_grades_db.setText(paths[3])
1389             self.groupBox_user.setEnabled(True)
1390
1391         if local and len(local) >= 4:
1392             self.input_grader_name.setText(local[0])
1393             self.spin_year.setValue(local[1])
1394             self.semester_comboBox.setCurrentIndex(int(local[2]))
1395             self.style_checkBox.setChecked(bool(local[3]))
1396             self.sync_command.setText(local[4])
1397
1398         if (paths and len(paths) >= 4) and (local and len(local) >= 4):
1399             self.spin_year.setEnabled(True)
1400             self.semester_comboBox.setEnabled(True)
1401             self.style_checkBox.setEnabled(True)
1402             self.input_grader_name.setEnabled(True)
1403             self.sync_command.setEnabled(True)
1404         # if (local and len(local) > 5) or len(paths):
1405         #     print('Obtained more settings than expected. Please check Ui_Create_settings_dialog.')
1406
1407         self.buttonBox.button(self.buttonBox.Reset).setDisabled(True)
1408         self.buttonBox.button(self.buttonBox.Apply).setDisabled(True)
1409

```

Here is the call graph for this function:



Here is the caller graph for this function:



7.10.3 Member Data Documentation

7.10.3.1 simple_diag

`main.Ui_Create_settings_dialog.simple_diag`

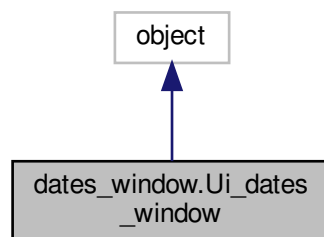
Definition at line 1544 of file main.py.

The documentation for this class was generated from the following file:

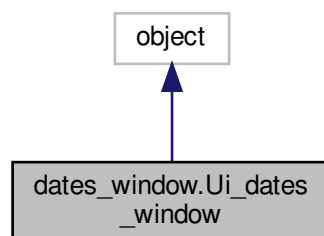
- [main.py](#)

7.11 dates_window.Ui_dates_window Class Reference

Inheritance diagram for `dates_window.Ui_dates_window`:



Collaboration diagram for `dates_window.Ui_dates_window`:



Public Member Functions

- def [setUpUi](#) (self, dates_window)
- def [retranslateUi](#) (self, dates_window)

Public Attributes

- [buttonBox](#)
- [calendarWidget](#)

7.11.1 Detailed Description

Definition at line 11 of file dates_window.py.

7.11.2 Member Function Documentation

7.11.2.1 retranslateUi()

```
def dates_window.Ui_dates_window.retranslateUi (
    self,
    dates_window )
```

Definition at line 29 of file dates_window.py.

```
29     def retranslateUi(self, dates_window):
30
31         _translate = QtCore.QCoreApplication.translate
32         dates_window.setWindowTitle(_translate("dates_window", "Check dates"))
33         self.calendarWidget.setAccessibleName(_translate("dates_window", "cal_diag"))
34
35
36
37
```

7.11.2.2 setUpUi()

```
def dates_window.Ui_dates_window.setUpUi (
    self,
    dates_window )
```

Definition at line 12 of file dates_window.py.

```
12     def setUpUi(self, dates_window):
13         dates_window.setObjectName("dates_window")
14         dates_window.resize(251, 314)
15         self.buttonBox = QtWidgets.QDialogButtonBox(dates_window)
16         self.buttonBox.setGeometry(QtCore.QRect(40, 260, 191, 32))
17         self.buttonBox.setOrientation(QtCore.Qt.Horizontal)
18         self.buttonBox.setStandardButtons(QtWidgets.QDialogButtonBox.Cancel|QtWidgets.QDialogButtonBox.Ok)
19         self.buttonBox.setObjectName("buttonBox")
20         self.calendarWidget = QtWidgets.QCalendarWidget(dates_window)
21         self.calendarWidget.setGeometry(QtCore.QRect(10, 10, 224, 232))
22         self.calendarWidget.setObjectName("calendarWidget")
23
24         self.retranslateUi(dates_window)
25         self.buttonBox.accepted.connect(dates_window.accept)
26         self.buttonBox.rejected.connect(dates_window.reject)
27         QtCore.QMetaObject.connectSlotsByName(dates_window)
28
```

7.11.3 Member Data Documentation

7.11.3.1 `buttonBox`

`dates_window.Ui_dates_window.buttonBox`

Definition at line 15 of file `dates_window.py`.

7.11.3.2 `calendarWidget`

`dates_window.Ui_dates_window.calendarWidget`

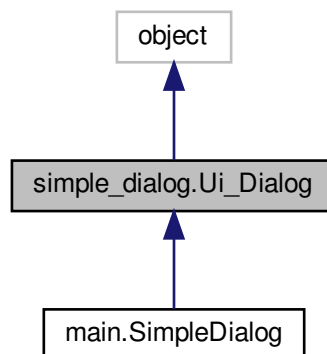
Definition at line 20 of file `dates_window.py`.

The documentation for this class was generated from the following file:

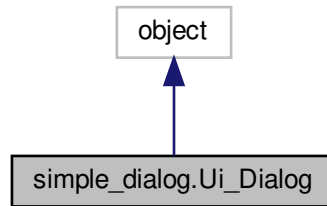
- [dates_window.py](#)

7.12 `simple_dialog.Ui_Dialog` Class Reference

Inheritance diagram for `simple_dialog.Ui_Dialog`:



Collaboration diagram for simple_dialog.Ui_Dialog:



Public Member Functions

- def [setupUi](#) (self, Dialog)
- def [retranslateUi](#) (self, Dialog)

Public Attributes

- [verticalLayout](#)
- [label_main_question](#)
- [buttonBox_simple_dial](#)

7.12.1 Detailed Description

Definition at line 11 of file simple_dialog.py.

7.12.2 Member Function Documentation

7.12.2.1 retranslateUi()

```
def simple_dialog.Ui_Dialog.retranslateUi (  
    self,  
    Dialog )
```

Definition at line 46 of file simple_dialog.py.

```
46     def retranslateUi(self, Dialog):  
47  
48         _translate = QtCore.QCoreApplication.translate  
51         Dialog.setWindowTitle(_translate("Dialog", "Create database ?"))  
52         self.label_main_question.setText(_translate("Dialog", "Database will be created.  Confirm.."))  
53  
54
```

7.12.2.2 setupUi()

```
def simple_dialog.Ui_Dialog.setupUi (
    self,
    Dialog )
```

Definition at line 12 of file simple_dialog.py.

```
12     def setupUi(self, Dialog):
13         Dialog.setObjectName("Dialog")
14         Dialog.resize(328, 76)
15         icon = QtGui.QIcon()
16         icon.addPixmap(QtGui.QPixmap("os_linux_1.ico"), QtGui.QIcon.Normal, QtGui.QIcon.Off)
17         Dialog.setWindowIcon(icon)
18         Dialog.setLocale(QtCore.QLocale(QtCore.QLocale.English, QtCore.QLocale.UnitedStates))
19         self.verticalLayout = QtWidgets.QVBoxLayout(Dialog)
20         self.verticalLayout.setObjectName("verticalLayout")
21         self.label_main_question = QtWidgets.QLabel(Dialog)
22         sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Minimum, QtWidgets.QSizePolicy.Minimum)
23         sizePolicy.setHorizontalStretch(0)
24         sizePolicy.setVerticalStretch(0)
25         sizePolicy.setHeightForWidth(self.label_main_question.sizePolicy().hasHeightForWidth())
26         self.label_main_question.setSizePolicy(sizePolicy)
27         self.label_main_question.setAlignment(QtCore.Qt.AlignCenter)
28         self.label_main_question.setObjectName("label_main_question")
29         self.verticalLayout.addWidget(self.label_main_question)
30         self.buttonBox_simple_dial = QtWidgets.QDialogButtonBox(Dialog)
31         sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Minimum, QtWidgets.QSizePolicy.Fixed)
32         sizePolicy.setHorizontalStretch(0)
33         sizePolicy.setVerticalStretch(0)
34         sizePolicy.setHeightForWidth(self.buttonBox_simple_dial.sizePolicy().hasHeightForWidth())
35         self.buttonBox_simple_dial.setSizePolicy(sizePolicy)
36         self.buttonBox_simple_dial.setOrientation(QtCore.Qt.Horizontal)
37         self.buttonBox_simple_dial.setStandardButtons(QtWidgets.QDialogButtonBox.Cancel|
QtWidgets.QDialogButtonBox.Ok)
38         self.buttonBox_simple_dial.setObjectName("buttonBox_simple_dial")
39         self.verticalLayout.addWidget(self.buttonBox_simple_dial)
40
41         self.retranslateUi(Dialog)
42         self.buttonBox_simple_dial.accepted.connect(Dialog.accept)
43         self.buttonBox_simple_dial.rejected.connect(Dialog.reject)
44         QtCore.QMetaObject.connectSlotsByName(Dialog)
45
```

7.12.3 Member Data Documentation

7.12.3.1 buttonBox_simple_dial

```
simple_dialog.Ui_Dialog.buttonBox_simple_dial
```

Definition at line 30 of file simple_dialog.py.

7.12.3.2 label_main_question

```
simple_dialog.Ui_Dialog.label_main_question
```

Definition at line 21 of file simple_dialog.py.

7.12.3.3 verticalLayout

`simple_dialog.Ui_Dialog.verticalLayout`

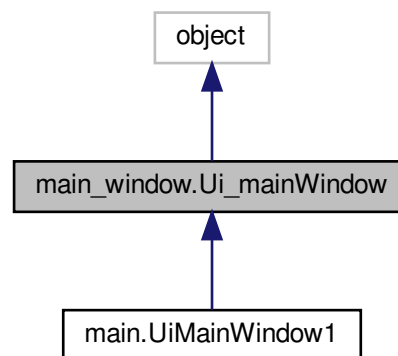
Definition at line 19 of file `simple_dialog.py`.

The documentation for this class was generated from the following file:

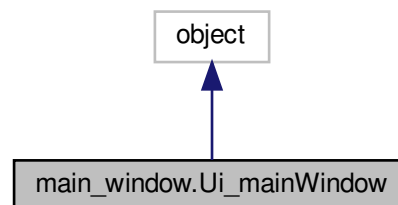
- [simple_dialog.py](#)

7.13 main_window.Ui_mainWindow Class Reference

Inheritance diagram for `main_window.Ui_mainWindow`:



Collaboration diagram for `main_window.Ui_mainWindow`:



Public Member Functions

- def [setUpUi](#) (self, mainWindow)
- def [retranslateUi](#) (self, mainWindow)

Public Attributes

- [centralwidget](#)
- [verticalLayout_7](#)
- [horizontalLayout_12](#)
- [input_file_location](#)
- [filename_lineEdit](#)
- [but_file_open](#)
- [but_begin](#)
- [horizontalLayout_7](#)
- [verticalLayout](#)
- [horizontalLayout](#)
- [label_from](#)
- [dateTimeEdit_from](#)
- [horizontalLayout_2](#)
- [label_submitted](#)
- [dateTimeEdit_submitted](#)
- [horizontalLayout_3](#)
- [label_to](#)
- [dateTimeEdit_to](#)
- [verticalLayout_3](#)
- [horizontalLayout_8](#)
- [input_current_id](#)
- [label_current_id](#)
- [horizontalLayout_9](#)
- [input_attempt](#)
- [label_attempt](#)
- [verticalLayout_2](#)
- [horizontalLayout_6](#)
- [input_max_pos_grade](#)
- [label_max_pos](#)
- [horizontalLayout_4](#)
- [input_subtract](#)
- [label_subtr](#)
- [horizontalLayout_5](#)
- [input_final_grade](#)
- [label_final](#)
- [verticalLayout_4](#)
- [but_regrade](#)
- [checkB_input_pin_status](#)
- [checkB_output_pin_status](#)
- [horizontalLayout_10](#)
- [but_prev](#)
- [checkB_wrong](#)
- [but_reset](#)

- [but_next](#)
- [popular_answers](#)
- [tabs_for_log_and_resp](#)
- [response_tab](#)
- [verticalLayout_9](#)
- [splitter](#)
- [input_response_browser](#)
- [input_response_browser_user](#)
- [tab_prev_resp](#)
- [verticalLayout_5](#)
- [input_prev_response](#)
- [tab_message_to_all](#)
- [verticalLayout_8](#)
- [input_message_to_all](#)
- [log_tab](#)
- [verticalLayout_6](#)
- [input_log_browser](#)
- [horizontalLayout_11](#)
- [but_save_response](#)
- [check_autosave](#)
- [manage_labs_but](#)
- [set_style_checkbox](#)
- [settings_but](#)
- [but_save_all](#)
- [but_create_report](#)
- [progressBar](#)

7.13.1 Detailed Description

Definition at line 11 of file main_window.py.

7.13.2 Member Function Documentation

7.13.2.1 retranslateUi()

```
def main_window.Ui_mainWindow.retranslateUi (  
    self,  
    mainWindow )
```

Definition at line 351 of file main_window.py.

```

351     def retranslateUi(self, mainWindow):
352
353         _translate = QtCore.QCoreApplication.translate
354         mainWindow.setWindowTitle(_translate("mainWindow", "CSCI3130 grader"))
355         self.input_file_location.setPlaceholderText(_translate("mainWindow", "Double click for path
356 selection or paste type path here"))
357
358         self.but_file_open.setText(_translate("mainWindow", "Open"))
359         self.but_begin.setText(_translate("mainWindow", "Begin"))
360         self.label_from.setText(_translate("mainWindow", "From"))
361         self.label_submitted.setText(_translate("mainWindow", "Submitted"))
362         self.label_to.setText(_translate("mainWindow", "To"))
363         self.label_current_id.setText(_translate("mainWindow", "current id"))
364         self.label_attempt.setText(_translate("mainWindow", "attempt"))
365         self.label_max_pos.setText(_translate("mainWindow", "lab max grade"))
366         self.label_subtr.setText(_translate("mainWindow", "subtract"))
367         self.label_final.setText(_translate("mainWindow", "final grade"))
368         self.but_regrade.setText(_translate("mainWindow", "GRADE"))
369         self.checkBox_input_pin_status.setText(_translate("mainWindow", "Input direction"))
370         self.checkBox_output_pin_status.setText(_translate("mainWindow", "Output direction"))
371         self.but_prev.setText(_translate("mainWindow", "prev"))
372         self.checkBox_wrong.setText(_translate("mainWindow", "WRONG"))
373         self.but_reset.setText(_translate("mainWindow", "Reset"))
374         self.but_next.setText(_translate("mainWindow", "next"))
375         self.input_response_browser.setPlaceholderText(_translate("mainWindow", "Auto answer"))
376         self.input_response_browser_user.setPlaceholderText(_translate("mainWindow", "User comment"))
377         self.tabs_for_log_and_resp.setTabText(self.tabs_for_log_and_resp.indexOf(self.response_tab),
378 _translate("mainWindow", "Response"))
379         self.tabs_for_log_and_resp.setTabText(self.tabs_for_log_and_resp.indexOf(self.tab_prev_resp),
380 _translate("mainWindow", "Previous Response"))
381         self.tabs_for_log_and_resp.setTabText(self.tabs_for_log_and_resp.indexOf(self.tab_message_to_all),
382 _translate("mainWindow", "Message to all"))
383         self.tabs_for_log_and_resp.setTabText(self.tabs_for_log_and_resp.indexOf(self.log_tab), _translate(
384 "mainWindow", "Log"))
385         self.but_save_response.setText(_translate("mainWindow", "save response"))
386         self.checkBox_autosave.setText(_translate("mainWindow", "autosave"))
387         self.manage_labs_but.setText(_translate("mainWindow", "Manage labs"))
388         self.setStyleCheckBox.setText(_translate("mainWindow", "style"))
389         self.settings_but.setText(_translate("mainWindow", "Settings"))
390         self.but_save_all.setText(_translate("mainWindow", "save all"))
391         self.but_create_report.setText(_translate("mainWindow", "Create reports"))
392         self.progressBar.setFormat(_translate("mainWindow", "%v/%m (%p)"))

```

7.13.2.2 setupUi()

```

def main_window.Ui_mainWindow.setupUi (
    self,
    mainWindow )

```

Definition at line 12 of file main_window.py.

```

12     def setupUi(self, mainWindow):
13         mainWindow.setObjectName("mainWindow")
14         mainWindow.setEnabled(True)
15         mainWindow.resize(888, 584)
16         icon = QtGui.QIcon()
17         icon.addPixmap(QtGui.QPixmap("os_linux_1.ico"), QtGui.QIcon.Normal, QtGui.QIcon.Off)
18         mainWindow.setWindowIcon(icon)
19         mainWindow.setAccessibleName("")
20         self.centralwidget = QtWidgets.QWidget(mainWindow)
21         self.centralwidget.setObjectName("centralwidget")
22         self.verticalLayout_7 = QtWidgets.QVBoxLayout(self.centralwidget)
23         self.verticalLayout_7.setObjectName("verticalLayout_7")
24         self.horizontalLayout_12 = QtWidgets.QHBoxLayout()
25         self.horizontalLayout_12.setObjectName("horizontalLayout_12")
26         self.input_file_location = BetterLineEdit(self.centralwidget)
27         self.input_file_location.setEnabled(False)
28         self.input_file_location.setLocale(QtCore.QLocale(QtCore.QLocale.English,
QtCore.QLocale.UnitedStates))

```

```

29     self.input_file_location.setText("")
30     self.input_file_location.setObjectName("input_file_location")
31     self.horizontalLayout_12.addWidget(self.input_file_location)
32     self.filename_lineEdit = QtWidgets.QLineEdit(self.centralwidget)
33     self.filename_lineEdit.setMaximumSize(QtCore.QSize(90, 16777215))
34     self.filename_lineEdit.setReadOnly(True)
35     self.filename_lineEdit.setObjectName("filename_lineEdit")
36     self.horizontalLayout_12.addWidget(self.filename_lineEdit)
37     self.but_file_open = QtWidgets.QPushButton(self.centralwidget)
38     self.but_file_open.setEnabled(False)
39     self.but_file_open.setLocale(QtCore.QLocale(QtCore.QLocale.English, QtCore.QLocale.UnitedStates))
40     self.but_file_open.setObjectName("but_file_open")
41     self.horizontalLayout_12.addWidget(self.but_file_open)
42     self.but_begin = QtWidgets.QPushButton(self.centralwidget)
43     self.but_begin.setEnabled(False)
44     self.but_begin.setLocale(QtCore.QLocale(QtCore.QLocale.English, QtCore.QLocale.UnitedStates))
45     self.but_begin.setCheckable(False)
46     self.but_begin.setAutoDefault(False)
47     self.but_begin.setDefault(False)
48     self.but_begin.setFlat(False)
49     self.but_begin.setObjectName("but_begin")
50     self.horizontalLayout_12.addWidget(self.but_begin)
51     self.verticalLayout_7.addLayout(self.horizontalLayout_12)
52     self.horizontalLayout_7 = QtWidgets.QHBoxLayout()
53     self.horizontalLayout_7.setSpacing(6)
54     self.horizontalLayout_7.setObjectName("horizontalLayout_7")
55     self.verticalLayout = QtWidgets.QVBoxLayout()
56     self.verticalLayout.setObjectName("verticalLayout")
57     self.horizontalLayout = QtWidgets.QHBoxLayout()
58     self.horizontalLayout.setObjectName("horizontalLayout")
59     self.label_from = QtWidgets.QLabel(self.centralwidget)
60     self.label_from.setObjectName("label_from")
61     self.horizontalLayout.addWidget(self.label_from)
62     self.dateTimeEdit_from = QtWidgets.QDateTimeEdit(self.centralwidget)
63     self.dateTimeEdit_from.setEnabled(True)
64     self.dateTimeEdit_from.setWrapping(False)
65     self.dateTimeEdit_from.setReadOnly(True)
66     self.dateTimeEdit_from.setAccelerated(False)
67     self.dateTimeEdit_from.setCalendarPopup(True)
68     self.dateTimeEdit_from.setObjectName("dateTimeEdit_from")
69     self.horizontalLayout.addWidget(self.dateTimeEdit_from)
70     self.verticalLayout.addLayout(self.horizontalLayout)
71     self.horizontalLayout_2 = QtWidgets.QHBoxLayout()
72     self.horizontalLayout_2.setObjectName("horizontalLayout_2")
73     self.label_submitted = QtWidgets.QLabel(self.centralwidget)
74     self.label_submitted.setObjectName("label_submitted")
75     self.horizontalLayout_2.addWidget(self.label_submitted)
76     self.dateTimeEdit_submitted = QtWidgets.QDateTimeEdit(self.centralwidget)
77     self.dateTimeEdit_submitted.setEnabled(True)
78     self.dateTimeEdit_submitted.setWrapping(False)
79     self.dateTimeEdit_submitted.setFrame(True)
80     self.dateTimeEdit_submitted.setReadOnly(True)
81     self.dateTimeEdit_submitted.setKeyboardTracking(False)
82     self.dateTimeEdit_submitted.setCalendarPopup(True)
83     self.dateTimeEdit_submitted.setObjectName("dateTimeEdit_submitted")
84     self.horizontalLayout_2.addWidget(self.dateTimeEdit_submitted)
85     self.verticalLayout.addLayout(self.horizontalLayout_2)
86     self.horizontalLayout_3 = QtWidgets.QHBoxLayout()
87     self.horizontalLayout_3.setObjectName("horizontalLayout_3")
88     self.label_to = QtWidgets.QLabel(self.centralwidget)
89     self.label_to.setObjectName("label_to")
90     self.horizontalLayout_3.addWidget(self.label_to)
91     self.dateTimeEdit_to = QtWidgets.QDateTimeEdit(self.centralwidget)
92     self.dateTimeEdit_to.setEnabled(True)
93     self.dateTimeEdit_to.setReadOnly(True)
94     self.dateTimeEdit_to.setCalendarPopup(True)
95     self.dateTimeEdit_to.setObjectName("dateTimeEdit_to")
96     self.horizontalLayout_3.addWidget(self.dateTimeEdit_to)
97     self.verticalLayout.addLayout(self.horizontalLayout_3)
98     self.horizontalLayout_7.addLayout(self.verticalLayout)
99     self.verticalLayout_3 = QtWidgets.QVBoxLayout()
100    self.verticalLayout_3.setObjectName("verticalLayout_3")
101    self.horizontalLayout_8 = QtWidgets.QHBoxLayout()
102    self.horizontalLayout_8.setObjectName("horizontalLayout_8")
103    self.input_current_id = QtWidgets.QLineEdit(self.centralwidget)
104    self.input_current_id.setEnabled(False)
105    self.input_current_id.setMaximumSize(QtCore.QSize(60, 40))
106    self.input_current_id.setReadOnly(True)
107    self.input_current_id.setObjectName("input_current_id")
108    self.horizontalLayout_8.addWidget(self.input_current_id)
109    self.label_current_id = QtWidgets.QLabel(self.centralwidget)

```

