CSCI3130 Grader

Generated by Doxygen 1.8.14

Contents

Chapter 1

README

Chapter 2

Namespace Index

2.1 Namespace List

Here is a list of all namespaces with brief descriptions:

Chapter 3

Hierarchical Index

3.1 Class Hierarchy

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

Chapter 4

Class Index

4.1 Class List

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

Chapter 5

File Index

5.1 File List

Here is a list of all files with brief descriptions:

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.

```
def commit_gen_report(grade_id, db_name='./grades.sqlite3'):
    if not os.path.isfile(db_name):
        raise Exception("DB not found")
    with lite.connect(db_name) as con:
        cur = con.cursor()
        cur.execute("UPDATE grades SET report_generated=strftime('%s','now') WHERE id=?", (grade_id,))
        con.commit()



```

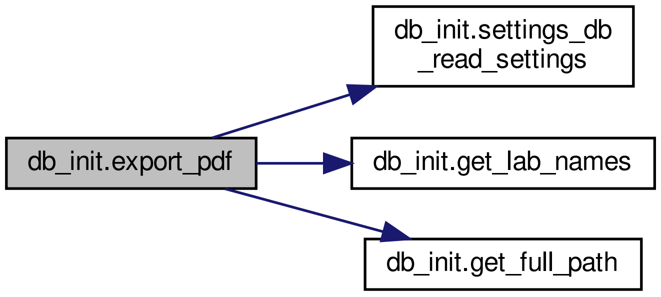
6.3.1.2 export_pdf()

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

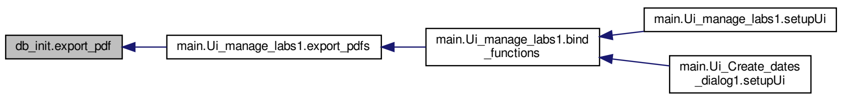
Definition at line 709 of file db_init.py.

```
def export_pdf(self=None):
    import subprocess
    import os

    paths, local = settings_db_read_settings()
    lab_ids, lab_types, lab_nums = get_lab_names()
    lab_names = []
    for i in range(len(lab_types)):
        lab_names.append(lab_types[i] + '_Lab_' + str(lab_nums[i]))

    full_path = get_full_path(paths, local) + "/"
    for lab_name in lab_names:
        nums_to_sync = '_{'
        i = 1
        while os.path.isdir(full_path + lab_name + '_' + str(i) + '/Answers'):
            nums_to_sync += str(i) + ','
            i += 1
        if i == 1:
            continue
        nums_to_sync = nums_to_sync[0:-1] + '}'
        # for case when we have only one directory to sync
        if len(nums_to_sync) == 4:
            nums_to_sync = '_1'
        if len(nums_to_sync) > 1:
            command = local[4] + ' ' + full_path + lab_name + nums_to_sync + '/Answers/*.pdf ' +
      os.path.expanduser(paths[2]) + lab_name + '/'
            process = subprocess.Popen(os.path.expandvars(command), stdout=subprocess.PIPE, shell=True)
            process.communicate()
            # print(output)
            # print(error)


```

Here is the call graph for this function:



Here is the caller graph for this function:


db_init.export_pdf
main.Ui_manage_labs1.export_pdfs
main.Ui_manage_labs1.bind_functions
main.Ui_manage_labs1.setupUi
main.Ui_Create_dates_dialog1.setupUi

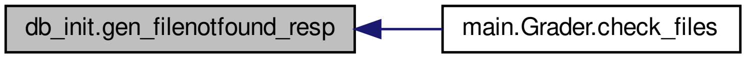
6.3.1.3 gen_filenotfound_resp()

```
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.

```
def gen_filenotfound_resp(lab_id, stud_path, corr_file, grader, att=None,
  next_date=None, db_name='./grades.sqlite3'):
    resp_text = 'file with name "{}" was not found.</br>'.format(corr_file)
    file_found = os.listdir(stud_path)
    potential_files = list()
    for file in file_found:
        if file not in ['grade.txt', 'penalty.txt', 'responce.txt', 'tech_info.txt', ]:
            potential_files.append(file)
    if potential_files:
        resp_text += '\nNext files|folders were found:</br>\n'
    for file in potential_files:
        if os.path.isdir(os.path.join(stud_path, file)):
            resp_text += file + ' - directory.</br>\n'
        else:
            resp_text += file + ' - regular file.</br>\n'

    if att and att < 4 and next_date:
        resp_text += 'Please submit your file by next due date ({}).</br>\n'.format(next_date)

    if not os.path.isfile(db_name):
        raise Exception("DB not found")
    with lite.connect(db_name) as con:
        cur = con.cursor()
        cur.execute("UPDATE grades SET graded=strftime('%s','now'), pass_fail=FALSE, grader_comment=?,
   grader=? WHERE id=?", (resp_text, grader, lab_id))
        con.commit()


```

Here is the caller graph for this function:



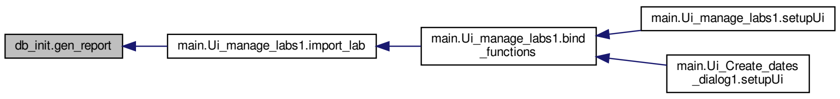
6.3.1.4 gen_report()

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

Definition at line 810 of file db_init.py.

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


```

Here is the caller graph for this function:


db_init.gen_report
main.Ui_manage_labs1.import_lab
main.Ui_manage_labs1.bind_functions
main.Ui_manage_labs1.setupUi
main.Ui_Create_dates_dialog1.setupUi

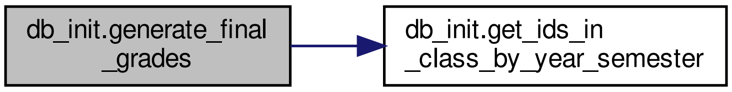
6.3.1.5 generate_final_grades()

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

Definition at line 595 of file db_init.py.

```
def generate_final_grades(db_name, year, semester):
    ids = get_ids_in_class_by_year_semester(year, semester, db_name)
    with lite.connect(db_name) as con:
        cur = con.cursor()

        labs = list()
        for sid in ids.values():  # using JOIN here will add too much extra data
            result = cur.execute('SELECT lab, MAX(grade * (select percent from penalties where
 id=GRADES.attempt)/100) '
                        'FROM GRADES WHERE class_id=? and attempt > 0 group by lab order by lab', (str(
sid),))
            labs.append(result.fetchall() )

        stud_info = list()
        for sid in ids.keys():
            result = cur.execute('SELECT first_name, second_name FROM students WHERE pipeline_id=?', (str(
sid),))
            stud_info.append(result.fetchall() )

    df_stud_info = pd.DataFrame(dict(zip(ids.keys(), stud_info)))
    df_grades = pd.DataFrame(dict(zip(ids.keys(), labs)))
    # id_list = list(ids.keys())
    # a = id_list[list(ids.values()).index(class_id)]


```

Here is the call graph for this function:


db_init.generate_final_grades
db_init.get_ids_in_class_by_year_semester

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.

```
def get_all_grades_by_lid(lab_id, att, db_name='./grades.sqlite3'):
    with lite.connect(db_name) as con:
        cur = con.cursor()
        result = cur.execute("SELECT submitted, class_id, id, lab_path FROM grades WHERE lab=? AND attempt=? ", (lab_id, att))
        try:
            subm, class_id, lab_id, lab_path = zip(*result.fetchall())
        except Exception as e:
            print(e)
            return None, None

    return subm, class_id, lab_id, lab_path


```

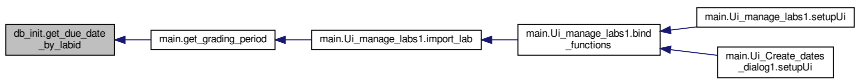
6.3.1.7 get_due_date_by_labid()

```
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.

```
def get_due_date_by_labid(lid_sem, att=None, db_name='./grades.sqlite3'):
    with lite.connect(db_name) as con:
        cur = con.cursor()
        if att:
            result = cur.execute('SELECT due_date_{} FROM lab_schedule WHERE id=?'.format(int(att)), (lid_sem,))
        else:
            result = cur.execute('SELECT due_date_1, due_date_2, due_date_3, due_date_4 FROM lab_schedule WHERE id=?', (lid_sem,))
        return result.fetchone()
    return None


```

Here is the caller graph for this function:


db_init.get_due_date_by_labid
main.get_grading_period
main.Ui_manage_labs1.import_lab
main.Ui_manage_labs1.bind_functions
main.Ui_manage_labs1.setupUi
main.Ui_Create_dates_dialog1.setupUi

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.

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

    return subm, class_id, lab_id, lab_path
```

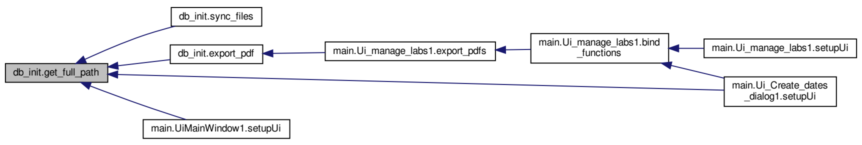
6.3.1.9 get_full_path()

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

Definition at line 672 of file db_init.py.

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

Here is the caller graph for this function:


db_init.sync_files
db_init.export_pdf
main.Ui_manage_labs1.export_pdfs
main.Ui_manage_labs1.bind_functions
main.Ui_manage_labs1.setupUi
db_init.get_full_path
main.Ui_Create_dates_dialog1.setupUi
main.UiMainWindow1.setupUi

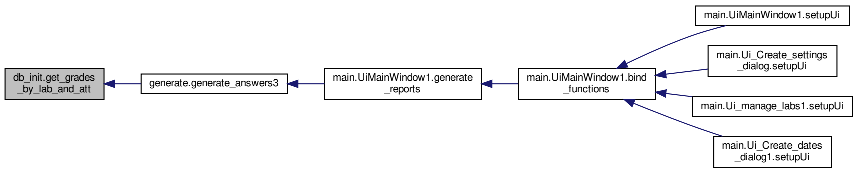
6.3.1.10 get_grades_by_lab_and_att()

```
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,
      c.grader_comment, c.extra_comment, c.grader, c.lab_path, '
641                              'd.pipeline_id, e.first_name, e.second_name, f.percent, c.grade*f.percent/100
      as final_grade '
642                              'from lab_schedule a '
643                              'join lab_names b on a.lab_id=b.id '
644                              'join grades c on c.lab=a.id '
645                              'join class d on d.id=c.class_id '
646                              'join students e on e.pipeline_id=d.pipeline_id '
647                              'join penalties f on f.id=c.attempt '
648                              'where c.attempt={0} AND a.id=? ORDER BY d.pipeline_id'.format(int(att)), (lid
     ,))
649         info_tup = result.fetchall()
650         info_desc = result.description
651     return info_tup, info_desc
652 
653 
```

Here is the caller graph for this function:


db_init.get_grades_by_lab_and_att
generate.generate_answers3
main.UiMainWindow1.generate_reports
main.UiMainWindow1.bind_functions
main.UiMainWindow1.setupUi
main.Ui_Create_settings_dialog.setupUi
main.Ui_manage_labs1.setupUi
main.Ui_Create_dates_dialog1.setupUi

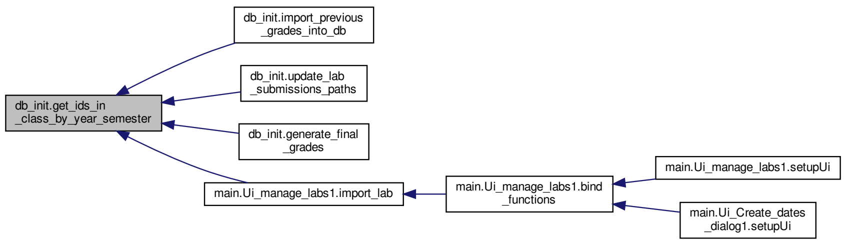
6.3.1.11 get_ids_in_class_by_year_semester()

```
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:


db_init.get_ids_in_class_by_year_semester
db_init.import_previous_grades_into_db
db_init.update_lab_submissions_paths
db_init.generate_final_grades
main.Ui_manage_labs1.import_lab
main.Ui_manage_labs1.bind_functions
main.Ui_manage_labs1.setupUi
main.Ui_Create_dates_dialog1.setupUi

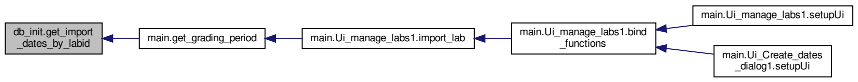
6.3.1.12 get_import_dates_by_labid()

```
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_{} FROM lab_schedule WHERE id=?'.format(int(att)), (
      lid_sem,))
803         else:
804             result = cur.execute('SELECT imported_1, imported_2, imported_3, imported_4 FROM lab_schedule
       WHERE id=?', (lid_sem,))
805         return result.fetchone()
806     return None
807 
808 
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:


db_init.get_import_dates_by_labid
main.get_grading_period
main.Ui_manage_labs1.import_lab
main.Ui_manage_labs1.bind_functions
main.Ui_manage_labs1.setupUi
main.Ui_Create_dates_dialog1.setupUi

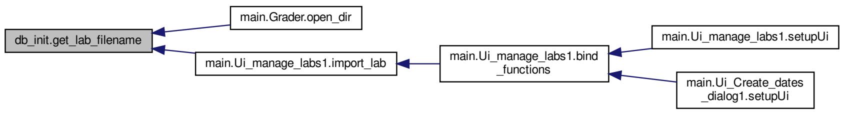
6.3.1.13 get_lab_filename()

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

Definition at line 654 of file db_init.py.

```
def get_lab_filename(lab_id, db_name='./grades.sqlite3'):
    with lite.connect(db_name) as con:
        cur = con.cursor()

        result = cur.execute('SELECT mandatory_files FROM lab_names WHERE id=? ', (str(lab_id),))
        return result.fetchall()[0]
    return None


```

Here is the caller graph for this function:


db_init.get_lab_filename
main.Grader.open_dir
main.Ui_manage_labs1.import_lab
main.Ui_manage_labs1.bind_functions
main.Ui_manage_labs1.setupUi
main.Ui_Create_dates_dialog1.setupUi

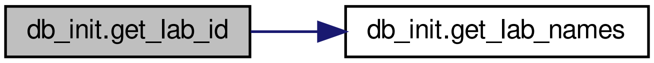
6.3.1.14 get_lab_id()

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

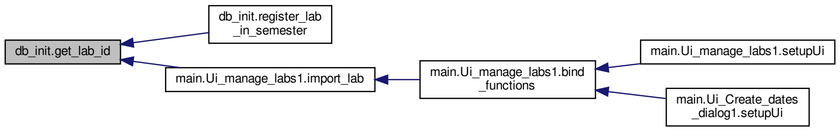
Definition at line 759 of file db_init.py.

```
def get_lab_id(ltype, lab_num):
    lab_ids, lab_types, lab_nums = get_lab_names()
    for i, lid in enumerate(lab_ids):
        if lab_types[i] == ltype and lab_num == lab_nums[i]:
            return lid
    return None


```

Here is the call graph for this function:



Here is the caller graph for this function:


db_init.get_lab_id
db_init.register_lab_in_semester
main.Ui_manage_labs1.import_lab
main.Ui_manage_labs1.bind_functions
main.Ui_manage_labs1.setupUi
main.Ui_Create_dates_dialog1.setupUi

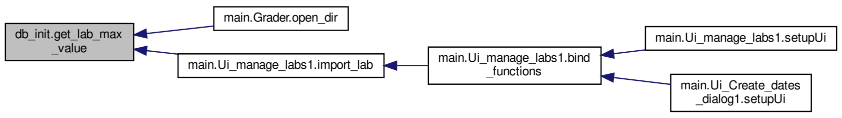
6.3.1.15 get_lab_max_value()

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

Definition at line 663 of file db_init.py.

```
def get_lab_max_value(lab_id, db_name='./grades.sqlite3'):
    with lite.connect(db_name) as con:
        cur = con.cursor()

        result = cur.execute('SELECT max_grade FROM lab_names WHERE id=? ', (str(lab_id),))
        return int(result.fetchone()[0])
    return None


```

Here is the caller graph for this function:


db_init.get_lab_max _value
main.Grader.open_dir
main.Ui_manage_labs1.import_lab
main.Ui_manage_labs1.bind _functions
main.Ui_manage_labs1.setupUi
main.Ui_Create_dates _dialog1.setupUi

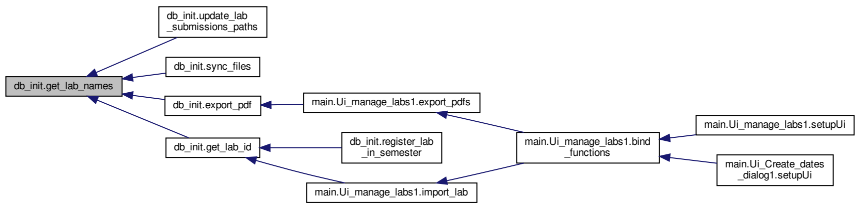
6.3.1.16 get_lab_names()

```
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:


db_init.get_lab_names
db_init.update_lab _submissions_paths
db_init.sync_files
db_init.export_pdf
db_init.get_lab_id
main.Ui_manage_labs1.export_pdfs
db_init.register_lab _in_semester
main.Ui_manage_labs1.import_lab
main.Ui_manage_labs1.bind _functions
main.Ui_manage_labs1.setupUi
main.Ui_Create_dates _dialog1.setupUi

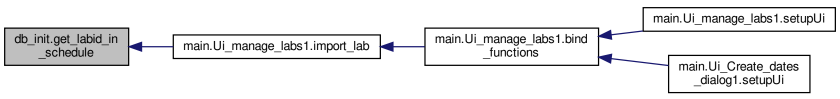
6.3.1.17 get_labid_in_schedule()

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

Definition at line 779 of file db_init.py.

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


```

Here is the caller graph for this function:


db_init.get_labid_in_schedule
main.Ui_manage_labs1.import_lab
main.Ui_manage_labs1.bind_functions
main.Ui_manage_labs1.setupUi
main.Ui_Create_dates_dialog1.setupUi

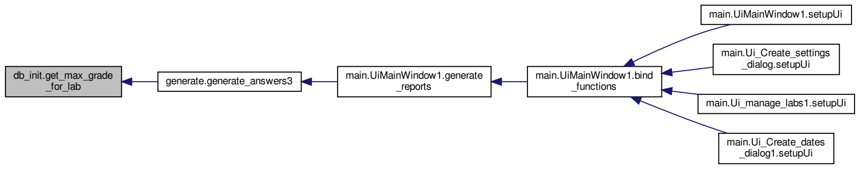
6.3.1.18 get_max_grade_for_lab()

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

Definition at line 617 of file db_init.py.

```
def get_max_grade_for_lab(lid, year, semester, db_name='./grades.sqlite3'):
    with lite.connect(db_name) as con:
        cur = con.cursor()
        result = cur.execute('SELECT e.pipeline_id as pipid, IFNULL(MAX(k.final_grade), 0) as grade '
                             'FROM class e '
                             'LEFT OUTER JOIN '
                             '   (SELECT d.pipeline_id, c.grade*f.percent/100 AS final_grade '
                             '   FROM grades c '
                             '      JOIN class d ON d.id = c.class_id '
                             '      JOIN penalties f ON f.id = c.attempt '
                             '   WHERE c.lab = ? ) k '
                             'ON e.pipeline_id = k.pipeline_id '
                             'WHERE year=? AND semester=? '
                             'GROUP BY e.pipeline_id '
                             'ORDER BY e.pipeline_id ', (lid, int(year), int(semester)))
        return result.fetchall()


```

Here is the caller graph for this function:


db_init.get_max_grade _for_lab
generate.generate_answers3
main.UiMainWindow1.generate _reports
main.UiMainWindow1.bind _functions
main.UiMainWindow1.setupUi
main.Ui_Create_settings _dialog.setupUi
main.Ui_manage_labs1.setupUi
main.Ui_Create_dates _dialog1.setupUi

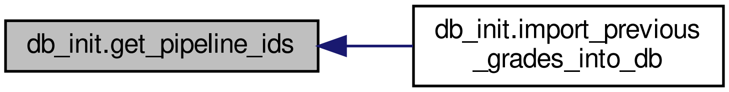
6.3.1.19 get_pipeline_ids()

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

Definition at line 290 of file db_init.py.

```
def get_pipeline_ids(db_name='./grades.sqlite3'):
    with lite.connect(db_name) as con:
        cur = con.cursor()
        result = cur.execute("SELECT pipeline_id FROM students")
        try:
            resut = (ids[0] for ids in result.fetchall())
        except Exception as e:
            print(e)
            return None
    return resut


#     :param year:
#     :param semester:
#     :param db_name:
#     :return:
#


```

Here is the caller graph for this function:



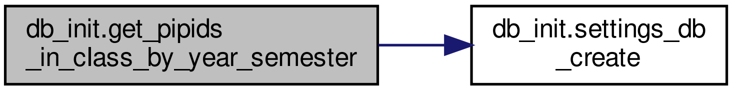
6.3.1.20 get_pipids_in_class_by_year_semester()

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

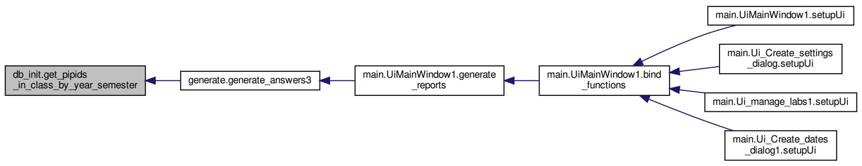
Definition at line 819 of file db_init.py.

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



```

Here is the call graph for this function:


db_init.get_pipids _in_class_by_year_semester
db_init.settings_db _create

Here is the caller graph for this function:


db_init.get_pipids _in_class_by_year_semester
generate.generate_answers3
main.UiMainWindow1.generate _reports
main.UiMainWindow1.bind _functions
main.UiMainWindow1.setupUi
main.Ui_Create_settings _dialog.setupUi
main.Ui_manage_labs1.setupUi
main.Ui_Create_dates _dialog1.setupUi

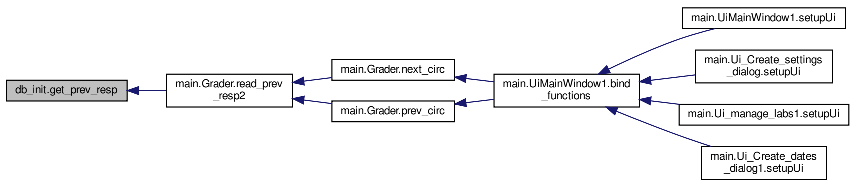
6.3.1.21 get_prev_resp()

```
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.

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


```

Here is the caller graph for this function:


db_init.get_prev_resp
main.Grader.read_prev_resp2
main.Grader.next_circ
main.Grader.prev_circ
main.UiMainWindow1.bind_functions
main.UiMainWindow1.setupUi
main.Ui_Create_settings_dialog.setupUi
main.Ui_manage_labs1.setupUi
main.Ui_Create_dates_dialog1.setupUi

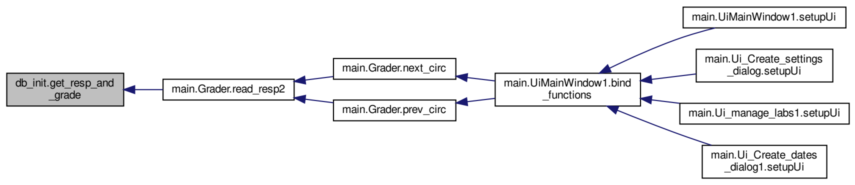
6.3.1.22 get_resp_and_grade()

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

Definition at line 434 of file db_init.py.

```
def get_resp_and_grade(grade_id, db_name='./grades.sqlite3'):
    with lite.connect(db_name) as con:
        cur = con.cursor()
        result = cur.execute("SELECT grade, grader_comment, extra_comment, graded FROM grades WHERE id=?",
      (grade_id,))
        grade, resp, uresp, graded = result.fetchone()

    return grade, resp, uresp, graded


```

Here is the caller graph for this function:


db_init.get_resp_and_grade
main.Grader.read_resp2
main.Grader.next_circ
main.Grader.prev_circ
main.UiMainWindow1.bind_functions
main.UiMainWindow1.setupUi
main.Ui_Create_settings_dialog.setupUi
main.Ui_manage_labs1.setupUi
main.Ui_Create_dates_dialog1.setupUi

6.3.1.23 grades_db_create()

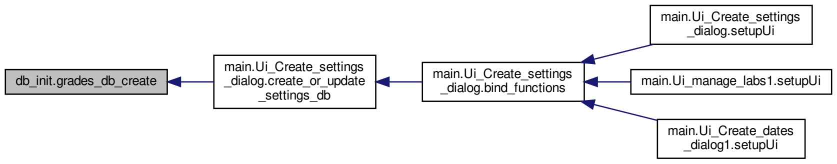
```
def db_init.grades_db_create (
              db_name,
              force = False )
```

Definition at line 94 of file db_init.py.