```

110         self.label_current_id.setObjectName("label_current_id")
111         self.horizontalLayout_8.addWidget(self.label_current_id)
112         self.verticalLayout_3.addLayout(self.horizontalLayout_8)
113         self.horizontalLayout_9 = QtWidgets.QHBoxLayout()
114         self.horizontalLayout_9.setObjectName("horizontalLayout_9")
115         self.input_attempt = QtWidgets.QLineEdit(self.centralwidget)
116         self.input_attempt.setEnabled(False)
117         self.input_attempt.setMaximumSize(QtCore.QSize(40, 40))
118         self.input_attempt.setReadOnly(True)
119         self.input_attempt.setObjectName("input_attempt")
120         self.horizontalLayout_9.addWidget(self.input_attempt)
121         spacerItem = QtWidgets.QSpacerItem(20, 20, QtWidgets.QSizePolicy.Fixed,
QtWidgets.QSizePolicy.Minimum)
122         self.horizontalLayout_9.addItem(spacerItem)
123         self.label_attempt = QtWidgets.QLabel(self.centralwidget)
124         self.label_attempt.setObjectName("label_attempt")
125         self.horizontalLayout_9.addWidget(self.label_attempt)
126         self.verticalLayout_3.addLayout(self.horizontalLayout_9)
127         spacerItem1 = QtWidgets.QSpacerItem(20, 40, QtWidgets.QSizePolicy.Minimum,
QtWidgets.QSizePolicy.Fixed)
128         self.verticalLayout_3.addItem(spacerItem1)
129         self.horizontalLayout_7.addLayout(self.verticalLayout_3)
130         self.verticalLayout_2 = QtWidgets.QVBoxLayout()
131         self.verticalLayout_2.setObjectName("verticalLayout_2")
132         self.horizontalLayout_6 = QtWidgets.QHBoxLayout()
133         self.horizontalLayout_6.setObjectName("horizontalLayout_6")
134         self.input_max_pos_grade = QtWidgets.QLineEdit(self.centralwidget)
135         self.input_max_pos_grade.setEnabled(False)
136         self.input_max_pos_grade.setMaximumSize(QtCore.QSize(40, 40))
137         self.input_max_pos_grade.setLocale(QtCore.QLocale(QtCore.QLocale.English,
QtCore.QLocale.UnitedStates))
138         self.input_max_pos_grade.setText("")
139         self.input_max_pos_grade.setReadOnly(True)
140         self.input_max_pos_grade.setObjectName("input_max_pos_grade")
141         self.horizontalLayout_6.addWidget(self.input_max_pos_grade)
142         self.label_max_pos = QtWidgets.QLabel(self.centralwidget)
143         self.label_max_pos.setEnabled(True)
144         self.label_max_pos.setLocale(QtCore.QLocale(QtCore.QLocale.English, QtCore.QLocale.UnitedStates))
145         self.label_max_pos.setObjectName("label_max_pos")
146         self.horizontalLayout_6.addWidget(self.label_max_pos)
147         self.verticalLayout_2.addLayout(self.horizontalLayout_6)
148         self.horizontalLayout_4 = QtWidgets.QHBoxLayout()
149         self.horizontalLayout_4.setObjectName("horizontalLayout_4")
150         self.input_subtract = QtWidgets.QLineEdit(self.centralwidget)
151         self.input_subtract.setEnabled(False)
152         self.input_subtract.setMaximumSize(QtCore.QSize(40, 40))
153         self.input_subtract.setLocale(QtCore.QLocale(QtCore.QLocale.English, QtCore.QLocale.UnitedStates))
154         self.input_subtract.setReadOnly(True)
155         self.input_subtract.setObjectName("input_subtract")
156         self.horizontalLayout_4.addWidget(self.input_subtract)
157         self.label_subtr = QtWidgets.QLabel(self.centralwidget)
158         self.label_subtr.setLocale(QtCore.QLocale(QtCore.QLocale.English, QtCore.QLocale.UnitedStates))
159         self.label_subtr.setObjectName("label_subtr")
160         self.horizontalLayout_4.addWidget(self.label_subtr)
161         self.verticalLayout_2.addLayout(self.horizontalLayout_4)
162         self.horizontalLayout_5 = QtWidgets.QHBoxLayout()
163         self.horizontalLayout_5.setObjectName("horizontalLayout_5")
164         self.input_final_grade = QtWidgets.QLineEdit(self.centralwidget)
165         self.input_final_grade.setEnabled(False)
166         self.input_final_grade.setMaximumSize(QtCore.QSize(40, 40))
167         self.input_final_grade.setLocale(QtCore.QLocale(QtCore.QLocale.English, QtCore.QLocale.UnitedStates))
168     ))
169         self.input_final_grade.setText("")
170         self.input_final_grade.setReadOnly(True)
171         self.input_final_grade.setObjectName("input_final_grade")
172         self.horizontalLayout_5.addWidget(self.input_final_grade)
173         self.label_final = QtWidgets.QLabel(self.centralwidget)
174         self.label_final.setEnabled(True)
175         self.label_final.setLocale(QtCore.QLocale(QtCore.QLocale.English, QtCore.QLocale.UnitedStates))
176         self.label_final.setObjectName("label_final")
177         self.horizontalLayout_5.addWidget(self.label_final)
178         self.verticalLayout_2.addLayout(self.horizontalLayout_5)
179         self.horizontalLayout_7.addLayout(self.verticalLayout_2)
180         self.verticalLayout_4 = QtWidgets.QVBoxLayout()
181         self.verticalLayout_4.setObjectName("verticalLayout_4")
182         self.but_regrade = QtWidgets.QPushButton(self.centralwidget)
183         self.but_regrade.setEnabled(False)
184         self.but_regrade.setLocale(QtCore.QLocale(QtCore.QLocale.English, QtCore.QLocale.UnitedStates))
185         self.but_regrade.setObjectName("but_regrade")
186         self.verticalLayout_4.addWidget(self.but_regrade)
187         self.checkBox_input_pin_status = QtWidgets.QCheckBox(self.centralwidget)

```



```

187         self.checkB_input_pin_status.setEnabled(False)
188         self.checkB_input_pin_status.setLocale(QtCore.QLocale(QtCore.QLocale.English,
QtCore.QLocale.UnitedStates))
189         self.checkB_input_pin_status.setObjectName("checkB_input_pin_status")
190         self.verticalLayout_4.addWidget(self.checkB_input_pin_status)
191         self.checkB_output_pin_status = QtWidgets.QCheckBox(self.centralwidget)
192         self.checkB_output_pin_status.setEnabled(False)
193         self.checkB_output_pin_status.setLocale(QtCore.QLocale(QtCore.QLocale.English,
QtCore.QLocale.UnitedStates))
194         self.checkB_output_pin_status.setObjectName("checkB_output_pin_status")
195         self.verticalLayout_4.addWidget(self.checkB_output_pin_status)
196         self.horizontalLayout_7.addLayout(self.verticalLayout_4)
197         self.verticalLayout_7.addLayout(self.horizontalLayout_7)
198         self.horizontalLayout_10 = QtWidgets.QHBoxLayout()
199         self.horizontalLayout_10.setSpacing(65)
200         self.horizontalLayout_10.setObjectName("horizontalLayout_10")
201         self.but_prev = QtWidgets.QPushButton(self.centralwidget)
202         self.but_prev.setEnabled(False)
203         self.but_prev.setMinimumSize(QtCore.QSize(60, 30))
204         self.but_prev.setMaximumSize(QtCore.QSize(200, 16777215))
205         self.but_prev.setLocale(QtCore.QLocale(QtCore.QLocale.English, QtCore.QLocale.UnitedStates))
206         self.but_prev.setObjectName("but_prev")
207         self.horizontalLayout_10.addWidget(self.but_prev)
208         self.checkB_wrong = QtWidgets.QCheckBox(self.centralwidget)
209         self.checkB_wrong.setEnabled(False)
210         self.checkB_wrong.setMinimumSize(QtCore.QSize(80, 20))
211         self.checkB_wrong.setMaximumSize(QtCore.QSize(75, 16777215))
212         self.checkB_wrong.setLocale(QtCore.QLocale(QtCore.QLocale.English, QtCore.QLocale.UnitedStates))
213         self.checkB_wrong.setObjectName("checkB_wrong")
214         self.horizontalLayout_10.addWidget(self.checkB_wrong)
215         self.but_reset = QtWidgets.QPushButton(self.centralwidget)
216         self.but_reset.setEnabled(False)
217         self.but_reset.setMinimumSize(QtCore.QSize(60, 20))
218         self.but_reset.setMaximumSize(QtCore.QSize(90, 16777215))
219         self.but_reset.setObjectName("but_reset")
220         self.horizontalLayout_10.addWidget(self.but_reset)
221         self.but_next = QtWidgets.QPushButton(self.centralwidget)
222         self.but_next.setEnabled(False)
223         self.but_next.setMinimumSize(QtCore.QSize(60, 30))
224         self.but_next.setMaximumSize(QtCore.QSize(200, 16777215))
225         self.but_next.setLocale(QtCore.QLocale(QtCore.QLocale.English, QtCore.QLocale.UnitedStates))
226         self.but_next.setObjectName("but_next")
227         self.horizontalLayout_10.addWidget(self.but_next)
228         self.verticalLayout_7.addLayout(self.horizontalLayout_10)
229         self.popular_answers = QtWidgets.QComboBox(self.centralwidget)
230         self.popular_answers.setEnabled(False)
231         self.popular_answers.setLocale(QtCore.QLocale(QtCore.QLocale.English, QtCore.QLocale.UnitedStates))
232         self.popular_answers.setEditable(False)
233         self.popular_answers.setCurrentText("")
234         self.popular_answers.setObjectName("popular_answers")
235         self.popular_answers.addItem("")
236         self.popular_answers.setItemText(0, "")
237         self.verticalLayout_7.addWidget(self.popular_answers)
238         self.tabs_for_log_and_resp = QtWidgets.QTabWidget(self.centralwidget)
239         self.tabs_for_log_and_resp.setEnabled(True)
240         self.tabs_for_log_and_resp.setMinimumSize(QtCore.QSize(770, 30))
241         self.tabs_for_log_and_resp.setMaximumSize(QtCore.QSize(20000, 3700))
242         self.tabs_for_log_and_resp.setLocale(QtCore.QLocale(QtCore.QLocale.English,
QtCore.QLocale.UnitedStates))
243         self.tabs_for_log_and_resp.setTabShape(QtWidgets.QTabWidget.Rounded)
244         self.tabs_for_log_and_resp.setObjectName("tabs_for_log_and_resp")
245         self.response_tab = QtWidgets.QWidget()
246         self.response_tab.setMinimumSize(QtCore.QSize(0, 180))
247         self.response_tab.setMaximumSize(QtCore.QSize(16777215, 300))
248         self.response_tab.setObjectName("response_tab")
249         self.verticalLayout_9 = QtWidgets.QVBoxLayout(self.response_tab)
250         self.verticalLayout_9.setObjectName("verticalLayout_9")
251         self.splitter = QtWidgets.QSplitter(self.response_tab)
252         self.splitter.setSizePolicy(QtWidgets.QSizePolicy.Expanding, QtWidgets.QSizePolicy.Expanding)
253         self.splitter.setHorizontalStretch(0)
254         self.splitter.setVerticalStretch(0)
255         self.splitter.setHeightForWidth(self.splitter.sizePolicy().hasHeightForWidth())
256         self.splitter.setSizePolicy(self.splitter.sizePolicy())
257         self.splitter.setOrientation(QtCore.Qt.Vertical)
258         self.splitter.setObjectName("splitter")
259         self.input_response_browser = QtWidgets.QPlainTextEdit(self.splitter)
260         self.input_response_browser.setEnabled(True)
261         self.input_response_browser.setMinimumSize(QtCore.QSize(0, 30))
262         self.input_response_browser.setReadOnly(True)
263         self.input_response_browser.setTextInteractionFlags(QtCore.Qt.TextSelectableByKeyboard|

```

```

    QtCore.Qt.TextSelectableByMouse)
264     self.input_response_browser.setObjectName("input_response_browser")
265     self.input_response_browser_user = BetterPlainTextEdit(self.splitter)
266     self.input_response_browser_user.setEnabled(False)
267     self.input_response_browser_user.setMinimumSize(QtCore.QSize(0, 30))
268     self.input_response_browser_user.setObjectName("input_response_browser_user")
269     self.verticalLayout_9.addWidget(self.splitter)
270     self.tabs_for_log_and_resp.addTab(self.response_tab, "")
271     self.tab_prev_resp = QtWidgets.QWidget()
272     self.tab_prev_resp.setObjectName("tab_prev_resp")
273     self.verticalLayout_5 = QtWidgets.QVBoxLayout(self.tab_prev_resp)
274     self.verticalLayout_5.setObjectName("verticalLayout_5")
275     self.input_prev_response = QtWidgets.QPlainTextEdit(self.tab_prev_resp)
276     self.input_prev_response.setEnabled(True)
277     self.input_prev_response.setTextInteractionFlags(QtCore.Qt.TextSelectableByKeyboard|
    QtCore.Qt.TextSelectableByMouse)
278     self.input_prev_response.setObjectName("input_prev_response")
279     self.verticalLayout_5.addWidget(self.input_prev_response)
280     self.tabs_for_log_and_resp.addTab(self.tab_prev_resp, "")
281     self.tab_message_to_all = QtWidgets.QWidget()
282     self.tab_message_to_all.setObjectName("tab_message_to_all")
283     self.verticalLayout_8 = QtWidgets.QVBoxLayout(self.tab_message_to_all)
284     self.verticalLayout_8.setObjectName("verticalLayout_8")
285     self.input_message_to_all = QtWidgets.QPlainTextEdit(self.tab_message_to_all)
286     self.input_message_to_all.setSizePolicy(QtWidgets.QSizePolicy.Expanding, QtWidgets.QSizePolicy.Expanding)
287
288     self.sizePolicy.setHorizontalStretch(0)
289     self.sizePolicy.setVerticalStretch(0)
290     self.sizePolicy.setHeightForWidth(self.input_message_to_all.sizePolicy().hasHeightForWidth())
291     self.input_message_to_all.setSizePolicy(self.sizePolicy)
292     self.input_message_to_all.setObjectName("input_message_to_all")
293     self.verticalLayout_8.addWidget(self.input_message_to_all)
294     self.tabs_for_log_and_resp.addTab(self.tab_message_to_all, "")
295     self.log_tab = QtWidgets.QWidget()
296     self.log_tab.setObjectName("log_tab")
297     self.verticalLayout_6 = QtWidgets.QVBoxLayout(self.log_tab)
298     self.verticalLayout_6.setObjectName("verticalLayout_6")
299     self.input_log_browser = QtWidgets.QTextBrowser(self.log_tab)
300     self.input_log_browser.setLocale(QtCore.QLocale(QtCore.QLocale.English, QtCore.QLocale.UnitedStates))
301
302     self.input_log_browser.setObjectName("input_log_browser")
303     self.verticalLayout_6.addWidget(self.input_log_browser)
304     self.tabs_for_log_and_resp.addTab(self.log_tab, "")
305     self.verticalLayout_7.addWidget(self.tabs_for_log_and_resp)
306     self.horizontalLayout_11 = QtWidgets.QHBoxLayout()
307     self.horizontalLayout_11.setObjectName("horizontalLayout_11")
308     self.but_save_response = QtWidgets.QPushButton(self.centralwidget)
309     self.but_save_response.setEnabled(False)
310     self.but_save_response.setLocale(QtCore.QLocale(QtCore.QLocale.English, QtCore.QLocale.UnitedStates))
311
312     self.but_save_response.setObjectName("but_save_response")
313     self.horizontalLayout_11.addWidget(self.but_save_response)
314     self.check_autosave = QtWidgets.QCheckBox(self.centralwidget)
315     self.check_autosave.setEnabled(False)
316     self.check_autosave.setObjectName("check_autosave")
317     self.horizontalLayout_11.addWidget(self.check_autosave)
318     self.manage_labs_but = QtWidgets.QPushButton(self.centralwidget)
319     self.manage_labs_but.setEnabled(False)
320     self.manage_labs_but.setObjectName("manage_labs_but")
321     self.horizontalLayout_11.addWidget(self.manage_labs_but)
322     self.set_style_checkbox = QtWidgets.QCheckBox(self.centralwidget)
323     self.set_style_checkbox.setObjectName("set_style_checkbox")
324     self.horizontalLayout_11.addWidget(self.set_style_checkbox)
325     self.settings_but = QtWidgets.QToolButton(self.centralwidget)
326     self.settings_but.setEnabled(True)
327     self.settings_but.setObjectName("settings_but")
328     self.horizontalLayout_11.addWidget(self.settings_but)
329     self.but_save_all = QtWidgets.QPushButton(self.centralwidget)
330     self.but_save_all.setEnabled(False)
331     self.but_save_all.setLocale(QtCore.QLocale(QtCore.QLocale.English, QtCore.QLocale.UnitedStates))
332     self.but_save_all.setObjectName("but_save_all")
333     self.horizontalLayout_11.addWidget(self.but_save_all)
334     self.but_create_report = QtWidgets.QPushButton(self.centralwidget)
335     self.but_create_report.setEnabled(False)
336     self.but_create_report.setObjectName("but_create_report")
337     self.horizontalLayout_11.addWidget(self.but_create_report)
338     self.verticalLayout_7.addLayout(self.horizontalLayout_11)
339     self.progressBar = QtWidgets.QProgressBar(self.centralwidget)
340     self.progressBar.setEnabled(True)
341     self.progressBar.setAutoFillBackground(False)
342     self.progressBar.setLocale(QtCore.QLocale(QtCore.QLocale.English, QtCore.QLocale.UnitedStates))

```