```
def grades_db_create(db_name, force=False):
    # from pathlib import Path
    print("I am going to create a grades DB with next name: ", db_name)
    db_name = str(db_name)
    if not os.path.isfile(db_name) or force:
        # compute some vars before the connection
        lab_names = list()
        for i in range(1, 13):
            lab_names.append(('CLA' + str(i), 'Closed', i, 10))
        for i in range(1, 9):
            lab_names.append(('OLA' + str(i), 'Open', i, 20))
        lab_names.append(('OLA9', 'Open', 9, 100))

        with lite.connect(db_name) as con:
            cur = con.cursor()
            # TODO: force should remove 'IF NOT EXISTS' and add 'DROP TABLE' to ensure new table creation
            # WISH: add TRY blocks for each CREATE and spawn new info window in case of error
            print('Creating students...')
            cur.execute("""CREATE TABLE students (
                            pipeline_id    TEXT    NOT NULL
                                                   PRIMARY KEY,
                            first_name     TEXT    NOT NULL,
                            second_name    TEXT    NOT NULL,
                            comment        TEXT,
                            cheating_ratio INTEGER DEFAULT (0)    );""")
            con.commit()
            print('Done.')
            print('Creating semesters...')
            cur.execute("""CREATE TABLE semesters (
                            semester CHAR (1) NOT NULL PRIMARY KEY,
                            name     VARCHAR      );""")
            con.commit()
            print('Done.')
            print('Creating class...')
            cur.execute("""CREATE TABLE class (
                            id             INTEGER PRIMARY KEY AUTOINCREMENT,
                            pipeline_id    TEXT    REFERENCES students (pipeline_id),
                            year           INTEGER,
                            semester       INTEGER REFERENCES semesters (semester),
                            cheating_ratio INTEGER DEFAULT (0),
                            UNIQUE (
                                pipeline_id,
                                year,
                                semester)     );""")
            con.commit()
            print(’Done.’)
            print(’Creating labs...’)
            cur.execute("""CREATE TABLE lab_names (
                            id              INT     NOT NULL PRIMARY KEY,
                            type            TEXT    NOT NULL,
                            num             INTEGER NOT NULL,
                            max_grade       INTEGER NOT NULL,
                            name            VARCHAR,
                            description     VARCHAR,
                            grader_comment  VARCHAR,
                            mandatory_files VARCHAR );""")
            con.commit()
            print(’Done.’)
            print(’Creating grades...’)
            cur.execute("""CREATE TABLE grades (
                            id               INTEGER PRIMARY KEY AUTOINCREMENT,
                            class_id                 NOT NULL
                                                     REFERENCES class (id) ON UPDATE CASCADE,
                            lab                      NOT NULL
                                                     REFERENCES lab_names (id) ON UPDATE CASCADE,
                            attempt          INT     DEFAULT (0),
                            submitted        INTEGER,
                            graded           INTEGER,
                            grade            INTEGER NOT NULL
                                                     DEFAULT (0),
                            pass_fail        BOOLEAN NOT NULL
                                                     DEFAULT (0),
                            grader_comment   TEXT,
                            extra_comment    TEXT,
                            report_generated BOOLEAN,
                            report_time      INTEGER,
                            lab_path         VARCHAR,
                            UNIQUE (
                                class_id,
                                lab,
                                attempt,
                                pass_fail) ON CONFLICT REPLACE );""")
            con.commit()
            print(’Done.’)

            print(’Creating lab schedule...’)
            cur.execute("""CREATE TABLE lab_schedule (
                            id         INTEGER PRIMARY KEY AUTOINCREMENT,
                            lab_id             REFERENCES lab_names (id),
                            year       INTEGER NOT NULL,
                            semester   INTEGER REFERENCES semesters (semester)
                                               NOT NULL,
                            due_date_1 INTEGER,
                            due_date_2 INTEGER,
                            due_date_3 INTEGER,
                            due_date_4 INTEGER,
                            imported_1 INTEGER,
                            imported_2 INTEGER,
                            imported_3 INTEGER,
                            imported_4 INTEGER,
                            posted_1   INTEGER,
                            posted_2   INTEGER,
                            posted_3   INTEGER,
                            posted_4   INTEGER
                        );""")
            con.commit()
            print(’Done.’)



            print(’Filling semesters...’)
            cur.executemany(’INSERT OR REPLACE INTO semesters\
                        (semester, name) VALUES (?, ?)’, [(1, ’SPRING’), (2, ’SUMMER’), (3, ’FALL’)])
            con.commit()
            print(’Done.’)
            print(’Filling labs...’)
            cur.executemany(’INSERT OR REPLACE INTO lab_names\
                        (id, type, num, max_grade) VALUES (?, ?, ?, ?)’, lab_names)
            con.commit()
            print(’Done.’)
            print(’Vacuuming...’)

            cur.execute(’VACUUM;’)
            con.commit()
```

```

            print('Done.')
            print('Creation of GRADES DB finished.')

            return True


#    Imports list of students from file in format: 'id % lname, fname' into Grades DB.
#    Should be called before first grading.
#    :param db_name: db that contains grades and student info
#    :param year: grading (current) year
#    :param semester: grading (current) semester
#    :param filename: file that contains student list
#    :return: nothing
#

```

Here is the caller graph for this function:


db_init.grades_db_create
main.Ui_Create_settings _dialog.create_or_update _settings_db
main.Ui_Create_settings _dialog.bind_functions
main.Ui_Create_settings _dialog.setupUi
main.Ui_manage_labs1.setupUi
main.Ui_Create_dates _dialog1.setupUi

6.3.1.24 import_previous_grades_into_db()

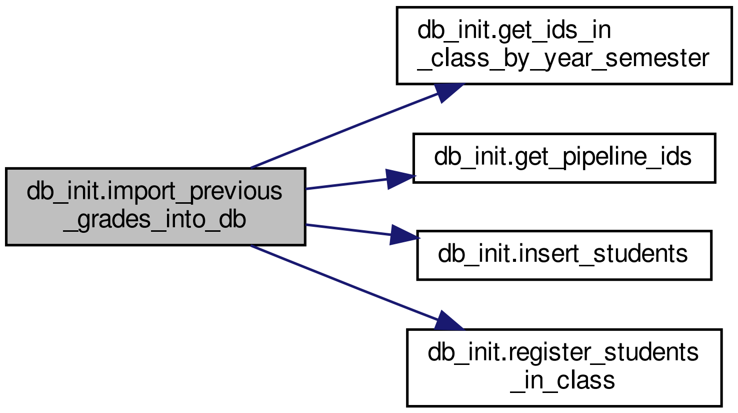
```
def db_init.import_previous_grades_into_db (
              year,
              semester,
              db_name = './grades.sqlite3',
              filename = './grades.xls' )
```

Definition at line 333 of file db_init.py.

```
def import_previous_grades_into_db(year, semester,
    db_name='./grades.sqlite3', filename='./grades.xls'):
    if not os.path.isfile(db_name):
        raise Exception("DB not found")

    df1 = pd.read_excel(filename)

    try:
        cls = df1.filter(like='CL')
    except Exception as e:
        print(e)
        cls = None  # no CLA's found

    try:
        ols = df1.filter(like='OL')
    except Exception as e:
        print(e)
        ols = None  # no OLAs found

    try:
        ids = df1.filter(like='sername').values.ravel().tolist()
```

```
      ids_len = len(ids)
  except Exception as e:
      print('Was not able to parse user ids, check xls file you are trying to import: ', e)
      raise e  # may be improved in the future - strange case
  try:
      names = df1.filter(like='Name').values.ravel().tolist()
  except Exception as e:  # either does not exist or has different name
      print(e)
      names = None

  class_dict = get_ids_in_class_by_year_semester(year, semester, db_name
)

  if (not class_dict and not names) or (class_dict and len(class_dict) < ids_len and not names):
      raise Exception('Did not find ids in table CLASS and did not find names in xls file')
  elif names and (not class_dict or (class_dict and len(class_dict) < ids_len)):
      print('Did not find existing students, but found names in xsl\nAdding new students...\n')
      existing_ids = get_pipeline_ids(db_name)
      need_to_update_students = False
      # otherwise just add ids to the class list
      if existing_ids:
          for sid in ids:
              if sid not in existing_ids:
                  need_to_update_students = True
      else:
          need_to_update_students = True

      if need_to_update_students:
          fname, lname = zip(*(name.split(', ') for name in names))
          fname = (name.strip() for name in fname)
          lname = (name.strip() for name in lname)
          insert_students(ids, fname, lname, db_name)
      register_students_in_class(ids, year, semester, db_name)

  class_ids = [class_dict[sid] for sid in ids]
  if ols is None and cls is None or len(class_ids) == 0:
      raise Exception('No grades to load')

  grades_tupples = list()
  if ols is not None:
      for lab_name in ols:
          grades = (str(grade) for grade in ols[lab_name].values)
          grades_tupples += list(zip(class_ids, [lab_name] * ids_len, [-1] * ids_len, grades, ['TRUE'] *
ids_len))

  if cls is not None:
      for lab_name in cls:
          grades = (str(grade) for grade in cls[lab_name].values)
          grades_tupples += list(zip(class_ids, [lab_name] * ids_len, [-1] * ids_len, grades, ['TRUE'] *
ids_len))

  with lite.connect(db_name) as con:
      cur = con.cursor()
      cur.executemany('INSERT OR REPLACE INTO grades\
                  (class_id, lab, attempt, grade, pass_fail) VALUES (?, ?, ?, ?, ?)', grades_tupples)
      con.commit()


```

Here is the call graph for this function:



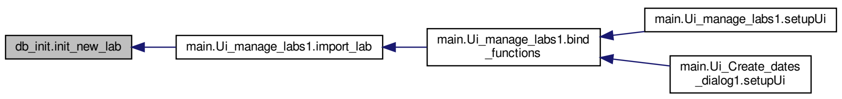
6.3.1.25 init_new_lab()

```
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.

```
def init_new_lab(stud_id, lab_name, att, submitted, lab_path, db_name='./grades.sqlite3'):
    if not os.path.isfile(db_name):
        raise Exception("DB not found")
    with lite.connect(db_name) as con:
        cur = con.cursor()
        cur.execute('INSERT INTO grades (class_id, lab, attempt, submitted, lab_path) VALUES (?, ?, ?, ?,
    ?)', (stud_id, lab_name, att, submitted, lab_path))
        con.commit()


#    :param db_name:
#    :return:
#


```

Here is the caller graph for this function:


db_init.init_new_lab
main.Ui_manage_labs1.import_lab
main.Ui_manage_labs1.bind_functions
main.Ui_manage_labs1.setupUi
main.Ui_Create_dates_dialog1.setupUi

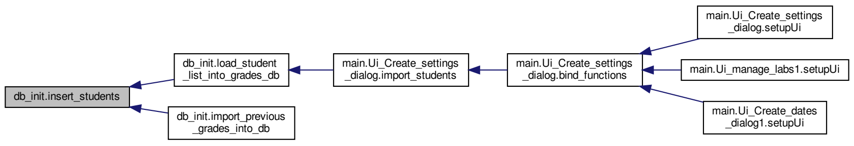
6.3.1.26 insert_students()

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

Definition at line 257 of file db_init.py.

```
def insert_students(ids, fname, lname, db_name='./grades.sqlite3'):
    names_tupple = list(zip(ids, fname, lname, [0] * len(ids)))
    with lite.connect(db_name) as con:
        cur = con.cursor()
        cur.executemany('INSERT OR REPLACE INTO STUDENTS \
                    (pipeline_id, first_name, second_name, cheating_ratio)'
                        ' VALUES (?, ?, ?, ?)', names_tupple)
        con.commit()


#     :param pipeline_ids:
#     :param year:
#     :param semester:
#     :param db_name:
#     :return:
#


```

Here is the caller graph for this function:


db_init.insert_students
db_init.load_student_list_into_grades_db
db_init.import_previous_grades_into_db
main.Ui_Create_settings_dialog.import_students
main.Ui_Create_settings_dialog.bind_functions
main.Ui_Create_settings_dialog.setupUi
main.Ui_manage_labs1.setupUi
main.Ui_Create_dates_dialog1.setupUi

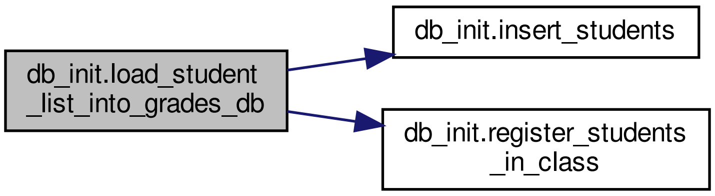
6.3.1.27 load_student_list_into_grades_db()

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

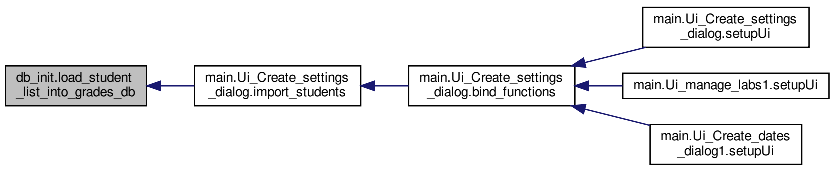
Definition at line 234 of file db_init.py.

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

    with open(filename, 'r') as sl:        ids, names = zip(*(line.strip().split('%') for line in sl))
        ids = list(sid.strip() for sid in ids)
        names = (name.strip() for name in names)  # for case when file contains extra whitespaces
        lname, fname = zip(*(namer.split(',') for namer in names))
        lname = (name.strip() for name in lname)
        fname = (name.strip() for name in fname)

    if os.path.isfile(db_name):
        insert_students(ids, fname, lname, db_name)
        register_students_in_class(ids, year, semester, db_name)


#    Takes students' info from the parameters and insert them into grades DB
#    :param ids: pipeline ids
#    :param fname: first name
#    :param lname: last name
#    :param db_name: specific name for grades DB
#    :return: nothing
#


```

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.

```
def reconstruct_grades_and_comments(db_name='./grades.sqlite3'):
    lab_id, lab_path = get_empty_grades(db_name)
    updated_grades = list()
    for l_iter in range(len(lab_path)):
        lpath = lab_path[l_iter]
        submition_t = int(lpath.split('-')[-1])
        try:
            with open(lpath+'/grade.txt', 'r') as gfile:
                cur_grade = int(gfile.readline().strip())
        except Exception as e:
            print("Error during grade file reading :", e)
            cur_grade = 0
        try:
            cur_t_graded = int(os.path.getmtime(lpath + '/grade.txt'))
        except Exception as e:
            print("Error during grade file statistics retrieval: ", e)
            cur_t_graded = None

        pass_fail = 'TRUE' if cur_grade else 'FALSE'
        try:
            with open(lpath+'/responce.txt', 'r') as rfile:
                cur_resp = rfile.readlines()
                if type(cur_resp) == list:
                    cur_resp = ' '.join(cur_resp)
        except Exception as e:
            print("Error during grade file reading", e)
            cur_resp = 'NULL'
        updated_grades.append((submition_t, cur_grade, cur_t_graded, pass_fail, cur_resp, lab_id[l_iter]))


    with lite.connect(db_name) as con:
        cur = con.cursor()
        cur.executemany('UPDATE grades SET submitted=?, grade=?, graded=?, pass_fail=?, grader_comment=? '
                        'WHERE id=?', updated_grades)
        con.commit()

    with lite.connect(db_name) as con:
        cur = con.cursor()
        cur.execute('VACUUM;')
        con.commit()


```

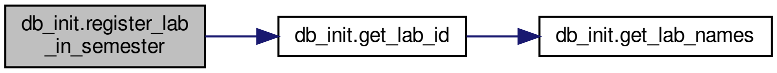
6.3.1.29 register_lab_in_semester()

```
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:



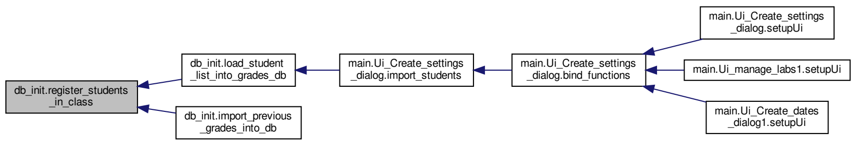
6.3.1.30 register_students_in_class()

```
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:


db_init.register_students_in_class
db_init.load_student_list_into_grades_db
db_init.import_previous_grades_into_db
main.Ui_Create_settings_dialog.import_students
main.Ui_Create_settings_dialog.bind_functions
main.Ui_Create_settings_dialog.setupUi
main.Ui_manage_labs1.setupUi
main.Ui_Create_dates_dialog1.setupUi

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
```

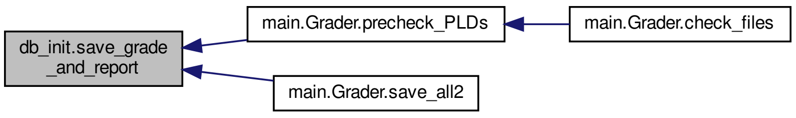
6.3.1.32 save_grade_and_report()

```
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.

```
def save_grade_and_report(grade_id, grade, report, user_comment, grader,
  db_name='./grades.sqlite3'):
    if not os.path.isfile(db_name):
        raise Exception("DB not found")
    with lite.connect(db_name) as con:
        cur = con.cursor()
        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))
        con.commit()


```

Here is the caller graph for this function:



6.3.1.33 settings_db_create()

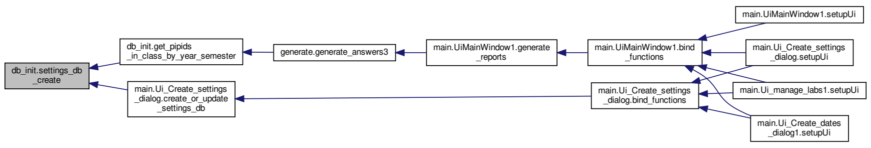
```
def db_init.settings_db_create (
                db_name = SETTINGS_DB_NAME,
                force = False )
```

Definition at line 18 of file db_init.py.

```
def settings_db_create(db_name=SETTINGS_DB_NAME, force=False):
    if not force and os.path.isfile(db_name):
        user_choice = input('Do you really want to drop database ? Type "yes" to continue\n ')
        if not user_choice.isalpha() or not user_choice.lower() == 'yes':
            return False

    # DB creation logic goes here
    with lite.connect(db_name) as con:
        cur = con.cursor()
        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:


db_init.settings_db _create
db_init.get_pipids _in_class_by_year_semester
main.Ui_Create_settings _dialog.create_or_update _settings_db
generate.generate_answers3
main.UiMainWindow1.generate _reports
main.UiMainWindow1.bind _functions
main.Ui_Create_settings _dialog.bind_functions
main.UiMainWindow1.setupUi
main.Ui_Create_settings _dialog.setupUi
main.Ui_manage_labs1.setupUi
main.Ui_Create_dates _dialog1.setupUi

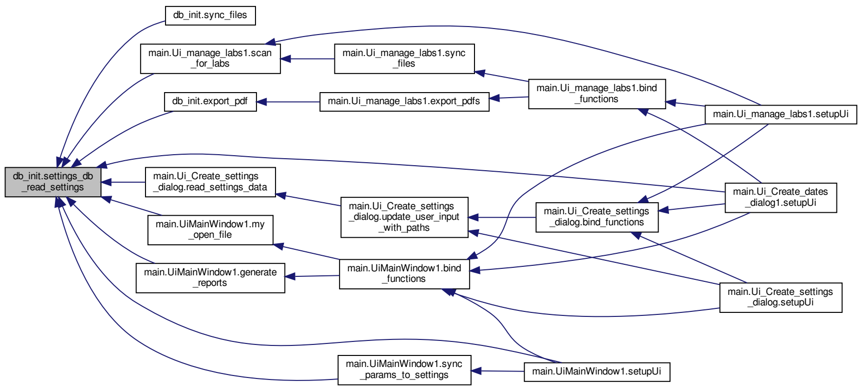
6.3.1.34 settings_db_read_settings()

```
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:


db_init.settings_db_read_settings
db_init.sync_files
main.Ui_manage_labs1.scan_for_labs
main.Ui_manage_labs1.sync_files
main.Ui_manage_labs1.bind_functions
main.Ui_manage_labs1.setupUi
db_init.export_pdf
main.Ui_manage_labs1.export_pdfs
main.Ui_Create_settings_dialog.read_settings_data
main.Ui_Create_settings_dialog.update_user_input_with_paths
main.Ui_Create_settings_dialog.bind_functions
main.Ui_Create_dates_dialog1.setupUi
main.UiMainWindow1.my_open_file
main.UiMainWindow1.generate_reports
main.UiMainWindow1.bind_functions
main.Ui_Create_settings_dialog.setupUi
main.UiMainWindow1.sync_params_to_settings
main.UiMainWindow1.setupUi

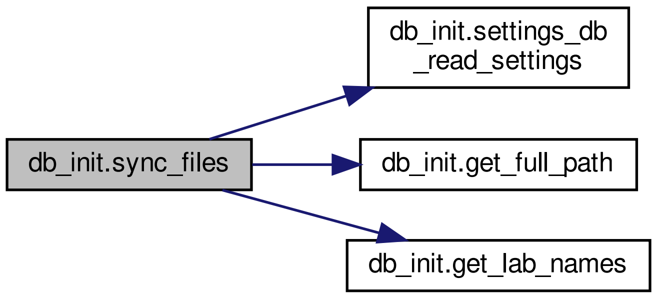
6.3.1.35 sync_files()

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

Definition at line 677 of file db_init.py.

```
def sync_files(self=None):
    import subprocess
    import os

    paths, local = settings_db_read_settings()
    full_path = get_full_path(paths, local) + "/server_sync/"
    lab_ids, lab_types, lab_nums = get_lab_names()
    lab_names = []
    for i in range(len(lab_types)):
        lab_names.append(lab_types[i] + '_Lab_' + str(lab_nums[i]))

    if not os.path.isdir(full_path):
        os.makedirs(full_path)
        for lab_name in lab_names:
            os.makedirs(full_path + lab_name)

    proc_arr = []
    for lab_name in lab_names:
        command = local[4] + ' ' + os.path.expanduser(paths[2] + lab_name) + '/*.zip' + ' ' + full_path +
      lab_name + '/'
        try:
            proc_arr.append(subprocess.Popen(os.path.expandvars(command), stdout=subprocess.PIPE, shell=
      True))
            proc_arr[-1].communicate()
        except Exception as e:
            print('Error in rsync: ', e)
        # output, error = process.communicate()
        # print(output)
        # print(error)

    for proc_elem in proc_arr:
        proc_elem.wait()


```

Here is the call graph for this function:



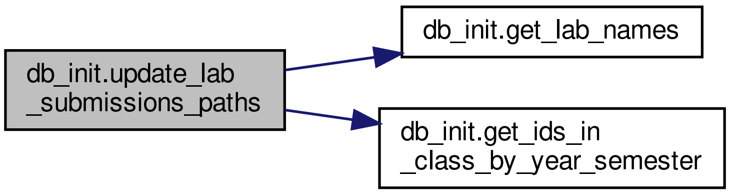
6.3.1.36 update_lab_submissions_paths()

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

Definition at line 499 of file db_init.py.

```
def update_lab_submissions_paths(db_name, repository_root, year, semester):
    import fnmatch
    import glob
    # import_previous_grades_into_db(year, semester, db_name, repository_root+'grades.xlsx')
    lab_id, lab_type, lab_num = get_lab_names()
    if lab_id is None or lab_type is None or lab_num is None:
        raise Exception("Error during lab type/num import: ")
    class_dict = get_ids_in_class_by_year_semester(year, semester, db_name
  )
    total_labs = len(lab_type)

    all_dirs = list()
    for lab_iter in range(total_labs):
        for attempt in range(1, 5):  # class rule - 4 attempts
            full_lab_name = repository_root + lab_type[lab_iter] + '_Lab_' + str(lab_num[lab_iter]) + '_' +
  str(attempt) + '/'
            print('Processing ', full_lab_name)
            for stud_id in class_dict.keys():
                found_dir = glob.glob(full_lab_name+stud_id+'*')
                if found_dir:
                    # since it is initial pass, we do not set pass/fail. It will be set later with grade
  and comment.
                    all_dirs.append((class_dict[stud_id], lab_id[lab_iter], attempt, 'FALSE', found_dir[-1]
  ))

    with lite.connect(db_name) as con:
        cur = con.cursor()
        cur.executemany('INSERT OR REPLACE INTO grades (class_id, lab, attempt, pass_fail, lab_path)'
                        ' VALUES (?, ?, ?, ?, ?)', all_dirs)
        con.commit()


```

Here is the call graph for this function:



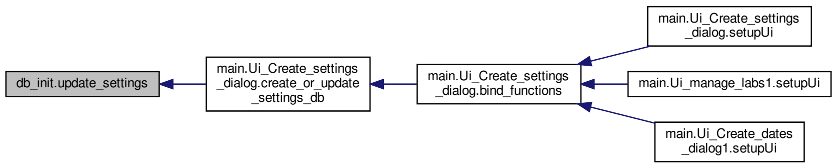
6.3.1.37 update_settings()

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

Definition at line 70 of file db_init.py.

```
def update_settings(paths, local, db_name=SETTINGS_DB_NAME):
    if os.path.isfile(db_name):
        with lite.connect(db_name) as con:
            cur = con.cursor()
            cur.execute('DELETE FROM PATHS;')
            cur.execute('INSERT OR REPLACE INTO PATHS (LOGISIM_HOME, GRADING_PATH, IMPORT_PATH, GRADES_DB)'
                        ' VALUES (?, ?, ?, ?);', paths)
            cur.execute('DELETE FROM LOCAL;')
            cur.execute('INSERT OR REPLACE INTO LOCAL (GRADER_NAME, YEAR, SEMESTER, USE_STYLE,
    SYNC_COMMAND)'
                        'VALUES (?, ?, ?, ?, ?);', local)
            con.commit()

        with lite.connect(db_name) as con:
            cur = con.cursor()
            cur.execute('VACUUM;')
            con.commit()


#    Will create database that contains all information about grades
#    :param db_name: path and name of the database
#    :param force: flag to overwrite existing db
#    :return: Unknown
#

```

Here is the caller graph for this function:


db_init.update_settings
main.Ui_Create_settings _dialog.create_or_update _settings_db
main.Ui_Create_settings _dialog.bind_functions
main.Ui_Create_settings _dialog.setupUi
main.Ui_manage_labs1.setupUi
main.Ui_Create_dates _dialog1.setupUi

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

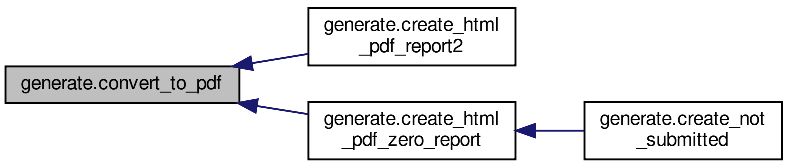
6.4.1.1 convert_to_pdf()

```
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 == "pdfkit":  # best margins
24         import pdfkit
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         pdfkit.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 #     Creates nice html report for submitted labs and converts it to pdf format.
47 #     TODO: use latex instead of ugly html.
48 #     :param joined_path: working directory
49 #     :param stud_name: full student name(first, last)
50 #     :param cur_dir: directory with all labs(usually same as joined_path)
51 #     :param grade: what grade to assign.
52 #     :param max_points: max possible grade for this lab.
53 #     :param penalty: usually for resubmission, like 90%, 70%...
54 #     :param final_score: final grade = grade * penalty
55 #     :param top_part: predefined top part of html document
56 #     :param bot_part: predefined bottom part of html document
57 #     :param generated_time: some extra statistics for curious students.
58 #     :return: nothing, pdf is generated instead.
59 #     """
60 #     with open(joined_path + '-returned.html', 'w') as stud_report:
61 #         stud_report.writelines(top_part)
62 #         stud_report.write('<p>Grading directory : ' + cur_dir + ' </br>')
63 #
64 #         with open(joined_path + '/tech_info.txt', 'r') as tech_file:
65 #             stud_report.writelines(tech_file.readlines())
66 #
67 #         stud_report.write('</p><p><i>Dear ' + stud_name + ', ')
68 #
69 #         with open(joined_path + '/responce.txt', 'r') as resp_file:
70 #             stud_report.writelines(resp_file.readlines())
71 #
72 #         stud_report.write("</i></p>\n"
73 #                           "<p>According to the comment above, next grade was assigned: "
74 #                           "%d of %d <br/>\n \
75 #                           Your final grade is %d*%.1f=<b>%d</b> of %d <br/>\n"
76 #                           % (grade, max_points, grade, penalty, final_score, max_points))
77 #         stud_report.write('This report was generated {} </p>'.format(generated_time))
78 #         # TODO add current date/time
79 #         stud_report.writelines(bot_part)
80 #
81 #     convert_to_pdf(joined_path + '-returned.html', "pdfkit")
82 #     os.remove(joined_path + '-returned.html')
83 #
84 
```

Here is the caller graph for this function:


generate.convert_to_pdf
generate.create_html_pdf_report2
generate.create_html_pdf_zero_report
generate.create_not_submitted

6.4.1.2 create_html_pdf_report2()

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

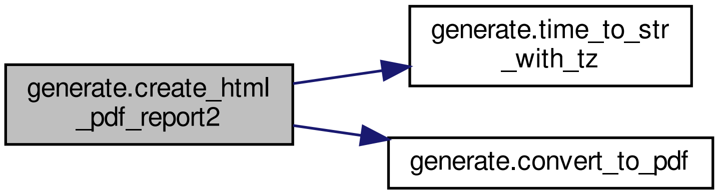
Creates nice html report for submitted labs and converts it to pdf format.

:return: nothing, pdf is generated instead.

Definition at line 90 of file generate.py.