```
340         self.progressBar.setProperty("value", 0)
341         self.progressBar.setTextVisible(True)
342         self.progressBar.setInvertedAppearance(False)
343         self.progressBar.setObjectName("progressBar")
344         self.verticalLayout_7.addWidget(self.progressBar)
345         mainWindow.setCentralWidget(self.centralwidget)
346
347         self.retranslateUi(mainWindow)
348         self.tabs_for_log_and_resp.setCurrentIndex(0)
349         QtCore.QMetaObject.connectSlotsByName(mainWindow)
350
```

7.13.3 Member Data Documentation

7.13.3.1 but_begin

main_window.Ui_mainWindow.but_begin

Definition at line 42 of file main_window.py.

7.13.3.2 but_create_report

main_window.Ui_mainWindow.but_create_report

Definition at line 331 of file main_window.py.

7.13.3.3 but_file_open

main_window.Ui_mainWindow.but_file_open

Definition at line 37 of file main_window.py.

7.13.3.4 but_next

main_window.Ui_mainWindow.but_next

Definition at line 221 of file main_window.py.

7.13.3.5 `but_prev`

`main_window.Ui_mainWindow.but_prev`

Definition at line 201 of file `main_window.py`.

7.13.3.6 `but_regrade`

`main_window.Ui_mainWindow.but_regrade`

Definition at line 181 of file `main_window.py`.

7.13.3.7 `but_reset`

`main_window.Ui_mainWindow.but_reset`

Definition at line 215 of file `main_window.py`.

7.13.3.8 `but_save_all`

`main_window.Ui_mainWindow.but_save_all`

Definition at line 326 of file `main_window.py`.

7.13.3.9 `but_save_response`

`main_window.Ui_mainWindow.but_save_response`

Definition at line 306 of file `main_window.py`.

7.13.3.10 `centralwidget`

`main_window.Ui_mainWindow.centralwidget`

Definition at line 20 of file `main_window.py`.

7.13.3.11 check_autosave

`main_window.Ui_mainWindow.check_autosave`

Definition at line 311 of file `main_window.py`.

7.13.3.12 checkB_input_pin_status

`main_window.Ui_mainWindow.checkB_input_pin_status`

Definition at line 186 of file `main_window.py`.

7.13.3.13 checkB_output_pin_status

`main_window.Ui_mainWindow.checkB_output_pin_status`

Definition at line 191 of file `main_window.py`.

7.13.3.14 checkB_wrong

`main_window.Ui_mainWindow.checkB_wrong`

Definition at line 208 of file `main_window.py`.

7.13.3.15 dateTimeEdit_from

`main_window.Ui_mainWindow.dateTimeEdit_from`

Definition at line 62 of file `main_window.py`.

7.13.3.16 dateTimeEdit_submitted

`main_window.Ui_mainWindow.dateTimeEdit_submitted`

Definition at line 76 of file `main_window.py`.

7.13.3.17 dateTimeEdit_to

`main_window.Ui_mainWindow.dateTimeEdit_to`

Definition at line 91 of file `main_window.py`.

7.13.3.18 filename_lineEdit

`main_window.Ui_mainWindow.filename_lineEdit`

Definition at line 32 of file `main_window.py`.

7.13.3.19 horizontalLayout

`main_window.Ui_mainWindow.horizontalLayout`

Definition at line 57 of file `main_window.py`.

7.13.3.20 horizontalLayout_10

`main_window.Ui_mainWindow.horizontalLayout_10`

Definition at line 198 of file `main_window.py`.

7.13.3.21 horizontalLayout_11

`main_window.Ui_mainWindow.horizontalLayout_11`

Definition at line 304 of file `main_window.py`.

7.13.3.22 horizontalLayout_12

`main_window.Ui_mainWindow.horizontalLayout_12`

Definition at line 24 of file `main_window.py`.

7.13.3.23 horizontalLayout_2

`main_window.Ui_mainWindow.horizontalLayout_2`

Definition at line 71 of file `main_window.py`.

7.13.3.24 horizontalLayout_3

`main_window.Ui_mainWindow.horizontalLayout_3`

Definition at line 86 of file `main_window.py`.

7.13.3.25 horizontalLayout_4

`main_window.Ui_mainWindow.horizontalLayout_4`

Definition at line 148 of file `main_window.py`.

7.13.3.26 horizontalLayout_5

`main_window.Ui_mainWindow.horizontalLayout_5`

Definition at line 162 of file `main_window.py`.

7.13.3.27 horizontalLayout_6

`main_window.Ui_mainWindow.horizontalLayout_6`

Definition at line 132 of file `main_window.py`.

7.13.3.28 horizontalLayout_7

`main_window.Ui_mainWindow.horizontalLayout_7`

Definition at line 52 of file `main_window.py`.

7.13.3.29 `horizontalLayout_8`

`main_window.Ui_mainWindow.horizontalLayout_8`

Definition at line 101 of file `main_window.py`.

7.13.3.30 `horizontalLayout_9`

`main_window.Ui_mainWindow.horizontalLayout_9`

Definition at line 113 of file `main_window.py`.

7.13.3.31 `input_attempt`

`main_window.Ui_mainWindow.input_attempt`

Definition at line 115 of file `main_window.py`.

7.13.3.32 `input_current_id`

`main_window.Ui_mainWindow.input_current_id`

Definition at line 103 of file `main_window.py`.

7.13.3.33 `input_file_location`

`main_window.Ui_mainWindow.input_file_location`

Definition at line 26 of file `main_window.py`.

7.13.3.34 `input_final_grade`

`main_window.Ui_mainWindow.input_final_grade`

Definition at line 164 of file `main_window.py`.

7.13.3.35 input_log_browser

`main_window.Ui_mainWindow.input_log_browser`

Definition at line 298 of file `main_window.py`.

7.13.3.36 input_max_pos_grade

`main_window.Ui_mainWindow.input_max_pos_grade`

Definition at line 134 of file `main_window.py`.

7.13.3.37 input_message_to_all

`main_window.Ui_mainWindow.input_message_to_all`

Definition at line 285 of file `main_window.py`.

7.13.3.38 input_prev_response

`main_window.Ui_mainWindow.input_prev_response`

Definition at line 275 of file `main_window.py`.

7.13.3.39 input_response_browser

`main_window.Ui_mainWindow.input_response_browser`

Definition at line 259 of file `main_window.py`.

7.13.3.40 input_response_browser_user

`main_window.Ui_mainWindow.input_response_browser_user`

Definition at line 265 of file `main_window.py`.

7.13.3.41 input_subtract

`main_window.Ui_mainWindow.input_subtract`

Definition at line 150 of file `main_window.py`.

7.13.3.42 label_attempt

`main_window.Ui_mainWindow.label_attempt`

Definition at line 123 of file `main_window.py`.

7.13.3.43 label_current_id

`main_window.Ui_mainWindow.label_current_id`

Definition at line 109 of file `main_window.py`.

7.13.3.44 label_final

`main_window.Ui_mainWindow.label_final`

Definition at line 172 of file `main_window.py`.

7.13.3.45 label_from

`main_window.Ui_mainWindow.label_from`

Definition at line 59 of file `main_window.py`.

7.13.3.46 label_max_pos

`main_window.Ui_mainWindow.label_max_pos`

Definition at line 142 of file `main_window.py`.

7.13.3.47 label_submitted

`main_window.Ui_mainWindow.label_submitted`

Definition at line 73 of file `main_window.py`.

7.13.3.48 label_subtr

`main_window.Ui_mainWindow.label_subtr`

Definition at line 157 of file `main_window.py`.

7.13.3.49 label_to

`main_window.Ui_mainWindow.label_to`

Definition at line 88 of file `main_window.py`.

7.13.3.50 log_tab

`main_window.Ui_mainWindow.log_tab`

Definition at line 294 of file `main_window.py`.

7.13.3.51 manage_labs_but

`main_window.Ui_mainWindow.manage_labs_but`

Definition at line 315 of file `main_window.py`.

7.13.3.52 popular_answers

`main_window.Ui_mainWindow.popular_answers`

Definition at line 229 of file `main_window.py`.

7.13.3.53 progressBar

`main_window.Ui_mainWindow.progressBar`

Definition at line 336 of file `main_window.py`.

7.13.3.54 response_tab

`main_window.Ui_mainWindow.response_tab`

Definition at line 245 of file `main_window.py`.

7.13.3.55 set_style_checkbox

`main_window.Ui_mainWindow.set_style_checkbox`

Definition at line 319 of file `main_window.py`.

7.13.3.56 settings_but

`main_window.Ui_mainWindow.settings_but`

Definition at line 322 of file `main_window.py`.

7.13.3.57 splitter

`main_window.Ui_mainWindow.splitter`

Definition at line 251 of file `main_window.py`.

7.13.3.58 tab_message_to_all

`main_window.Ui_mainWindow.tab_message_to_all`

Definition at line 281 of file `main_window.py`.

7.13.3.59 tab_prev_resp

`main_window.Ui_mainWindow.tab_prev_resp`

Definition at line 271 of file `main_window.py`.

7.13.3.60 tabs_for_log_and_resp

`main_window.Ui_mainWindow.tabs_for_log_and_resp`

Definition at line 238 of file `main_window.py`.

7.13.3.61 verticalLayout

`main_window.Ui_mainWindow.verticalLayout`

Definition at line 55 of file `main_window.py`.

7.13.3.62 verticalLayout_2

`main_window.Ui_mainWindow.verticalLayout_2`

Definition at line 130 of file `main_window.py`.

7.13.3.63 verticalLayout_3

`main_window.Ui_mainWindow.verticalLayout_3`

Definition at line 99 of file `main_window.py`.

7.13.3.64 verticalLayout_4

`main_window.Ui_mainWindow.verticalLayout_4`

Definition at line 179 of file `main_window.py`.

7.13.3.65 verticalLayout_5

```
main_window.Ui_mainWindow.verticalLayout_5
```

Definition at line 273 of file main_window.py.

7.13.3.66 verticalLayout_6

```
main_window.Ui_mainWindow.verticalLayout_6
```

Definition at line 296 of file main_window.py.

7.13.3.67 verticalLayout_7

```
main_window.Ui_mainWindow.verticalLayout_7
```

Definition at line 22 of file main_window.py.

7.13.3.68 verticalLayout_8

```
main_window.Ui_mainWindow.verticalLayout_8
```

Definition at line 283 of file main_window.py.

7.13.3.69 verticalLayout_9

```
main_window.Ui_mainWindow.verticalLayout_9
```

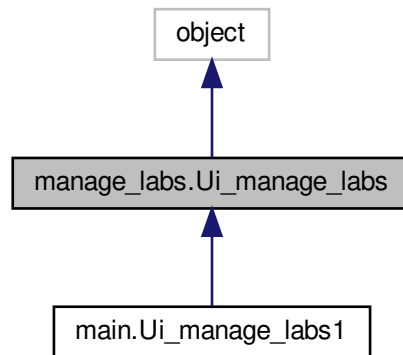
Definition at line 249 of file main_window.py.

The documentation for this class was generated from the following file:

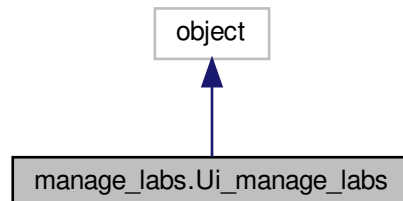
- [main_window.py](#)

7.14 manage_labs.Ui_manage_labs Class Reference

Inheritance diagram for manage_labs.Ui_manage_labs:



Collaboration diagram for manage_labs.Ui_manage_labs:



Public Member Functions

- def [setupUi](#) (self, manage_labs)
- def [retranslateUi](#) (self, manage_labs)

Public Attributes

- [verticalLayout](#)
- [horizontalLayout](#)

- [labs_select_comboBox](#)
- [sync_but](#)
- [import_but](#)
- [create_due_dates_but](#)
- [export_but](#)
- [status_bar](#)

7.14.1 Detailed Description

Definition at line 11 of file `manage_labs.py`.

7.14.2 Member Function Documentation

7.14.2.1 `retranslateUi()`

```
def manage_labs.Ui_manage_labs.retranslateUi (  
    self,  
    manage_labs )
```

Definition at line 47 of file `manage_labs.py`.

```
47     def retranslateUi(self, manage_labs):  
48  
51         _translate = QtCore.QCoreApplication.translate  
52         manage_labs.setWindowTitle(_translate("manage_labs", "Manage labs"))  
53         self.sync_but.setText(_translate("manage_labs", "Sync to local storage"))  
54         self.import_but.setText(_translate("manage_labs", "import labs"))  
55         self.create_due_dates_but.setText(_translate("manage_labs", "Create due dates"))  
56         self.export_but.setText(_translate("manage_labs", "Export pdfs"))  
57  
58
```

7.14.2.2 `setupUi()`

```
def manage_labs.Ui_manage_labs.setupUi (  
    self,  
    manage_labs )
```

Definition at line 12 of file `manage_labs.py`.


```
12     def setupUi(self, manage_labs):
13         manage_labs.setObjectName("manage_labs")
14         manage_labs.resize(753, 90)
15         manage_labs.setWindowFilePath("")
16         self.verticalLayout = QtWidgets.QVBoxLayout(manage_labs)
17         self.verticalLayout.setObjectName("verticalLayout")
18         self.horizontalLayout = QtWidgets.QHBoxLayout()
19         self.horizontalLayout.setObjectName("horizontalLayout")
20         self.labs_select_comboBox = QtWidgets.QComboBox(manage_labs)
21         self.labs_select_comboBox.setEnabled(False)
22         self.labs_select_comboBox.setObjectName("labs_select_comboBox")
23         self.horizontalLayout.addWidget(self.labs_select_comboBox)
24         self.sync_but = QtWidgets.QPushButton(manage_labs)
25         self.sync_but.setObjectName("sync_but")
26         self.horizontalLayout.addWidget(self.sync_but)
27         self.import_but = QtWidgets.QPushButton(manage_labs)
28         self.import_but.setEnabled(False)
29         self.import_but.setObjectName("import_but")
30         self.horizontalLayout.addWidget(self.import_but)
31         self.create_due_dates_but = QtWidgets.QPushButton(manage_labs)
32         self.create_due_dates_but.setEnabled(False)
33         self.create_due_dates_but.setObjectName("create_due_dates_but")
34         self.horizontalLayout.addWidget(self.create_due_dates_but)
35         self.export_but = QtWidgets.QPushButton(manage_labs)
36         self.export_but.setEnabled(False)
37         self.export_but.setObjectName("export_but")
38         self.horizontalLayout.addWidget(self.export_but)
39         self.verticalLayout.addLayout(self.horizontalLayout)
40         self.status_bar = QtWidgets.QLineEdit(manage_labs)
41         self.status_bar.setObjectName("status_bar")
42         self.verticalLayout.addWidget(self.status_bar)
43
44         self.retranslateUi(manage_labs)
45         QtCore.QMetaObject.connectSlotsByName(manage_labs)
46
```

7.14.3 Member Data Documentation

7.14.3.1 create_due_dates_but

`manage_labs.Ui_manage_labs.create_due_dates_but`

Definition at line 31 of file `manage_labs.py`.

7.14.3.2 export_but

`manage_labs.Ui_manage_labs.export_but`

Definition at line 35 of file `manage_labs.py`.

7.14.3.3 `horizontalLayout`

`manage_labs.Ui_manage_labs.horizontalLayout`

Definition at line 18 of file `manage_labs.py`.

7.14.3.4 `import_but`

`manage_labs.Ui_manage_labs.import_but`

Definition at line 27 of file `manage_labs.py`.

7.14.3.5 `labs_select_comboBox`

`manage_labs.Ui_manage_labs.labs_select_comboBox`

Definition at line 20 of file `manage_labs.py`.

7.14.3.6 `status_bar`

`manage_labs.Ui_manage_labs.status_bar`

Definition at line 40 of file `manage_labs.py`.

7.14.3.7 `sync_but`

`manage_labs.Ui_manage_labs.sync_but`

Definition at line 24 of file `manage_labs.py`.

7.14.3.8 `verticalLayout`

`manage_labs.Ui_manage_labs.verticalLayout`

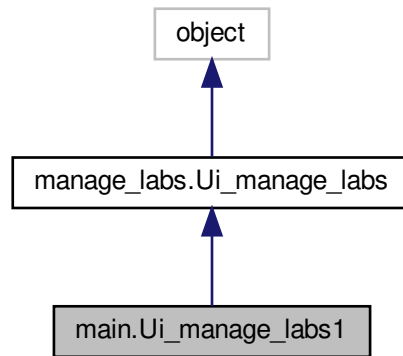
Definition at line 16 of file `manage_labs.py`.

The documentation for this class was generated from the following file:

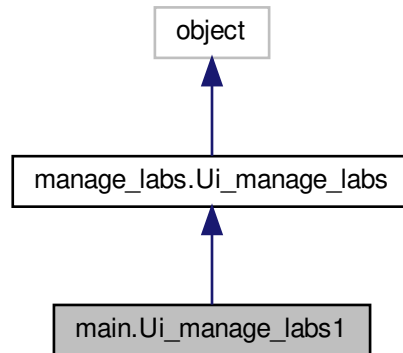
- [manage_labs.py](#)

7.15 main.Ui_manage_labs1 Class Reference

Inheritance diagram for main.Ui_manage_labs1:



Collaboration diagram for main.Ui_manage_labs1:



Public Member Functions

- def [bind_functions](#) (self)
- def [setupUi](#) (self, manage_labs)
- def [set_local_vars](#) (self)

- [def update_status_bar](#) (self, force=False)
- [def sync_files](#) (self)
- [def scan_for_labs](#) (self)
- [def import_lab](#) (self)
- [def check_for_due_dates](#) (self, dir)
- [def open_dates_dialog](#) (self)
- [def due_date_creator](#) (self, due_location, due_dates)
- [def export_pdfs](#) (self)

Public Attributes

- [pdf_files_len](#)
- [main_lab_path](#)
- [cal_window](#)

Static Public Attributes

- [srv_sync_path](#) = None
- [selected_path](#) = None
- [selected_lab_name](#) = None
- [zip_files_len](#) = None

7.15.1 Detailed Description

Definition at line 1574 of file main.py.

7.15.2 Member Function Documentation

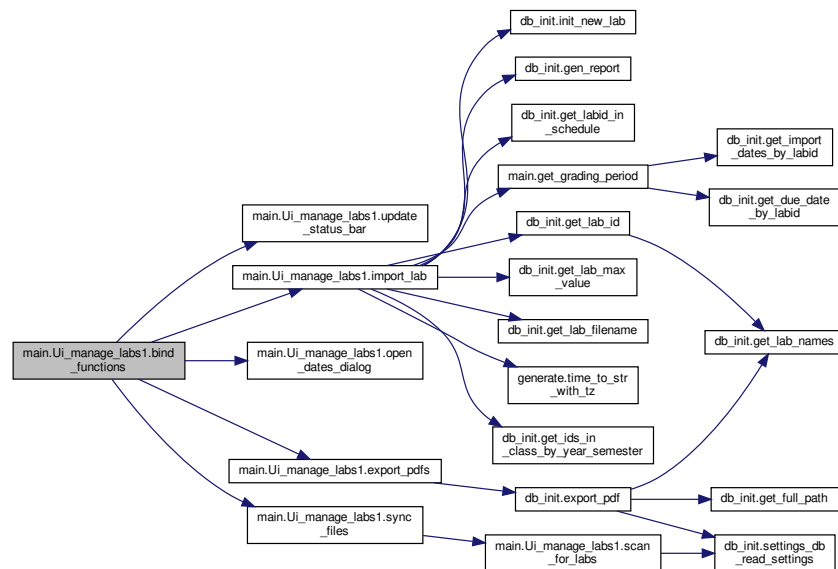
7.15.2.1 bind_functions()

```
def main.Ui_manage_labs1.bind_functions (
    self )
```

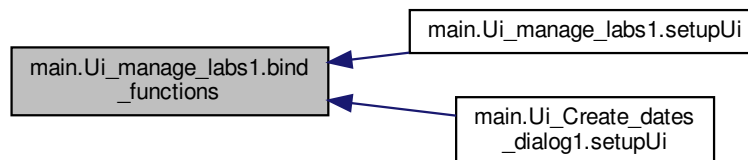
Definition at line 1580 of file main.py.

```
1580     def bind_functions(self):
1581         self.labs_select_comboBox.currentIndexChanged.connect(self.update_status_bar)
1582         self.import_but.clicked.connect(self.import_lab)
1583         self.create_due_dates_but.clicked.connect(self.open_dates_dialog)
1584         # self.sync_but.clicked.connect(lambda i: self.sync_but.setDisabled(True))
1585         self.sync_but.clicked.connect(self.sync_files)
1586         self.export_but.clicked.connect(self.export_pdfs)
1587
```

Here is the call graph for this function:



Here is the caller graph for this function:



7.15.2.2 check_for_due_dates()

```
def main.Ui_manage_labs1.check_for_due_dates (
    self,
    dir )
```

Definition at line 1818 of file main.py.

```
1818     def check_for_due_dates(self, dir):
1819         return sorted([f for f in os.listdir(dir) if 'due_' in f])
1820
1821
```

7.15.2.3 due_date_creator()

```
def main.Ui_manage_labs1.due_date_creator (
    self,
    due_location,
    due_dates )
```

Definition at line 1856 of file main.py.

```
1856     def due_date_creator(self, due_location, due_dates):
1857         if len(due_location) > 1:
1858             i = 1
1859             for due_date in due_dates:
1860                 with open('%sdue_%d_%d' % (due_location, i, due_date), 'w'):
1861                     i += 1
1862         else:
1863             print('Location was not specified.')
1864
```

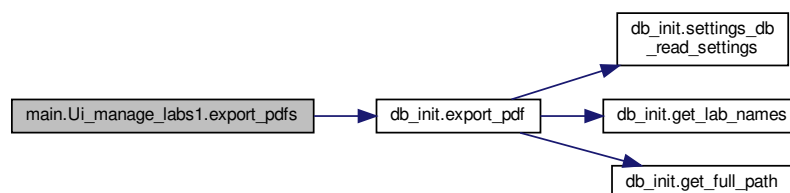
7.15.2.4 export_pdfs()

```
def main.Ui_manage_labs1.export_pdfs (
    self )
```

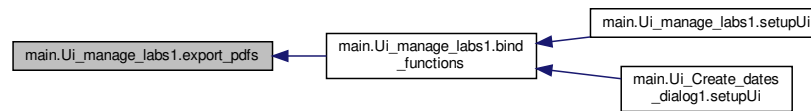
Definition at line 1865 of file main.py.

```
1865     def export_pdfs(self):
1866         self.export_but.setDisabled(True)
1867         self.export_but.setText('Exporting..')
1868         self.export_but.repaint()
1869         export_pdf()
1870         self.export_but.setText('Export pdfs')
1871         self.export_but.setEnabled(True)
1872
1873
```

Here is the call graph for this function:



Here is the caller graph for this function:



7.15.2.5 import_lab()

```
def main.Ui_manage_labs1.import_lab (
    self )
```

Definition at line 1664 of file main.py.

```

1664     def import_lab(self):
1665         if self.selected_path:
1666             self.import_but.setDisabled(True)
1667             self.import_but.setText('Importing..')
1668             self.import_but.repaint()
1669
1670             # due_file = self.check_for_due_dates(self.selected_path)
1671             if False:
1672                 # if len(due_file) < 4:
1673                     self.status_bar.setText('Create due dates !')
1674                     self.import_but.setText('Import labs')
1675                     self.import_but.setEnabled(True)
1676                     return False
1677             else:
1678                 from shutil import copy2 as cp2
1679                 zip_files = [f for f in os.listdir(self.selected_path) if 'zip' in f]
1680                 real_zip_files_rev = sorted([f for f in zip_files if os.path.isfile(os.path.join(self.
1681 selected_path, f))], reverse=True)
1682
1683                 year, semester = self.main_lab_path.split('/')[-1].split('_')
1684                 ltype, _, lab_num = self.selected_lab_name.split('_')
1685                 lid = get_labid_in_schedule(get_lab_id(ltype, int(lab_num)),
1686 year, semester)
1687                 if lid is None:
1688                     self.status_bar.setText('Create due dates ! Lab is not initialised in lab_schedule')
1689                     self.import_but.setText('Import labs')
1690                     self.import_but.setEnabled(True)
1691                     return False
1692                 current_check, prev_due, next_due, current_timestamp =
1693 get_grading_period(lid)
1694
1695                 if current_check > 4:
1696                     self.status_bar.setText('This lab has no more resubmissions (graded 4 times).')
1697                     self.import_but.setText('Import labs')
1698                     self.import_but.setEnabled(True)
1699                     return False
1700                 if current_timestamp < next_due:
1701                     # we cannot grade before the due date
1702                     self.status_bar.setText('Current date is less than next due date. It is too early to
1703 import.')
1704                     self.import_but.setText('Import labs')
1705                     self.import_but.setEnabled(True)
1706                     return False

```

```

1706
1707     penalty_mess = ''
1708     if current_check == 1:
1709         penalty_mess = '100% - this is your max point(no resubmissions)'
1710     elif current_check == 2:
1711         penalty_mess = '90% - first resubmission'
1712     elif current_check == 3:
1713         penalty_mess = '70% - second resubmission'
1714     elif current_check == 4:
1715         penalty_mess = '50% - third resubmission'
1716
1717     lab_type, _, lab_num = self.selected_lab_name.split('_')
1718     lab_corr_name = lab_type[0] + 'LA' + lab_num
1719     max_points = get_lab_max_value(lab_corr_name)
1720     lab_filename = get_lab_filename(lab_corr_name)
1721
1722     # temporary solution. path should be stored as local var
1723     paths_to_grading_dir = self.main_lab_path + '/' + self.selected_lab_name + '_' + str(
current_check) + '/'
1724
1725     # proc_time = datetime.datetime.fromtimestamp(current_timestamp).strftime('%Y-%m-%d %H:%M:%S')
1726     proc_time = time_to_str_with_tz(current_timestamp)
1727
1728     # File manipulations goes below:
1729
1730     if not os.path.isdir(paths_to_grading_dir):
1731         os.makedirs(paths_to_grading_dir)
1732
1733     cur_year, cur_sem = paths_to_grading_dir.split('/')[3].split('_')
1734     id_to_classId = get_ids_in_class_by_year_semester(cur_year
, cur_sem)[0]
1735
1736     imported_files_counter = 0
1737
1738     selected_files = []
1739     for file in real_zip_files_rev:
1740         parts = file.split('.')[0].split('-')
1741         if int(parts[2]) > prev_due and int(parts[2]) <= next_due:
1742             if len(selected_files) == 0:
1743                 selected_files.append(file)
1744             elif selected_files[-1].split('.')[0].split('-')[0] != parts[0]:
1745                 selected_files.append(file)
1746
1747     for file in reversed(selected_files):
1748         zipped_file = zipfile.ZipFile(self.selected_path + file)
1749         extraction_dir = paths_to_grading_dir + file.split('.')[0]
1750         try:
1751             zipped_file.extractall(paths_to_grading_dir + file.split('.')[0])
1752         except Exception as e:
1753             print(self.selected_path + file)
1754             print(e)
1755         finally:
1756             zipped_file.close()
1757         parts = file.split('.')[0].split('-')
1758         subm_int = int(extraction_dir.split('-')[-1])
1759         # subm_time =
datetime.datetime.fromtimestamp(subm_int).replace(tzinfo=tz.tzutc()).astimezone(tz.tzlocal()).strftime('%Y-%m-%d %H:%M:%S')
1759         subm_time = time_to_str_with_tz(subm_int)
1760         # check for required files
1761         if not lab_filename[0] or os.path.isfile(extraction_dir + '/' + lab_filename[0]):
1762             lab_response = 'I did not find any errors. Good job !'
1763             cur_grade = max_points
1764         else:
1765             lab_response = 'File "' + lab_filename[0] + '" was not found.\nThese files were
found: ' + \
1766                 " ".join(os.listdir(extraction_dir))
1767             cur_grade = 0
1768
1769     # This check is for a case when you graded the lab and trying to import it again.
1770     # No existing files should be wiped
1771     if not os.path.isfile(extraction_dir+'penalty.txt'):
1772         with open(extraction_dir+'penalty.txt', 'w') as f:
1773             f.write(penalty_mess)
1774
1775     if not os.path.isfile(extraction_dir + '/grade.txt'):
1776         with open(extraction_dir + '/grade.txt', 'w') as f:
1777             f.write(str(cur_grade))
1778
1779     if not os.path.isfile(extraction_dir + '/responce.txt'):
1780         with open(extraction_dir + '/responce.txt', 'w') as f:
1781             f.write(lab_response)
1782

```

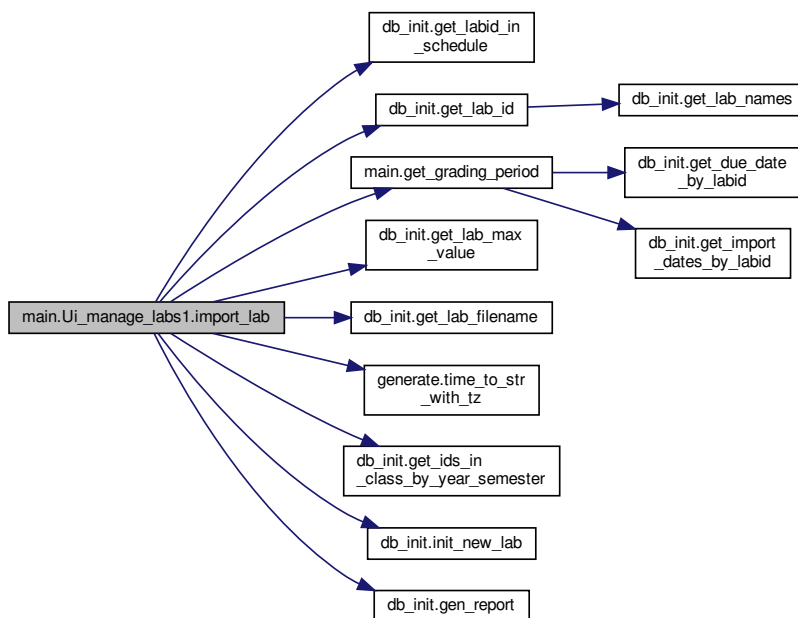


```

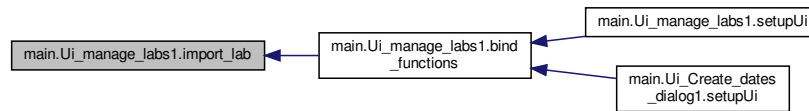
1783         if not os.path.isfile(extraction_dir + '/tech_info.txt'):
1784             with open(extraction_dir + '/tech_info.txt', 'w') as f:
1785                 f.writelines(['File was submitted at %s<br/>\n' % subm_time,
1786                             'I started processing your file at %s<br/>\n' % proc_time,
1787                             "I found that your lab type is '%s' and it's number is %s <br/>"
1788                             % (lab_type, lab_num),
1789                             'So max points for this lab type is <u>%d</u><br/>' % max_points,
1790                             'Theoretical max points: %s)' % penalty_mess])
1791         init_new_lab(id_to_classId[parts[0]], lid, current_check, subm_int,
1792 extraction_dir)
1793         imported_files_counter += 1
1794         # cp2(self.selected_path + due_file[current_check-1], paths_to_grading_dir)
1795         # check_filename = paths_to_grading_dir + 'check_' + str(current_check) + '_' +
1796         str(current_timestamp)
1797         # with open(check_filename, 'w'): pass
1798         gen_report(lid, att=current_check)
1799         # cp2(check_filename, self.selected_path)
1800         self.import_but.setEnabled(True)
1801         self.import_but.setText('Import labs')
1802         self.status_bar.setText("Imported " + str(imported_files_counter) + " files.")
1803         return True
1804     return False
1805
1806
1807
1808
1809
1810

```

Here is the call graph for this function:



Here is the caller graph for this function:



7.15.2.6 open_dates_dialog()

```
def main.Ui_manage_labs1.open_dates_dialog (
    self )
```

Definition at line 1827 of file main.py.

```

1827     def open_dates_dialog(self):
1828         self.create_due_dates_but.setDisabled(True)
1829         self.create_due_dates_but.repaint()
1830         self.cal_window = QtWidgets.QDialog()
1831         dui = Ui_Create_dates_dialog1()
1832         dui.setupUi(self.cal_window, self.selected_lab_name)
1833         # self.cal_window.finished.connect(self.check_new_win_result)
1834         self.cal_window.show()
1835         accepted = self.cal_window.exec_()
1836         if accepted:
1837             due_dates = list()
1838             due_dates.append(dui.init_subm_date_time.dateTime().toTime_t())
1839             due_dates.append(dui.first_subm_date_time.dateTime().toTime_t())
1840             due_dates.append(dui.second_subm_date_time.dateTime().toTime_t())
1841             due_dates.append(dui.third_subm_date_time.dateTime().toTime_t())
1842             due_location = dui.lab_path.text()
1843             self.due_date_creator(due_location, due_dates)
1844             year, semester = self.main_lab_path.split('/')[-1].split('_')
1845             ltype, _, lab_num = self.selected_lab_name.split('_')
1846             register_lab_in_semester(ltype, lab_num, year, semester, due_dates)
1847         self.create_due_dates_but.setEnabled(True)
1848

```

Here is the caller graph for this function:



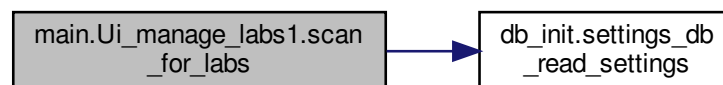
7.15.2.7 scan_for_labs()

```
def main.Ui_manage_labs1.scan_for_labs (
    self )
```

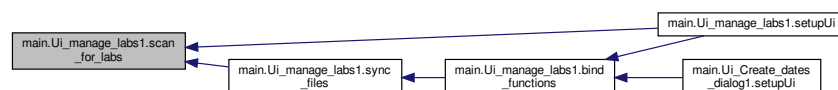
Definition at line 1651 of file main.py.

```
1651     def scan_for_labs(self):
1652         paths, local = settings_db_read_settings()
1653         # self.local_path = paths[1] + str(local[1]) + '_' + str(local[2]) + '/'
1654         self.main_lab_path = get_full_path(paths, local)
1655         self.srv_sync_path = self.main_lab_path + "/server_sync/"
1656         dirs = os.walk(self.srv_sync_path).__next__()[1]
1657         if len(dirs) > 0:
1658             self.labs_select_comboBox.addItem(sorted(dirs))
1659             self.labs_select_comboBox.setCurrentIndex(0)
1660             self.labs_select_comboBox.setFocus(True)
1661             self.update_status_bar(force=True)
1662
1663
```

Here is the call graph for this function:



Here is the caller graph for this function:



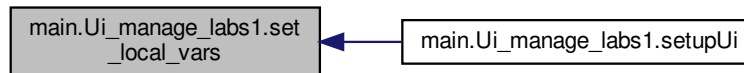
7.15.2.8 set_local_vars()

```
def main.Ui_manage_labs1.set_local_vars (
    self )
```

Definition at line 1604 of file main.py.

```
1604     def set_local_vars(self):
1605         pass
1606
```

Here is the caller graph for this function:



7.15.2.9 setupUi()

```
def main.Ui_manage_labs1.setupUi (
    self,
    manage_labs )
```

Definition at line 1588 of file main.py.

```
1588     def setupUi(self, manage_labs):
1589         super().setupUi(manage_labs)
1590         self.bind_functions()
1591         self.set_local_vars()
1592
1593         try:
1594             self.scan_for_labs()
1595             if self.labs_select_comboBox.count() > 0:
1596                 self.labs_select_comboBox.setEnabled(True)
1597                 self.import_but.setEnabled(True)
1598                 self.create_due_dates_but.setEnabled(True)
1599                 self.export_but.setEnabled(True)
1600         except Exception as e:
1601             print('Error in manage labs. Probably your grading path was not set properly: ', e)
1602
1603
```

[illegible]

```
def main.Ui_manage_labs1.sync_files (
    self )
```

```

1626     def sync_files(self):
1627         self.sync_but.setDisabled(True)
1628         self.sync_but.setText('Synchronizing...')
1629         self.sync_but.repaint()
1630         self.status_bar.setText("Synchronizing...")
1631         self.status_bar.repaint()

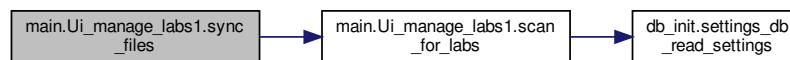
```

```

1632     sync_files()
1633     self.status_bar.setText("Done.")
1634     self.sync_but.setText('Sync to local storage')
1635     self.sync_but.setEnabled(True)
1636
1637     sync_success = True # there are no tools to check it at this point.
1638     if sync_success and not self.labs_select_comboBox.isEnabled():
1639         self.labs_select_comboBox.setEnabled(True)
1640         self.create_due_dates_but.setEnabled(True)
1641         self.scan_for_labs()
1642         # TODO: There should be additional checks to enable import and export, but I do not have
1643         # enough time to implement them.
1644         self.import_but.setEnabled(True)
1645         self.export_but.setEnabled(True)

```

Here is the call graph for this function:



Here is the caller graph for this function:



7.15.2.11 update_status_bar()

```

def main.Ui_manage_labs1.update_status_bar (
    self,
    force = False )

```

Definition at line 1607 of file main.py.

```

1607     def update_status_bar(self, force=False):
1608         # no need to scan files in background, but only when user selects it intentionally, or if it is
1609         # first run
1610         if self.labs_select_comboBox.hasFocus() or force:
1611             self.selected_lab_name = self.labs_select_comboBox.currentText()
1612             self.selected_path = self.srv_sync_path + self.selected_lab_name + '/'
1613             zip_pdf_files = [f for f in os.listdir(self.selected_path) if '.zip' in f or '.pdf' in f]
1614             self.pdf_files_len = len([f for f in zip_pdf_files if f.split('.')[-1] == 'pdf'])
1615             self.zip_files_len = len([f for f in zip_pdf_files if f.split('.')[-1] == 'zip'])

```

```

1616
1617         self.status_bar.setText("Contains " + str(self.zip_files_len) + ' zip files and ' + str(self.
pdf_files_len) + ' pdf files.')
1618
1619         if self.zip_files_len > 0 and not self.create_due_dates_but.isEnabled():
1620             self.export_but.setEnabled(True)
1621             self.import_but.setEnabled(True)
1622             self.labs_select_comboBox.setEnabled(True)
1623
1624         # good_zip_files_size = len([f for f in zip_files if os.isfile(os.path.join(selected_path,
f))])
1625

```

Here is the caller graph for this function:



7.15.3 Member Data Documentation

7.15.3.1 cal_window

`main.Ui_manage_labs1.cal_window`

Definition at line 1830 of file `main.py`.

7.15.3.2 main_lab_path

`main.Ui_manage_labs1.main_lab_path`

Definition at line 1654 of file `main.py`.

7.15.3.3 pdf_files_len

`main.Ui_manage_labs1.pdf_files_len`

Definition at line 1614 of file `main.py`.

7.15.3.4 selected_lab_name

```
main.Ui_manage_labs1.selected_lab_name = None [static]
```

Definition at line 1577 of file main.py.

7.15.3.5 selected_path

```
main.Ui_manage_labs1.selected_path = None [static]
```

Definition at line 1576 of file main.py.

7.15.3.6 srv_sync_path

```
main.Ui_manage_labs1.srv_sync_path = None [static]
```

Definition at line 1575 of file main.py.

7.15.3.7 zip_files_len

```
main.Ui_manage_labs1.zip_files_len = None [static]
```

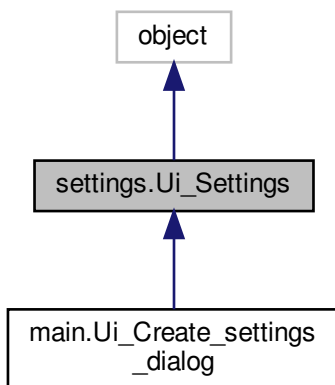
Definition at line 1578 of file main.py.

The documentation for this class was generated from the following file:

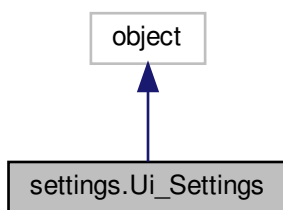
- [main.py](#)

7.16 settings.Ui_Settings Class Reference

Inheritance diagram for settings.Ui_Settings:



Collaboration diagram for settings.Ui_Settings:



Public Member Functions

- def [setupUi](#) (self, Settings)
- def [retranslateUi](#) (self, Settings)

Public Attributes

- [verticalLayout](#)
- [groupBox_db](#)
- [formLayout](#)
- [label_settings_db](#)
- [input_settings_db](#)
- [label_grades_db](#)
- [input_grades_db](#)
- [groupBox_user](#)
- [formLayout_2](#)
- [label_logisim_path](#)
- [input_logisim_path](#)
- [label_local_stor](#)
- [input_local_stor](#)
- [label_rem_stor](#)
- [input_rem_stor](#)
- [groupBox_local](#)
- [gridLayout](#)
- [spin_year](#)
- [label_grad_year](#)
- [input_grader_name](#)
- [label_semester](#)
- [label_style](#)
- [label_sync_comm](#)
- [label_grader_name](#)
- [style_checkBox](#)
- [semester_comboBox](#)
- [sync_command](#)
- [import_students_btn](#)
- [buttonBox](#)

7.16.1 Detailed Description

Definition at line 11 of file settings.py.

7.16.2 Member Function Documentation

7.16.2.1 retranslateUi()

```
def settings.Ui_Settings.retranslateUi (
    self,
    Settings )
```

Definition at line 227 of file settings.py.