```
def create_html_pdf_report2(lab_dict):
    with open('./answer.top', 'r') as partial_html:
        top_part = partial_html.readlines()

    with open('./answer.bottom', 'r') as partial_html:
        bot_part = partial_html.readlines()

    with open(lab_dict['lab_path'] + '-returned.html', 'w') as stud_report:
        stud_report.writelines(top_part)

        stud_report.write('<p>Grading directory : {} </br>'.format(lab_dict['lab_path'].split('/')[-1]))
        stud_report.write('Due date was {} <br/>'.format(time_to_str_with_tz(lab_dict['
due_date'])))
        stud_report.write('File was submited at {} <br/>'.format(
time_to_str_with_tz(lab_dict['submitted'])))
        stud_report.write('I imported your file at {} <br/>'.format(
time_to_str_with_tz(lab_dict['import_date'])))
        if lab_dict['graded'] is not None:
            stud_report.write('I graded your lab at {} <br/>'.format(
time_to_str_with_tz(lab_dict['graded'])))
        else:
            stud_report.write('I did not grad your lab or grade timestamp was not set.<br/>')
        stud_report.write('Lab type : \'{}\' and it\'s number is \'{}\' <br/>'.format(lab_dict['type'],
lab_dict['num']))
        stud_report.write('</p><p><i>Dear {} {}, '.format(lab_dict['first_name'], lab_dict['second_name']))
        if lab_dict['grader_comment'] is None or len(lab_dict['grader_comment']) < 2:
            stud_report.write('There were no comments.')
        else:
            stud_report.write(lab_dict['grader_comment'])
        if lab_dict['extra_comment'] is not None and len(lab_dict['extra_comment']) > 0:
            stud_report.write('<br/>\nExtra comment: {}'.format(lab_dict['extra_comment']))

        stud_report.write("</i></p>\n"
                          "<p>According to the comment above, next grade was assigned: {} of {} <br/>\n"
                          " Your final grade is computed as {}*{:.1f}=<b>{}</b> of {} <br/>\n"
                          "".format(lab_dict['final_grade'], lab_dict['max_grade'], lab_dict['grade'],
  lab_dict['percent']/100, lab_dict['final_grade'], lab_dict['max_grade']))
        if lab_dict['grade'] == 0:
            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
        stud_report.write('This report was generated {} </p>\n'.format(QDateTime.currentDateTime().toString
  (Qt.DefaultLocaleLongDate)))

        stud_report.writelines(bot_part)

    convert_to_pdf(lab_dict['lab_path'] + '-returned.html', "pdfkit")
    os.remove(lab_dict['lab_path'] + '-returned.html')


#    Creates nice html report for nonsubmitted labs and converts it to pdf format.
#     :param filename: filename with correct naming(zeroes instead of timestamp)
#     :param stud_name: full student name(first, last)
#     :param top_part: predefined top part of html document
#     :param bot_part: predefined bottom part of html document
#     :return:
#

```

Here is the call graph for this function:


generate.create_html_pdf_report2
generate.time_to_str_with_tz
generate.convert_to_pdf

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.

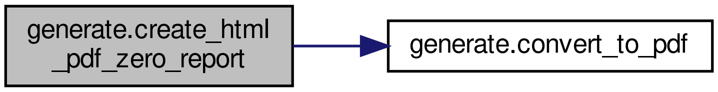
```
139 def create_html_pdf_zero_report(filename, stud_name, top_part, bot_part):
140     with open(filename, 'w') as zeroes_file:
141         zeroes_file.writelines(top_part)
142         zeroes_file.write(stud_name + ' : You did not submit your lab. :(</p>\n')
143         zeroes_file.write("<p>According 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, "pdfkit")
147     os.remove(filename)
```

```


# def generate_answers(resubmit_num, dir_name, lab_type, lab_num, year, semester, grader_name):
#     """
#     general function that figures out max points, filenames, etc
#     and calls generate function with appropriate parameters
#     :param resubmit_num: resubmission attempt
#     :param dir_name: working dir
#     :param lab_type: open or closed lab
#     :param lab_num: just lab identifier
#     :param year: used wit semester to identify correct class list
#     :param semester: used wit year to identify correct class list
#     :param grader_name: name that will be displayed in the report
#     :return:
#     """
#     students = {}
#     # select
#
#     ids = get_pipids_in_class_by_year_semester(year, semester, 'grades.sqlite3')
#     with lite.connect('grades.sqlite3') as con:
#         cur = con.cursor()
#
#         for sid in ids.keys():
#             result = cur.execute('SELECT first_name, second_name FROM students WHERE pipeline_id=?',
    (str(sid),))
#             students[sid] = " ".join(result.fetchall()[0])
#
#     if not students:
#         with open('students_list1.txt', 'r') as stud_list_file:
#             temp_arr = stud_list_file.readlines()
#             for line in temp_arr:
#                 sid, name = line.split('%')
#                 students[sid.strip()] = name.strip()
#         del temp_arr
#
#
#     if lab_type == 'Closed':
#         max_points = 10
#         type_for_name = 'CL'
#     elif lab_type == 'Open':
#         max_points = 20
#         type_for_name = 'OL'
#     else:
#         raise Exception('Unknown lab type')
#
#     if resubmit_num == 1:
#         penalty = 1.0
#     elif resubmit_num == 2:
#         penalty = 0.9
#     elif resubmit_num == 3:
#         penalty = 0.7
#     elif resubmit_num == 4:
#         penalty = 0.5
#     else:
#         penalty = 0.0
#
#     generated_time = QDateTime.currentDateTime().toString(Qt.DefaultLocaleLongDate)
#
#     print('This is ', type_for_name, ' lab, so max points is ', max_points)
#
#     try:
#         shutil.rmtree(dir_name + 'Answers', ignore_errors=True)
#         os.remove(dir_name + "grades.csv")
#         os.remove(dir_name + "grades_for_" + type_for_name + "lab_num.csv")
#     except Exception as e:
#         print('Exception during dir preparetion : ', e)
#
#     dirs = os.walk(dir_name).__next__()[1]
#
#     with open('./answer.top', 'r') as partial_html:
#         top_part = partial_html.readlines()
#
#     with open('./answer.bottom', 'r') as partial_html:
#         bot_part = partial_html.readlines()
#
#     grades = list()
#     for cur_dir in dirs:
#         student_id = cur_dir.split('-')[0]
#         joined_path = os.path.join(dir_name, cur_dir)
#         with open(joined_path + '/grade.txt', 'r') as grade_file:
#             grade = grade_file.readlines()
```

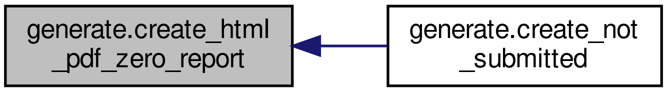
```
#
#         grade = int(grade[0].strip())
#         final_score = grade * penalty
#         grades.append((student_id, final_score))
#         create_html_pdf_report(joined_path, students[student_id], cur_dir, grade,
#                                max_points, penalty, final_score, top_part, bot_part, generated_time)
#
#     submitted = [x.split('-')[0] for x in dirs]
#
#     zeroes = list()
#     for student in students:
#         if student not in submitted:
#             grades.append((student, 0))
#             zeroes.append(student)
#
#     if resubmit_num == 1:
#         for student_id in zeroes:
#             filename = '%s/%s-%s%d-0000000000-returned' % \
#                        (dir_name, student_id, type_for_name, lab_num)
#             create_html_pdf_zero_report(filename+'.html', students[student_id], top_part, bot_part)
#
#     with open(dir_name + '/' + 'grades.csv', 'w') as grades_file:
#         for grade in sorted(grades):
#             grades_file.write("%s, %f \n" % grade)
#
#     os.mkdir(dir_name + '/Answers')
#     files = os.walk(dir_name).__next__()[2]
#     for file in files:
#         if file[-3:] == 'pdf':
#             shutil.move(dir_name + '/' + file, dir_name + '/Answers/' + file)
#
#     print('Done')
#
#
# def generate_answers2(resubmit_num, dir_name, lab_type, lab_num, year, semester, grader_name):
#     """
#     general function that figures out max points, filenames, etc
#     and calls generate function with appropriate parameters
#     :param resubmit_num: resubmission attempt
#     :param dir_name: working dir
#     :param lab_type: open or closed lab
#     :param lab_num: just lab identifier
#     :param year: used wit semester to identify correct class list
#     :param semester: used wit year to identify correct class list
#     :param grader_name: name that will be displayed in the report
#     :return:
#     """
#     students = {}
#     # select
#     import sqlite3 as lite
#     from db_init import get_ids_in_class_by_year_semester
#     ids = get_ids_in_class_by_year_semester(year, semester, 'grades.sqlite3')
#     with lite.connect('grades.sqlite3') as con:
#         cur = con.cursor()
#
#         for sid in ids.keys():
#             result = cur.execute('SELECT first_name, second_name FROM students WHERE pipeline_id=?',
  (str(sid),))
#             students[sid] = " ".join(result.fetchall()[0])
#
#     if not students:
#         with open('students_list1.txt', 'r') as stud_list_file:
#             temp_arr = stud_list_file.readlines()
#             for line in temp_arr:
#                 sid, name = line.split('%')
#                 students[sid.strip()] = name.strip()
#         del temp_arr
#
#
#     if lab_type == 'Closed':
#         max_points = 10
#         type_for_name = 'CL'
#     elif lab_type == 'Open':
#         max_points = 20
#         type_for_name = 'OL'
#     else:
#         raise Exception('Unknown lab type')
#
#     if resubmit_num == 1:
#         penalty = 1.0
#     elif resubmit_num == 2:
```

```
#         penalty = 0.9
#     elif resubmit_num == 3:
#         penalty = 0.7
#     elif resubmit_num == 4:
#         penalty = 0.5
#     else:
#         penalty = 0.0
#
#     generated_time = QDateTime.currentDateTime().toString(Qt.DefaultLocaleLongDate)
#
#     print('This is ', type_for_name, ' lab, so max points is ', max_points)
#
#     try:
#         shutil.rmtree(dir_name + 'Answers', ignore_errors=True)
#         os.remove(dir_name + "grades.csv")
#         os.remove(dir_name + "grades_for_" + type_for_name + "lab_num.csv")
#     except Exception as e:
#         print('Exception during dir preparetion : ', e)
#
#     dirs = os.walk(dir_name).__next__()[1]
#
#     with open('./answer.top', 'r') as partial_html:
#         top_part = partial_html.readlines()
#
#     with open('./answer.bottom', 'r') as partial_html:
#         bot_part = partial_html.readlines()
#
#     grades = list()
#     for cur_dir in dirs:
#         student_id = cur_dir.split('-')[0]
#         joined_path = os.path.join(dir_name, cur_dir)
#         with open(joined_path + '/grade.txt', 'r') as grade_file:
#             grade = grade_file.readlines()
#
#         grade = int(grade[0].strip())
#         final_score = grade * penalty
#         grades.append((student_id, final_score))
#         create_html_pdf_report(joined_path, students[student_id], cur_dir, grade,
#                                max_points, penalty, final_score, top_part, bot_part, generated_time)
#
#     submitted = [x.split('-')[0] for x in dirs]
#
#     zeroes = list()
#     for student in students:
#         if student not in submitted:
#             grades.append((student, 0))
#             zeroes.append(student)
#
#     if resubmit_num == 1:
#         for student_id in zeroes:
#             filename = '%s/%s-%s%d-0000000000-returned' % \
#                        (dir_name, student_id, type_for_name, lab_num)
#             create_html_pdf_zero_report(filename+'.html', students[student_id], top_part, bot_part)
#
#     with open(dir_name + '/' + 'grades.csv', 'w') as grades_file:
#         for grade in sorted(grades):
#             grades_file.write("%s, %f \n" % grade)
#
#     os.mkdir(dir_name + '/Answers')
#     files = os.walk(dir_name).__next__()[2]
#     for file in files:
#         if file[-3:] == 'pdf':
#             shutil.move(dir_name + '/' + file, dir_name + '/Answers/' + file)
#
#     print('Done')

#
# def create_a_report(lab_dict):
#
#     create_html_pdf_report2( lab_dict)


```

Here is the call graph for this function:



Here is the caller graph for this function:


generate.create_html_pdf_zero_report
generate.create_not_submitted

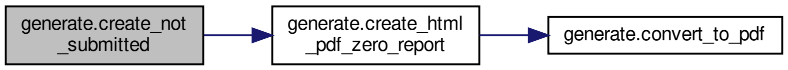
6.4.1.4 create_not_submitted()

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

Definition at line 380 of file generate.py.

```
def create_not_submitted(stud_id, lab_type, lab_num, dir_name):
    with open('./answer.top', 'r') as partial_html:
        top_part = partial_html.readlines()

    with open('./answer.bottom', 'r') as partial_html:
        bot_part = partial_html.readlines()
    filename = '%s/%s-%s%d-0000000000-returned' % \
               (dir_name, stud_id, lab_type, lab_num)
    create_html_pdf_zero_report(filename + '.html', stud_id, top_part, bot_part)


```

Here is the call graph for this function:


generate.create_not_submitted
generate.create_html_pdf_zero_report
generate.convert_to_pdf

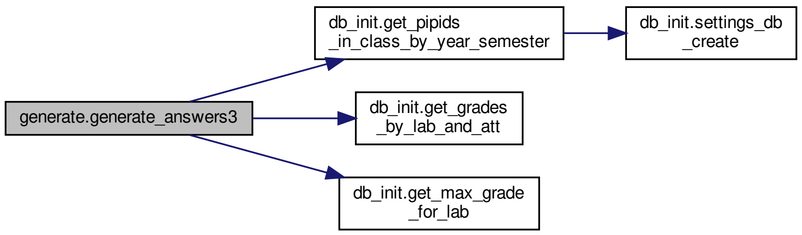
6.4.1.5 generate_answers3()

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

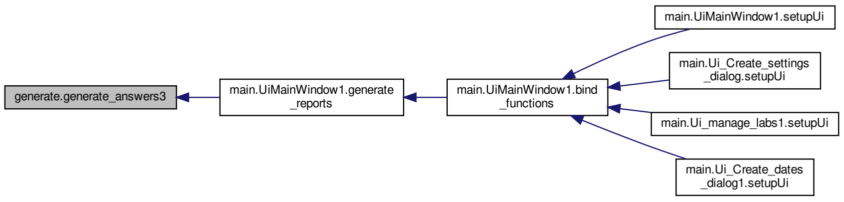
Definition at line 391 of file generate.py.

```
def generate_answers3(lid, att, year, semester, db_name='./grades.sqlite3'):
    all_ids = get_pipids_in_class_by_year_semester(year, semester)
    info_tup, info_desc = get_grades_by_lab_and_att(lid, att)
    col_names = [elem[0] for elem in info_desc]
    main_list = list()
    for tup in info_tup:
        a = dict()
        for i, elem in enumerate(tup):
            a[col_names[i]] = elem
        main_list.append(a)
    graded_students = [elem['pipeline_id'] for elem in main_list]
    grades = [elem['final_grade'] for elem in main_list]
    grade_dict = dict(zip(graded_students, grades))
    lab_type = main_list[0]['type']
    lab_num = main_list[0]['num']
    dir_name = main_list[0]['lab_path']
    dir_name = dir_name[:dir_name.rfind('/')]
    corrctd_lab_type = 'CL' if lab_type == 'Closed' else 'OL'

    # for elem in main_list:
    #     create_a_report(elem)
    #
    # for elem in main_list:
    #     commit_gen_report(elem['grade_id'])

    not_subm_ids = [stud_id for stud_id in all_ids if stud_id not in graded_students]

    if len(main_list) + len(not_subm_ids) == 0:
        return

    ans_dir = os.path.join(dir_name, 'Answers')
    if os.path.exists(ans_dir):
        shutil.rmtree(ans_dir, ignore_errors=True)
    gr_file = os.path.join(dir_name, 'grades.csv')
    if os.path.exists(gr_file):
        os.remove(gr_file)
    gr_long_file = os.path.join(dir_name, "grades_for_{}lab_num.csv".format(corrctd_lab_type))
    if os.path.exists(gr_long_file):
        os.remove(gr_long_file)
    files_to_rem = (os.path.join(dir_name, file) for file in (el for el in os.walk(dir_name).__next__()[2]
      if el[-3:] in ['pdf', 'html']))

    with mp.Pool() as pool:
        pool.map(os.remove, files_to_rem)
        r1 = pool.map_async(create_html_pdf_report2, main_list)
        r2 = pool.map_async(commit_gen_report, (elem['grade_id'] for elem in main_list))
        if att == 1:
            pool.starmap(create_not_submitted, ((stud_id, correctd_lab_type, lab_num, dir_name) for stud_id
   in not_subm_ids))
        r1.wait()
        r2.wait()

    with open(os.path.join(dir_name, '{}_lab_{}_grades.csv'.format(lab_num, lab_type)), 'w') as grades_file
   :
        grades_file.write("{1} Lab {0}, {1} Lab {0}\n".format(lab_num, lab_type))
        for stud_id in all_ids:
            if stud_id not in not_subm_ids:
                grades_file.write("{:s}, {:d}\n".format(stud_id, int(grade_dict[stud_id])))
            else:
                grades_file.write("{:s}, {:d}\n".format(stud_id, 0))


    best_grade_list = get_max_grade_for_lab(lid, year, semester)
    with open(os.path.join(dir_name, '{}_lab_{}_grades_best_so_far.csv'.format(lab_num, lab_type)), 'w') as
   grades_file:
        grades_file.write("{1} Lab {0}, {1} Lab {0}\n".format(lab_num, lab_type))
        for stud_tup in best_grade_list:
            grades_file.write('{}, {}\n'.format(stud_tup[0], stud_tup[1]))

    # for elem in main_list:
    #     create_html_pdf_report2(elem)
    # for elem in main_list:
    #     commit_gen_report(elem['grade_id'])

    # if att == 1:  # we do not form report for second attempt since most people are happy with previous
   grade
        # for stud_id in not_subm_ids:
        #     create_not_submitted(stud_id, correctd_lab_type, lab_num, dir_name)

    os.mkdir(os.path.join(dir_name, 'Answers'))
    files = os.walk(dir_name).__next__()[2]
    for file in files:
        if file[-3:] == 'pdf':
            shutil.move(os.path.join(dir_name, file), os.path.join(dir_name, 'Answers/{}'.format(file)))

    print('Done')


```

Here is the call graph for this function:


generate.generate_answers3
db_init.get_pipids_in_class_by_year_semester
db_init.settings_db_create
db_init.get_grades_by_lab_and_att
db_init.get_max_grade_for_lab

Here is the caller graph for this function:


generate.generate_answers3
main.UiMainWindow1.generate_reports
main.UiMainWindow1.bind_functions
main.UiMainWindow1.setupUi
main.Ui_Create_settings_dialog.setupUi
main.Ui_manage_labs1.setupUi
main.Ui_Create_dates_dialog1.setupUi

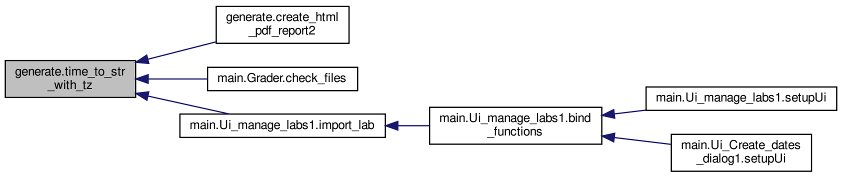
6.4.1.6 time_to_str_with_tz()

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

Definition at line 474 of file generate.py.

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

```

Here is the caller graph for this function:


generate.time_to_str_with_tz
generate.create_html_pdf_report2
main.Grader.check_files
main.Ui_manage_labs1.import_lab
main.Ui_manage_labs1.bind_functions
main.Ui_manage_labs1.setupUi
main.Ui_Create_dates_dialog1.setupUi

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

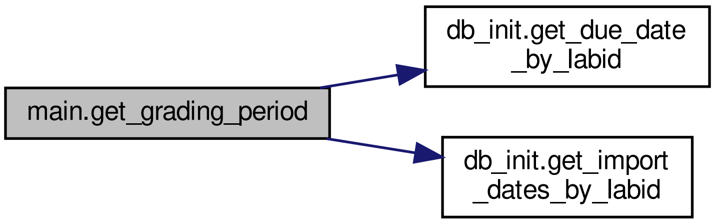
6.5.1.1 get_grading_period()

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

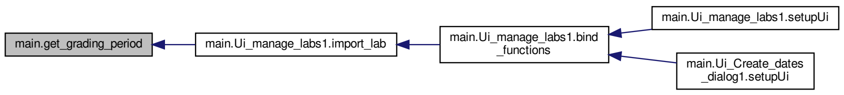
Definition at line 1874 of file main.py.

```
def get_grading_period(lid, cur_only=False):
    # should comput correct grading period and return the due date in Unix timestamp format
    import time
    # due_timestamps = [int(f.split('_')[2]) for f in due_files]

    current_timestamp = int(time.time())
    due_timestamps1 = get_due_date_by_labid(lid)
    import_timestamps1 = get_import_dates_by_labid(lid)
    cur_check = len(due_timestamps1)
    for i, ts in enumerate(import_timestamps1):
        if ts is None:
            cur_check = i
            break
    i = 0
    if cur_check:
        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]:
            i += 1

    if cur_only:  # neede for CLA2-2
        i = max(0, i-1)

    if i == 0:
        from_time = 0
        to_time = due_timestamps1[i]
    elif i > len(due_timestamps1)-1:
        from_time = due_timestamps1[i-1]
        to_time = int(time.time())
    else:
        from_time = due_timestamps1[i - 1]
        to_time = due_timestamps1[i]

    cur_check_num = i+1
    # cur_check += 1


    #
    # check_files = [int(f.split('_')[2]) for f in os.listdir(dir) if 'check_' in f]
    # if len(check_files) > 0:
    #     if len(check_files) >= 4:
    #         cur_check_num = 0
    #         from_time = 0
    #         to_time = 0
    #     else:
    #         cur_check_num = len(check_files) + 1           # 1 + 1
    #         from_time = due_timestamps[cur_check_num - 2]  # 0 => after first due date
    #         to_time = due_timestamps[cur_check_num - 1]    # 1 => before second due date
    # else:
    #     from_time = 0
    #     to_time = due_timestamps[0]
    #     cur_check_num = 1

    return cur_check_num, from_time, to_time, current_timestamp


```

Here is the call graph for this function:


main.get_grading_period
db_init.get_due_date_by_labid
db_init.get_import_dates_by_labid

Here is the caller graph for this function:


main.get_grading_period
main.Ui_manage_labs1.import_lab
main.Ui_manage_labs1.bind_functions
main.Ui_manage_labs1.setupUi
main.Ui_Create_dates_dialog1.setupUi

6.5.1.2 read_settings()

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

Definition at line 63 of file main.py.

```
def read_settings(db_name = 'settings.sqlite3' ):
    import os.path

    if os.path.exists(db_name):
        with lite.connect(db_name) as con:
            cur = con.cursor()
            try:
                cur.execute('SELECT * FROM PATHS')
                result = cur.fetchone()
                for row in result:
                    print(row)
                logisim_path = result[0][0]
                working_dir = result[0][1]
                # since import is not implemented - ignore import path: import_path = result[0][2]
                return logisim_path, working_dir
            except Exception as e:
                print('Was not able to get results from settings DB: ', e)
    return None


```

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:

```
=  """
/* https://stackoverflow.com/questions/22332106/python-qtgui-qprogressbar-color */
QProgressBar
{
    border: 1px solid grey;
    border-radius: 5px;
    text-align: center;
    font-weight: bold;
}
QProgressBar::chunk
{
    background-color: #d7801a;
    width: 2.15px;
    margin: 0.5px;
}
"""
```

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

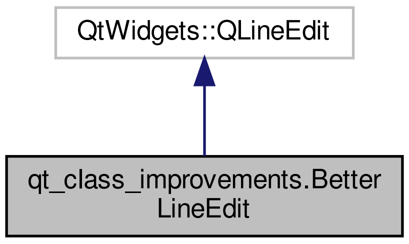
- class Ui_Dialog

Chapter 7

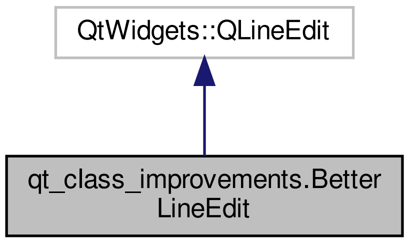
Class Documentation

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.

```
    def __init__(self, *args, **kwargs):
        QtWidgets.QLineEdit.__init__(self, *args, **kwargs)

        self.installEventFilter(self)

```

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.

```
      def eventFilter(self, obj, event):
          if event.type() == QtCore.QEvent.MouseButtonDblClick:
              self.dclicked.emit()
          return False


  #    Overloaded QPlainTextEdit to track focus out.
  #    Needed to implement autosaving of user answer.
  #

```

7.1.4 Member Data Documentation

7.1.4.1 dclicked

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

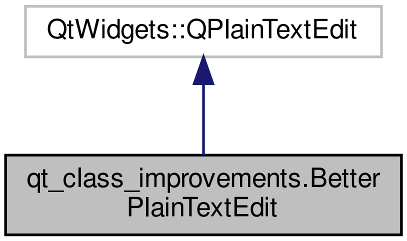
Definition at line 12 of file qt_class_improvements.py.

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

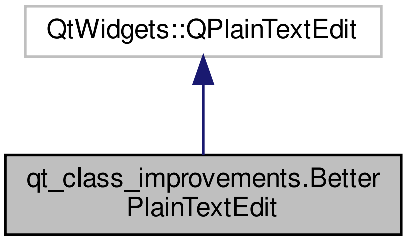
- qt_class_improvements.py

7.2 qt_class_improvements.BetterPlainTextEdit Class Reference

Inheritance diagram for qt_class_improvements.BetterPlainTextEdit:


QtWidgets::QPlainTextEdit
qt_class_improvements.Better PlainTextEdit

Collaboration diagram for qt_class_improvements.BetterPlainTextEdit:


QtWidgets::QPlainTextEdit
qt_class_improvements.Better PlainTextEdit

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.

```
    def __init__(self, *args, **kwargs):
        QtWidgets.QPlainTextEdit.__init__(self, *args, **kwargs)

        self.installEventFilter(self)

```

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.

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

```

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.

```
        def __init__(self, name):
            self.name = name
            self.input_pins = list()
            self.output_pins = list()

    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 )
```

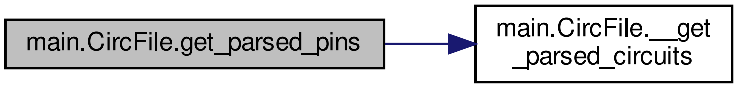
:return:

Definition at line 124 of file main.py.

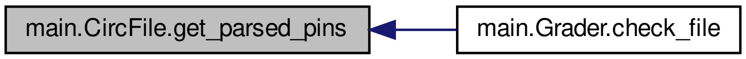
```
124     def get_parsed_pins(self):
125         self.__get_parsed_circuits()
126         arr = self.__all_circuits
127         all_pins = list()
128         for elem in arr:
129             pins = list()
130             for child in elem.findall('comp'):
131                 if child.get('name') == '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()
```

```
        input_pins = list()
        other_pins = list()

        if clean_data:
            for pin in clean_data:
                if pin.type == 'output':
                    output_pins.append(pin)
                elif pin.type == 'input' or pin.type == 'tristate':
                    input_pins.append(pin)
                else:
                    other_pins.append(pin)
        else:
            raise Exception('Error in pin parsing(clean data)')

        return input_pins, output_pins, other_pins


```

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.

```
    def get_parsed_pins2(self, what_to_grade):

        tree = ET.parse(self.filename)
        root = tree.getroot()
        arr=list()
        for child in root:
            # print(child.tag)
            if child.tag == 'circuit':
                arr.append(child)
            # if child.attrib["name"] == what_to_grade:
            #     a = child
            #     b =1

        all_circs = list()
        good_arr = list()
        for node in arr:
            if node.get('name').upper() in what_to_grade:
                good_arr.append(node)
                circ_instance = self.circ_type(node.get('name'))
                all_circs.append(circ_instance)
                # print(list(node)[0].items()[0][1])

        all_pins = list()
        for elem in good_arr:
            pins = list()
            for child in elem.findall('comp'):
                if child.get('name') == 'Pin':
                    pins.append(child)
                    # print(child.tag, child.attrib)
            all_pins.append(pins)


        clean_all_pins = list()
        for pins in all_pins:
            clean_data = list()
            for pin in pins:
                name = '0'
                type = '0'
                for elem in list(pin):
                    if elem.get('name') in ['output', 'input', 'tristate']:
                        type = elem.get('name')
                    elif elem.get('name') == 'label':
                        name = elem.get('val')
                clean_data.append(self.PinType(name, type))
            clean_all_pins.append(clean_data)
        for i in range(len(clean_all_pins)):
            for pin in clean_all_pins[i]:
                if pin.type == 'output':
                    all_circs[i].output_pins.append(pin.name)
                else:
                    all_circs[i].input_pins.append(pin.name)
        return all_circs


```

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.