```
227     def retranslateUi(self, Settings):
228
229         _translate = QtCore.QCoreApplication.translate
230         Settings.setWindowTitle(_translate("Settings", "Settings"))
231         self.groupBox_db.setTitle(_translate("Settings", "&Database paths:"))
232         self.label_settings_db.setText(_translate("Settings", "Settings"))
233         self.input_settings_db.setText(_translate("Settings", "./settings.sqlite3"))
234         self.label_grades_db.setText(_translate("Settings", "Grades"))
235         self.input_grades_db.setText(_translate("Settings", "
~/Documents/3130_labs/grades.sqlite3"))
236         self.groupBox_user.setTitle(_translate("Settings", "User paths"))
237         self.label_logisim_path.setText(_translate("Settings", "Logisim path"))
238         self.input_logisim_path.setPlaceholderText(_translate("Settings", "path to logisim executable
logisim.jar"))
239         self.label_local_stor.setText(_translate("Settings", "Local lab storage"))
240         self.input_local_stor.setPlaceholderText(_translate("Settings", "local directory that contains
labs, reports, and other working files"))
241         self.label_rem_stor.setText(_translate("Settings", "Remote lab storage"))
242         self.input_rem_stor.setPlaceholderText(_translate("Settings", "sshfs mounted dir that points to
submission directory on the remote server"))
243         self.groupBox_local.setTitle(_translate("Settings", "&Local settings"))
244         self.label_grad_year.setText(_translate("Settings", "Grading year"))
245         self.label_semester.setText(_translate("Settings", "Grading semester"))
246         self.label_style.setText(_translate("Settings", "Use styles"))
247         self.label_sync_comm.setText(_translate("Settings", "Sync command"))
248         self.label_grader_name.setText(_translate("Settings", "Grader name"))
249         self.semester_comboBox.setItemText(0, _translate("Settings", "Spring"))
250         self.semester_comboBox.setItemText(1, _translate("Settings", "Summer"))
251         self.semester_comboBox.setItemText(2, _translate("Settings", "Fall"))
252         self.sync_command.setPlaceholderText(_translate("Settings", "rsync -avz ? cp -v ? dd ... ?"))
253         self.import_students_btn.setText(_translate("Settings", "Import students"))
254
255
256
257
```

7.16.2.2 setupUi()

```
def settings.Ui_Settings.setupUi (
    self,
    Settings )
```

Definition at line 12 of file settings.py.

```
12     def setupUi(self, Settings):
13         Settings.setObjectName("Settings")
14         Settings.setEnabled(True)
15         Settings.resize(800, 487)
16         Settings.setMinimumSize(QtCore.QSize(600, 0))
17         icon = QtGui.QIcon()
18         icon.addPixmap(QtGui.QPixmap("os_linux_1.ico"), QtGui.QIcon.Normal, QtGui.QIcon.Off)
19         Settings.setWindowIcon(icon)
20         Settings.setLocale(QtCore.QLocale(QtCore.QLocale.English, QtCore.QLocale.UnitedStates))
21         self.verticalLayout = QtWidgets.QVBoxLayout(Settings)
22         self.verticalLayout.setObjectName("verticalLayout")
23         self.groupBox_db = QtWidgets.QGroupBox(Settings)
24         sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Preferred, QtWidgets.QSizePolicy.Preferred
```

```

    )
25     sizePolicy.setHorizontalStretch(0)
26     sizePolicy.setVerticalStretch(0)
27     sizePolicy.setHeightForWidth(self.groupBox_db.sizePolicy().hasHeightForWidth())
28     self.groupBox_db.setSizePolicy(sizePolicy)
29     self.groupBox_db.setMinimumSize(QSize(0, 0))
30     self.groupBox_db.setAutoFillBackground(False)
31     self.groupBox_db.setAlignment(Qt.AlignLeading|Qt.AlignLeft|Qt.AlignTop)
32     self.groupBox_db.setFlat(False)
33     self.groupBox_db.setCheckable(False)
34     self.groupBox_db.setObjectName("groupBox_db")
35     self.formLayout = QtWidgets.QFormLayout(self.groupBox_db)
36     self.formLayout.setObjectName("formLayout")
37     self.label_settings_db = QtWidgets.QLabel(self.groupBox_db)
38     self.label_settings_db.setMinimumSize(QSize(110, 0))
39     self.label_settings_db.setObjectName("label_settings_db")
40     self.formLayout.addWidget(0, QtWidgets.QFormLayout.LabelRole, self.label_settings_db)
41     self.input_settings_db = QtWidgets.QLineEdit(self.groupBox_db)
42     self.input_settings_db.setEnabled(False)
43     self.input_settings_db.setMinimumSize(QSize(550, 31))
44     self.input_settings_db.setObjectName("input_settings_db")
45     self.formLayout.addWidget(0, QtWidgets.QFormLayout.FieldRole, self.input_settings_db)
46     self.label_grades_db = QtWidgets.QLabel(self.groupBox_db)
47     self.label_grades_db.setMinimumSize(QSize(110, 0))
48     self.label_grades_db.setObjectName("label_grades_db")
49     self.formLayout.addWidget(1, QtWidgets.QFormLayout.LabelRole, self.label_grades_db)
50     self.input_grades_db = QtWidgets.QLineEdit(self.groupBox_db)
51     self.input_grades_db.setMinimumSize(QSize(550, 31))
52     self.input_grades_db.setText("")
53     self.input_grades_db.setObjectName("input_grades_db")
54     self.formLayout.addWidget(1, QtWidgets.QFormLayout.FieldRole, self.input_grades_db)
55     self.verticalLayout.addWidget(self.groupBox_db)
56     self.groupBox_user = QtWidgets.QGroupBox(Settings)
57     self.groupBox_user.setEnabled(False)
58     sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Preferred, QtWidgets.QSizePolicy.Preferred)
    )

59     sizePolicy.setHorizontalStretch(0)
60     sizePolicy.setVerticalStretch(0)
61     sizePolicy.setHeightForWidth(self.groupBox_user.sizePolicy().hasHeightForWidth())
62     self.groupBox_user.setSizePolicy(sizePolicy)
63     self.groupBox_user.setMinimumSize(QSize(0, 0))
64     self.groupBox_user.setObjectName("groupBox_user")
65     self.formLayout_2 = QtWidgets.QFormLayout(self.groupBox_user)
66     self.formLayout_2.setObjectName("formLayout_2")
67     self.label_logisim_path = QtWidgets.QLabel(self.groupBox_user)
68     sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Preferred, QtWidgets.QSizePolicy.Preferred)
    )

69     sizePolicy.setHorizontalStretch(0)
70     sizePolicy.setVerticalStretch(0)
71     sizePolicy.setHeightForWidth(self.label_logisim_path.sizePolicy().hasHeightForWidth())
72     self.label_logisim_path.setSizePolicy(sizePolicy)
73     self.label_logisim_path.setMinimumSize(QSize(110, 0))
74     self.label_logisim_path.setObjectName("label_logisim_path")
75     self.formLayout_2.addWidget(0, QtWidgets.QFormLayout.LabelRole, self.label_logisim_path)
76     self.input_logisim_path = QtWidgets.QLineEdit(self.groupBox_user)
77     sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Expanding, QtWidgets.QSizePolicy.Fixed)
78     sizePolicy.setHorizontalStretch(0)
79     sizePolicy.setVerticalStretch(0)
80     sizePolicy.setHeightForWidth(self.input_logisim_path.sizePolicy().hasHeightForWidth())
81     self.input_logisim_path.setSizePolicy(sizePolicy)
82     self.input_logisim_path.setMinimumSize(QSize(637, 31))
83     self.input_logisim_path.setText("")
84     self.input_logisim_path.setObjectName("input_logisim_path")
85     self.formLayout_2.addWidget(0, QtWidgets.QFormLayout.FieldRole, self.input_logisim_path)
86     self.label_local_stor = QtWidgets.QLabel(self.groupBox_user)
87     self.label_local_stor.setMinimumSize(QSize(110, 0))
88     self.label_local_stor.setObjectName("label_local_stor")
89     self.formLayout_2.addWidget(1, QtWidgets.QFormLayout.LabelRole, self.label_local_stor)
90     self.input_local_stor = QtWidgets.QLineEdit(self.groupBox_user)
91     self.input_local_stor.setMinimumSize(QSize(637, 31))
92     self.input_local_stor.setText("")
93     self.input_local_stor.setObjectName("input_local_stor")
94     self.formLayout_2.addWidget(1, QtWidgets.QFormLayout.FieldRole, self.input_local_stor)
95     self.label_rem_stor = QtWidgets.QLabel(self.groupBox_user)
96     self.label_rem_stor.setMinimumSize(QSize(110, 0))
97     self.label_rem_stor.setObjectName("label_rem_stor")
98     self.formLayout_2.addWidget(2, QtWidgets.QFormLayout.LabelRole, self.label_rem_stor)
99     self.input_rem_stor = QtWidgets.QLineEdit(self.groupBox_user)
100    self.input_rem_stor.setMinimumSize(QSize(637, 31))
101    self.input_rem_stor.setInputMask("")
102    self.input_rem_stor.setText("")

```

```

103     self.input_rem_stor.setObjectName("input_rem_stor")
104     self.formLayout_2.addWidget(2, QtWidgets.QFormLayout.FieldRole, self.input_rem_stor)
105     self.verticalLayout.addWidget(self.groupBox_user)
106     self.groupBox_local = QtWidgets.QGroupBox(Settings)
107     sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Preferred,
QtWidgets.QSizePolicy.MinimumExpanding)
108     sizePolicy.setHorizontalStretch(0)
109     sizePolicy.setVerticalStretch(0)
110     sizePolicy.setHeightForWidth(self.groupBox_local.sizePolicy().hasHeightForWidth())
111     self.groupBox_local.setSizePolicy(sizePolicy)
112     self.groupBox_local.setMinimumSize(QtCore.QSize(0, 145))
113     self.groupBox_local.setMaximumSize(QtCore.QSize(16777215, 300))
114     self.groupBox_local.setFlat(False)
115     self.groupBox_local.setCheckable(False)
116     self.groupBox_local.setObjectName("groupBox_local")
117     self.gridLayout = QtWidgets.QGridLayout(self.groupBox_local)
118     self.gridLayout.setObjectName("gridLayout")
119     self.spin_year = QtWidgets.QSpinBox(self.groupBox_local)
120     self.spin_year.setEnabled(False)
121     self.spin_year.setMinimumSize(QtCore.QSize(110, 31))
122     self.spin_year.setMaximumSize(QtCore.QSize(110, 16777215))
123     self.spin_year.setWrapping(True)
124     self.spin_year.setReadOnly(False)
125     self.spin_year.setButtonSymbols(QtWidgets.QAbstractSpinBox.PlusMinus)
126     self.spin_year.setAccelerated(True)
127     self.spin_year.setProperty("showGroupSeparator", False)
128     self.spin_year.setMinimum(2012)
129     self.spin_year.setMaximum(2026)
130     self.spin_year.setProperty("value", 2018)
131     self.spin_year.setObjectName("spin_year")
132     self.gridLayout.addWidget(self.spin_year, 0, 1, 1, 1)
133     self.label_grad_year = QtWidgets.QLabel(self.groupBox_local)
134     sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Preferred, QtWidgets.QSizePolicy.Fixed)
135     sizePolicy.setHorizontalStretch(0)
136     sizePolicy.setVerticalStretch(0)
137     sizePolicy.setHeightForWidth(self.label_grad_year.sizePolicy().hasHeightForWidth())
138     self.label_grad_year.setSizePolicy(sizePolicy)
139     self.label_grad_year.setMinimumSize(QtCore.QSize(110, 31))
140     self.label_grad_year.setMaximumSize(QtCore.QSize(110, 16777215))
141     self.label_grad_year.setObjectName("label_grad_year")
142     self.gridLayout.addWidget(self.label_grad_year, 0, 0, 1, 1)
143     self.input_grader_name = QtWidgets.QLineEdit(self.groupBox_local)
144     self.input_grader_name.setEnabled(False)
145     self.input_grader_name.setMinimumSize(QtCore.QSize(110, 31))
146     self.input_grader_name.setMaximumSize(QtCore.QSize(110, 16777215))
147     self.input_grader_name.setObjectName("input_grader_name")
148     self.gridLayout.addWidget(self.input_grader_name, 2, 1, 1, 1)
149     self.label_semester = QtWidgets.QLabel(self.groupBox_local)
150     sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Preferred, QtWidgets.QSizePolicy.Fixed)
151     sizePolicy.setHorizontalStretch(0)
152     sizePolicy.setVerticalStretch(0)
153     sizePolicy.setHeightForWidth(self.label_semester.sizePolicy().hasHeightForWidth())
154     self.label_semester.setSizePolicy(sizePolicy)
155     self.label_semester.setMinimumSize(QtCore.QSize(110, 31))
156     self.label_semester.setMaximumSize(QtCore.QSize(110, 16777215))
157     self.label_semester.setObjectName("label_semester")
158     self.gridLayout.addWidget(self.label_semester, 0, 3, 1, 1)
159     self.label_style = QtWidgets.QLabel(self.groupBox_local)
160     self.label_style.setMinimumSize(QtCore.QSize(110, 31))
161     self.label_style.setObjectName("label_style")
162     self.gridLayout.addWidget(self.label_style, 1, 0, 1, 1)
163     self.label_sync_comm = QtWidgets.QLabel(self.groupBox_local)
164     self.label_sync_comm.setObjectName("label_sync_comm")
165     self.gridLayout.addWidget(self.label_sync_comm, 2, 3, 1, 1)
166     self.label_grader_name = QtWidgets.QLabel(self.groupBox_local)
167     sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Preferred, QtWidgets.QSizePolicy.Fixed)
168     sizePolicy.setHorizontalStretch(0)
169     sizePolicy.setVerticalStretch(0)
170     sizePolicy.setHeightForWidth(self.label_grader_name.sizePolicy().hasHeightForWidth())
171     self.label_grader_name.setSizePolicy(sizePolicy)
172     self.label_grader_name.setMinimumSize(QtCore.QSize(110, 31))
173     self.label_grader_name.setObjectName("label_grader_name")
174     self.gridLayout.addWidget(self.label_grader_name, 2, 0, 1, 1)
175     self.style_checkBox = QtWidgets.QCheckBox(self.groupBox_local)
176     self.style_checkBox.setEnabled(False)
177     sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.MinimumExpanding,
QtWidgets.QSizePolicy.Fixed)
178     sizePolicy.setHorizontalStretch(0)
179     sizePolicy.setVerticalStretch(0)
180     sizePolicy.setHeightForWidth(self.style_checkBox.sizePolicy().hasHeightForWidth())
181     self.style_checkBox.setSizePolicy(sizePolicy)

```

```

182         self.style_checkBox.setMinimumSize(QtCore.QSize(0, 31))
183         self.style_checkBox.setMaximumSize(QtCore.QSize(110, 16777215))
184         self.style_checkBox.setLayoutDirection(QtCore.Qt.LeftToRight)
185         self.style_checkBox.setText("")
186         self.style_checkBox.setObjectName("style_checkBox")
187         self.gridLayout.addWidget(self.style_checkBox, 1, 1, 1, 1)
188         self.semester_comboBox = QtWidgets.QComboBox(self.groupBox_local)
189         self.semester_comboBox.setEnabled(False)
190         self.semester_comboBox.setMinimumSize(QtCore.QSize(110, 31))
191         self.semester_comboBox.setMaximumSize(QtCore.QSize(110, 16777215))
192         self.semester_comboBox.setMaxVisibleItems(3)
193         self.semester_comboBox.setMaxCount(5)
194         self.semester_comboBox.setObjectName("semester_comboBox")
195         self.semester_comboBox.addItem("")
196         self.semester_comboBox.addItem("")
197         self.semester_comboBox.addItem("")
198         self.gridLayout.addWidget(self.semester_comboBox, 0, 4, 1, 1)
199         self.sync_command = QtWidgets.QLineEdit(self.groupBox_local)
200         self.sync_command.setEnabled(False)
201         self.sync_command.setMinimumSize(QtCore.QSize(0, 31))
202         self.sync_command.setInputMask("")
203         self.sync_command.setObjectName("sync_command")
204         self.gridLayout.addWidget(self.sync_command, 2, 4, 1, 4)
205         self.import_stuents_btn = QtWidgets.QPushButton(self.groupBox_local)
206         self.import_stuents_btn.setObjectName("import_stuents_btn")
207         self.gridLayout.addWidget(self.import_stuents_btn, 0, 6, 1, 1)
208         spacerItem = QtWidgets.QSpacerItem(40, 20, QtWidgets.QSizePolicy.Expanding,
QtWidgets.QSizePolicy.Minimum)
209         self.gridLayout.addItem(spacerItem, 0, 5, 1, 1)
210         self.verticalLayout.addWidget(self.groupBox_local)
211         self.buttonBox = QtWidgets.QDialogButtonBox(Settings)
212         sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Expanding, QtWidgets.QSizePolicy.Fixed)
213         sizePolicy.setHorizontalStretch(0)
214         sizePolicy.setVerticalStretch(0)
215         sizePolicy.setHeightForWidth(self.buttonBox.sizePolicy().hasHeightForWidth())
216         self.buttonBox.setSizePolicy(sizePolicy)
217         self.buttonBox.setOrientation(QtCore.Qt.Horizontal)
218         self.buttonBox.setStandardButtons(QtWidgets.QDialogButtonBox.Apply|
QtWidgets.QDialogButtonBox.Cancel|QtWidgets.QDialogButtonBox.Ok|QtWidgets.QDialogButtonBox.Reset|QtWidgets.QDialogButtonBox
219         self.buttonBox.setObjectName("buttonBox")
220         self.verticalLayout.addWidget(self.buttonBox)
221
222         self.retranslateUi(Settings)
223         self.buttonBox.accepted.connect(Settings.accept)
224         self.buttonBox.rejected.connect(Settings.reject)
225         QtCore.QMetaObject.connectSlotsByName(Settings)
226

```

7.16.3 Member Data Documentation

7.16.3.1 buttonBox

settings.Ui_Settings.buttonBox

Definition at line 211 of file settings.py.

7.16.3.2 formLayout

settings.Ui_Settings.formLayout

Definition at line 35 of file settings.py.

7.16.3.3 formLayout_2

```
settings.Ui_Settings.formLayout_2
```

Definition at line 65 of file settings.py.

7.16.3.4 gridLayout

```
settings.Ui_Settings.gridLayout
```

Definition at line 117 of file settings.py.

7.16.3.5 groupBox_db

```
settings.Ui_Settings.groupBox_db
```

Definition at line 23 of file settings.py.

7.16.3.6 groupBox_local

```
settings.Ui_Settings.groupBox_local
```

Definition at line 106 of file settings.py.

7.16.3.7 groupBox_user

```
settings.Ui_Settings.groupBox_user
```

Definition at line 56 of file settings.py.

7.16.3.8 import_students_btn

```
settings.Ui_Settings.import_students_btn
```

Definition at line 205 of file settings.py.

7.16.3.9 input_grader_name

```
settings.Ui_Settings.input_grader_name
```

Definition at line 143 of file settings.py.

7.16.3.10 input_grades_db

```
settings.Ui_Settings.input_grades_db
```

Definition at line 50 of file settings.py.

7.16.3.11 input_local_stor

```
settings.Ui_Settings.input_local_stor
```

Definition at line 90 of file settings.py.

7.16.3.12 input_logisim_path

```
settings.Ui_Settings.input_logisim_path
```

Definition at line 76 of file settings.py.

7.16.3.13 input_rem_stor

```
settings.Ui_Settings.input_rem_stor
```

Definition at line 99 of file settings.py.

7.16.3.14 input_settings_db

```
settings.Ui_Settings.input_settings_db
```

Definition at line 41 of file settings.py.

7.16.3.15 label_grad_year

```
settings.Ui_Settings.label_grad_year
```

Definition at line 133 of file settings.py.

7.16.3.16 label_grader_name

```
settings.Ui_Settings.label_grader_name
```

Definition at line 166 of file settings.py.

7.16.3.17 label_grades_db

```
settings.Ui_Settings.label_grades_db
```

Definition at line 46 of file settings.py.

7.16.3.18 label_local_stor

```
settings.Ui_Settings.label_local_stor
```

Definition at line 86 of file settings.py.

7.16.3.19 label_logisim_path

```
settings.Ui_Settings.label_logisim_path
```

Definition at line 67 of file settings.py.

7.16.3.20 label_rem_stor