```
    def __init__(self, working_directory, grader='Ivan'):
        self.__from_date = 0
        self.to_date = 0
        self.attempt = 0
        self.timestamps = list()
        self.stud_ids = list()
        self.stud_id = ''
        self.submitted = 0
        self.input_correct = False
        self.output_correct = False
        self.lab_max_grade = 0
        self.subtract = 0
        self.__wrong_clicked = False
        self.final_grade = 0
        self.__possible_answers_dict = {}
        self.global_log = ''
        self.previous_responses = ''
        self.__message_to_all = ''
        self.__graded_idlist = list()
        self.file_list = list()
        self.resp_text = 'I did not find any errors. Good job!\n'
        self.user_comment = ''
        self.cur_idx = 0
        self.working_dir = working_directory
        self.input_suggestion = set('',)
        self.resp_len = 38
        self.logisim_pid = -1
        self.circ_file_name = MAIN_FILE_NAME
        self.lab_type = ''
        self.lab_num = 0
        self.time = 0
        self.circ_obj_ref = None
        self.tot_elem = 0
        self.lab_id = ''
        self.grader = grader

```

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.

```
    def add_to_common_answers(self, typed):
        self.input_suggestion.add(typed)


```

7.5.3.2 check_circ_exist()

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

Definition at line 519 of file main.py.

```
    def check_circ_exist(self):
        if not os.path.isfile(self.file_list[self.cur_idx] + '/' + self.circ_file_name):
            self.resp_text = 'File was not found'
            file_found = os.listdir(self.file_list[self.cur_idx])
            potential_files = list()
            for file in file_found:
                if file not in ['grade.txt', 'penalty.txt', 'responce.txt', 'tech_info.txt', ]:
                    potential_files.append(file)
            if potential_files:
                self.resp_text += '\nNext files|folders were found:\n'
            for file in potential_files:
                if os.path.isdir(self.file_list[self.cur_idx] + '/' + file):
                    self.resp_text += file + ' - directory.\n'
                else:
                    self.resp_text += file + ' - regular file.\n'
            self.resp_len = len(self.resp_text)
            self.final_grade = 0
            return False
        return True

```

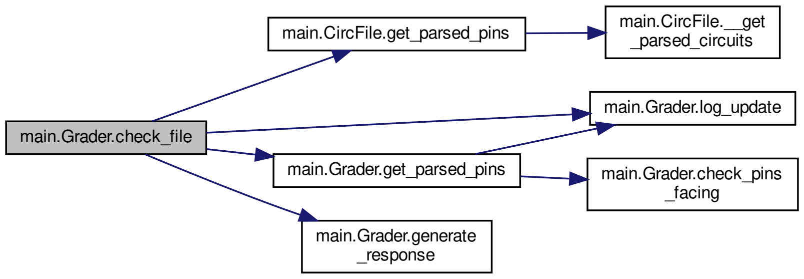
7.5.3.3 check_file()

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

Definition at line 498 of file main.py.

```
    def check_file(self):
        file = os.path.join(self.file_list[self.cur_idx], MAIN_FILE_NAME)

        circ_obj = CircFile(file)
        self.circ_obj_ref = circ_obj
        self.subtract = 0
        try:
            self.get_parsed_pins()

            self.log_update('Pins successfully parsed.')
            self.final_grade = self.lab_max_grade - self.subtract
            self.generate_response()
        except Exception as e:
            print(e)
            self.log_update(sys.exc_info()[0])

```

Here is the call graph for this function:


main.CircFile.get_parsed_pins
main.CircFile.__get_parsed_circuits
main.Grader.log_update
main.Grader.check_file
main.Grader.get_parsed_pins
main.Grader.check_pins_facing
main.Grader.generate_response

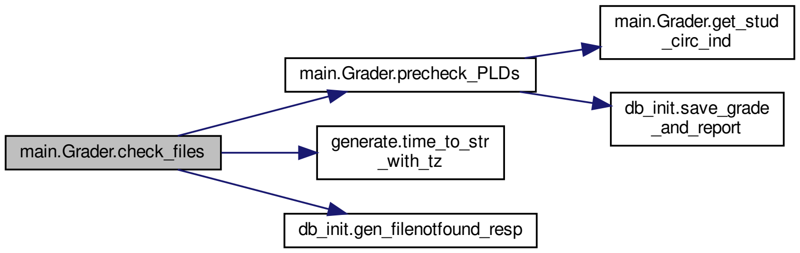
7.5.3.4 check_files()

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

Definition at line 348 of file main.py.

```
    def check_files(self):
        paths_with_files_list = list()
        good_ids = list()
        good_sids = list()
        good_tss = list()

        for i, stud_path in enumerate(self.lab_paths):
            cur_path = os.path.join(stud_path, self.circ_file_name)
            if os.path.exists(cur_path):
                paths_with_files_list.append(stud_path)
                good_ids.append(self.grade_ids[i])
                good_sids.append(self.stud_ids[i])
                good_tss.append(self.timestamps[i])
                if self.lab_num > 8 and self.lab_type == 'Closed':
                    self.precheck_PLDs(i)
            else:
                if self.attempt > 1:
                    next_date = time_to_str_with_tz(self.time_to + self.time_to - self.
      time_from)
                else:
                    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
                gen_filenotfound_resp(self.grade_ids[i], stud_path, self.
      circ_file_name, self.grader, self.attempt, next_date)
        # self.grade_ids = good_ids
        # self.stud_ids = good_sids
        # self.timestamps = good_tss
        return good_tss, good_sids, good_ids, paths_with_files_list

```

Here is the call graph for this function:


main.Grader.check_files
main.Grader.precheck_PLDs
generate.time_to_str_with_tz
db_init.gen_filenotfound_resp
main.Grader.get_stud_circ_ind
db_init.save_grade_and_report

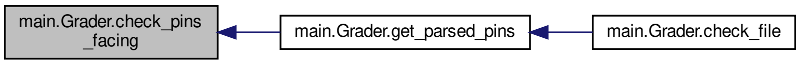
7.5.3.5 check_pins_facing()

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

Definition at line 487 of file main.py.

```
    def check_pins_facing(self, pins, corr_facing):
        for pin in pins:
            if pin.facing != corr_facing and pin.facing != '':
                return False
        return True
```

Here is the caller graph for this function:



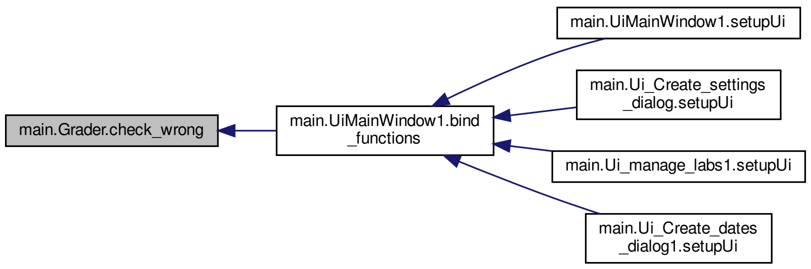
7.5.3.6 check_wrong()

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

Definition at line 643 of file main.py.

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

```

Here is the caller graph for this function:


main.Grader.check_wrong
main.UiMainWindow1.bind_functions
main.UiMainWindow1.setupUi
main.Ui_Create_settings_dialog.setupUi
main.Ui_manage_labs1.setupUi
main.Ui_Create_dates_dialog1.setupUi

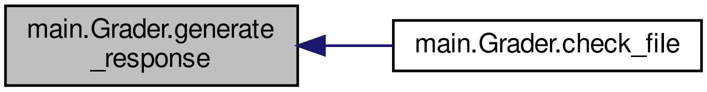
7.5.3.7 generate_response()

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

Definition at line 693 of file main.py.

```
    def generate_response(self):
        self.resp_text = ''
        self.user_comment = ''
        if self.input_correct and self.output_correct:
            self.resp_text = 'I did not find any errors. Good job!'
        else:
            if not self.input_correct:
                self.resp_text += 'Your input pins have wrong orientation.\n'

            if not self.output_correct:
                self.resp_text += 'Your output pins have wrong orientation.\n'
        self.resp_len = len(self.resp_text)

```

Here is the caller graph for this function:



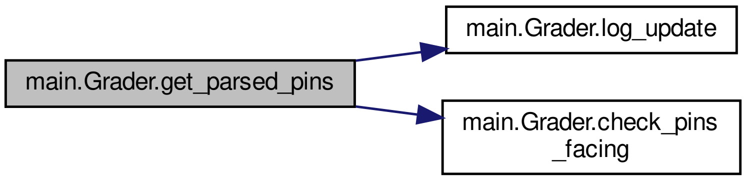
7.5.3.8 get_parsed_pins()

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

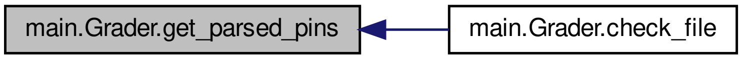
Definition at line 459 of file main.py.

```
    def get_parsed_pins(self):
        try:
            input_pins, output_pins, other_pins = self.circ_obj_ref.get_parsed_pins()
            if other_pins:
                self.log_update('I was not able to recognize ' + str(len(other_pins)) + " pins.")
            self.input_correct = True
            self.output_correct = True
            if not self.check_pins_facing(pins=input_pins, corr_facing='east'):
                self.subtract += 1
                self.input_correct = False
            if not self.check_pins_facing(pins=output_pins, corr_facing='west'):
                self.subtract += 1
                self.output_correct = False
        except Exception as e:  # TODO check for FileNotFoundError and assign ZERO
            print(e)
            # self.log_update(sys.exc_info()[0])
            # print(sys.exc_info()[0])
            raise
        # self.log_update('Done checking: ' + self.filename)


```

Here is the call graph for this function:



Here is the caller graph for this function:



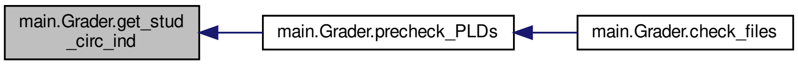
7.5.3.9 get_stud_circ_ind()

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

Definition at line 374 of file main.py.

```
    def get_stud_circ_ind(self, student_circuits, circ_to_grade):
        for stud_circ in student_circuits:
            if stud_circ.name.upper() == circ_to_grade.upper():
                return student_circuits.index(stud_circ)
        for stud_circ in student_circuits:
            print(stud_circ.name.upper())
        return -1

```

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.

```
    def get_stud_id(self):
        return self.stud_id

```

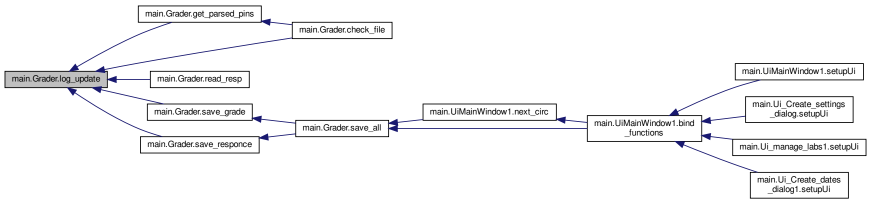
7.5.3.11 log_update()

```
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:


main.Grader.log_update
main.Grader.get_parsed_pins
main.Grader.check_file
main.Grader.read_resp
main.Grader.save_grade
main.Grader.save_responce
main.Grader.save_all
main.UiMainWindow1.next_circ
main.UiMainWindow1.bind_functions
main.UiMainWindow1.setupUi
main.Ui_Create_settings_dialog.setupUi
main.Ui_manage_labs1.setupUi
main.Ui_Create_dates_dialog1.setupUi

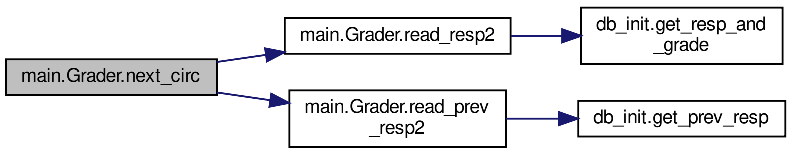
7.5.3.12 next_circ()

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

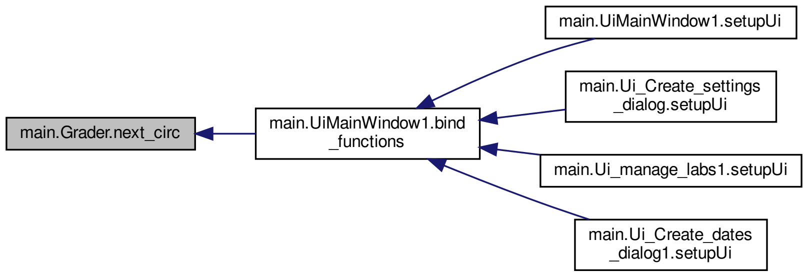
Definition at line 601 of file main.py.

```
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:


main.Grader.next_circ
main.UiMainWindow1.bind_functions
main.UiMainWindow1.setupUi
main.Ui_Create_settings_dialog.setupUi
main.Ui_manage_labs1.setupUi
main.Ui_Create_dates_dialog1.setupUi

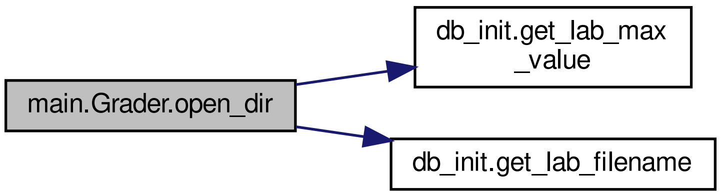
7.5.3.13 open_dir()

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

Definition at line 270 of file main.py.

```
    def open_dir(self):
        # TODO check behaviour when directory is wrong.
        # if len(self.working_directory) < 3:
        #     wdir = './'
        # else:
        #     wdir = self.working_directory


        root, dirs, files = os.walk(self.working_dir).__next__()
        files.sort()
        # check_file = files[0]  # not used at this time
        # if len(files) < 1:
        #     raise Exception("No due files ? Extra files in working directory ?")
        # due_file = files[1] # TODO: change this to a better design. - Already changed

        self.lab_type = self.working_dir.split('/')[-2].split('_')[0]
        self.lab_num = int(self.working_dir.split('/')[-2].split('_')[2])
        self.attempt = int(self.working_dir.split('/')[-2].split('_')[3])

        if self.lab_type == 'Closed':
            self.lab_id = 'CLA{}'.format(self.lab_num)
            # self.lab_max_grade = 10
        else:  # Open
            # self.lab_max_grade = 20
            self.lab_id = 'OLA{}'.format(self.lab_num)

        self.lab_max_grade = get_lab_max_value(self.lab_id)

        # self.time = int(due_file[6:])

        # dirs.sort()  # sort list of submitted labs
        # if dirs[0] == 'Answers':
        #     dirs.pop(0)

        self.circ_file_name = get_lab_filename(self.lab_id)[0]
        self.year, self.semester = self.working_dir.split('/')[-3].split('_')
        self.lid = get_labid_in_schedule(get_lab_id(self.lab_type, self.lab_num), self.year, self.semester)
        self.timestamps, self.stud_ids, self.grade_ids, self.lab_paths = get_empty_grades_by_lid(self.lid, self.attempt)

        atime = get_grading_period(self.lid, cur_only=True)
        self.time_from = atime[1]
        self.time_to = atime[2]
        self.time_cur = atime[3]

        self.time_from_qt = QDateTime.fromSecsSinceEpoch(self.time_from)
        self.time_to_qt = QDateTime.fromSecsSinceEpoch(self.time_to)
        self.time_cur_qt = QDateTime.fromSecsSinceEpoch(self.time_cur)

        if self.lab_num > 8 and self.lab_type == 'Closed':
            if self.lab_num == 9:
                self.what_to_grade = ['PC_BUS', 'AR_LD', 'PC_LD', 'PC_INC', 'DR_LD', 'DR_BUS']
            elif self.lab_num == 10:
                self.what_to_grade = ["R_LD", "R_BUS", "S_LD", "ACC_CLR", "ACC_LD", "ACC_BUS", "ALU_SEL"]
            elif self.lab_num == 11:
                self.what_to_grade = ["Z_LD", "OUTR_LD", "RAM_RW", "RAM_EN", "IR_LD", "SC_CLR"]
            circ = CircFile('/home/vanya/Documents/3130_labs/2018_2/PLDs.circ')
            self.all_my_circuits = circ.get_parsed_pins2(self.what_to_grade)

        if self.lab_paths is not None and len(self.lab_paths) > 0:
            self.timestamps, self.stud_ids, self.grade_ids, self.lab_paths = self.check_files()

        if self.lab_paths is None or len(self.lab_paths) == 0:  # if there are no ungraded labs - display all labs
            self.timestamps, self.stud_ids, self.grade_ids, self.lab_paths = get_all_grades_by_lid(self.lid, self.attempt)

        # self.grades = [self.lab_max_grade]*len(self.grade_ids)
        # self.stud_ids = dirs
        # self.stud_ids = list()
        # self.timestamps = list()
        # # directory_list = list()
        # for name in dirs:
        #     self.file_list.append(os.path.join(root, name))
        #     temp_arr = name.split('-')
        #     self.stud_ids.append(temp_arr[0])
        #     self.timestamps.append(int(temp_arr[2]))

        # for file in self.file_list:
        #     print(file)

```

Here is the call graph for this function:



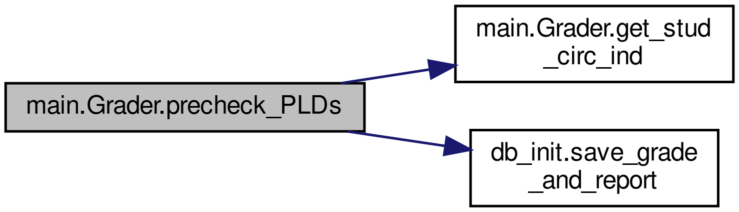
7.5.3.14 precheck_PLDs()

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

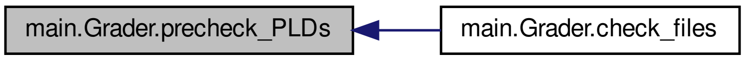
Definition at line 382 of file main.py.

```
    def precheck_PLDs(self, stud_ind):
        file = os.path.join(self.lab_paths[stud_ind], self.circ_file_name)

        student_circuits = CircFile(file).get_parsed_pins2(self.what_to_grade)
        errors = 0

        out_str = '<br> Next part was generated by automatic grader that I wrote several years ago.' \
                  '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 ' \
                  'even if not everything was correct.<br>'
        for circ_to_grade in self.what_to_grade:
            for good_circ in self.all_my_circuits:
                if good_circ.name.upper() == circ_to_grade.upper():
                    cur_ind = self.get_stud_circ_ind(student_circuits, circ_to_grade)
                    out_str += '<br>'
                    if cur_ind == -1:
                        out_str += '<font color="red">{}  NOT FOUND!<br> </font>'.format(circ_to_grade)
                        errors += 1
                    else:
                        check_pins = student_circuits[cur_ind].input_pins
                        for i in range(len(check_pins)):
                            if check_pins[i][0].lower() != 'c':
                                # print(check_pins[i])
                                if len(check_pins[i][1:]) > 0:
                                    try:
                                        pos = None
                                        for ch in check_pins[i]:
                                            if not ch.isalpha():
                                                pos = check_pins[i].index(ch)
                                                break
                                        num = int(check_pins[i][pos:])
                                    except Exception as e:
                                        print(e)
                                        continue
                                    check_pins[i] = check_pins[i][0:1] + str(num)
                        student_circuits_sorted = sorted(check_pins)
                        good_circ_sorted = sorted(good_circ.input_pins)
                        sm = difflib.SequenceMatcher(None, student_circuits_sorted, good_circ_sorted)
                        res_ratio = sm.ratio()

                        if res_ratio > 0.99:
                            out_str += '<font color="green"> {} : PERFECT MATCH!<br> </font>'.format(circ_to_grade)
                        elif res_ratio > 0.15:
                            out_str += '{} :Great news : you match ratio is {:.1%} (>75%)<br>{} : <b>FOUND</b> {}  <br>{} : <b>EXPECTED</b> {} <br>'\
                                .format(circ_to_grade, res_ratio, circ_to_grade, ' '.join(student_circuits_sorted), circ_to_grade, ' '.join(good_circ_sorted))
                            errors += 1
                        else:
                            out_str += '<font color="red">{} Bad news : you match ratio is only', \
                                       '{:.1%} - this means that you have to significantly change your circuit. <br> ' \
                                       'Please send me a message if you need some advice.<br> </font>'.format(circ_to_grade, res_ratio)
                            errors += 1

        final_grade = math.ceil(10 * (len(self.what_to_grade) - errors) / len(self.what_to_grade))
        # out_str += '<br> Bad grade confidence: ' + conf + ' (this is for Ivan)<br>' + '<br> Next part will be typed manually: <br>'
        save_grade_and_report(self.grade_ids[stud_ind], final_grade, out_str, None, self.grader)
        return final_grade, out_str


```

Here is the call graph for this function:


main.Grader.get_stud_circ_ind
main.Grader.precheck_PLDs
db_init.save_grade_and_report

Here is the caller graph for this function:



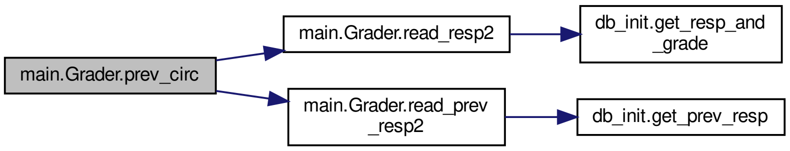
7.5.3.15 prev_circ()

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

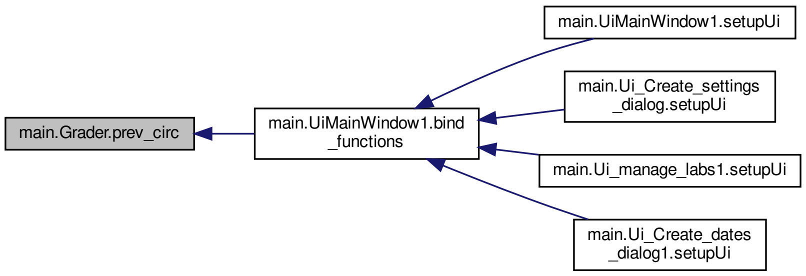
Definition at line 622 of file main.py.

```
    def prev_circ(self):
        self.cur_idx -= 1
        # self.check_file(self.cur_idx)
        self.user_comment = ''
        graded = self.read_resp2()
        if graded:
            self.read_prev_resp2()
        # if self.check_circ_exist():
        #     self.read_resp()
        self.stud_id = self.stud_ids[self.cur_idx]
        # try:
        #     self.read_prev_resp()
        # except Exception as e:
        #     print('Error during attempt to read prev resp when opening prev circuit: ', e)
        #     # TODO add handler
        return self.cur_idx

```

Here is the call graph for this function:



Here is the caller graph for this function:


main.UiMainWindow1.setupUi
main.Ui_Create_settings_dialog.setupUi
main.Grader.prev_circ
main.UiMainWindow1.bind_functions
main.Ui_manage_labs1.setupUi
main.Ui_Create_dates_dialog1.setupUi

7.5.3.16 read_prev_resp()

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

Definition at line 581 of file main.py.

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

```

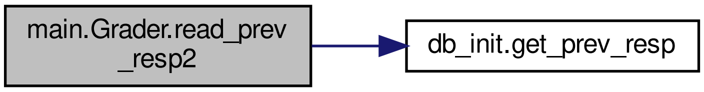
7.5.3.17 read_prev_resp2()

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

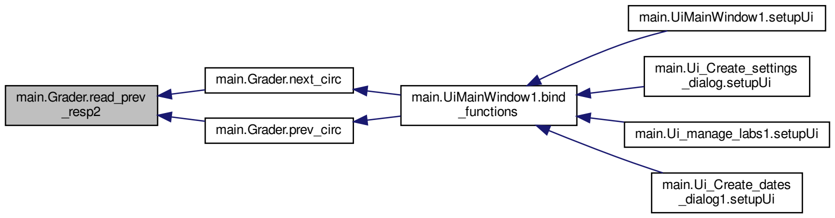
Definition at line 573 of file main.py.

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

```

Here is the call graph for this function:



Here is the caller graph for this function:


main.Grader.read_prev_resp2
main.Grader.next_circ
main.Grader.prev_circ
main.UiMainWindow1.bind_functions
main.UiMainWindow1.setupUi
main.Ui_Create_settings_dialog.setupUi
main.Ui_manage_labs1.setupUi
main.Ui_Create_dates_dialog1.setupUi

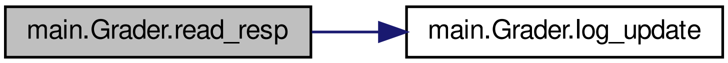
7.5.3.18 read_resp()

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

Definition at line 544 of file main.py.

```
    def read_resp(self):
        self.submitted = self.timestamps[self.cur_idx]
        try:
            with open(os.path.join(self.file_list[self.cur_idx], 'responce.txt'), 'r') as resp_file:
                a = resp_file.readlines()
                self.resp_text = ''.join(a)
                self.resp_len = len(self.resp_text)
        except Exception as e:
            print(e)
            self.log_update(sys.exc_info()[0])

        try:
            with open(os.path.join(self.file_list[self.cur_idx], 'grade.txt'), 'r') as grade_file:
                self.final_grade = int(grade_file.readline())
        except Exception as e:
            print(e)
            self.log_update(sys.exc_info()[0])

        # self.read_prev_resp()

```

Here is the call graph for this function:



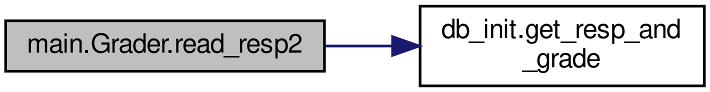
7.5.3.19 read_resp2()

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

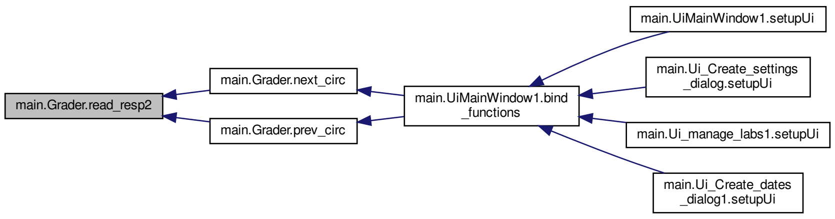
Definition at line 564 of file main.py.

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

```

Here is the call graph for this function:



Here is the caller graph for this function:


main.Grader.read_resp2
main.Grader.next_circ
main.Grader.prev_circ
main.UiMainWindow1.bind_functions
main.UiMainWindow1.setupUi
main.Ui_Create_settings_dialog.setupUi
main.Ui_manage_labs1.setupUi
main.Ui_Create_dates_dialog1.setupUi

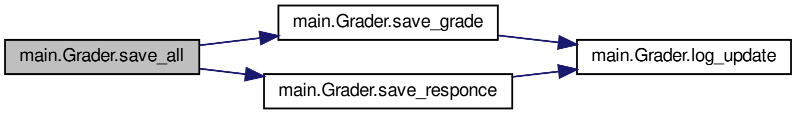
7.5.3.20 save_all()

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

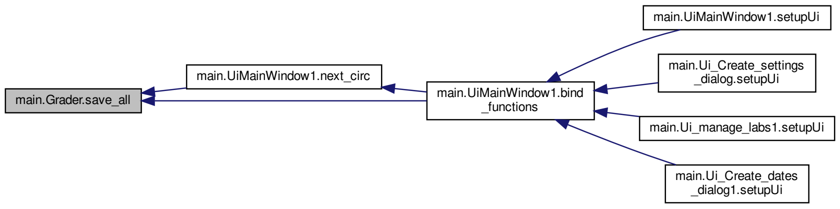
Definition at line 677 of file main.py.

```
    def save_all(self):
        self.save_grade()
        self.save_responce()


```

Here is the call graph for this function:



Here is the caller graph for this function:


main.Grader.save_all
main.UiMainWindow1.next_circ
main.UiMainWindow1.bind_functions
main.UiMainWindow1.setupUi
main.Ui_Create_settings_dialog.setupUi
main.Ui_manage_labs1.setupUi
main.Ui_Create_dates_dialog1.setupUi

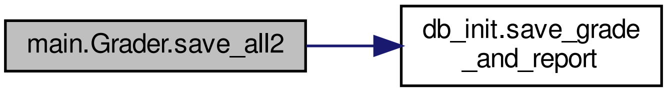
7.5.3.21 save_all2()

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

Definition at line 686 of file main.py.

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

```

Here is the call graph for this function:



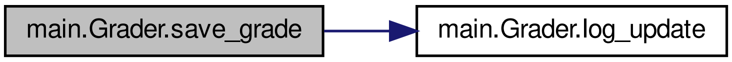
7.5.3.22 save_grade()

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

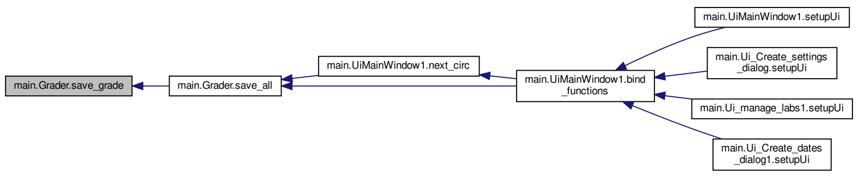
Definition at line 652 of file main.py.

```
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:



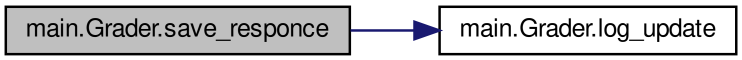
7.5.3.23 save_responce()

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

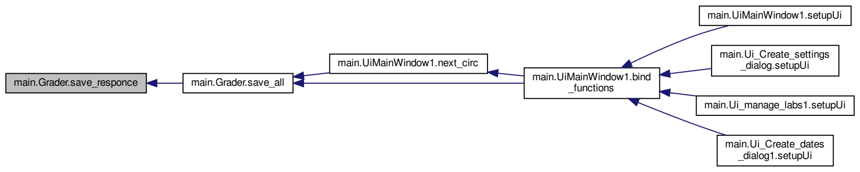
Definition at line 664 of file main.py.

```
664     def save_responce(self):
665         file = os.path.join(self.lab_paths[self.cur_idx], 'responce.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('Responce saved')
671
```

Here is the call graph for this function:



Here is the caller graph for this function:


main.Grader.save_responce
main.Grader.save_all
main.UiMainWindow1.next_circ
main.UiMainWindow1.bind_functions
main.UiMainWindow1.setupUi
main.Ui_Create_settings_dialog.setupUi
main.Ui_manage_labs1.setupUi
main.Ui_Create_dates_dialog1.setupUi

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.

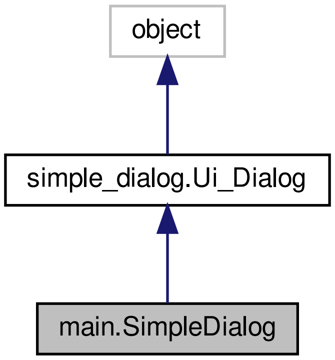
The documentation for this class was generated from the following file:

- main.py

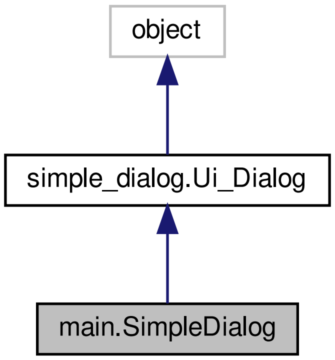
7.7 main.SimpleDialog Class Reference

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.

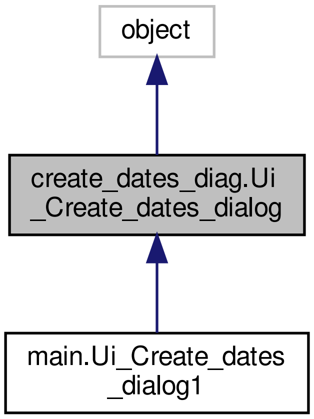
```
    def setupUi(self, Dialog, phrase):
        super().setupUi(Dialog)
        self.label_main_question.setText(phrase)


```

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

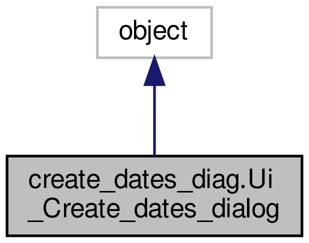
- main.py

7.8 create_dates_diag.Ui_Create_dates_dialog Class Reference

Inheritance diagram for create_dates_diag.Ui_Create_dates_dialog:


object
create_dates_diag.Ui _Create_dates_dialog
main.Ui_Create_dates _dialog1

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.

```
    def retranslateUi(self, Create_dates_dialog):

         _translate = QtCore.QCoreApplication.translate
         Create_dates_dialog.setWindowTitle(_translate("Create_dates_dialog", "Dialog"))
         self.lab_path.setToolTip(_translate("Create_dates_dialog", "Tripple for file dialog"))
         self.lab_path.setPlaceholderText(_translate("Create_dates_dialog", "DoubleClick to select path"))
         self.init_label.setText(_translate("Create_dates_dialog", "Submission date"))
         self.first_label.setText(_translate("Create_dates_dialog", "1st resubmission"))
         self.second_label.setText(_translate("Create_dates_dialog", "2nd resubmission"))
         self.third_label.setText(_translate("Create_dates_dialog", "3rd resubmission"))

```

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.

```
    def setupUi(self, Create_dates_dialog):
        Create_dates_dialog.setObjectName("Create_dates_dialog")
        Create_dates_dialog.resize(589, 250)
        Create_dates_dialog.setMinimumSize(QtCore.QSize(500, 250))
        Create_dates_dialog.setMaximumSize(QtCore.QSize(1000, 300))
        icon = QtGui.QIcon()
        icon.addPixmap(QtGui.QPixmap("os_linux_1.ico"), QtGui.QIcon.Normal, QtGui.QIcon.Off)
        Create_dates_dialog.setWindowIcon(icon)
        self.verticalLayout = QtWidgets.QVBoxLayout(Create_dates_dialog)
        self.verticalLayout.setObjectName("verticalLayout")
        self.horizontalLayout_5 = QtWidgets.QHBoxLayout()
        self.horizontalLayout_5.setObjectName("horizontalLayout_5")
        self.lab_path = BetterLineEdit(Create_dates_dialog)
        self.lab_path.setFocusPolicy(QtCore.Qt.StrongFocus)
        self.lab_path.setStatusTip("")
        self.lab_path.setWhatsThis("")
        self.lab_path.setAccessibleName("")
        self.lab_path.setAccessibleDescription("")
        self.lab_path.setInputMask("")
        self.lab_path.setReadOnly(False)
        self.lab_path.setCursorMoveStyle(QtCore.Qt.LogicalMoveStyle)
        self.lab_path.setClearButtonEnabled(False)
        self.lab_path.setObjectName("lab_path")
        self.horizontalLayout_5.addWidget(self.lab_path)
        self.verticalLayout.addLayout(self.horizontalLayout_5)
        self.horizontalLayout = QtWidgets.QHBoxLayout()
        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
```

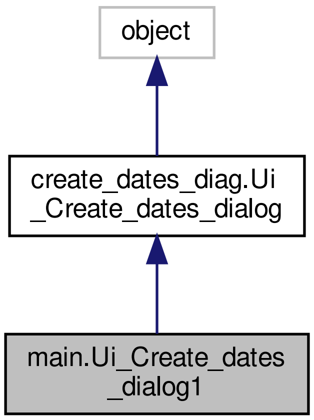
Definition at line 20 of file create_dates_diag.py.

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

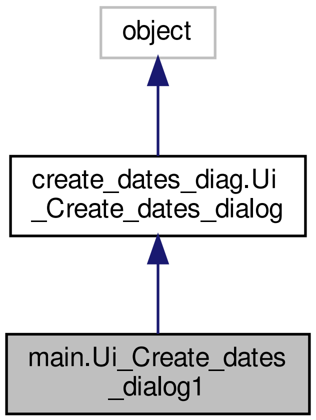
- create_dates_diag.py

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

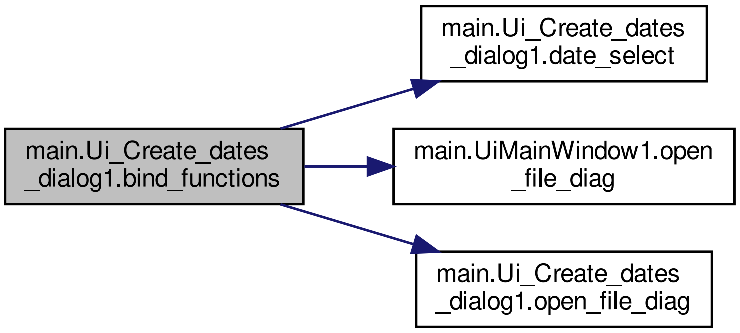
7.9.2.1 bind_functions()

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

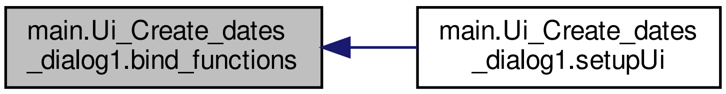
Definition at line 1934 of file main.py.

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

```

Here is the call graph for this function:


main.Ui_Create_dates _dialog1.bind_functions
main.Ui_Create_dates _dialog1.date_select
main.UiMainWindow1.open _file_diag
main.Ui_Create_dates _dialog1.open_file_diag

Here is the caller graph for this function:


main.Ui_Create_dates _dialog1.bind_functions
main.Ui_Create_dates _dialog1.setupUi

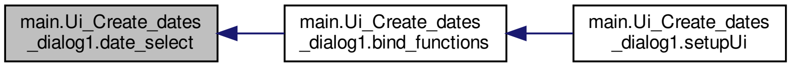
7.9.2.2 date_select()

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

Definition at line 1987 of file main.py.

```
    def date_select(self):
        self.first_subm_date_time.setDateTime(self.init_subm_date_time.dateTime().addDays(7))
        self.second_subm_date_time.setDateTime(self.init_subm_date_time.dateTime().addDays(14))
        self.third_subm_date_time.setDateTime(self.init_subm_date_time.dateTime().addDays(21))

```

Here is the caller graph for this function:


main.Ui_Create_dates_dialog1.date_select
main.Ui_Create_dates_dialog1.bind_functions
main.Ui_Create_dates_dialog1.setupUi

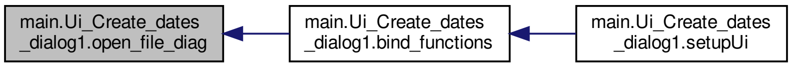
7.9.2.3 open_file_diag()

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

Definition at line 1996 of file main.py.

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


```

Here is the caller graph for this function:


main.Ui_Create_dates_dialog1.open_file_diag
main.Ui_Create_dates_dialog1.bind_functions
main.Ui_Create_dates_dialog1.setupUi

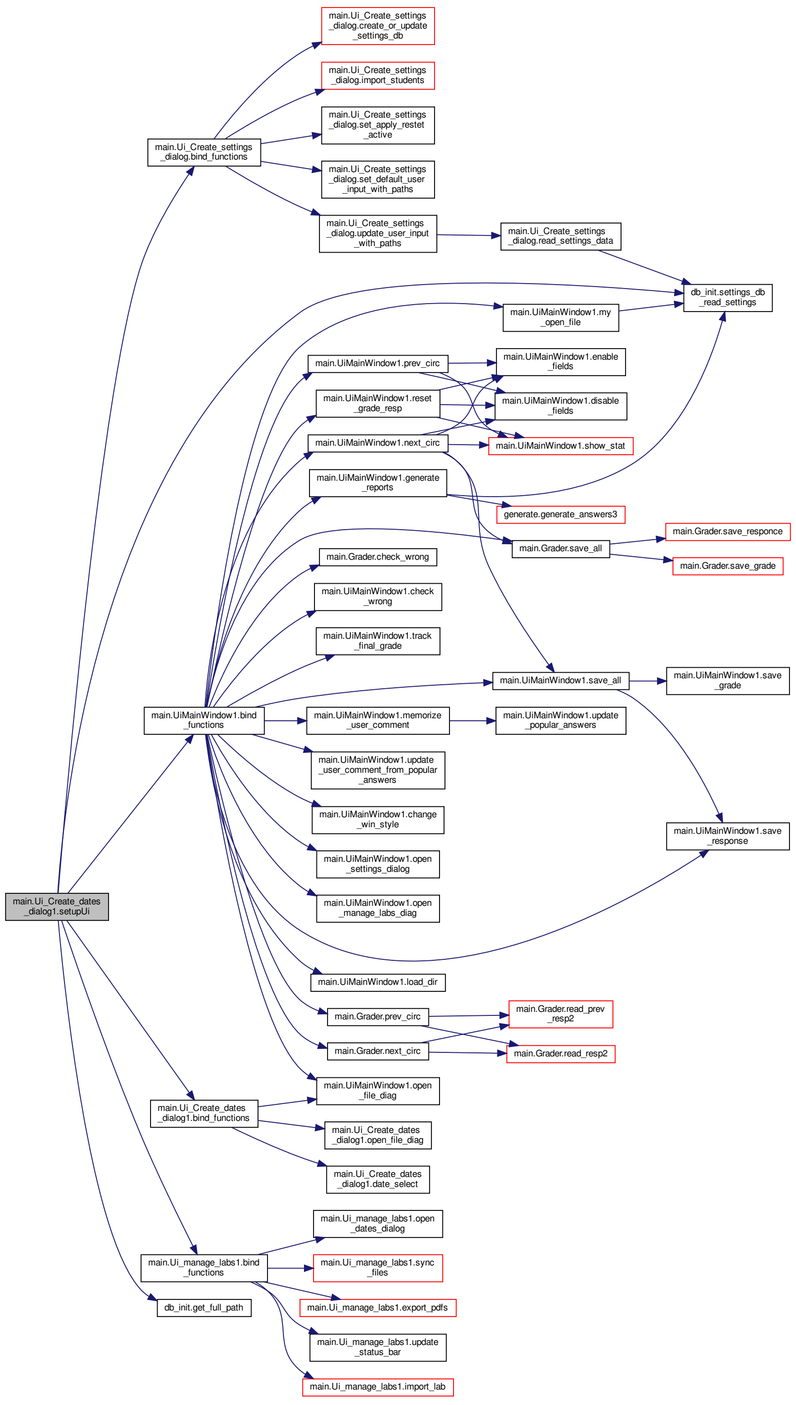
7.9.2.4 setupUi()

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

Definition at line 1971 of file main.py.

```
    def setupUi(self, Create_dates_dialog, selected_lab=''):
        super().setupUi(Create_dates_dialog)
        self.bind_functions()
        self.init_subm_date_time.setDateTime(QDateTime.currentDateTime())
        paths, local = settings_db_read_settings()
        good_path = get_full_path(paths, local) + '/server_sync/'
        try:
            good_path += selected_lab + '/'
        except Exception as e:
            print('Exception when tried to append selected folder from Manage labs. ', e)
        self.lab_path.setText(good_path)

```

Here is the call graph for this function:


main.Ui_Create_settings_dialog.create_or_update_settings_db
main.Ui_Create_settings_dialog.import_students
main.Ui_Create_settings_dialog.set_apply_restet_active
main.Ui_Create_settings_dialog.bind_functions
main.Ui_Create_settings_dialog.set_default_user_input_with_paths
main.Ui_Create_settings_dialog.update_user_input_with_paths
main.Ui_Create_settings_dialog.read_settings_data
db_init.settings_db_read_settings
main.UiMainWindow1.my_open_file
main.UiMainWindow1.prev_circ
main.UiMainWindow1.enable_fields
main.UiMainWindow1.reset_grade_resp
main.UiMainWindow1.disable_fields
main.UiMainWindow1.next_circ
main.UiMainWindow1.show_stat
main.UiMainWindow1.generate_reports
generate.generate_answers3
main.Grader.save_responce
main.Grader.save_all
main.Grader.check_wrong
main.Grader.save_grade
main.UiMainWindow1.check_wrong
main.UiMainWindow1.track_final_grade
main.UiMainWindow1.save_all
main.UiMainWindow1.save_grade
main.UiMainWindow1.bind_functions
main.UiMainWindow1.memorize_user_comment
main.UiMainWindow1.update_popular_answers
main.UiMainWindow1.update_user_comment_from_popular_answers
main.UiMainWindow1.change_win_style
main.UiMainWindow1.save_response
main.UiMainWindow1.open_settings_dialog
main.Ui_Create_dates_dialog1.setupUi
main.UiMainWindow1.open_manage_labs_diag
main.UiMainWindow1.load_dir
main.Grader.prev_circ
main.Grader.read_prev_resp2
main.Grader.next_circ
main.Grader.read_resp2
main.UiMainWindow1.open_file_diag
main.Ui_Create_dates_dialog1.bind_functions
main.Ui_Create_dates_dialog1.open_file_diag
main.Ui_Create_dates_dialog1.date_select
main.Ui_manage_labs1.open_dates_dialog
main.Ui_manage_labs1.bind_functions
main.Ui_manage_labs1.sync_files
main.Ui_manage_labs1.export_pdfs
db_init.get_full_path
main.Ui_manage_labs1.update_status_bar
main.Ui_manage_labs1.import_lab

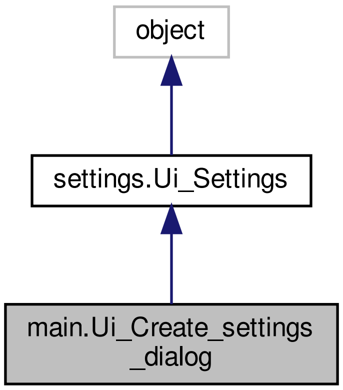
The documentation for this class was generated from the following file:

- main.py

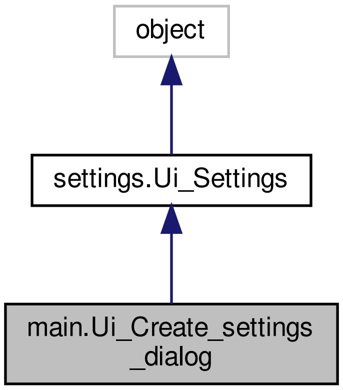
7.10 main.Ui_Create_settings_dialog Class Reference

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

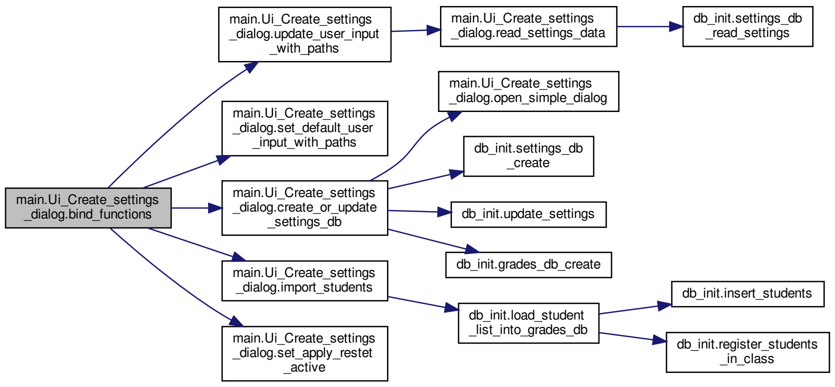
7.10.2.1 bind_functions()

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

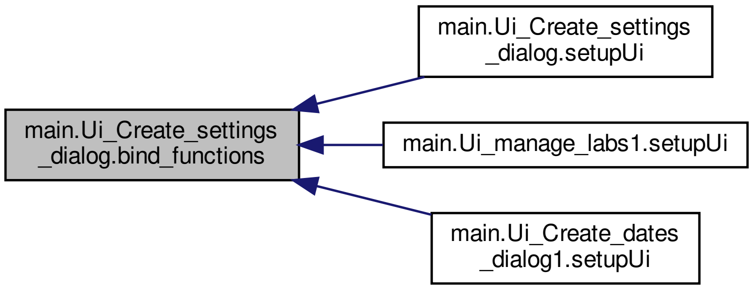
Definition at line 1347 of file main.py.

```
    def bind_functions(self):
        self.buttonBox.button(self.buttonBox.Reset).clicked.connect(self.update_user_input_with_paths)
        self.buttonBox.button(self.buttonBox.RestoreDefaults).clicked.connect(self.
set_default_user_input_with_paths)
        self.buttonBox.button(self.buttonBox.Apply).clicked.connect(self.create_or_update_settings_db)
        self.buttonBox.button(self.buttonBox.Ok).clicked.connect(self.create_or_update_settings_db)

        self.import_stuents_btn.clicked.connect(self.import_students)

        # TODO: make 'personal' events and update only fields that have been changed
        self.input_logisim_path.textChanged.connect(self.set_apply_restet_active)
        self.input_local_stor.textChanged.connect(self.set_apply_restet_active)
        self.input_rem_stor.textChanged.connect(self.set_apply_restet_active)
        self.input_grader_name.textChanged.connect(self.set_apply_restet_active)
        self.spin_year.valueChanged.connect(self.set_apply_restet_active)
        self.semester_comboBox.currentIndexChanged.connect(self.set_apply_restet_active)
        self.style_checkBox.stateChanged.connect(self.set_apply_restet_active)
        self.sync_command.textChanged.connect(self.set_apply_restet_active)
        self.input_grades_db.textChanged.connect(self.set_apply_restet_active)

```

Here is the call graph for this function:


main.Ui_Create_settings_dialog.bind_functions
main.Ui_Create_settings_dialog.update_user_input_with_paths
main.Ui_Create_settings_dialog.read_settings_data
db_init.settings_db_read_settings
main.Ui_Create_settings_dialog.open_simple_dialog
main.Ui_Create_settings_dialog.set_default_user_input_with_paths
db_init.settings_db_create
main.Ui_Create_settings_dialog.create_or_update_settings_db
db_init.update_settings
db_init.grades_db_create
main.Ui_Create_settings_dialog.import_students
db_init.insert_students
db_init.load_student_list_into_grades_db
db_init.register_students_in_class
main.Ui_Create_settings_dialog.set_apply_restet_active

Here is the caller graph for this function:


main.Ui_Create_settings_dialog.setupUi
main.Ui_Create_settings_dialog.bind_functions
main.Ui_manage_labs1.setupUi
main.Ui_Create_dates_dialog1.setupUi

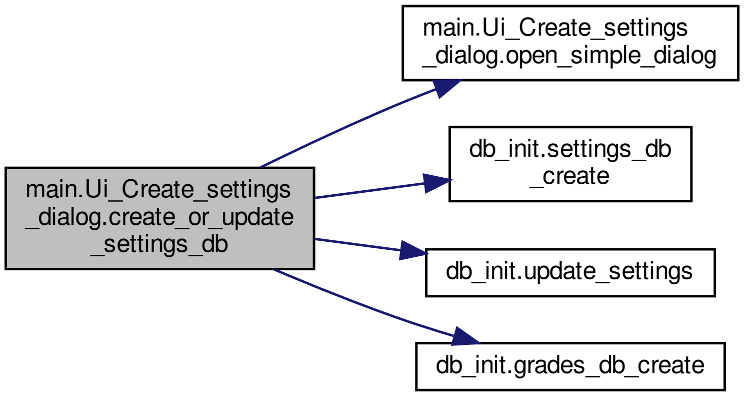
7.10.2.2 create_or_update_settings_db()

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

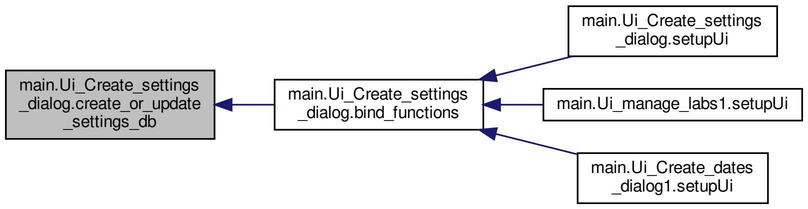
Definition at line 1439 of file main.py.