```
settings.Ui_Settings.label_rem_stor
```

Definition at line 95 of file settings.py.

7.16.3.21 label_semester

```
settings.Ui_Settings.label_semester
```

Definition at line 149 of file settings.py.

7.16.3.22 label_settings_db

```
settings.Ui_Settings.label_settings_db
```

Definition at line 37 of file settings.py.

7.16.3.23 label_style

```
settings.Ui_Settings.label_style
```

Definition at line 159 of file settings.py.

7.16.3.24 label_sync_comm

```
settings.Ui_Settings.label_sync_comm
```

Definition at line 163 of file settings.py.

7.16.3.25 semester_comboBox

```
settings.Ui_Settings.semester_comboBox
```

Definition at line 188 of file settings.py.

7.16.3.26 spin_year

```
settings.Ui_Settings.spin_year
```

Definition at line 119 of file settings.py.

7.16.3.27 style_checkBox

`settings.Ui_Settings.style_checkBox`

Definition at line 175 of file settings.py.

7.16.3.28 sync_command

`settings.Ui_Settings.sync_command`

Definition at line 199 of file settings.py.

7.16.3.29 verticalLayout

`settings.Ui_Settings.verticalLayout`

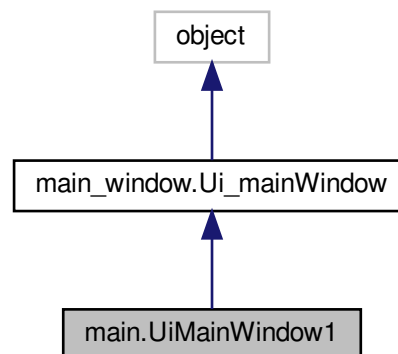
Definition at line 21 of file settings.py.

The documentation for this class was generated from the following file:

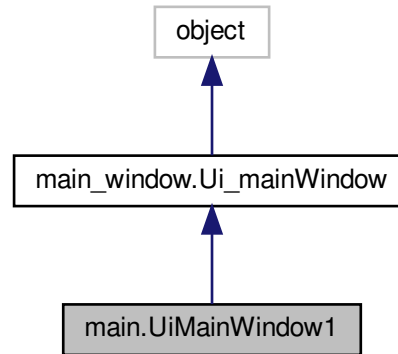
- [settings.py](#)

7.17 main.UiMainWindow1 Class Reference

Inheritance diagram for main.UiMainWindow1:



Collaboration diagram for `main.UiMainWindow1`:



Public Member Functions

- `def __init__ (self)`
- `def disable_fields (self)`
- `def enable_fields (self)`
- `def load_dir (self)`
- `def my_open_file (self)`
- `def show_stat (self)`
- `def check_file (self)`
- `def next_circ (self)`
- `def prev_circ (self)`
- `def check_wrong (self)`
- `def regrade (self)`
- `def reset_grade_resp (self)`
- `def update_popular_answers (self)`
- `def save_grade (self)`
- `def save_response (self)`
- `def save_all (self)`
- `def track_final_grade (self)`
- `def setupUi (self, main_window)`
- `def sync_params_to_settings (self)`
- `def bind_functions (self)`
- `def change_win_style (self)`
- `def dummy_d_1 (self)`
- `def update_user_comment_from_popular_answers (self)`
- `def open_file_diag (self)`
- `def memorize_user_comment (self)`
- `def kill_logisim (self)`
- `def run_logisim (self, filename)`
- `def generate_reports (self)`
- `def open_settings_dialog (self)`
- `def open_manage_labs_diag (self)`

Public Attributes

- [grader_ref](#)
- [cal_window](#)
- [working_dir](#)
- [class_id_to_id](#)
- [current_tz](#)
- [logisim_path](#)
- [grader_name](#)
- [settings_window](#)
- [manage_labs_window](#)

7.17.1 Detailed Description

Definition at line 719 of file main.py.

7.17.2 Constructor & Destructor Documentation

7.17.2.1 `__init__()`

```
def main.UiMainWindow1.__init__ (
    self )
```

Definition at line 721 of file main.py.

```
721     def __init__(self):
722         Ui_mainWindow.__init__(self)
723         self.grader_ref = None
724         self.cal_window = None
725         self.working_dir = None
726
727
728
```

7.17.3 Member Function Documentation

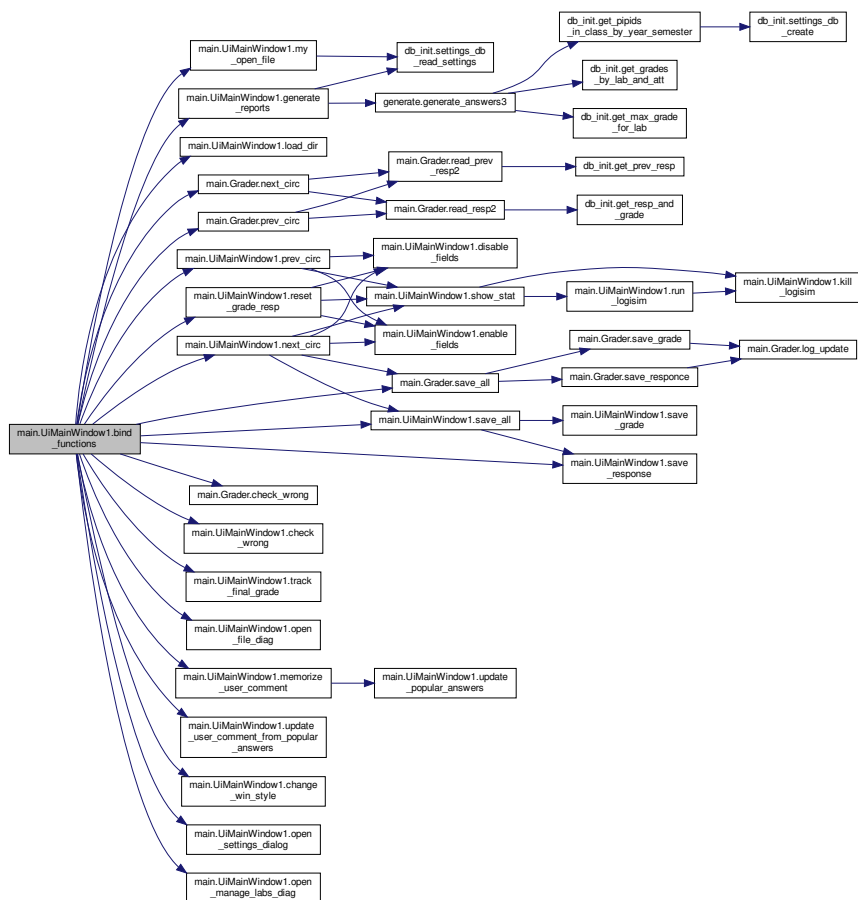
7.17.3.1 bind_functions()

```
def main.UiMainWindow1.bind_functions (
    self )
```

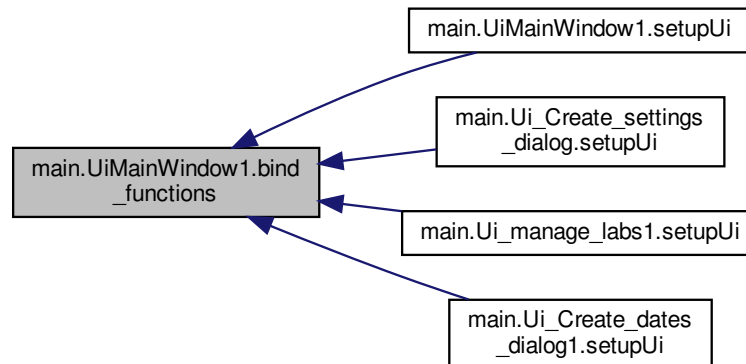
Definition at line 1124 of file main.py.

```
1124     def bind_functions(self):
1125         self.but_file_open.clicked.connect(self.my_open_file)
1126         self.but_begin.clicked.connect(self.load_dir)
1127         self.but_next.clicked.connect(self.next_circ)
1128         self.but_prev.clicked.connect(self.prev_circ)
1129         self.checkB_wrong.clicked.connect(self.check_wrong)
1130         # self.but_regrade.clicked.connect(self.regrade)
1131         self.but_save_all.clicked.connect(self.save_all)
1132         self.but_save_response.clicked.connect(self.save_response)
1133         self.input_final_grade.textEdited.connect(self.track_final_grade)
1134         # self.but_edit_done.clicked.connect(self.resp_edit_done)
1135         # self.popular_answers.activated.connect(self.select_saved_answer)
1136         # self.but_create_report.setEnabled(True) # Debug
1137         self.but_create_report.clicked.connect(self.generate_reports)
1138         # self.new_window_but.clicked.connect(self.open_dates_dialog)
1139         # self.input_response_browser_user.focusInEvent(self, self.memorize_user_comment)
1140         # self.custom_but_test.rightClicked[int].connect(self.dummy_d)
1141         self.input_file_location.dclicked.connect(self.open_file_diag)
1142         self.input_response_browser_user.focus_lost.connect(self.memorize_user_comment)
1143         self.popular_answers.currentIndexChanged.connect(self.update_user_comment_from_popular_answers)
1144         self.set_style_checkbox.stateChanged.connect(self.change_win_style)
1145         self.but_reset.clicked.connect(self.reset_grade_resp)
1146         self.settings_but.clicked.connect(self.open_settings_dialog)
1147         self.manage_labs_but.clicked.connect(self.open_manage_labs_diag)
1148         # self.sync_but.clicked.connect(self.sync_files)
1149
1150
1151
```

Here is the call graph for this function:



Here is the caller graph for this function:



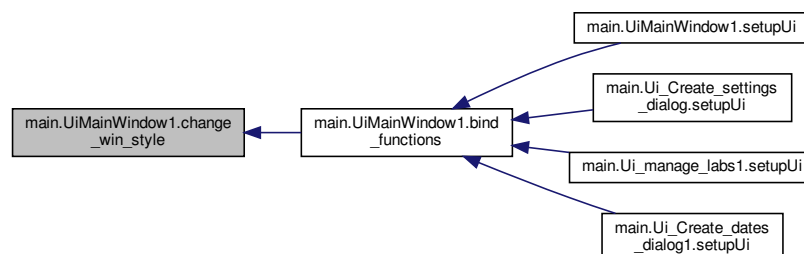
7.17.3.2 change_win_style()

```
def main.UiMainWindow1.change_win_style (
    self )
```

Definition at line 1157 of file main.py.

```
1157     def change_win_style(self):
1158         if self.set_style_checkbox.isChecked():
1159             self.progressBar.setStyleSheet(styleData)
1160         else:
1161             self.progressBar.setStyleSheet('')
1162
```

Here is the caller graph for this function:



7.17.3.3 check_file()

```
def main.UiMainWindow1.check_file (
    self )
```

Definition at line 900 of file main.py.

```
900     def check_file(self):
901         self.input_subtract.setText(str(self.grader_ref.subtract))
902         self.input_final_grade.setText(str(self.grader_ref.final_grade))
903
904         self.input_log_browser.setText(self.grader_ref.global_log)
905         # self.input_log_browser.append(self.grader_ref.global_log)
906
907         if self.grader_ref.input_correct:
908             self.checkB_input_pin_status.setChecked(True)
909         if self.grader_ref.output_correct:
910             self.checkB_output_pin_status.setChecked(True)
911
912         # self.but_save_response.setDisabled(True)
913         # self.but_save_all.setDisabled(True)
914
915         # self.but_edit_done.setDisabled(True)
916     try:
917         # self.grader_ref.generate_response() #TODO this overwrites File not found.
918         self.input_response_browser.setPlainText(self.grader_ref.resp_text)
919         # self.but_edit_done.setEnabled(True)
920         # self.but_save_response.setEnabled(True)
921         # self.but_save_all.setEnabled(True)
922     except Exception as e:
923         print('Error in generate response:', e)
924
```

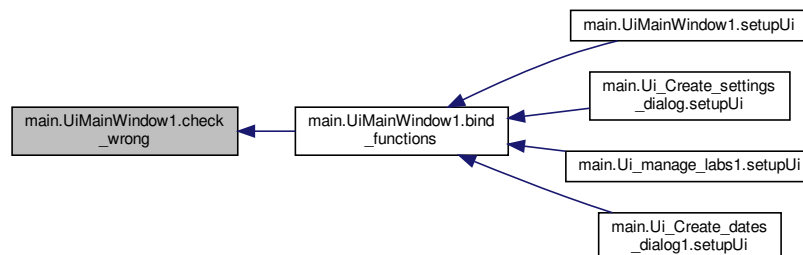
7.17.3.4 check_wrong()

```
def main.UiMainWindow1.check_wrong (
    self )
```

Definition at line 976 of file main.py.

```
976     def check_wrong(self):
977         if self.checkB_wrong.isEnabled():
978             self.grader_ref.check_wrong()
979             self.input_final_grade.setText(str(self.grader_ref.final_grade))
980             self.grader_ref.log_update('Lab was marked as wrong manually. Zero was assigned to final grade.
981         ')
982             self.input_response_browser.setPlainText(self.grader_ref.resp_text)
983             self.checkB_wrong.setDisabled(True)
```

Here is the caller graph for this function:



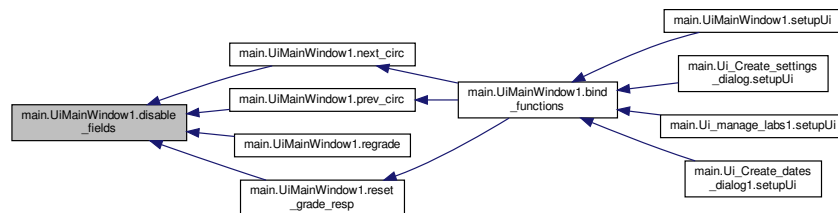
7.17.3.5 disable_fields()

```
def main.UiMainWindow1.disable_fields (
    self )
```

Definition at line 733 of file main.py.

```
733     def disable_fields(self):
734         self.checkBox_input_pin_status.setDisabled(True)
735         self.checkBox_output_pin_status.setDisabled(True)
736         # self.input_response_browser.setDisabled(True)
737         self.checkBox_wrong.setDisabled(True)
738
739         # self.input_subtract.setDisabled(True)
740         self.but_regrade.setDisabled(True)
741         self.popular_answers.setDisabled(True)
742         self.input_final_grade.setDisabled(True)
743         self.checkBox_wrong.setChecked(False)
744         self.checkBox_autosave.setDisabled(True)
745         self.input_current_id.setText('')
746
```

Here is the caller graph for this function:



7.17.3.6 dummy_d_1()

```
def main.UiMainWindow1.dummy_d_1 (
    self )
```

Definition at line 1164 of file main.py.

```
1164     def dummy_d_1(self):
1165         print('dummy_1 activated')
1166
```

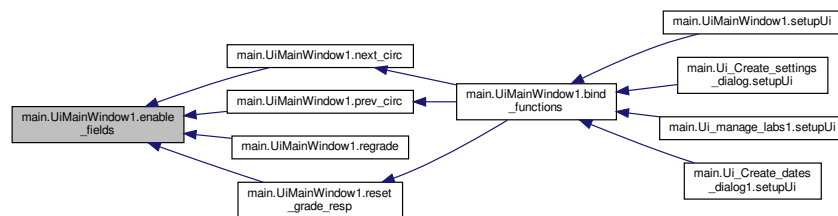
7.17.3.7 enable_fields()

```
def main.UiMainWindow1.enable_fields (
    self )
```

Definition at line 751 of file main.py.

```
751     def enable_fields(self):
752         self.checkB_input_pin_status.setEnabled(True)
753         self.checkB_output_pin_status.setEnabled(True)
754         # self.input_response_browser.setEnabled(True)
755         self.checkB_wrong.setEnabled(True)
756         self.input_final_grade.setEnabled(True)
757         self.check_autosave.setEnabled(True)
758
759         # self.input_subtract.setEnabled(True)
760         # self.but_regrade.setEnabled(True)
761         self.popular_answers.setEnabled(True)
762
```

Here is the caller graph for this function:



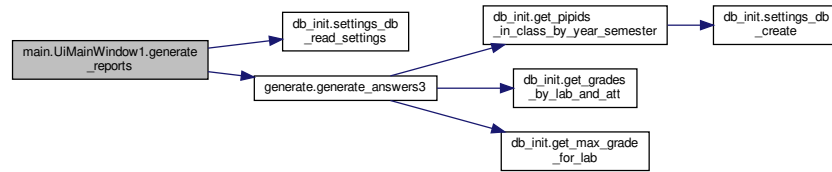
7.17.3.8 generate_reports()

```
def main.UiMainWindow1.generate_reports (
    self )
```

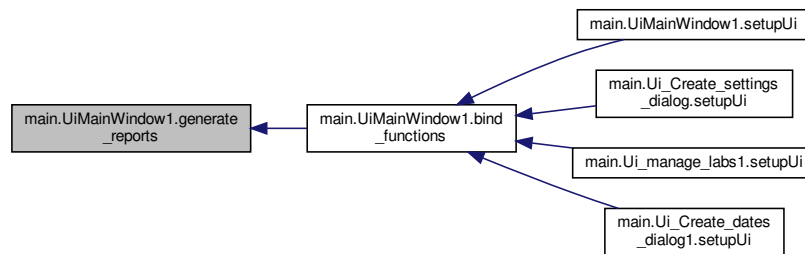
Definition at line 1241 of file main.py.

```
1241     def generate_reports(self):
1242         self.but_create_report.setDisabled(True)
1243         self.but_create_report.setText('Generating..')
1244         self.but_create_report.repaint()
1245         # from generate import generate_answers
1246         # (resubmit_num, dir_name, lab_type, lab_num)
1247         if hasattr(self, 'grader_ref'):
1248             loc_settings = settings_db_read_settings()[1]
1249             generate_answers3(self.grader_ref.lid, self.grader_ref.attempt, self.
grader_ref.year, self.grader_ref.semester)
1250             # generate_answers(self.grader_ref.attempt, self.grader_ref.working_dir,
self.grader_ref.lab_type, self.grader_ref.lab_num, loc_settings[1], loc_settings[2], self.grader_name)
1251             # generate_answers2(self.grader_ref.attempt, self.grader_ref.working_dir,
self.grader_ref.lab_type, self.grader_ref.lab_num, loc_settings[1], loc_settings[2], self.grader_name)
1252             self.but_create_report.setEnabled(True)
1253             self.but_create_report.setText('Create reports')
1254
1255
1256
```

Here is the call graph for this function:



Here is the caller graph for this function:



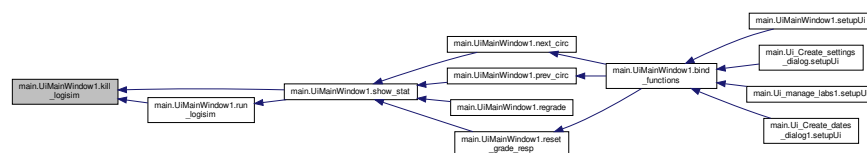
7.17.3.9 kill_logisim()

```
def main.UiMainWindow1.kill_logisim (
    self )
```

Definition at line 1216 of file main.py.

```
1216     def kill_logisim(self):
1217         try:
1218             self.grader_ref.logisim_pid.kill()
1219         except Exception as e:
1220             print("was not able to kill : ", e)
1221
```

Here is the caller graph for this function:



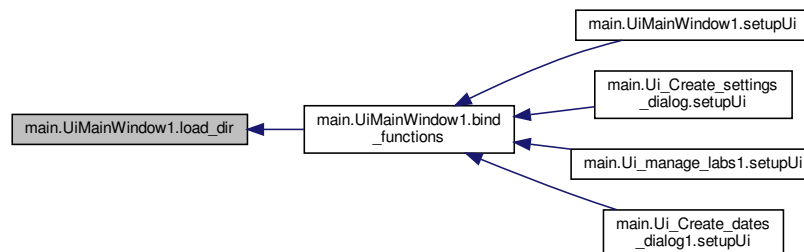
7.17.3.10 load_dir()

```
def main.UiMainWindow1.load_dir (
    self )
```

Definition at line 767 of file main.py.