```
    def create_or_update_settings_db(self):
        from pathlib import Path
        settings_location = str(Path(os.path.expandvars(os.path.expanduser('./settings.sqlite3'))).absolute
    ())
        if not os.path.isfile(settings_location):
            if self.open_simple_dialog("Do you want to create settings database ?"):
                if not settings_db_create(force=True):
                    raise Exception('Was not able to create SETTINGS db.')
        if len(self.input_local_stor.text()) > 0:
            if self.input_local_stor.text()[-1] != '/':
                self.input_local_stor.setText(self.input_local_stor.text() + '/')
        if len(self.input_rem_stor.text()) > 0:
            if self.input_rem_stor.text()[-1] != '/':
                self.input_rem_stor.setText(self.input_rem_stor.text() + '/')
        if len(self.input_logisim_path.text()) > 0:
            if self.input_logisim_path.text()[-1] != '/':
                self.input_logisim_path.setText(self.input_logisim_path.text() + '/')


        paths = (self.input_logisim_path.text(), self.input_local_stor.text(), self.input_rem_stor.text(),
                 self.input_grades_db.text())
        if os.path.isfile(settings_location):
            local = (self.input_grader_name.text(), int(self.spin_year.text()),
                    self.semester_comboBox.currentIndex(), self.style_checkBox.checkState(), self.
sync_command.text())
            if len(self.input_local_stor.text()) > 0:
                local_stor = str(Path(os.path.expanduser(os.path.expandvars(self.input_local_stor.text())))
.absolute())
                if local_stor[-1] != '/':
                    local_stor += '/'
                if not os.path.isdir(local_stor):
                    os.mkdir(local_stor)
                local_grading_path = local_stor + self.spin_year.text() + '_' +\
                                     str(self.semester_comboBox.currentIndex())
                if not os.path.isdir(local_grading_path):
                    os.mkdir(local_grading_path)
            update_settings(paths, local)


        grades_location = str(Path(os.path.expandvars(os.path.expanduser(self.input_grades_db.text()))).
absolute())
        if len(self.input_grades_db.text()) > 1 and not os.path.isfile(grades_location):
            if self.open_simple_dialog("Do you want to create GRADES database ?"):
                print('Before grades creation.')
                if not grades_db_create(grades_location, force=True):
                    raise Exception('Was not able to create GRADES db.')

        if os.path.isfile(settings_location) and os.path.isfile(grades_location):
            self.buttonBox.button(self.buttonBox.Apply).setDisabled(True)
            self.buttonBox.button(self.buttonBox.Apply).repaint()
            self.buttonBox.button(self.buttonBox.Reset).setDisabled(True)
            self.buttonBox.button(self.buttonBox.Reset).repaint()
            if not self.groupBox_user.isEnabled():
                self.groupBox_user.setEnabled(True)
            if not self.input_logisim_path.isEnabled():
                self.input_logisim_path.setEnabled(True)
                self.label_logisim_path.setEnabled(True)
            if not self.input_local_stor.isEnabled():
                self.input_local_stor.setEnabled(True)
                self.label_local_stor.setEnabled(True)
            if not self.input_rem_stor.isEnabled():
                self.input_rem_stor.setEnabled(True)
                self.label_rem_stor.setEnabled(True)
            if not self.spin_year.isEnabled():
                self.spin_year.setEnabled(True)
            if not self.semester_comboBox.isEnabled():
                self.semester_comboBox.setEnabled(True)
            if not self.style_checkBox.isEnabled():
                self.style_checkBox.setEnabled(True)
            if not self.input_grader_name.isEnabled():
                self.input_grader_name.setEnabled(True)
            if not self.sync_command.isEnabled():
                self.sync_command.setEnabled(True)

        # if len(self.input_local_stor.text()) > 1:
        #     full_path = Path(self.input_local_stor.text()).absolute()
        #     if not os.path.exists(full_path) or not os.path.isdir(full_path):
        #         os.makedirs(full_path)

```

Here is the call graph for this function:


main.Ui_Create_settings _dialog.open_simple_dialog
db_init.settings_db _create
main.Ui_Create_settings _dialog.create_or_update _settings_db
db_init.update_settings
db_init.grades_db_create

Here is the caller graph for this function:


main.Ui_Create_settings _dialog.setupUi
main.Ui_Create_settings _dialog.create_or_update _settings_db
main.Ui_Create_settings _dialog.bind_functions
main.Ui_manage_labs1.setupUi
main.Ui_Create_dates _dialog1.setupUi

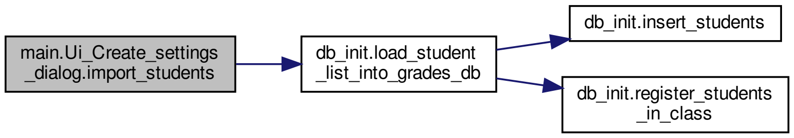
7.10.2.3 import_students()

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

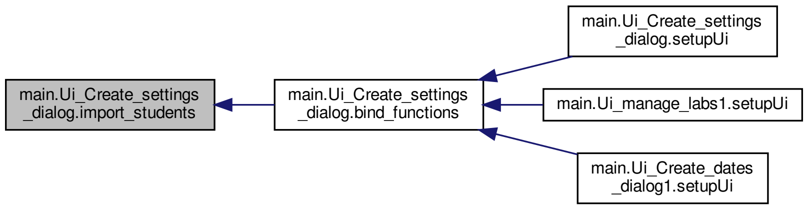
Definition at line 1518 of file main.py.

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


        self.import_stuents_btn.setEnabled(True)

```

Here is the call graph for this function:



Here is the caller graph for this function:



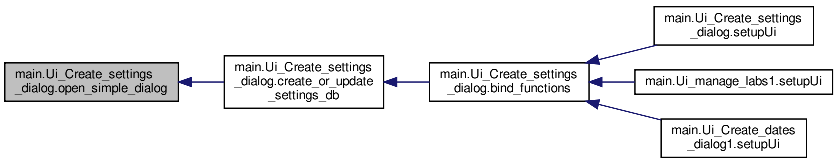
7.10.2.4 open_simple_dialog()

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

Definition at line 1543 of file main.py.

```
def open_simple_dialog(self, phrase):
    self.simple_diag = QtWidgets.QDialog()
    dui = SimpleDialog()
    dui.setupUi(self.simple_diag, phrase)

    self.buttonBox.setDisabled(True)
    self.buttonBox.repaint()

    self.simple_diag.setWindowTitle('Settings confirmation')
    self.simple_diag.show()

    result = self.simple_diag.exec_()

    self.buttonBox.setEnabled(True)

    return result


```

Here is the caller graph for this function:


main.Ui_Create_settings _dialog.open_simple_dialog
main.Ui_Create_settings _dialog.create_or_update _settings_db
main.Ui_Create_settings _dialog.bind_functions
main.Ui_Create_settings _dialog.setupUi
main.Ui_manage_labs1.setupUi
main.Ui_Create_dates _dialog1.setupUi

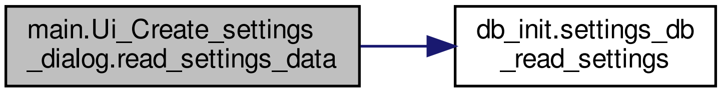
7.10.2.5 read_settings_data()

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

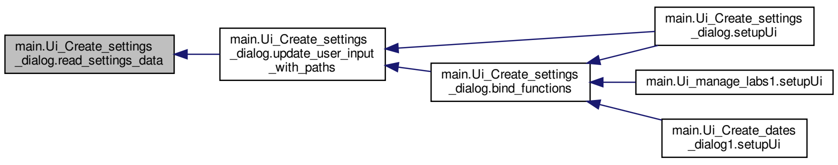
Definition at line 1428 of file main.py.

```
    def read_settings_data(self):
        self.buttonBox.button(self.buttonBox.Reset).setEnabled(True)
        return settings_db_read_settings()

```

Here is the call graph for this function:


main.Ui_Create_settings _dialog.read_settings_data
db_init.settings_db _read_settings

Here is the caller graph for this function:


main.Ui_Create_settings _dialog.read_settings_data
main.Ui_Create_settings _dialog.update_user_input _with_paths
main.Ui_Create_settings _dialog.bind_functions
main.Ui_Create_settings _dialog.setupUi
main.Ui_manage_labs1.setupUi
main.Ui_Create_dates _dialog1.setupUi

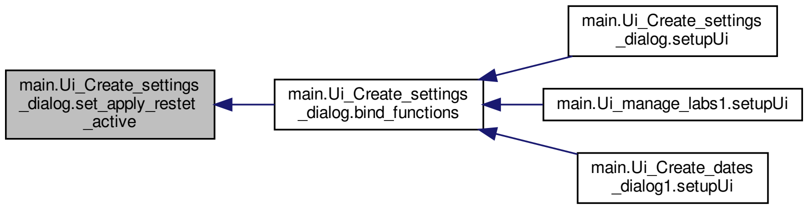
7.10.2.6 set_apply_restet_active()

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

Definition at line 1531 of file main.py.

```
def set_apply_restet_active(self):
    self.buttonBox.button(self.buttonBox.Reset).setEnabled(True)
    self.buttonBox.button(self.buttonBox.Apply).setEnabled(True)

```

Here is the caller graph for this function:


main.Ui_Create_settings_dialog.set_apply_restet_active
main.Ui_Create_settings_dialog.bind_functions
main.Ui_Create_settings_dialog.setupUi
main.Ui_manage_labs1.setupUi
main.Ui_Create_dates_dialog1.setupUi

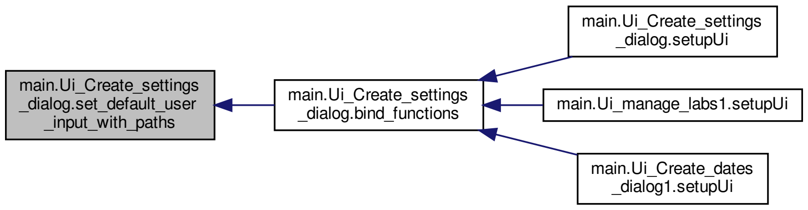
7.10.2.7 set_default_user_input_with_paths()

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

Definition at line 1415 of file main.py.

```
def set_default_user_input_with_paths(self):
    self.input_logisim_path.setText("~/Downloads/")
    self.input_local_stor.setText("~/Documents/3130_labs/")
    self.input_grades_db.setText("~/Documents/3130_labs/grades.sqlite3")
    self.input_rem_stor.setText("")  # impossible to predict
    self.groupBox_user.setEnabled(True)
    self.buttonBox.button(self.buttonBox.Reset).setEnabled(True)
    self.buttonBox.button(self.buttonBox.Apply).setEnabled(True)

```

Here is the caller graph for this function:


main.Ui_Create_settings_dialog.set_default_user_input_with_paths
main.Ui_Create_settings_dialog.bind_functions
main.Ui_Create_settings_dialog.setupUi
main.Ui_manage_labs1.setupUi
main.Ui_Create_dates_dialog1.setupUi

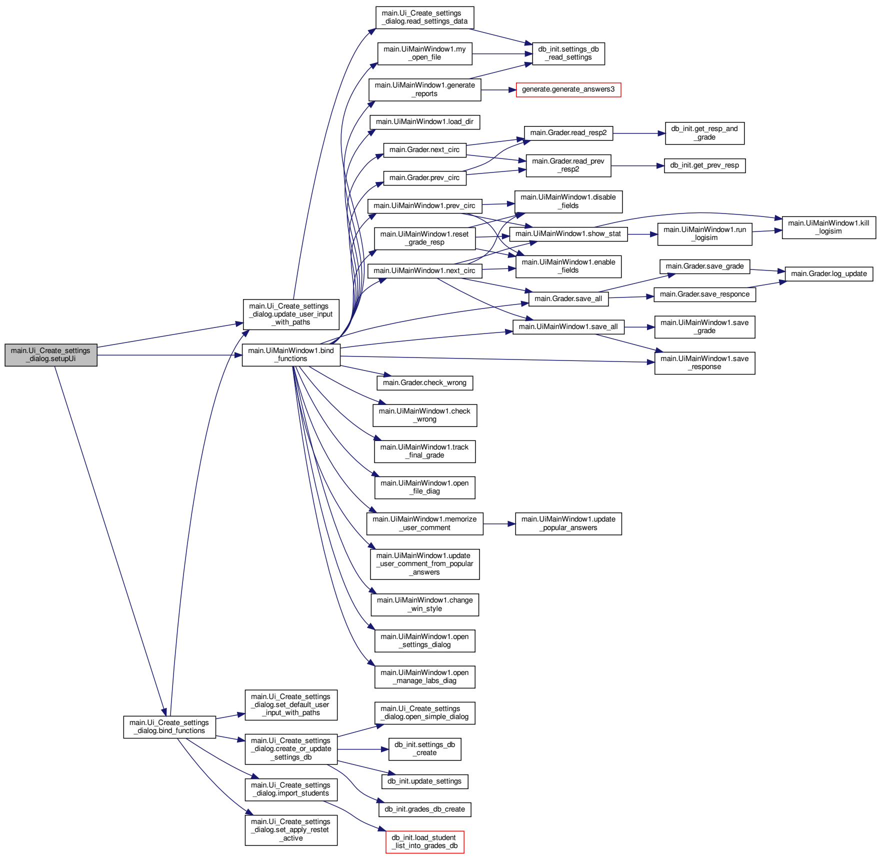
7.10.2.8 setupUi()

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

Definition at line 1370 of file main.py.

```
    def setupUi(self, Settings):
        super().setupUi(Settings)
        self.buttonBox.button(self.buttonBox.Reset).setDisabled(True)
        self.buttonBox.button(self.buttonBox.Apply).setDisabled(True)
        self.bind_functions()
        self.update_user_input_with_paths()

```

Here is the call graph for this function:


main.Ui_Create_settings_dialog.setupUi
main.Ui_Create_settings_dialog.update_user_input_with_paths
main.UiMainWindow1.bind_functions
main.Ui_Create_settings_dialog.read_settings_data
main.UiMainWindow1.my_open_file
main.UiMainWindow1.generate_reports
main.UiMainWindow1.load_dir
main.Grader.next_circ
main.Grader.prev_circ
main.UiMainWindow1.prev_circ
main.UiMainWindow1.reset_grade_resp
main.UiMainWindow1.next_circ
db_init.settings_db_read_settings
generate.generate_answers3
main.Grader.read_resp2
main.Grader.read_prev_resp2
db_init.get_resp_and_grade
db_init.get_prev_resp
main.UiMainWindow1.disable_fields
main.UiMainWindow1.show_stat
main.UiMainWindow1.enable_fields
main.UiMainWindow1.run_logisim
main.UiMainWindow1.kill_logisim
main.Grader.save_grade
main.Grader.log_update
main.Grader.save_all
main.Grader.save_responce
main.UiMainWindow1.save_all
main.UiMainWindow1.save_grade
main.UiMainWindow1.save_response
main.Grader.check_wrong
main.UiMainWindow1.check_wrong
main.UiMainWindow1.track_final_grade
main.UiMainWindow1.open_file_diag
main.UiMainWindow1.memorize_user_comment
main.UiMainWindow1.update_popular_answers
main.UiMainWindow1.update_user_comment_from_popular_answers
main.UiMainWindow1.change_win_style
main.UiMainWindow1.open_settings_dialog
main.UiMainWindow1.open_manage_labs_diag
main.Ui_Create_settings_dialog.bind_functions
main.Ui_Create_settings_dialog.set_default_user_input_with_paths
main.Ui_Create_settings_dialog.create_or_update_settings_db
main.Ui_Create_settings_dialog.import_students
main.Ui_Create_settings_dialog.set_apply_restet_active
main.Ui_Create_settings_dialog.open_simple_dialog
db_init.settings_db_create
db_init.update_settings
db_init.grades_db_create
db_init.load_student_list_into_grades_db

7.10.2.9 update_user_input_with_paths()

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

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

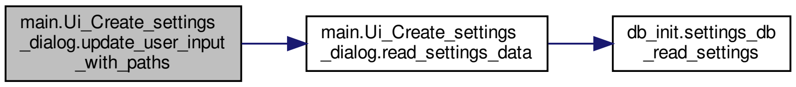
Warning

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

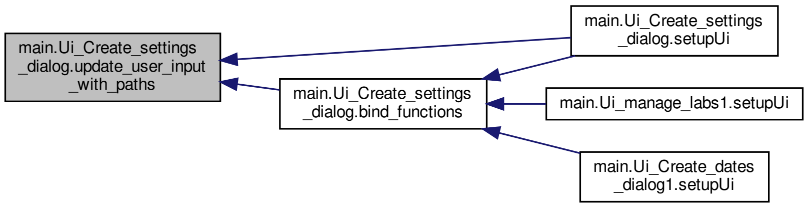
Definition at line 1382 of file main.py.

```
    def update_user_input_with_paths(self):
        paths, local = self.read_settings_data()
        if paths and len(paths) >= 4:
            self.input_logisim_path.setText(paths[0])
            self.input_local_stor.setText(paths[1])
            self.input_rem_stor.setText(paths[2])
            self.input_grades_db.setText(paths[3])
            self.groupBox_user.setEnabled(True)

        if local and len(local) >= 4:
            self.input_grader_name.setText(local[0])
            self.spin_year.setValue(local[1])
            self.semester_comboBox.setCurrentIndex(int(local[2]))
            self.style_checkBox.setChecked(bool(local[3]))
            self.sync_command.setText(local[4])

        if (paths and len(paths) >= 4 ) and (local and len(local) >= 4):
            self.spin_year.setEnabled(True)
            self.semester_comboBox.setEnabled(True)
            self.style_checkBox.setEnabled(True)
            self.input_grader_name.setEnabled(True)
            self.sync_command.setEnabled(True)
        # if (local and len(local) > 5) or len(paths):
        #         print('Obtained more settings than expected. Please check Ui_Create_settings_dialog.')

        self.buttonBox.button(self.buttonBox.Reset).setDisabled(True)
        self.buttonBox.button(self.buttonBox.Apply).setDisabled(True)

```

Here is the call graph for this function:


main.Ui_Create_settings _dialog.update_user_input _with_paths
main.Ui_Create_settings _dialog.read_settings_data
db_init.settings_db _read_settings

Here is the caller graph for this function:


main.Ui_Create_settings _dialog.update_user_input _with_paths
main.Ui_Create_settings _dialog.bind_functions
main.Ui_Create_settings _dialog.setupUi
main.Ui_manage_labs1.setupUi
main.Ui_Create_dates _dialog1.setupUi

7.10.3 Member Data Documentation

7.10.3.1 simple_diag

```
main.Ui_Create_settings_dialog.simple_diag
```

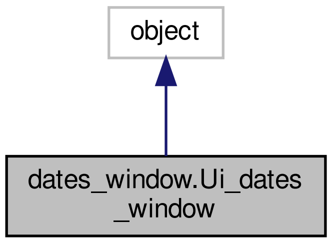
Definition at line 1544 of file main.py.

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

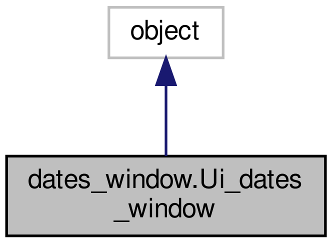
- main.py

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.

```
    def retranslateUi(self, dates_window):

        _translate = QtCore.QCoreApplication.translate
        dates_window.setWindowTitle(_translate("dates_window", "Check dates"))
        self.calendarWidget.setAccessibleName(_translate("dates_window", "cal_diag"))


```

7.11.2.2 setupUi()

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

Definition at line 12 of file dates_window.py.

```
    def setupUi(self, dates_window):
        dates_window.setObjectName("dates_window")
        dates_window.resize(251, 314)
        self.buttonBox = QtWidgets.QDialogButtonBox(dates_window)
        self.buttonBox.setGeometry(QtCore.QRect(40, 260, 191, 32))
        self.buttonBox.setOrientation(QtCore.Qt.Horizontal)
        self.buttonBox.setStandardButtons(QtWidgets.QDialogButtonBox.Cancel|QtWidgets.QDialogButtonBox.Ok)
        self.buttonBox.setObjectName("buttonBox")
        self.calendarWidget = QtWidgets.QCalendarWidget(dates_window)
        self.calendarWidget.setGeometry(QtCore.QRect(10, 10, 224, 232))
        self.calendarWidget.setObjectName("calendarWidget")

        self.retranslateUi(dates_window)
        self.buttonBox.accepted.connect(dates_window.accept)
        self.buttonBox.rejected.connect(dates_window.reject)
        QtCore.QMetaObject.connectSlotsByName(dates_window)

```

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
```

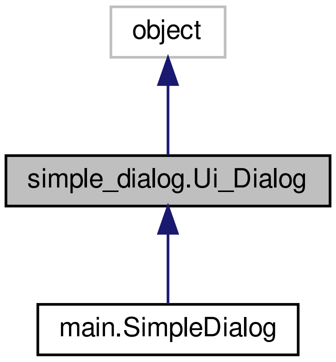
Definition at line 20 of file dates_window.py.

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

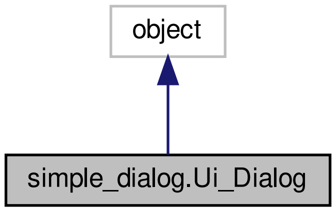
- dates_window.py

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.

```
    def retranslateUi(self, Dialog):

        _translate = QtCore.QCoreApplication.translate
        Dialog.setWindowTitle(_translate("Dialog", "Create database ?"))
        self.label_main_question.setText(_translate("Dialog", "Database will be created.  Confirm.."))


```

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
```

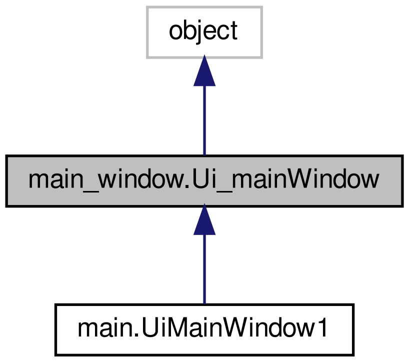
Definition at line 19 of file simple_dialog.py.

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

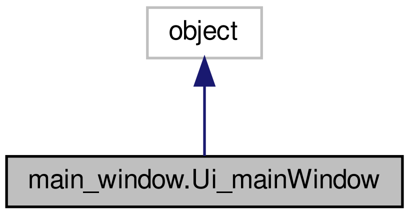
- simple_dialog.py

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.

```
    def retranslateUi(self, mainWindow):

        _translate = QtCore.QCoreApplication.translate
        mainWindow.setWindowTitle(_translate("mainWindow", "CSCI3130 grader"))
        self.input_file_location.setPlaceholderText(_translate("mainWindow", "Double click for path
    selection or paste|type path here"))
        self.but_file_open.setText(_translate("mainWindow", "Open"))
        self.but_begin.setText(_translate("mainWindow", "Begin"))
        self.label_from.setText(_translate("mainWindow", "From"))
        self.label_submitted.setText(_translate("mainWindow", "Submitted"))
        self.label_to.setText(_translate("mainWindow", "To"))
        self.label_current_id.setText(_translate("mainWindow", "current id"))
        self.label_attempt.setText(_translate("mainWindow", "attempt"))
        self.label_max_pos.setText(_translate("mainWindow", "lab max grade"))
        self.label_subtr.setText(_translate("mainWindow", "subtract"))
        self.label_final.setText(_translate("mainWindow", "final grade"))
        self.but_regrade.setText(_translate("mainWindow", "GRADE"))
        self.checkB_input_pin_status.setText(_translate("mainWindow", "Input direction"))
        self.checkB_output_pin_status.setText(_translate("mainWindow", "Output direction"))
        self.but_prev.setText(_translate("mainWindow", "prev"))
        self.checkB_wrong.setText(_translate("mainWindow", "WRONG"))
        self.but_reset.setText(_translate("mainWindow", "Reset"))
        self.but_next.setText(_translate("mainWindow", "next"))
        self.input_response_browser.setPlaceholderText(_translate("mainWindow", "Auto answer"))
        self.input_response_browser_user.setPlaceholderText(_translate("mainWindow", "User comment"))
        self.tabs_for_log_and_resp.setTabText(self.tabs_for_log_and_resp.indexOf(self.response_tab),
   _translate("mainWindow", "Response"))
        self.tabs_for_log_and_resp.setTabText(self.tabs_for_log_and_resp.indexOf(self.tab_prev_resp),
   _translate("mainWindow", "Previous Response"))
        self.tabs_for_log_and_resp.setTabText(self.tabs_for_log_and_resp.indexOf(self.tab_message_to_all),
   _translate("mainWindow", "Message to all"))
        self.tabs_for_log_and_resp.setTabText(self.tabs_for_log_and_resp.indexOf(self.log_tab), _translate(
   "mainWindow", "Log"))
        self.but_save_response.setText(_translate("mainWindow", "save responce"))
        self.check_autosave.setText(_translate("mainWindow", "autosave"))
        self.manage_labs_but.setText(_translate("mainWindow", "Manage labs"))
        self.set_style_checkbox.setText(_translate("mainWindow", "style"))
        self.settings_but.setText(_translate("mainWindow", "Settings"))
        self.but_save_all.setText(_translate("mainWindow", "save all"))
        self.but_create_report.setText(_translate("mainWindow", "Create reports"))
        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.

```
    def setupUi(self, mainWindow):
        mainWindow.setObjectName("mainWindow")
        mainWindow.setEnabled(True)
        mainWindow.resize(888, 584)
        icon = QtGui.QIcon()
        icon.addPixmap(QtGui.QPixmap("os_linux_1.ico"), QtGui.QIcon.Normal, QtGui.QIcon.Off)
        mainWindow.setWindowIcon(icon)
        mainWindow.setAccessibleName("")
        self.centralwidget = QtWidgets.QWidget(mainWindow)
        self.centralwidget.setObjectName("centralwidget")
        self.verticalLayout_7 = QtWidgets.QVBoxLayout(self.centralwidget)
        self.verticalLayout_7.setObjectName("verticalLayout_7")
        self.horizontalLayout_12 = QtWidgets.QHBoxLayout()
        self.horizontalLayout_12.setObjectName("horizontalLayout_12")
        self.input_file_location = BetterLineEdit(self.centralwidget)
        self.input_file_location.setEnabled(False)
        self.input_file_location.setLocale(QtCore.QLocale(QtCore.QLocale.English,
   QtCore.QLocale.UnitedStates))
```