```
767     def load_dir(self):
768         # activate elements
769         cur_year, cur_sem = self.grader_ref.working_dir.split('/')[ -3].split('_')
770         self.class_id_to_id = get_ids_in_class_by_year_semester(cur_year,
771         cur_sem)[1]
772         self.but_begin.setDisabled(True)
773         self.but_begin.repaint()
774         self.progressBar.setEnabled(True)
775         self.disable_fields()
776
777         self.grader_ref.tot_elem = len(self.grader_ref.lab_paths)
778         if self.grader_ref.tot_elem > 1:
779             self.but_next.setEnabled(True)
780
781         self.progressBar.setMaximum(self.grader_ref.tot_elem)
782         self.progressBar.setValue(0)
783         self.popular_answers.clear()
784
785         # self.grader_ref.check_file(0)
786         # self.grader_ref.stud_id = self.grader_ref.stud_ids[self.grader_ref.cur_idx]
787         self.grader_ref.cur_idx = -1
788         # graded = self.grader_ref.read_resp2()
789         # if graded:
790         #     self.grader_ref.read_prev_resp2()
791         self.next_circ()
792         # self.grader_ref.read_resp()
793         # self.grader_ref.read_prev_resp()
794         # self.show_stat()
795         # self.check_file()
796         # self.input_current_id.setPlainText(self.grader_ref.get_stud_id())
797
798         self.enable_fields()
799         self.input_response_browser_user.setEnabled(True)
800         self.but_regrade.setText('GRADE')
801         self.but_save_all.setEnabled(True)
802         self.but_save_response.setEnabled(True)
803         self.check_autosave.setEnabled(True)
804         self.but_reset.setEnabled(True)
805
```

Here is the caller graph for this function:



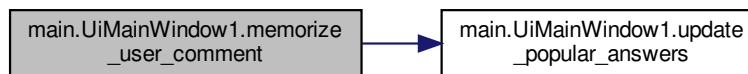
7.17.3.11 memorize_user_comment()

```
def main.UiMainWindow1.memorize_user_comment (
    self )
```

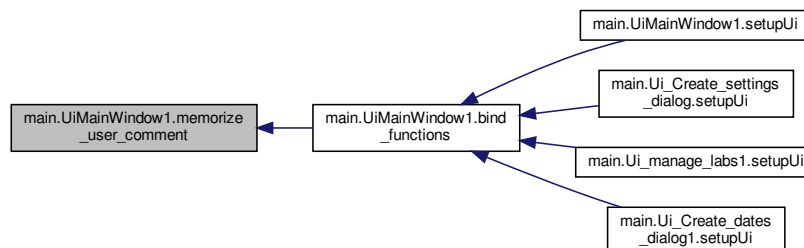
Definition at line 1191 of file main.py.

```
1191     def memorize_user_comment(self):
1192         typed = self.input_response_browser_user.toPlainText()
1193         if hasattr(self, 'grader_ref') and typed:
1194             try:
1195                 index = self.popular_answers.findText(self.input_response_browser_user.toPlainText(),
1196                                                         QtCore.Qt.MatchFixedString)
1197                 if index >= 0:
1198                     self.popular_answers.setCurrentIndex(index)
1199             else:
1200                 self.grader_ref.add_to_common_answers(typed)
1201                 self.update_popular_answers()
1202                 index = self.popular_answers.findText(self.input_response_browser_user.toPlainText(),
1203                                                         QtCore.Qt.MatchFixedString)
1204             try:
1205                 self.popular_answers.setCurrentIndex(index)
1206             except Exception as e:
1207                 print('Failed to select proper index: ', e)
1208                 raise
1209         except Exception as e:
1210             print('failed to add popular answer: ', e)
1211
```

Here is the call graph for this function:



Here is the caller graph for this function:



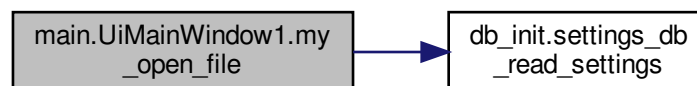
7.17.3.12 my_open_file()

```
def main.UiMainWindow1.my_open_file (
    self )
```

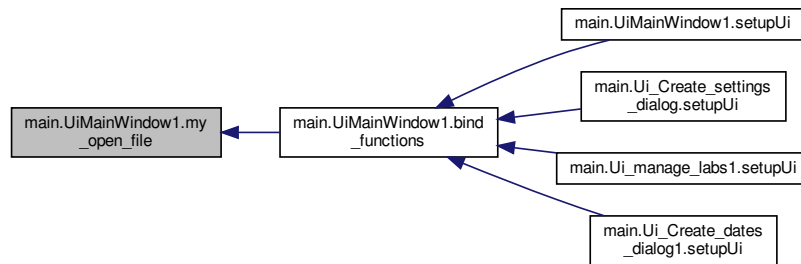
Definition at line 812 of file main.py.

```
812     def my_open_file(self):
813         working_dir = self.input_file_location.text()
814         # self.input_response_browser.clear()
815         # self.input_response_browser_user.clear()
816         self.input_response_browser.setPlainText('I did not find any errors. Good job!')
817         grader_name = settings_db_read_settings()[1][0]
818         self.current_tz = QDateTime.currentDateTime().timeZoneAbbreviation()
819
820     try:
821         my_grader = Grader(working_dir, grader_name)
822         my_grader.open_dir()
823
824         self.grader_ref = my_grader
825
826         self.input_max_pos_grade.setText(str(my_grader.lab_max_grade))
827         self.input_attempt.setText(str(my_grader.attempt))
828         self.dateTimeEdit_from.setDateTime(my_grader.time_from_qt)
829         self.dateTimeEdit_to.setDateTime(my_grader.time_to_qt)
830         self.grader_ref.add_to_common_answers('') # helps to remove all text in user comment section
831         # QDateTime.currentDateTime().timeZone()
832         # global MAIN_FILE_NAME, MAIN_FILE_NAME_OVERRIDE
833
834         # MAIN_FILE_NAME = get_lab_filename(my_grader.lab_id)[0]
835         # if not MAIN_FILE_NAME:
836         #     # Old way, I was determining filename as the most common submitted file.
837         #     if not MAIN_FILE_NAME_OVERRIDE:
838         #         a = []
839         #         for root, dirs, files in os.walk(working_dir):
840         #             for file in files:
841         #                 if file.endswith(".circ"):
842         #                     a.append(file)
843         #         a = np.array(a)
844         #         MAIN_FILE_NAME = Counter(a.flat).most_common(1)[0][0]
845         #     else:
846         #         MAIN_FILE_NAME = MAIN_FILE_NAME_OVERRIDE
847         #     # Now I can just read it from DB
848
849         # self.grader_ref.circ_file_name = MAIN_FILE_NAME
850         self.filename_lineEdit.setText(self.grader_ref.circ_file_name.split('.')[0])
851         # self.reset_grade_resp()
852         self.but_save_all.setChecked(False)
853
854         self.but_create_report.setEnabled(True)
855         self.but_begin.setEnabled(True)
856
857     except Exception as e: # TODO add log error
858         print('Error in open_file : ', e)
859         print(sys.exc_info()[0])
860
861
```

Here is the call graph for this function:



Here is the caller graph for this function:



7.17.3.13 next_circ()

```
def main.UiMainWindow1.next_circ (
    self )
```

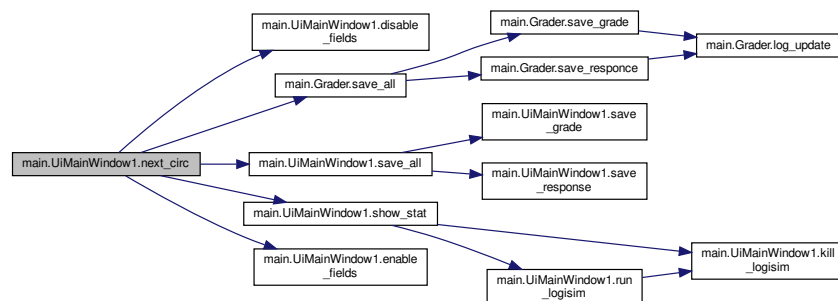
Definition at line 932 of file main.py.

```

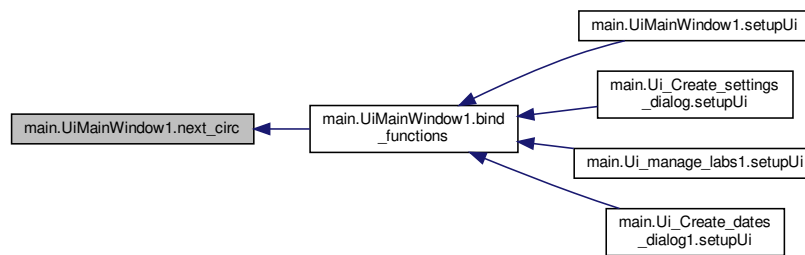
932     def next_circ(self):
933         self.disable_fields()
934         self.but_regrade.setText('GRADE')
935         if self.check_autosave.isChecked() and self.grader_ref.cur_idx >= 0:
936             self.save_all()
937         # else:
938         #     self.check_autosave.setDisabled(True)
939         next_idx = self.grader_ref.next_circ()
940         # self.check_file()
941         self.show_stat()
942         if next_idx >= self.grader_ref.tot_elem-1:
943             self.but_next.setDisabled(True)
944         if next_idx == 1:
945             self.but_prev.setEnabled(True)
946
947         self.progressBar.setValue(next_idx)
948         self.enable_fields()
949

```

Here is the call graph for this function:



Here is the caller graph for this function:



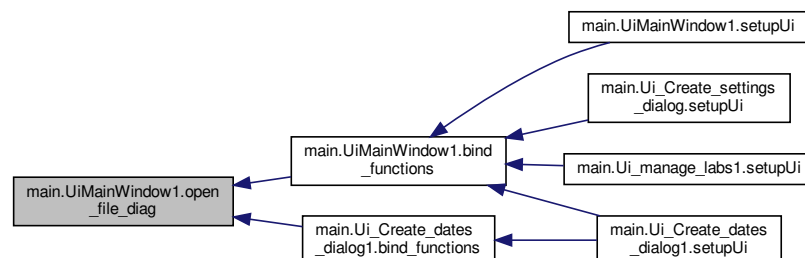
7.17.3.14 open_file_diag()

```
def main.UiMainWindow1.open_file_diag (
    self )
```

Definition at line 1180 of file main.py.

```
1180     def open_file_diag(self):
1181         obtained_dir = QFileDialog.getExistingDirectory(caption='Select directory with lab',
1182                                                         directory=self.input_file_location.text())
1183         if len(obtained_dir) > 1:
1184             self.input_file_location.setText(obtained_dir+'/')
1185
```

Here is the caller graph for this function:



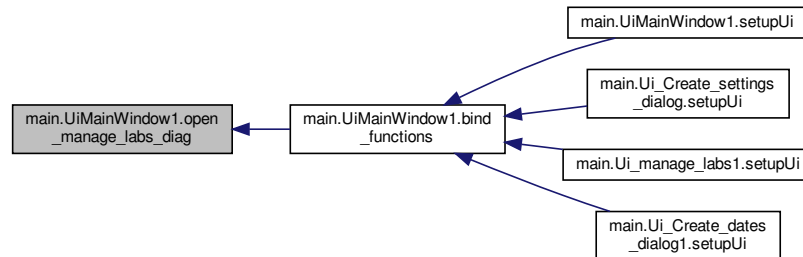
7.17.3.15 open_manage_labs_diag()

```
def main.UiMainWindow1.open_manage_labs_diag (
    self )
```

Definition at line 1315 of file main.py.

```
1315     def open_manage_labs_diag(self):
1316         self.manage_labs_but.setDisabled(True)
1317         self.manage_labs_but.repaint()
1318         self.centralwidget.setDisabled(True)
1319         self.centralwidget.repaint()
1320         self.manage_labs_window = QtWidgets.QDialog()
1321         dui = Ui_manage_labs1()
1322         dui.setupUi(self.manage_labs_window)
1323
1324         self.manage_labs_window.show()
1325         self.manage_labs_window.exec_()
1326
1327         self.centralwidget.setEnabled(True)
1328         self.manage_labs_but.setEnabled(True)
1329
1330         if not self.but_file_open.isEnabled():
1331             paths, local = settings_db_read_settings()
1332             # if there are some labs in server sync directory:
1333             if len(os.walk(get_full_path(paths, local) + "/server_sync/").__next__()[1]) > 0:
1334                 self.but_file_open.setEnabled(True)
1335                 self.input_file_location.setEnabled(True)
1336
1337
```

Here is the caller graph for this function:



7.17.3.16 open_settings_dialog()

```
def main.UiMainWindow1.open_settings_dialog (
    self )
```

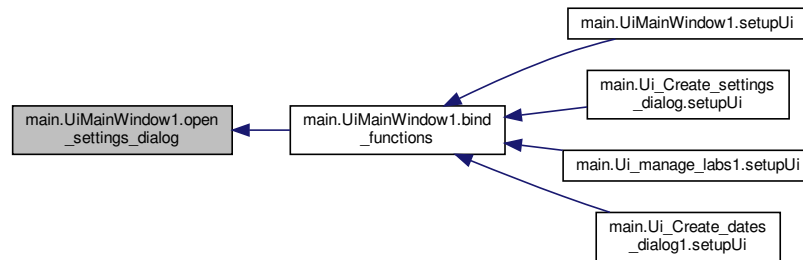
Definition at line 1285 of file main.py.

```

1285     def open_settings_dialog(self):
1286         self.settings_but.setDisabled(True)
1287         self.settings_but.repaint()
1288         self.settings_window = QtWidgets.QDialog()
1289         dui = Ui_Create_settings_dialog()
1290         dui.setupUi(self.settings_window)
1291
1292         self.centralwidget.setDisabled(True)
1293         self.centralwidget.repaint()
1294
1295         self.settings_window.show()
1296         self.settings_window.exec_()
1297
1298         self.sync_params_to_settings()
1299         self.centralwidget.setEnabled(True)
1300
1301         self.settings_but.setEnabled(True)
1302
1303         if not self.manage_labs_but.isEnabled():
1304             from pathlib import Path
1305             settings_location = str(Path(os.path.expandvars(os.path.expanduser('./settings.sqlite3'))).
absolute())
1306             if os.path.isfile(settings_location):
1307                 self.manage_labs_but.setEnabled(True)
1308
1309

```

Here is the caller graph for this function:



7.17.3.17 prev_circ()

```

def main.UiMainWindow1.prev_circ (
    self )

```

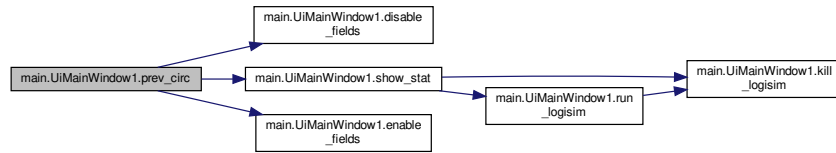
Definition at line 957 of file main.py.

```

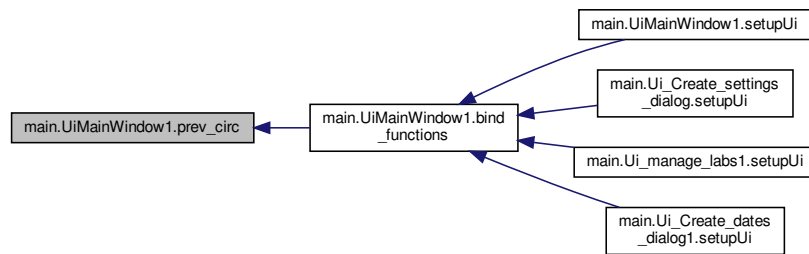
957     def prev_circ(self):
958         self.disable_fields()
959         self.but_regrade.setText('GRADE')
960         next_idx = self.grader_ref.prev_circ()
961         # self.check_file()
962         self.show_stat()
963         if next_idx <= self.grader_ref.tot_elem-1:
964             self.but_next.setEnabled(True)
965         if next_idx == 0:
966             self.but_prev.setDisabled(True)
967
968         self.progressBar.setValue(next_idx)
969         self.enable_fields()
970

```

Here is the call graph for this function:



Here is the caller graph for this function:



7.17.3.18 regrade()

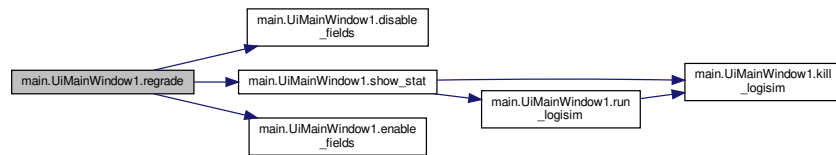
```
def main.UiMainWindow1.regrade (
    self )
```

Definition at line 988 of file main.py.

```

988     def regrade(self):
989         self.disable_fields()
990         self.but_regrade.setText('regrade')
991         # if self.lab_num > 8 and self.lab_type == 'Closed':
992         #     self.precheck_PLDs(i, cur_path)
993         self.show_stat()
994         # self.grader_ref.check_file()
995         # if self.grader_ref.check_circ_exist():
996         #     self.check_file()
997         self.input_response_browser.setPlainText(self.grader_ref.resp_text)
998         self.enable_fields()
999 
```

Here is the call graph for this function:



7.17.3.19 reset_grade_resp()

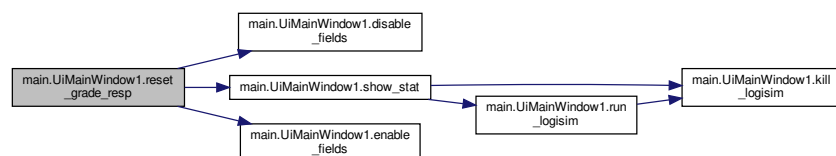
```
def main.UiMainWindow1.reset_grade_resp (
    self )
```

Definition at line 1004 of file main.py.

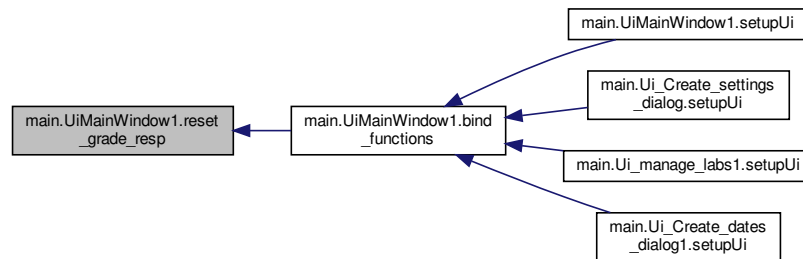
```

1004     def reset_grade_resp(self):
1005         self.disable_fields()
1006         self.show_stat()
1007         # self.grader_ref.check_file()
1008         # if self.grader_ref.check_circ_exist():
1009         if self.grader_ref.lab_num > 8 and self.grader_ref.lab_type == 'Closed':
1010             self.grader_ref.final_grade, report = self.grader_ref.precheck_PLDs(self.grader_ref.cur_idx)
1011             self.input_response_browser.setPlainText(report)
1012         else:
1013             self.grader_ref.final_grade = self.grader_ref.lab_max_grade
1014             self.input_response_browser.setPlainText('I did not find any errors. Good job!')
1015
1016         self.input_final_grade.setText(str(self.grader_ref.final_grade))
1017         self.enable_fields()
1018 
```

Here is the call graph for this function:



Here is the caller graph for this function:



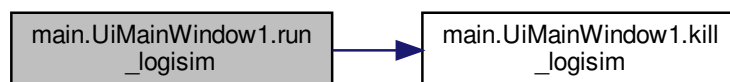
7.17.3.20 run_logisim()

```
def main.UiMainWindow1.run_logisim (
    self,
    filename )
```

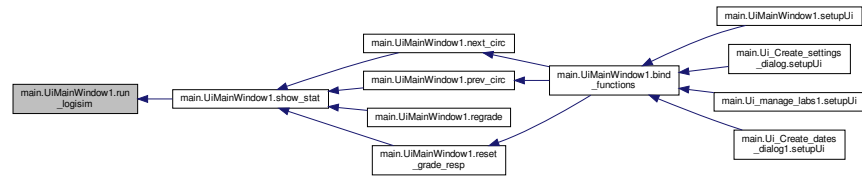
Definition at line 1228 of file main.py.