```
        self.input_file_location.setText("")
        self.input_file_location.setObjectName("input_file_location")
        self.horizontalLayout_12.addWidget(self.input_file_location)
        self.filename_lineEdit = QtWidgets.QLineEdit(self.centralwidget)
        self.filename_lineEdit.setMaximumSize(QtCore.QSize(90, 16777215))
        self.filename_lineEdit.setReadOnly(True)
        self.filename_lineEdit.setObjectName("filename_lineEdit")
        self.horizontalLayout_12.addWidget(self.filename_lineEdit)
        self.but_file_open = QtWidgets.QPushButton(self.centralwidget)
        self.but_file_open.setEnabled(False)
        self.but_file_open.setLocale(QtCore.QLocale(QtCore.QLocale.English, QtCore.QLocale.UnitedStates))
        self.but_file_open.setObjectName("but_file_open")
        self.horizontalLayout_12.addWidget(self.but_file_open)
        self.but_begin = QtWidgets.QPushButton(self.centralwidget)
        self.but_begin.setEnabled(False)
        self.but_begin.setLocale(QtCore.QLocale(QtCore.QLocale.English, QtCore.QLocale.UnitedStates))
        self.but_begin.setCheckable(False)
        self.but_begin.setAutoDefault(False)
        self.but_begin.setDefault(False)
        self.but_begin.setFlat(False)
        self.but_begin.setObjectName("but_begin")
        self.horizontalLayout_12.addWidget(self.but_begin)
        self.verticalLayout_7.addLayout(self.horizontalLayout_12)
        self.horizontalLayout_7 = QtWidgets.QHBoxLayout()
        self.horizontalLayout_7.setSpacing(6)
        self.horizontalLayout_7.setObjectName("horizontalLayout_7")
        self.verticalLayout = QtWidgets.QVBoxLayout()
        self.verticalLayout.setObjectName("verticalLayout")
        self.horizontalLayout = QtWidgets.QHBoxLayout()
        self.horizontalLayout.setObjectName("horizontalLayout")
        self.label_from = QtWidgets.QLabel(self.centralwidget)
        self.label_from.setObjectName("label_from")
        self.horizontalLayout.addWidget(self.label_from)
        self.dateTimeEdit_from = QtWidgets.QDateTimeEdit(self.centralwidget)
        self.dateTimeEdit_from.setEnabled(True)
        self.dateTimeEdit_from.setWrapping(False)
        self.dateTimeEdit_from.setReadOnly(True)
        self.dateTimeEdit_from.setAccelerated(False)
        self.dateTimeEdit_from.setCalendarPopup(True)
        self.dateTimeEdit_from.setObjectName("dateTimeEdit_from")
        self.horizontalLayout.addWidget(self.dateTimeEdit_from)
        self.verticalLayout.addLayout(self.horizontalLayout)
        self.horizontalLayout_2 = QtWidgets.QHBoxLayout()
        self.horizontalLayout_2.setObjectName("horizontalLayout_2")
        self.label_submitted = QtWidgets.QLabel(self.centralwidget)
        self.label_submitted.setObjectName("label_submitted")
        self.horizontalLayout_2.addWidget(self.label_submitted)
        self.dateTimeEdit_submitted = QtWidgets.QDateTimeEdit(self.centralwidget)
        self.dateTimeEdit_submitted.setEnabled(True)
        self.dateTimeEdit_submitted.setWrapping(False)
        self.dateTimeEdit_submitted.setFrame(True)
        self.dateTimeEdit_submitted.setReadOnly(True)
        self.dateTimeEdit_submitted.setKeyboardTracking(False)
        self.dateTimeEdit_submitted.setCalendarPopup(True)
        self.dateTimeEdit_submitted.setObjectName("dateTimeEdit_submitted")
        self.horizontalLayout_2.addWidget(self.dateTimeEdit_submitted)
        self.verticalLayout.addLayout(self.horizontalLayout_2)
        self.horizontalLayout_3 = QtWidgets.QHBoxLayout()
        self.horizontalLayout_3.setObjectName("horizontalLayout_3")
        self.label_to = QtWidgets.QLabel(self.centralwidget)
        self.label_to.setObjectName("label_to")
        self.horizontalLayout_3.addWidget(self.label_to)
        self.dateTimeEdit_to = QtWidgets.QDateTimeEdit(self.centralwidget)
        self.dateTimeEdit_to.setEnabled(True)
        self.dateTimeEdit_to.setReadOnly(True)
        self.dateTimeEdit_to.setCalendarPopup(True)
        self.dateTimeEdit_to.setObjectName("dateTimeEdit_to")
        self.horizontalLayout_3.addWidget(self.dateTimeEdit_to)
        self.verticalLayout.addLayout(self.horizontalLayout_3)
        self.horizontalLayout_7.addLayout(self.verticalLayout)
        self.verticalLayout_3 = QtWidgets.QVBoxLayout()
        self.verticalLayout_3.setObjectName("verticalLayout_3")
        self.horizontalLayout_8 = QtWidgets.QHBoxLayout()
        self.horizontalLayout_8.setObjectName("horizontalLayout_8")
        self.input_current_id = QtWidgets.QLineEdit(self.centralwidget)
        self.input_current_id.setEnabled(False)
        self.input_current_id.setMaximumSize(QtCore.QSize(60, 40))
        self.input_current_id.setReadOnly(True)
        self.input_current_id.setObjectName("input_current_id")
        self.horizontalLayout_8.addWidget(self.input_current_id)
        self.label_current_id = QtWidgets.QLabel(self.centralwidget)
        self.label_current_id.setObjectName("label_current_id")
        self.horizontalLayout_8.addWidget(self.label_current_id)
        self.verticalLayout_3.addLayout(self.horizontalLayout_8)
        self.horizontalLayout_9 = QtWidgets.QHBoxLayout()
        self.horizontalLayout_9.setObjectName("horizontalLayout_9")
        self.input_attempt = QtWidgets.QLineEdit(self.centralwidget)
        self.input_attempt.setEnabled(False)
        self.input_attempt.setMaximumSize(QtCore.QSize(40, 40))
        self.input_attempt.setReadOnly(True)
        self.input_attempt.setObjectName("input_attempt")
        self.horizontalLayout_9.addWidget(self.input_attempt)
        spacerItem = QtWidgets.QSpacerItem(20, 20, QtWidgets.QSizePolicy.Fixed,
QtWidgets.QSizePolicy.Minimum)
        self.horizontalLayout_9.addItem(spacerItem)
        self.label_attempt = QtWidgets.QLabel(self.centralwidget)
        self.label_attempt.setObjectName("label_attempt")
        self.horizontalLayout_9.addWidget(self.label_attempt)
        self.verticalLayout_3.addLayout(self.horizontalLayout_9)
        spacerItem1 = QtWidgets.QSpacerItem(20, 40, QtWidgets.QSizePolicy.Minimum,
QtWidgets.QSizePolicy.Fixed)
        self.verticalLayout_3.addItem(spacerItem1)
        self.horizontalLayout_7.addLayout(self.verticalLayout_3)
        self.verticalLayout_2 = QtWidgets.QVBoxLayout()
        self.verticalLayout_2.setObjectName("verticalLayout_2")
        self.horizontalLayout_6 = QtWidgets.QHBoxLayout()
        self.horizontalLayout_6.setObjectName("horizontalLayout_6")
        self.input_max_pos_grade = QtWidgets.QLineEdit(self.centralwidget)
        self.input_max_pos_grade.setEnabled(False)
        self.input_max_pos_grade.setMaximumSize(QtCore.QSize(40, 40))
        self.input_max_pos_grade.setLocale(QtCore.QLocale(QtCore.QLocale.English,
QtCore.QLocale.UnitedStates))
        self.input_max_pos_grade.setText("")
        self.input_max_pos_grade.setReadOnly(True)
        self.input_max_pos_grade.setObjectName("input_max_pos_grade")
        self.horizontalLayout_6.addWidget(self.input_max_pos_grade)
        self.label_max_pos = QtWidgets.QLabel(self.centralwidget)
        self.label_max_pos.setEnabled(True)
        self.label_max_pos.setLocale(QtCore.QLocale(QtCore.QLocale.English, QtCore.QLocale.UnitedStates))
        self.label_max_pos.setObjectName("label_max_pos")
        self.horizontalLayout_6.addWidget(self.label_max_pos)
        self.verticalLayout_2.addLayout(self.horizontalLayout_6)
        self.horizontalLayout_4 = QtWidgets.QHBoxLayout()
        self.horizontalLayout_4.setObjectName("horizontalLayout_4")
        self.input_subtract = QtWidgets.QLineEdit(self.centralwidget)
        self.input_subtract.setEnabled(False)
        self.input_subtract.setMaximumSize(QtCore.QSize(40, 40))
        self.input_subtract.setLocale(QtCore.QLocale(QtCore.QLocale.English, QtCore.QLocale.UnitedStates))
        self.input_subtract.setReadOnly(True)
        self.input_subtract.setObjectName("input_subtract")
        self.horizontalLayout_4.addWidget(self.input_subtract)
        self.label_subtr = QtWidgets.QLabel(self.centralwidget)
        self.label_subtr.setLocale(QtCore.QLocale(QtCore.QLocale.English, QtCore.QLocale.UnitedStates))
        self.label_subtr.setObjectName("label_subtr")
        self.horizontalLayout_4.addWidget(self.label_subtr)
        self.verticalLayout_2.addLayout(self.horizontalLayout_4)
        self.horizontalLayout_5 = QtWidgets.QHBoxLayout()
        self.horizontalLayout_5.setObjectName("horizontalLayout_5")
        self.input_final_grade = QtWidgets.QLineEdit(self.centralwidget)
        self.input_final_grade.setEnabled(False)
        self.input_final_grade.setMaximumSize(QtCore.QSize(40, 40))
        self.input_final_grade.setLocale(QtCore.QLocale(QtCore.QLocale.English, QtCore.QLocale.UnitedStates
))
        self.input_final_grade.setText("")
        self.input_final_grade.setReadOnly(True)
        self.input_final_grade.setObjectName("input_final_grade")
        self.horizontalLayout_5.addWidget(self.input_final_grade)
        self.label_final = QtWidgets.QLabel(self.centralwidget)
        self.label_final.setEnabled(True)
        self.label_final.setLocale(QtCore.QLocale(QtCore.QLocale.English, QtCore.QLocale.UnitedStates))
        self.label_final.setObjectName("label_final")
        self.horizontalLayout_5.addWidget(self.label_final)
        self.verticalLayout_2.addLayout(self.horizontalLayout_5)
        self.horizontalLayout_7.addLayout(self.verticalLayout_2)
        self.verticalLayout_4 = QtWidgets.QVBoxLayout()
        self.verticalLayout_4.setObjectName("verticalLayout_4")
        self.but_regrade = QtWidgets.QPushButton(self.centralwidget)
        self.but_regrade.setEnabled(False)
        self.but_regrade.setLocale(QtCore.QLocale(QtCore.QLocale.English, QtCore.QLocale.UnitedStates))
        self.but_regrade.setObjectName("but_regrade")
        self.verticalLayout_4.addWidget(self.but_regrade)
        self.checkB_input_pin_status = QtWidgets.QCheckBox(self.centralwidget)
        self.checkB_input_pin_status.setEnabled(False)
        self.checkB_input_pin_status.setLocale(QtCore.QLocale(QtCore.QLocale.English,
    QtCore.QLocale.UnitedStates))
        self.checkB_input_pin_status.setObjectName("checkB_input_pin_status")
        self.verticalLayout_4.addWidget(self.checkB_input_pin_status)
        self.checkB_output_pin_status = QtWidgets.QCheckBox(self.centralwidget)
        self.checkB_output_pin_status.setEnabled(False)
        self.checkB_output_pin_status.setLocale(QtCore.QLocale(QtCore.QLocale.English,
    QtCore.QLocale.UnitedStates))
        self.checkB_output_pin_status.setObjectName("checkB_output_pin_status")
        self.verticalLayout_4.addWidget(self.checkB_output_pin_status)
        self.horizontalLayout_7.addLayout(self.verticalLayout_4)
        self.verticalLayout_7.addLayout(self.horizontalLayout_7)
        self.horizontalLayout_10 = QtWidgets.QHBoxLayout()
        self.horizontalLayout_10.setSpacing(65)
        self.horizontalLayout_10.setObjectName("horizontalLayout_10")
        self.but_prev = QtWidgets.QPushButton(self.centralwidget)
        self.but_prev.setEnabled(False)
        self.but_prev.setMinimumSize(QtCore.QSize(60, 30))
        self.but_prev.setMaximumSize(QtCore.QSize(200, 16777215))
        self.but_prev.setLocale(QtCore.QLocale(QtCore.QLocale.English, QtCore.QLocale.UnitedStates))
        self.but_prev.setObjectName("but_prev")
        self.horizontalLayout_10.addWidget(self.but_prev)
        self.checkB_wrong = QtWidgets.QCheckBox(self.centralwidget)
        self.checkB_wrong.setEnabled(False)
        self.checkB_wrong.setMinimumSize(QtCore.QSize(80, 20))
        self.checkB_wrong.setMaximumSize(QtCore.QSize(75, 16777215))
        self.checkB_wrong.setLocale(QtCore.QLocale(QtCore.QLocale.English, QtCore.QLocale.UnitedStates))
        self.checkB_wrong.setObjectName("checkB_wrong")
        self.horizontalLayout_10.addWidget(self.checkB_wrong)
        self.but_reset = QtWidgets.QPushButton(self.centralwidget)
        self.but_reset.setEnabled(False)
        self.but_reset.setMinimumSize(QtCore.QSize(60, 20))
        self.but_reset.setMaximumSize(QtCore.QSize(90, 16777215))
        self.but_reset.setObjectName("but_reset")
        self.horizontalLayout_10.addWidget(self.but_reset)
        self.but_next = QtWidgets.QPushButton(self.centralwidget)
        self.but_next.setEnabled(False)
        self.but_next.setMinimumSize(QtCore.QSize(60, 30))
        self.but_next.setMaximumSize(QtCore.QSize(200, 16777215))
        self.but_next.setLocale(QtCore.QLocale(QtCore.QLocale.English, QtCore.QLocale.UnitedStates))
        self.but_next.setObjectName("but_next")
        self.horizontalLayout_10.addWidget(self.but_next)
        self.verticalLayout_7.addLayout(self.horizontalLayout_10)
        self.popular_answers = QtWidgets.QComboBox(self.centralwidget)
        self.popular_answers.setEnabled(False)
        self.popular_answers.setLocale(QtCore.QLocale(QtCore.QLocale.English, QtCore.QLocale.UnitedStates))
        self.popular_answers.setEditable(False)
        self.popular_answers.setCurrentText("")
        self.popular_answers.setObjectName("popular_answers")
        self.popular_answers.addItem("")
        self.popular_answers.setItemText(0, "")
        self.verticalLayout_7.addWidget(self.popular_answers)
        self.tabs_for_log_and_resp = QtWidgets.QTabWidget(self.centralwidget)
        self.tabs_for_log_and_resp.setEnabled(True)
        self.tabs_for_log_and_resp.setMinimumSize(QtCore.QSize(770, 30))
        self.tabs_for_log_and_resp.setMaximumSize(QtCore.QSize(20000, 3700))
        self.tabs_for_log_and_resp.setLocale(QtCore.QLocale(QtCore.QLocale.English,
    QtCore.QLocale.UnitedStates))
        self.tabs_for_log_and_resp.setTabShape(QtWidgets.QTabWidget.Rounded)
        self.tabs_for_log_and_resp.setObjectName("tabs_for_log_and_resp")
        self.response_tab = QtWidgets.QWidget()
        self.response_tab.setMinimumSize(QtCore.QSize(0, 180))
        self.response_tab.setMaximumSize(QtCore.QSize(16777215, 300))
        self.response_tab.setObjectName("response_tab")
        self.verticalLayout_9 = QtWidgets.QVBoxLayout(self.response_tab)
        self.verticalLayout_9.setObjectName("verticalLayout_9")
        self.splitter = QtWidgets.QSplitter(self.response_tab)
        sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Expanding, QtWidgets.QSizePolicy.Expanding
    )
        sizePolicy.setHorizontalStretch(0)
        sizePolicy.setVerticalStretch(0)
        sizePolicy.setHeightForWidth(self.splitter.sizePolicy().hasHeightForWidth())
        self.splitter.setSizePolicy(sizePolicy)
        self.splitter.setOrientation(QtCore.Qt.Vertical)
        self.splitter.setObjectName("splitter")
        self.input_response_browser = QtWidgets.QPlainTextEdit(self.splitter)
        self.input_response_browser.setEnabled(True)
        self.input_response_browser.setMinimumSize(QtCore.QSize(0, 30))
        self.input_response_browser.setReadOnly(True)
        self.input_response_browser.setTextInteractionFlags(QtCore.Qt.TextSelectableByKeyboard|
QtCore.Qt.TextSelectableByMouse)
        self.input_response_browser.setObjectName("input_response_browser")
        self.input_response_browser_user = BetterPlainTextEdit(self.splitter)
        self.input_response_browser_user.setEnabled(False)
        self.input_response_browser_user.setMinimumSize(QtCore.QSize(0, 30))
        self.input_response_browser_user.setObjectName("input_response_browser_user")
        self.verticalLayout_9.addWidget(self.splitter)
        self.tabs_for_log_and_resp.addTab(self.response_tab, "")
        self.tab_prev_resp = QtWidgets.QWidget()
        self.tab_prev_resp.setObjectName("tab_prev_resp")
        self.verticalLayout_5 = QtWidgets.QVBoxLayout(self.tab_prev_resp)
        self.verticalLayout_5.setObjectName("verticalLayout_5")
        self.input_prev_response = QtWidgets.QPlainTextEdit(self.tab_prev_resp)
        self.input_prev_response.setEnabled(True)
        self.input_prev_response.setTextInteractionFlags(QtCore.Qt.TextSelectableByKeyboard|
QtCore.Qt.TextSelectableByMouse)
        self.input_prev_response.setObjectName("input_prev_response")
        self.verticalLayout_5.addWidget(self.input_prev_response)
        self.tabs_for_log_and_resp.addTab(self.tab_prev_resp, "")
        self.tab_message_to_all = QtWidgets.QWidget()
        self.tab_message_to_all.setObjectName("tab_message_to_all")
        self.verticalLayout_8 = QtWidgets.QVBoxLayout(self.tab_message_to_all)
        self.verticalLayout_8.setObjectName("verticalLayout_8")
        self.input_message_to_all = QtWidgets.QPlainTextEdit(self.tab_message_to_all)
        sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Expanding, QtWidgets.QSizePolicy.Expanding
)
        sizePolicy.setHorizontalStretch(0)
        sizePolicy.setVerticalStretch(0)
        sizePolicy.setHeightForWidth(self.input_message_to_all.sizePolicy().hasHeightForWidth())
        self.input_message_to_all.setSizePolicy(sizePolicy)
        self.input_message_to_all.setObjectName("input_message_to_all")
        self.verticalLayout_8.addWidget(self.input_message_to_all)
        self.tabs_for_log_and_resp.addTab(self.tab_message_to_all, "")
        self.log_tab = QtWidgets.QWidget()
        self.log_tab.setObjectName("log_tab")
        self.verticalLayout_6 = QtWidgets.QVBoxLayout(self.log_tab)
        self.verticalLayout_6.setObjectName("verticalLayout_6")
        self.input_log_browser = QtWidgets.QTextBrowser(self.log_tab)
        self.input_log_browser.setLocale(QtCore.QLocale(QtCore.QLocale.English, QtCore.QLocale.UnitedStates
))
        self.input_log_browser.setObjectName("input_log_browser")
        self.verticalLayout_6.addWidget(self.input_log_browser)
        self.tabs_for_log_and_resp.addTab(self.log_tab, "")
        self.verticalLayout_7.addWidget(self.tabs_for_log_and_resp)
        self.horizontalLayout_11 = QtWidgets.QHBoxLayout()
        self.horizontalLayout_11.setObjectName("horizontalLayout_11")
        self.but_save_response = QtWidgets.QPushButton(self.centralwidget)
        self.but_save_response.setEnabled(False)
        self.but_save_response.setLocale(QtCore.QLocale(QtCore.QLocale.English, QtCore.QLocale.UnitedStates
))
        self.but_save_response.setObjectName("but_save_response")
        self.horizontalLayout_11.addWidget(self.but_save_response)
        self.check_autosave = QtWidgets.QCheckBox(self.centralwidget)
        self.check_autosave.setEnabled(False)
        self.check_autosave.setObjectName("check_autosave")
        self.horizontalLayout_11.addWidget(self.check_autosave)
        self.manage_labs_but = QtWidgets.QPushButton(self.centralwidget)
        self.manage_labs_but.setEnabled(False)
        self.manage_labs_but.setObjectName("manage_labs_but")
        self.horizontalLayout_11.addWidget(self.manage_labs_but)
        self.set_style_checkbox = QtWidgets.QCheckBox(self.centralwidget)
        self.set_style_checkbox.setObjectName("set_style_checkbox")
        self.horizontalLayout_11.addWidget(self.set_style_checkbox)
        self.settings_but = QtWidgets.QToolButton(self.centralwidget)
        self.settings_but.setEnabled(True)
        self.settings_but.setObjectName("settings_but")
        self.horizontalLayout_11.addWidget(self.settings_but)
        self.but_save_all = QtWidgets.QPushButton(self.centralwidget)
        self.but_save_all.setEnabled(False)
        self.but_save_all.setLocale(QtCore.QLocale(QtCore.QLocale.English, QtCore.QLocale.UnitedStates))
        self.but_save_all.setObjectName("but_save_all")
        self.horizontalLayout_11.addWidget(self.but_save_all)
        self.but_create_report = QtWidgets.QPushButton(self.centralwidget)
        self.but_create_report.setEnabled(False)
        self.but_create_report.setObjectName("but_create_report")
        self.horizontalLayout_11.addWidget(self.but_create_report)
        self.verticalLayout_7.addLayout(self.horizontalLayout_11)
        self.progressBar = QtWidgets.QProgressBar(self.centralwidget)
        self.progressBar.setEnabled(True)
        self.progressBar.setAutoFillBackground(False)
        self.progressBar.setLocale(QtCore.QLocale(QtCore.QLocale.English, QtCore.QLocale.UnitedStates))
        self.progressBar.setProperty("value", 0)
        self.progressBar.setTextVisible(True)
        self.progressBar.setInvertedAppearance(False)
        self.progressBar.setObjectName("progressBar")
        self.verticalLayout_7.addWidget(self.progressBar)
        mainWindow.setCentralWidget(self.centralwidget)

        self.retranslateUi(mainWindow)
        self.tabs_for_log_and_resp.setCurrentIndex(0)
        QtCore.QMetaObject.connectSlotsByName(mainWindow)

```

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`

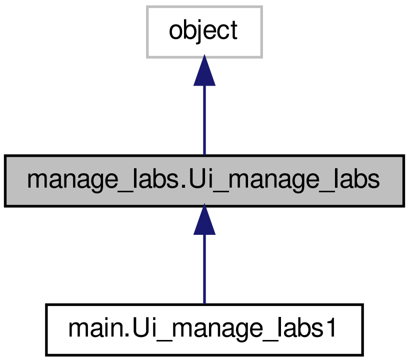
Definition at line 249 of file main_window.py.

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

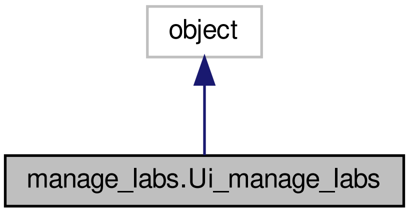
- main_window.py

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.

```
    def retranslateUi(self, manage_labs):

        _translate = QtCore.QCoreApplication.translate
        manage_labs.setWindowTitle(_translate("manage_labs", "Manage labs"))
        self.sync_but.setText(_translate("manage_labs", "Sync to local storage"))
        self.import_but.setText(_translate("manage_labs", "import labs"))
        self.create_due_dates_but.setText(_translate("manage_labs", "Create due dates"))
        self.export_but.setText(_translate("manage_labs", "Export pdfs"))


```

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`

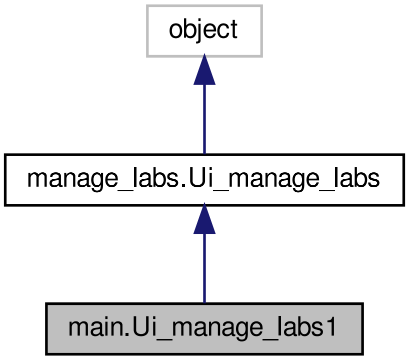
Definition at line 16 of file manage_labs.py.

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

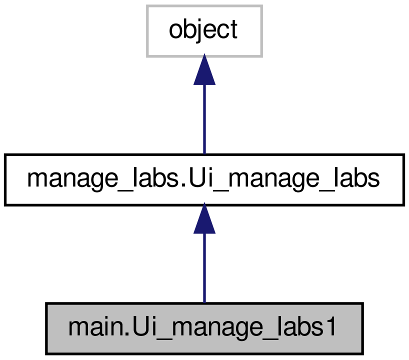
- manage_labs.py

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

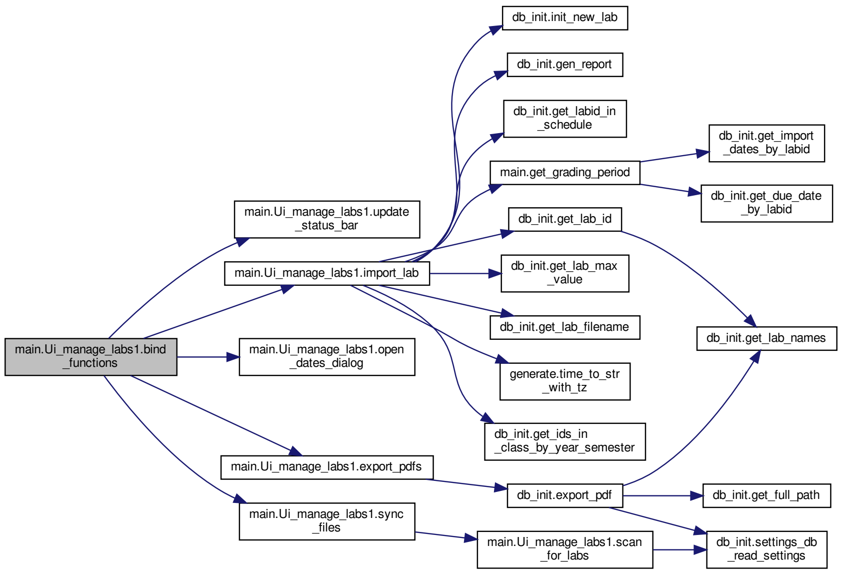
7.15.2.1 bind_functions()

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

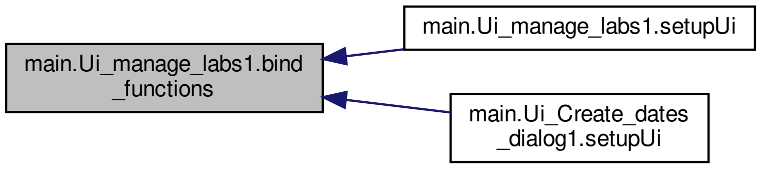
Definition at line 1580 of file main.py.

```
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:


db_init.init_new_lab
db_init.gen_report
db_init.get_labid_in_schedule
db_init.get_import_dates_by_labid
main.get_grading_period
db_init.get_due_date_by_labid
main.Ui_manage_labs1.update_status_bar
db_init.get_lab_id
main.Ui_manage_labs1.import_lab
db_init.get_lab_max_value
db_init.get_lab_filename
db_init.get_lab_names
main.Ui_manage_labs1.bind_functions
main.Ui_manage_labs1.open_dates_dialog
generate.time_to_str_with_tz
db_init.get_ids_in_class_by_year_semester
main.Ui_manage_labs1.export_pdfs
db_init.export_pdf
db_init.get_full_path
main.Ui_manage_labs1.sync_files
main.Ui_manage_labs1.scan_for_labs
db_init.settings_db_read_settings

Here is the caller graph for this function:


main.Ui_manage_labs1.bind_functions
main.Ui_manage_labs1.setupUi
main.Ui_Create_dates_dialog1.setupUi

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.

```
    def due_date_creator(self, due_location, due_dates):
        if len(due_location) > 1:
            i = 1
            for due_date in due_dates:
                with open('%sdue_%d_%d' % (due_location, i, due_date), 'w'):
                    i += 1
        else:
            print('Location was not specified.')