```
1228     def run_logisim(self, filename):
1229
1230         command = 'java -jar ' + self.logisim_path + 'logisim-generic-2.7.1.jar {}'.format(filename)
1231         # command_with_file = command + os.path.join(self.grader_ref.file_list[self.grader_ref.cur_idx],
1232         MAIN_FILE_NAME)
1233         # if self.grader_ref.logisim_pid.pid > 0:
1234         self.kill_logisim()
1235         self.grader_ref.logisim_pid = subprocess.Popen(command, shell=True)
```

Here is the call graph for this function:



Here is the caller graph for this function:



7.17.3.21 save_all()

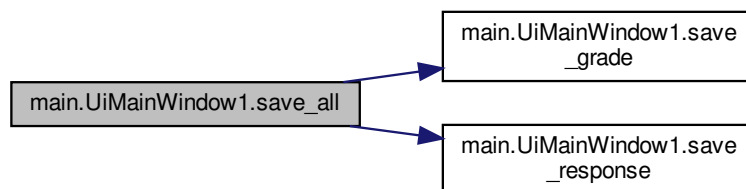
```
def main.UiMainWindow1.save_all (
    self )
```

Definition at line 1053 of file main.py.

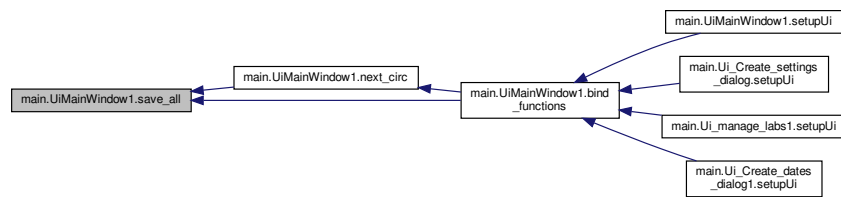
```

1053     def save_all(self):
1054         self.grader_ref.save_grade()
1055         # self.grader_ref.save_response()
1056         self.save_response()
1057         self.grader_ref.save_all2()
1058 
```

Here is the call graph for this function:



Here is the caller graph for this function:



7.17.3.22 save_grade()

```
def main.UiMainWindow1.save_grade (
    self )
```

Definition at line 1035 of file main.py.

```
1035     def save_grade(self):
1036         self.grader_ref.save_grade()
1037
```

Here is the caller graph for this function:



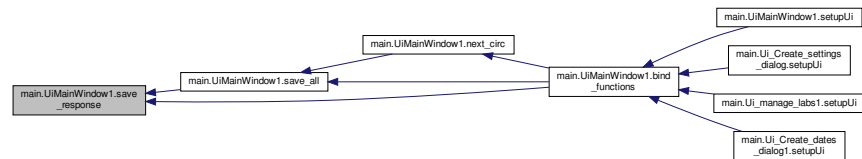
7.17.3.23 save_response()

```
def main.UiMainWindow1.save_response (
    self )
```

Definition at line 1043 of file main.py.

```
1043     def save_response(self):
1044         self.grader_ref.resp_text = self.input_response_browser.toPlainText()
1045         self.grader_ref.user_comment = self.input_response_browser_user.toPlainText()
1046         self.grader_ref.save_responce()
1047
```

Here is the caller graph for this function:



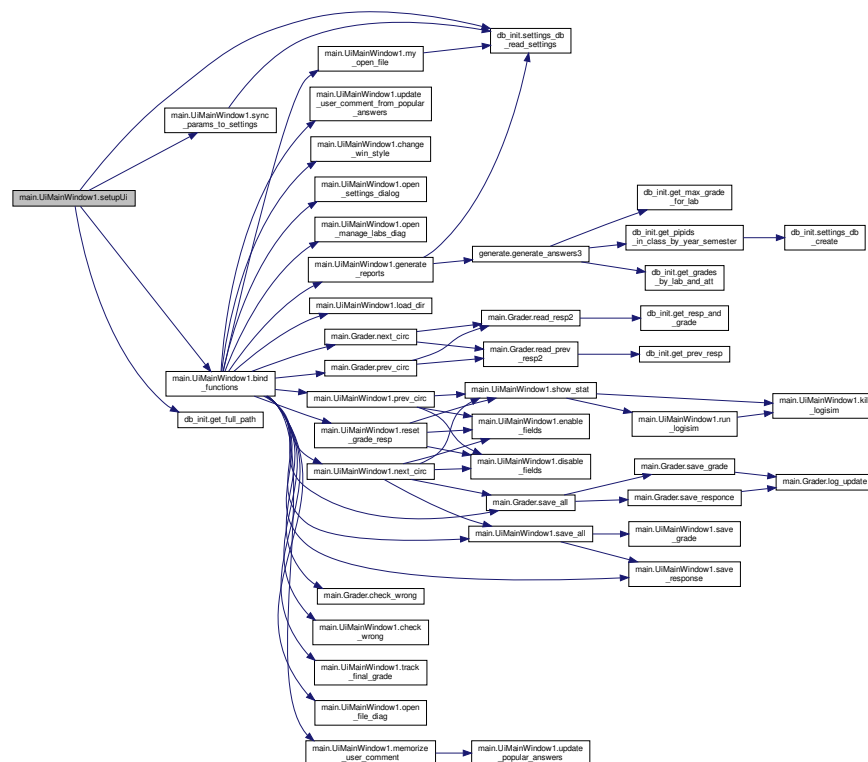
7.17.3.24 setupUi()

```
def main.UiMainWindow1.setupUi (
    self,
    main_window )
```

Definition at line 1076 of file main.py.

```
1076     def setupUi(self, main_window):
1077         super().setupUi(main_window)
1078
1079         self.bind_functions()
1080         self.sync_params_to_settings()
1081
1082         from pathlib import Path
1083         settings_location = str(Path(os.path.expandvars(os.path.expanduser('./settings.sqlite3'))).absolute
1084         ())
1085         if os.path.isfile(settings_location):
1086             paths, local = settings_db_read_settings()
1087             try:
1088                 if len(os.walk(get_full_path(paths, local) + "/server_sync/").__next__()[1]) >
1089                 0:
1090                     if not self.manage_labs_but.isEnabled():
1091                         self.manage_labs_but.setEnabled(True)
1092                     if not self.but_file_open.isEnabled():
1093                         self.but_file_open.setEnabled(True)
1094                     self.input_file_location.setEnabled(True)
1095             except Exception as e:
1096                 print("Most likely you did not fill all the settings: ", e)
```

Here is the call graph for this function:



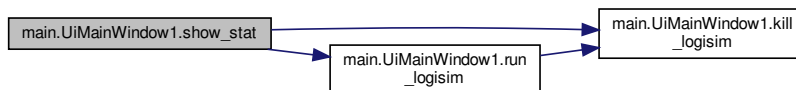
7.17.3.25 show_stat()

```
def main.UiMainWindow1.show_stat (
    self )
```

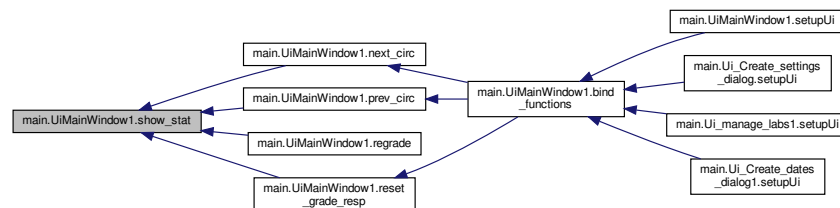
Definition at line 867 of file main.py.

```
867     def show_stat(self):
868         self.input_prev_response.setPlainText(self.grader_ref.previous_responses)
869         file_path = os.path.join(self.grader_ref.lab_paths[self.grader_ref.cur_idx], self.grader_ref.
circ_file_name)
870         if not Path(file_path).is_file():
871             self.kill_logisim()
872             self.grader_ref.final_grade = 0
873             self.input_response_browser.setPlainText('File does not exist.')
874             self.grader_ref.final_grade = 0
875         else:
876             if self.but_regrade.text() == '&GRADE' or self.but_regrade.text() == 'GRADE':
877                 try:
878                     self.run_logisim(file_path)
879                 except Exception as e:
880                     print('Error in run_logisim: ', e)
881                     print(sys.exc_info()[0])
882
883             self.input_current_id.setText(self.class_id_to_id[self.grader_ref.get_stud_id()])
884             self.dateTimeEdit_submitted.setDateTime(QDateTime.fromSecsSinceEpoch(self.grader_ref.timestamps[
self.grader_ref.cur_idx]))
885             self.input_subtract.setText('')
886             self.input_final_grade.setText(str(self.grader_ref.final_grade))
887             self.input_log_browser.setText(self.grader_ref.global_log)
888             self.input_response_browser.setPlainText(self.grader_ref.resp_text)
889             self.input_response_browser_user.setPlainText(self.grader_ref.user_comment)
890             self.checkBox_input_pin_status.setChecked(False)
891             self.checkBox_output_pin_status.setChecked(False)
892             self.popular_answers.setCurrentIndex(-1)
893
```

Here is the call graph for this function:



Here is the caller graph for this function:



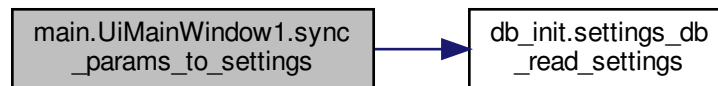
7.17.3.26 sync_params_to_settings()

```
def main.UiMainWindow1.sync_params_to_settings (
    self )
```

Definition at line 1101 of file main.py.

```
1101     def sync_params_to_settings(self):
1102         paths, local = settings_db_read_settings()
1103         working_dir = ''
1104         if paths and len(paths) == 4:
1105             self.logisim_path = paths[0]
1106             if len(paths[1]) > 0:
1107                 working_dir = paths[1]
1108             else:
1109                 working_dir = './'
1110         if local and len(local) >= 4:
1111             self.grader_name = local[0]
1112             working_dir += str(local[1])
1113             working_dir += '_' + local[2] + '/'
1114             self.set_style_checkbox.setChecked(bool(local[3]))
1115
1116         if len(working_dir) > 0:
1117             self.input_file_location.setText(os.path.expanduser(working_dir))
1118
1119
```

Here is the call graph for this function:



Here is the caller graph for this function:



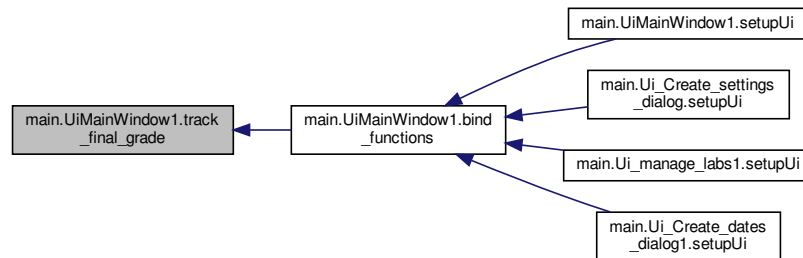
7.17.3.27 track_final_grade()

```
def main.UiMainWindow1.track_final_grade (
    self )
```

Definition at line 1063 of file main.py.

```
1063     def track_final_grade(self):
1064         grade = self.input_final_grade.text()
1065         self.grader_ref.log_update('Manual grade change from : ' + str(self.grader_ref.final_grade))
1066         self.input_log_browser.setText(self.grader_ref.global_log)
1067         self.grader_ref.final_grade = int(grade)
1068         self.grader_ref.log_update('Manual grade change to: ' + str(grade))
1069         self.input_log_browser.setText(self.grader_ref.global_log)
1070
```

Here is the caller graph for this function:



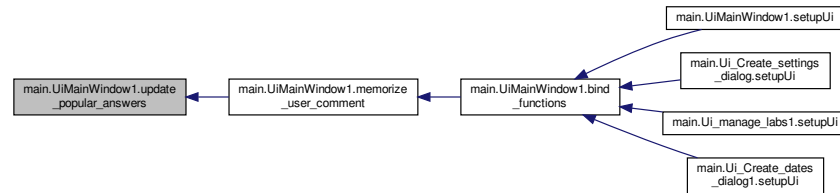
7.17.3.28 update_popular_answers()

```
def main.UiMainWindow1.update_popular_answers (
    self )
```

Definition at line 1024 of file main.py.

```
1024     def update_popular_answers(self):
1025         if len(self.popular_answers) != len(self.grader_ref.input_suggestion):
1026             self.popular_answers.clear()
1027             self.popular_answers.addItem(self.grader_ref.input_suggestion)
1028             # for item in self.grader_ref.input_suggestion:
1029
```

Here is the caller graph for this function:



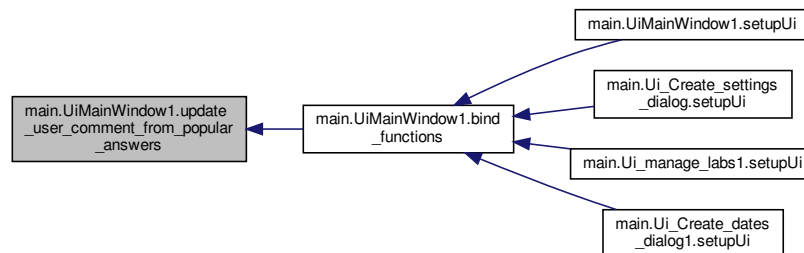
7.17.3.29 update_user_comment_from_popular_answers()

```
def main.UiMainWindow1.update_user_comment_from_popular_answers (
    self )
```

Definition at line 1171 of file main.py.

```
1171     def update_user_comment_from_popular_answers(self):
1172         if self.popular_answers.hasFocus():
1173             self.input_response_browser_user.setPlainText(self.popular_answers.currentText())
1174
```

Here is the caller graph for this function:



7.17.4 Member Data Documentation

7.17.4.1 cal_window

`main.UiMainWindow1.cal_window`

Definition at line 724 of file main.py.

7.17.4.2 class_id_to_id

`main.UiMainWindow1.class_id_to_id`

Definition at line 770 of file main.py.

7.17.4.3 current_tz

`main.UiMainWindow1.current_tz`

Definition at line 818 of file main.py.

7.17.4.4 grader_name

`main.UiMainWindow1.grader_name`

Definition at line 1111 of file main.py.

7.17.4.5 grader_ref

`main.UiMainWindow1.grader_ref`

Definition at line 723 of file main.py.

7.17.4.6 logisim_path

`main.UiMainWindow1.logisim_path`

Definition at line 1105 of file main.py.

7.17.4.7 manage_labs_window

`main.UiMainWindow1.manage_labs_window`

Definition at line 1320 of file main.py.

7.17.4.8 settings_window

`main.UiMainWindow1.settings_window`

Definition at line 1288 of file main.py.

7.17.4.9 working_dir

`main.UiMainWindow1.working_dir`

Definition at line 725 of file main.py.

The documentation for this class was generated from the following file:

- [main.py](#)

Chapter 8

File Documentation

8.1 create_dates_diag.py File Reference

Classes

- class [create_dates_diag.Ui_Create_dates_dialog](#)

Namespaces

- [create_dates_diag](#)

8.2 dates_window.py File Reference

Classes

- class [dates_window.Ui_dates_window](#)

Namespaces

- [dates_window](#)

8.3 db_init.py File Reference

Namespaces

- [db_init](#)

Functions

- `def db_init.settings_db_create` (db_name=SETTINGS_DB_NAME, force=False)
- `def db_init.settings_db_read_settings` (db_name=SETTINGS_DB_NAME)
- `def db_init.update_settings` (paths, local, db_name=SETTINGS_DB_NAME)
- `def db_init.grades_db_create` (db_name, force=False)
- `def db_init.load_student_list_into_grades_db` (db_name, year, semester, filename='students_list3.txt')
- `def db_init.insert_students` (ids, fname, lname, db_name='./grades.sqlite3')
- `def db_init.register_students_in_class` (pipeline_ids, year, semester, db_name='./grades.sqlite3')
- `def db_init.get_pipeline_ids` (db_name='./grades.sqlite3')
- `def db_init.get_ids_in_class_by_year_semester` (year, semester, db_name='./grades.sqlite3')
- `def db_init.import_previous_grades_into_db` (year, semester, db_name='./grades.sqlite3', filename='./grades.xls')
- `def db_init.gen_filenotfound_resp` (lab_id, stud_path, corr_file, grader, att=None, next_date=None, db_name='./grades.sqlite3')
- `def db_init.get_resp_and_grade` (grade_id, db_name='./grades.sqlite3')
- `def db_init.get_prev_resp` (grade_id, class_id, lab_id, db_name='./grades.sqlite3')
- `def db_init.save_a_grade_to_db` (grade_id, grade, grader_comment, extra_comment, grader_name, graded=True, pass_fail=True, db_name='./grades.sqlite3')
- `def db_init.init_new_lab` (stud_id, lab_name, att, submitted, lab_path, db_name='./grades.sqlite3')
- `def db_init.get_lab_names` (db_name='./grades.sqlite3')
- `def db_init.update_lab_submissions_paths` (db_name, repository_root, year, semester)
- `def db_init.get_empty_grades_by_lid` (lab_id, att, db_name='./grades.sqlite3')
- `def db_init.get_all_grades_by_lid` (lab_id, att, db_name='./grades.sqlite3')
- `def db_init.reconstruct_grades_and_comments` (db_name='./grades.sqlite3')
- `def db_init.generate_final_grades` (db_name, year, semester)
- `def db_init.get_max_grade_for_lab` (lid, year, semester, db_name='./grades.sqlite3')
- `def db_init.get_grades_by_lab_and_att` (lid, att, db_name='./grades.sqlite3')
- `def db_init.get_lab_filename` (lab_id, db_name='./grades.sqlite3')
- `def db_init.get_lab_max_value` (lab_id, db_name='./grades.sqlite3')
- `def db_init.get_full_path` (paths, local)
- `def db_init.sync_files` (self=None)
- `def db_init.export_pdf` (self=None)
- `def db_init.save_grade_and_report` (grade_id, grade, report, user_comment, grader, db_name='./grades.sqlite3')
- `def db_init.commit_gen_report` (grade_id, db_name='./grades.sqlite3')
- `def db_init.get_lab_id` (ltype, lab_num)
- `def db_init.register_lab_in_semester` (ltype, lab_num, year, semester, due_dates, db_name='./grades.sqlite3')
- `def db_init.get_labid_in_schedule` (lid, year, semester, db_name='./grades.sqlite3')
- `def db_init.get_due_date_by_labid` (lid_sem, att=None, db_name='./grades.sqlite3')
- `def db_init.get_import_dates_by_labid` (lid_sem, att=None, db_name='./grades.sqlite3')
- `def db_init.gen_report` (lid_sem, att=None, db_name='./grades.sqlite3')
- `def db_init.get_pipids_in_class_by_year_semester` (year, semester, db_name='./grades.sqlite3')

Variables

- string `db_init.SETTINGS_DB_NAME` = 'settings.sqlite3'

8.4 generate.py File Reference

Namespaces

- [generate](#)

Functions

- def [generate.convert_to_pdf](#) (html_file, func_type)
- def [generate.create_html_pdf_report2](#) (lab_dict)
Creates nice html report for submitted labs and converts it to pdf format.
- def [generate.create_html_pdf_zero_report](#) (filename, stud_name, top_part, bot_part)
- def [generate.create_not_submitted](#) (stud_id, lab_type, lab_num, dir_name)
- def [generate.generate_answers3](#) (lid, att, year, semester, db_name='./grades.sqlite3')
- def [generate.time_to_str_with_tz](#) (in_time)

8.5 main.py File Reference

Classes

- class [main.CircFile](#)
- class [main.CircFile.circ_type](#)
- class [main.CircFile.PinType](#)
- class [main.Grader](#)
- class [main.UiMainWindow1](#)
- class [main.Ui_Create_settings_dialog](#)
Creates window that provides user with convenient way of changing settings that are stored in sqlite3 db.
- class [main.SimpleDialog](#)
Wrapper class for very simple Ok|Cancel dialog.
- class [main.Ui_manage_labs1](#)
- class [main.Ui_Create_dates_dialog1](#)

Namespaces

- [main](#)

Functions

- def [main.read_settings](#) (db_name='settings.sqlite3')
- def [main.get_grading_period](#) (lid, cur_only=False)

Variables

- string [main.MAIN_FILE_NAME](#) = "
- string [main.MAIN_FILE_NAME_OVERRIDE](#) = "
- string [main.styleData](#)
- [main.app](#) = QtWidgets.QApplication(sys.argv)
- [main.MainWindow](#) = QtWidgets.QMainWindow()
- [main.ui](#) = UiMainWindow1()

8.6 main_window.py File Reference

Classes

- class [main_window.Ui_mainWindow](#)

Namespaces

- [main_window](#)

8.7 manage_labs.py File Reference

Classes

- class [manage_labs.Ui_manage_labs](#)

Namespaces

- [manage_labs](#)

8.8 qt_class_improvements.py File Reference

Classes

- class [qt_class_improvements.BetterLineEdit](#)
- class [qt_class_improvements.BetterPlainTextEdit](#)

Namespaces

- [qt_class_improvements](#)

8.9 README.md File Reference

8.10 settings.py File Reference

Classes

- class [settings.Ui_Settings](#)

Namespaces

- [settings](#)

8.11 simple_dialog.py File Reference

Classes

- class [simple_dialog.Ui_Dialog](#)

Namespaces

- [simple_dialog](#)