```

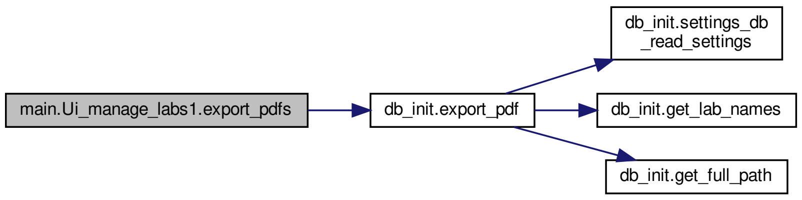
7.15.2.4 export_pdfs()

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

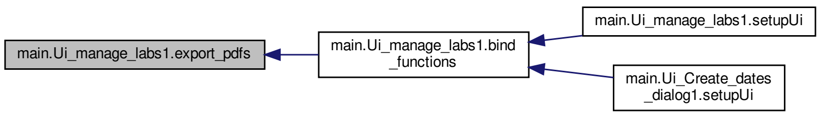
Definition at line 1865 of file main.py.

```
    def export_pdfs(self):
        self.export_but.setDisabled(True)
        self.export_but.setText('Exporting..')
        self.export_but.repaint()
        export_pdf()
        self.export_but.setText('Export pdfs')
        self.export_but.setEnabled(True)


```

Here is the call graph for this function:


main.Ui_manage_labs1.export_pdfs
db_init.export_pdf
db_init.settings_db _read_settings
db_init.get_lab_names
db_init.get_full_path

Here is the caller graph for this function:


main.Ui_manage_labs1.export_pdfs
main.Ui_manage_labs1.bind_functions
main.Ui_manage_labs1.setupUi
main.Ui_Create_dates_dialog1.setupUi

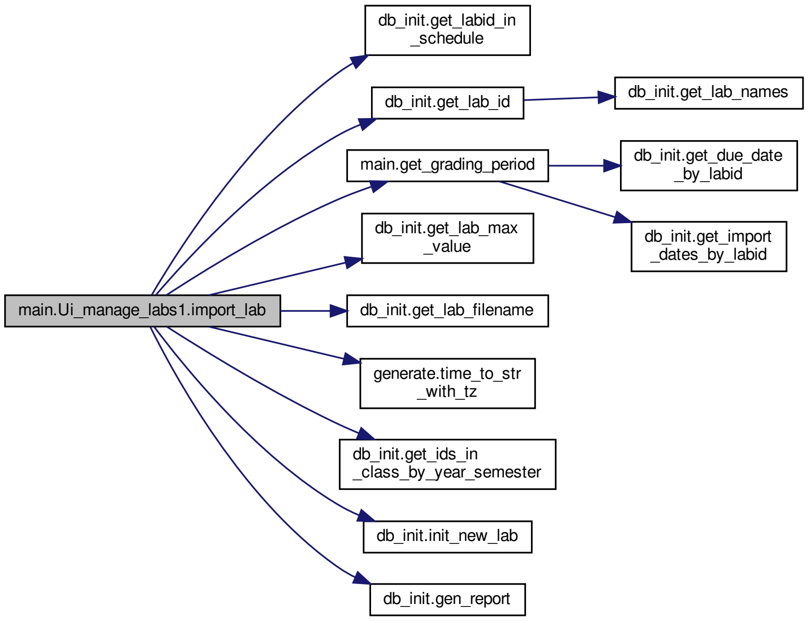
7.15.2.5 import_lab()

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

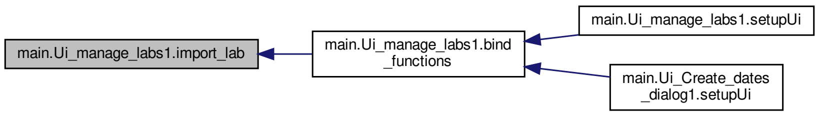
Definition at line 1664 of file main.py.

```
    def import_lab(self):
        if self.selected_path:
            self.import_but.setDisabled(True)
            self.import_but.setText('Importing..')
            self.import_but.repaint()

            # due_file = self.check_for_due_dates(self.selected_path)
            if False:
            # if len(due_file) < 4:
                self.status_bar.setText('Create due dates !')
                self.import_but.setText('Import labs')
                self.import_but.setEnabled(True)
                return False
            else:
                from shutil import copy2 as cp2
                zip_files = [f for f in os.listdir(self.selected_path) if 'zip' in f]
                real_zip_files_rev = sorted([f for f in zip_files if os.path.isfile(os.path.join(self.
      selected_path, f))], reverse=True)

                year, semester = self.main_lab_path.split('/')[-1].split('_')
                ltype, _, lab_num = self.selected_lab_name.split('_')
                lid = get_labid_in_schedule(get_lab_id(ltype, int(lab_num)),
       year, semester)
                if lid is None:
                    self.status_bar.setText('Create due dates ! Lab is not initialised in lab_schedule')
                    self.import_but.setText('Import labs')
                    self.import_but.setEnabled(True)
                    return False
                current_check, prev_due, next_due, current_timestamp =
      get_grading_period(lid)

                if current_check > 4:
                    self.status_bar.setText('This lab has no more resubmissions (graded 4 times).')
                    self.import_but.setText('Import labs')
                    self.import_but.setEnabled(True)
                    return False

                if current_timestamp < next_due:
                    # we cannot grade before the due date
                    self.status_bar.setText('Current date is less than next due date. It is too early to
       import.')
                    self.import_but.setText('Import labs')
                    self.import_but.setEnabled(True)
                    return False



                penalty_mess = ''
                if current_check == 1:
                    penalty_mess = '100% - this is your max point(no resubmissions)'
                elif current_check == 2:
                    penalty_mess = '90% - first resubmission'
                elif current_check == 3:
                    penalty_mess = '70% - second resubmission'
                elif current_check == 4:
                    penalty_mess = '50% - third resubmission'

                lab_type, _, lab_num = self.selected_lab_name.split('_')
                lab_corr_name = lab_type[0] + 'LA' + lab_num
                max_points = get_lab_max_value(lab_corr_name)
                lab_filename = get_lab_filename(lab_corr_name)

                # temporary solution. path should be stored as local var
                paths_to_grading_dir = self.main_lab_path + '/' + self.selected_lab_name + '_' + str(
current_check) + '/'

                # proc_time = datetime.utcfromtimestamp(current_timestamp).strftime('%Y-%m-%d %H:%M:%S')
                proc_time = time_to_str_with_tz(current_timestamp)

                # File manipulations goes below:

                if not os.path.isdir(paths_to_grading_dir):
                    os.makedirs(paths_to_grading_dir)

                cur_year, cur_sem = paths_to_grading_dir.split('/')[-3].split('_')
                id_to_classId = get_ids_in_class_by_year_semester(cur_year
, cur_sem)[0]
                imported_files_counter = 0

                selected_files = []
                for file in real_zip_files_rev:
                    parts = file.split('.')[0].split('-')
                    if int(parts[2]) > prev_due and int(parts[2]) <= next_due:
                        if len(selected_files) == 0:
                            selected_files.append(file)
                        elif selected_files[-1].split('.')[0].split('-')[0] != parts[0]:
                            selected_files.append(file)

                for file in reversed(selected_files):
                    zipped_file = zipfile.ZipFile(self.selected_path + file)
                    extraction_dir = paths_to_grading_dir + file.split('.')[0]
                    try:
                        zipped_file.extractall(paths_to_grading_dir + file.split('.')[0])
                    except Exception as e:
                        print(self.selected_path + file)
                        print(e)
                    finally:
                        zipped_file.close()
                    parts = file.split('.')[0].split('-')
                    subm_int = int(extraction_dir.split('-')[-1])
                    # subm_time =
 datetime.utcfromtimestamp(subm_int).replace(tzinfo=tz.tzutc()).astimezone(tz.tzlocal()).strftime('%Y-%m-%d %H:%M:%S')
                    subm_time = time_to_str_with_tz(subm_int)
                    # check for required files
                    if not lab_filename[0] or os.path.isfile(extraction_dir + '/' + lab_filename[0]):
                        lab_responce = 'I did not find any errors. Good job !'
                        cur_grade = max_points
                    else:
                        lab_responce = 'File "' + lab_filename[0] +'" was not found.\nThese files were
 found: ' +\
                                       " ".join(os.listdir(extraction_dir))
                        cur_grade = 0

                    # This check is for a case when you graded the lab and trying to import it again.
                    # No existing files should be wiped
                    if not os.path.isfile(extraction_dir+'/penalty.txt'):
                        with open(extraction_dir+'/penalty.txt', 'w') as f:
                            f.write(penalty_mess)

                    if not os.path.isfile(extraction_dir + '/grade.txt'):
                        with open(extraction_dir + '/grade.txt', 'w') as f:
                            f.write(str(cur_grade))

                    if not os.path.isfile(extraction_dir + '/responce.txt'):
                        with open(extraction_dir + '/responce.txt', 'w') as f:
                            f.write(lab_responce)

                    if not os.path.isfile(extraction_dir + '/tech_info.txt'):
                        with open(extraction_dir + '/tech_info.txt', 'w') as f:
                            f.writelines(['File was submited at %s<br/>\n' % subm_time,
                                          'I started processing your file at %s<br/>\n' % proc_time,
                                           "I found that your lab type is '%s' and it's number is %s <br/>"
   % (lab_type, lab_num),
                                           'So max points for this lab type is <u>%d</u><br/>' % max_points,
                                           'Theoretical max points: %s)' % penalty_mess])

                    init_new_lab(id_to_classId[parts[0]], lid, current_check, subm_int,
   extraction_dir)
                    imported_files_counter += 1

                # cp2(self.selected_path + due_file[current_check-1], paths_to_grading_dir)

                # check_filename = paths_to_grading_dir + 'check_' + str(current_check) + '_' +
    str(current_timestamp)
                # with open(check_filename, 'w'): pass
                gen_report(lid, att=current_check)

                # cp2(check_filename, self.selected_path)

                self.import_but.setEnabled(True)
                self.import_but.setText('Import labs')
                self.status_bar.setText("Imported " + str(imported_files_counter) + " files.")
                return True

        return False
```

Here is the call graph for this function:


db_init.get_labid_in _schedule
db_init.get_lab_id
db_init.get_lab_names
main.get_grading_period
db_init.get_due_date _by_labid
db_init.get_lab_max _value
db_init.get_import _dates_by_labid
main.Ui_manage_labs1.import_lab
db_init.get_lab_filename
generate.time_to_str _with_tz
db_init.get_ids_in _class_by_year_semester
db_init.init_new_lab
db_init.gen_report

Here is the caller graph for this function:


main.Ui_manage_labs1.import_lab
main.Ui_manage_labs1.bind_functions
main.Ui_manage_labs1.setupUi
main.Ui_Create_dates_dialog1.setupUi

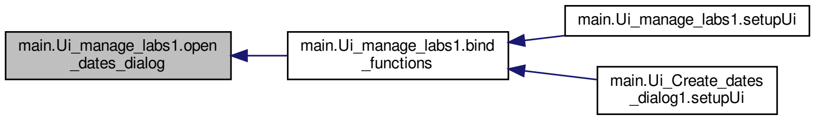
7.15.2.6 open_dates_dialog()

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

Definition at line 1827 of file main.py.

```
    def open_dates_dialog(self):
        self.create_due_dates_but.setDisabled(True)
        self.create_due_dates_but.repaint()
        self.cal_window = QtWidgets.QDialog()
        dui = Ui_Create_dates_dialog1()
        dui.setupUi(self.cal_window, self.selected_lab_name)
        # self.cal_window.finished.connect(self.check_new_win_result)
        self.cal_window.show()
        accepted = self.cal_window.exec_()
        if accepted:
            due_dates = list()
            due_dates.append(dui.init_subm_date_time.dateTime().toTime_t())
            due_dates.append(dui.first_subm_date_time.dateTime().toTime_t())
            due_dates.append(dui.second_subm_date_time.dateTime().toTime_t())
            due_dates.append(dui.third_subm_date_time.dateTime().toTime_t())
            due_location = dui.lab_path.text()
            self.due_date_creator(due_location, due_dates)
            year, semester = self.main_lab_path.split('/')[-1].split('_')
            ltype, _, lab_num = self.selected_lab_name.split('_')
            register_lab_in_semester(ltype, lab_num, year, semester, due_dates)
        self.create_due_dates_but.setEnabled(True)

```

Here is the caller graph for this function:


main.Ui_manage_labs1.open_dates_dialog
main.Ui_manage_labs1.bind_functions
main.Ui_manage_labs1.setupUi
main.Ui_Create_dates_dialog1.setupUi

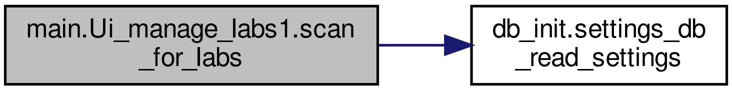
7.15.2.7 scan_for_labs()

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

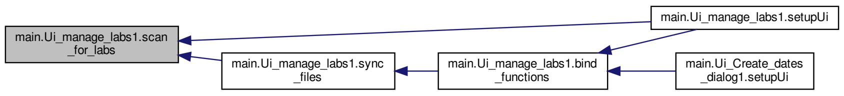
Definition at line 1651 of file main.py.

```
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.addItems(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:


main.Ui_manage_labs1.scan_for_labs
db_init.settings_db_read_settings

Here is the caller graph for this function:


main.Ui_manage_labs1.scan_for_labs
main.Ui_manage_labs1.setupUi
main.Ui_manage_labs1.sync_files
main.Ui_manage_labs1.bind_functions
main.Ui_Create_dates_dialog1.setupUi

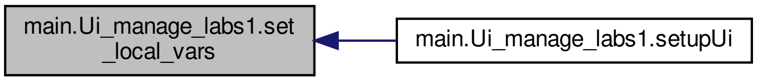
7.15.2.8 set_local_vars()

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

Definition at line 1604 of file main.py.

```
    def set_local_vars(self):
        pass

```

Here is the caller graph for this function:



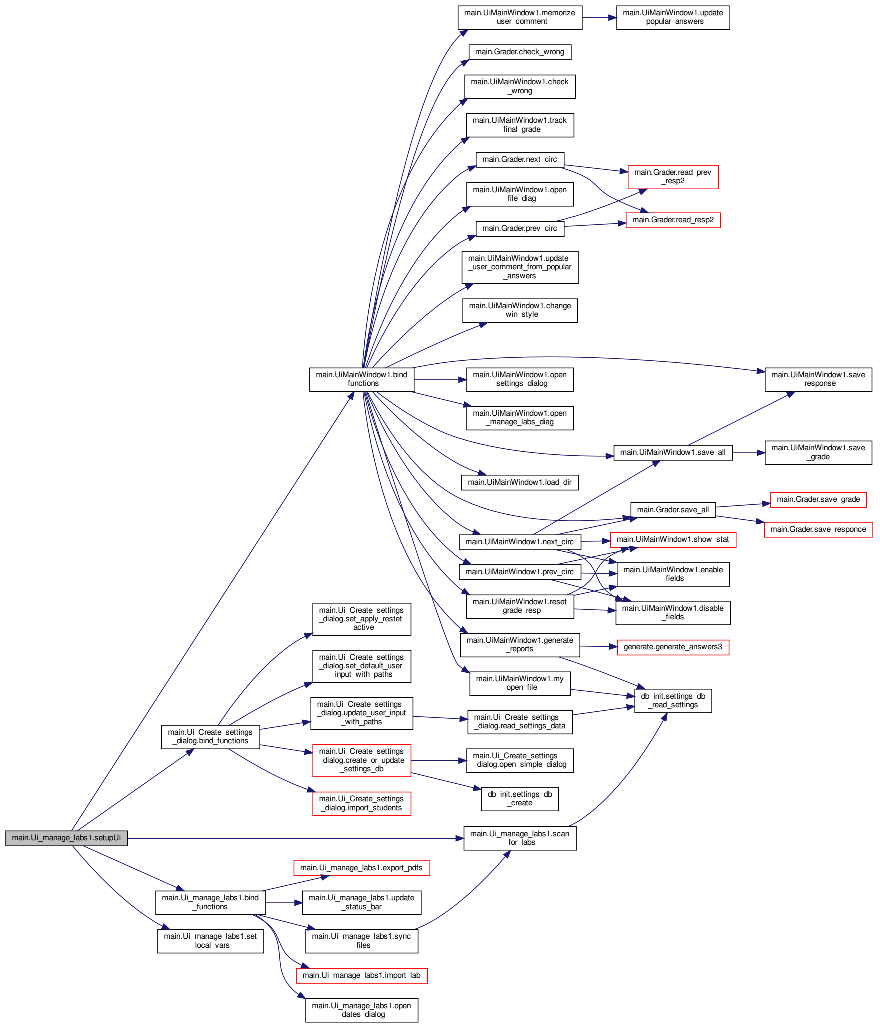
7.15.2.9 setupUi()

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

Definition at line 1588 of file main.py.

```
    def setupUi(self, manage_labs):
        super().setupUi(manage_labs)
        self.bind_functions()
        self.set_local_vars()

        try:
            self.scan_for_labs()
            if self.labs_select_comboBox.count() > 0:
                self.labs_select_comboBox.setEnabled(True)
                self.import_but.setEnabled(True)
                self.create_due_dates_but.setEnabled(True)
                self.export_but.setEnabled(True)
        except Exception as e:
            print('Error in manage labs. Probably your grading path was not set properly: ', e)


```

Here is the call graph for this function:


main.Ui_manage_labs1.setupUi
main.UiMainWindow1.bind_functions
main.UiMainWindow1.memorize_user_comment
main.UiMainWindow1.update_popular_answers
main.Grader.check_wrong
main.UiMainWindow1.check_wrong
main.UiMainWindow1.track_final_grade
main.Grader.next_circ
main.Grader.read_prev_resp2
main.UiMainWindow1.open_file_diag
main.Grader.prev_circ
main.Grader.read_resp2
main.UiMainWindow1.update_user_comment_from_popular_answers
main.UiMainWindow1.change_win_style
main.UiMainWindow1.open_settings_dialog
main.UiMainWindow1.save_response
main.UiMainWindow1.open_manage_labs_diag
main.UiMainWindow1.save_all
main.UiMainWindow1.save_grade
main.UiMainWindow1.load_dir
main.Grader.save_grade
main.Grader.save_all
main.Grader.save_responce
main.UiMainWindow1.next_circ
main.UiMainWindow1.show_stat
main.UiMainWindow1.prev_circ
main.UiMainWindow1.enable_fields
main.UiMainWindow1.reset_grade_resp
main.UiMainWindow1.disable_fields
main.UiMainWindow1.generate_reports
generate.generate_answers3
main.UiMainWindow1.my_open_file
db_init.settings_db_read_settings
main.Ui_Create_settings_dialog.set_apply_restet_active
main.Ui_Create_settings_dialog.set_default_user_input_with_paths
main.Ui_Create_settings_dialog.update_user_input_with_paths
main.Ui_Create_settings_dialog.read_settings_data
main.Ui_Create_settings_dialog.bind_functions
main.Ui_Create_settings_dialog.create_or_update_settings_db
main.Ui_Create_settings_dialog.open_simple_dialog
db_init.settings_db_create
main.Ui_Create_settings_dialog.import_students
main.Ui_manage_labs1.scan_for_labs
main.Ui_manage_labs1.export_pdfs
main.Ui_manage_labs1.bind_functions
main.Ui_manage_labs1.update_status_bar
main.Ui_manage_labs1.set_local_vars
main.Ui_manage_labs1.sync_files
main.Ui_manage_labs1.import_lab
main.Ui_manage_labs1.open_dates_dialog

7.15.2.10 sync_files()

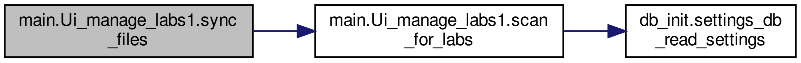
```
def main.Ui_manage_labs1.sync_files (
            self )
```

Definition at line 1626 of file main.py.

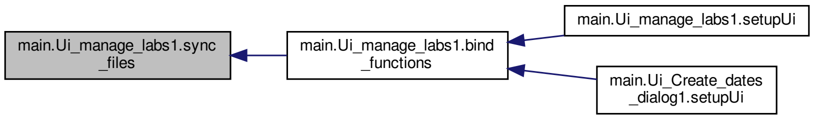
```
1626    def sync_files(self):
1627        self.sync_but.setDisabled(True)
1628        self.sync_but.setText('Synchronizing...')
1629        self.sync_but.repaint()
1630        self.status_bar.setText("Synchronizing...")
1631        self.status_bar.repaint()
```

```
        sync_files()
        self.status_bar.setText("Done.")
        self.sync_but.setText('Sync to local storage')
        self.sync_but.setEnabled(True)

        sync_success = True  # there are no tools to check it at this point.
        if sync_success and not self.labs_select_comboBox.isEnabled():
            self.labs_select_comboBox.setEnabled(True)
            self.create_due_dates_but.setEnabled(True)
            self.scan_for_labs()
            #  TODO: There should be additional checks to enable import and export, but I do not have
enough time to implement them.
            self.import_but.setEnabled(True)
            self.export_but.setEnabled(True)

```

Here is the call graph for this function:


main.Ui_manage_labs1.sync_files
main.Ui_manage_labs1.scan_for_labs
db_init.settings_db_read_settings

Here is the caller graph for this function:


main.Ui_manage_labs1.sync_files
main.Ui_manage_labs1.bind_functions
main.Ui_manage_labs1.setupUi
main.Ui_Create_dates_dialog1.setupUi

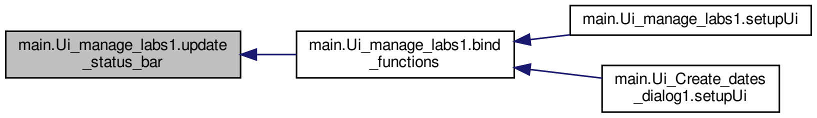
7.15.2.11 update_status_bar()

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

Definition at line 1607 of file main.py.

```
    def update_status_bar(self, force=False):
        # no need to scan files in background, but only when user selects it intentionally, or if it is
first run
        if self.labs_select_comboBox.hasFocus() or force:
            self.selected_lab_name = self.labs_select_comboBox.currentText()
            self.selected_path = self.srv_sync_path + self.selected_lab_name + '/'
            zip_pdf_files = [f for f in os.listdir(self.selected_path) if '.zip' in f or '.pdf' in f]

            self.pdf_files_len = len([f for f in zip_pdf_files if f.split('.')[1] == 'pdf'])
            self.zip_files_len = len([f for f in zip_pdf_files if f.split('.')[1] == 'zip'])

            self.status_bar.setText("Contains " + str(self.zip_files_len) + ' zip files and ' + str(self.
    pdf_files_len) + ' pdf files.')

            if self.zip_files_len > 0 and not self.create_due_dates_but.isEnabled():
                self.export_but.setEnabled(True)
                self.import_but.setEnabled(True)
                self.labs_select_comboBox.setEnabled(True)

            # good_zip_files_size = len([f for f in zip_files if os.isfile(os.path.join(selected_path,
     f))])

```

Here is the caller graph for this function:


main.Ui_manage_labs1.update
_status_bar
main.Ui_manage_labs1.bind
_functions
main.Ui_manage_labs1.setupUi
main.Ui_Create_dates
_dialog1.setupUi

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]
```

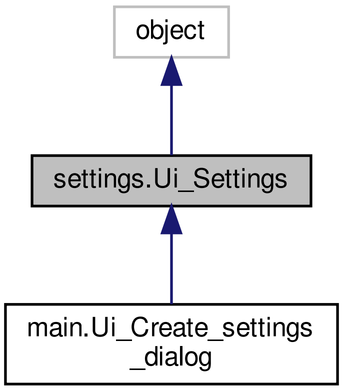
Definition at line 1578 of file main.py.

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

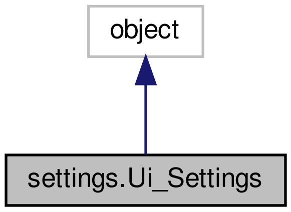
- main.py

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_stuents_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.

```
    def retranslateUi(self, Settings):

        _translate = QtCore.QCoreApplication.translate
        Settings.setWindowTitle(_translate("Settings", "Settings"))
        self.groupBox_db.setTitle(_translate("Settings", "&Database paths:"))
        self.label_settings_db.setText(_translate("Settings", "Settings"))
        self.input_settings_db.setText(_translate("Settings", "./settings.sqlite3"))
        self.label_grades_db.setText(_translate("Settings", "Grades"))
        self.input_grades_db.setPlaceholderText(_translate("Settings", "
      ~/Documents/3130_labs/grades.sqlite3"))
        self.groupBox_user.setTitle(_translate("Settings", "User paths"))
        self.label_logisim_path.setText(_translate("Settings", "Logisim path"))
        self.input_logisim_path.setPlaceholderText(_translate("Settings", "path to logisim executable
       logisim.jar"))
        self.label_local_stor.setText(_translate("Settings", "Local lab storage"))
        self.input_local_stor.setPlaceholderText(_translate("Settings", "local directory that contains
       labs, reports, and other working files"))
        self.label_rem_stor.setText(_translate("Settings", "Remote lab storage"))
        self.input_rem_stor.setPlaceholderText(_translate("Settings", "sshfs mounted dir that points to
       submission directory on the remote server"))
        self.groupBox_local.setTitle(_translate("Settings", "&Local settings"))
        self.label_grad_year.setText(_translate("Settings", "Grading year"))
        self.label_semester.setText(_translate("Settings", "Grading semester"))
        self.label_style.setText(_translate("Settings", "Use styles"))
        self.label_sync_comm.setText(_translate("Settings", "Sync command"))
        self.label_grader_name.setText(_translate("Settings", "Grader name"))
        self.semester_comboBox.setItemText(0, _translate("Settings", "Spring"))
        self.semester_comboBox.setItemText(1, _translate("Settings", "Summer"))
        self.semester_comboBox.setItemText(2, _translate("Settings", "Fall"))
        self.sync_command.setPlaceholderText(_translate("Settings", "rsync -avz ? cp -v ? dd ... ?"))
        self.import_stuents_btn.setText(_translate("Settings", "Import_students"))


```

7.16.2.2 setupUi()

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

Definition at line 12 of file settings.py.

```
    def setupUi(self, Settings):
        Settings.setObjectName("Settings")
        Settings.setEnabled(True)
        Settings.resize(800, 487)
        Settings.setMinimumSize(QtCore.QSize(600, 0))
        icon = QtGui.QIcon()
        icon.addPixmap(QtGui.QPixmap("os_linux_1.ico"), QtGui.QIcon.Normal, QtGui.QIcon.Off)
        Settings.setWindowIcon(icon)
        Settings.setLocale(QtCore.QLocale(QtCore.QLocale.English, QtCore.QLocale.UnitedStates))
        self.verticalLayout = QtWidgets.QVBoxLayout(Settings)
        self.verticalLayout.setObjectName("verticalLayout")
        self.groupBox_db = QtWidgets.QGroupBox(Settings)
        sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Preferred, QtWidgets.QSizePolicy.Preferred
)
        sizePolicy.setHorizontalStretch(0)
        sizePolicy.setVerticalStretch(0)
        sizePolicy.setHeightForWidth(self.groupBox_db.sizePolicy().hasHeightForWidth())
        self.groupBox_db.setSizePolicy(sizePolicy)
        self.groupBox_db.setMinimumSize(QtCore.QSize(0, 0))
        self.groupBox_db.setAutoFillBackground(False)
        self.groupBox_db.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignTop)
        self.groupBox_db.setFlat(False)
        self.groupBox_db.setCheckable(False)
        self.groupBox_db.setObjectName("groupBox_db")
        self.formLayout = QtWidgets.QFormLayout(self.groupBox_db)
        self.formLayout.setObjectName("formLayout")
        self.label_settings_db = QtWidgets.QLabel(self.groupBox_db)
        self.label_settings_db.setMinimumSize(QtCore.QSize(110, 0))
        self.label_settings_db.setObjectName("label_settings_db")
        self.formLayout.setWidget(0, QtWidgets.QFormLayout.LabelRole, self.label_settings_db)
        self.input_settings_db = QtWidgets.QLineEdit(self.groupBox_db)
        self.input_settings_db.setEnabled(False)
        self.input_settings_db.setMinimumSize(QtCore.QSize(550, 31))
        self.input_settings_db.setObjectName("input_settings_db")
        self.formLayout.setWidget(0, QtWidgets.QFormLayout.FieldRole, self.input_settings_db)
        self.label_grades_db = QtWidgets.QLabel(self.groupBox_db)
        self.label_grades_db.setMinimumSize(QtCore.QSize(110, 0))
        self.label_grades_db.setObjectName("label_grades_db")
        self.formLayout.setWidget(1, QtWidgets.QFormLayout.LabelRole, self.label_grades_db)
        self.input_grades_db = QtWidgets.QLineEdit(self.groupBox_db)
        self.input_grades_db.setMinimumSize(QtCore.QSize(550, 31))
        self.input_grades_db.setText("")
        self.input_grades_db.setObjectName("input_grades_db")
        self.formLayout.setWidget(1, QtWidgets.QFormLayout.FieldRole, self.input_grades_db)
        self.verticalLayout.addWidget(self.groupBox_db)
        self.groupBox_user = QtWidgets.QGroupBox(Settings)
        self.groupBox_user.setEnabled(False)
        sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Preferred, QtWidgets.QSizePolicy.Preferred
)
        sizePolicy.setHorizontalStretch(0)
        sizePolicy.setVerticalStretch(0)
        sizePolicy.setHeightForWidth(self.groupBox_user.sizePolicy().hasHeightForWidth())
        self.groupBox_user.setSizePolicy(sizePolicy)
        self.groupBox_user.setMinimumSize(QtCore.QSize(0, 0))
        self.groupBox_user.setObjectName("groupBox_user")
        self.formLayout_2 = QtWidgets.QFormLayout(self.groupBox_user)
        self.formLayout_2.setObjectName("formLayout_2")
        self.label_logisim_path = QtWidgets.QLabel(self.groupBox_user)
        sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Preferred, QtWidgets.QSizePolicy.Preferred
)
        sizePolicy.setHorizontalStretch(0)
        sizePolicy.setVerticalStretch(0)
        sizePolicy.setHeightForWidth(self.label_logisim_path.sizePolicy().hasHeightForWidth())
        self.label_logisim_path.setSizePolicy(sizePolicy)
        self.label_logisim_path.setMinimumSize(QtCore.QSize(110, 0))
        self.label_logisim_path.setObjectName("label_logisim_path")
        self.formLayout_2.setWidget(0, QtWidgets.QFormLayout.LabelRole, self.label_logisim_path)
        self.input_logisim_path = QtWidgets.QLineEdit(self.groupBox_user)
        sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Expanding, QtWidgets.QSizePolicy.Fixed)
        sizePolicy.setHorizontalStretch(0)
        sizePolicy.setVerticalStretch(0)
        sizePolicy.setHeightForWidth(self.input_logisim_path.sizePolicy().hasHeightForWidth())
        self.input_logisim_path.setSizePolicy(sizePolicy)
        self.input_logisim_path.setMinimumSize(QtCore.QSize(637, 31))
        self.input_logisim_path.setText("")
        self.input_logisim_path.setObjectName("input_logisim_path")
        self.formLayout_2.setWidget(0, QtWidgets.QFormLayout.FieldRole, self.input_logisim_path)
        self.label_local_stor = QtWidgets.QLabel(self.groupBox_user)
        self.label_local_stor.setMinimumSize(QtCore.QSize(110, 0))
        self.label_local_stor.setObjectName("label_local_stor")
        self.formLayout_2.setWidget(1, QtWidgets.QFormLayout.LabelRole, self.label_local_stor)
        self.input_local_stor = QtWidgets.QLineEdit(self.groupBox_user)
        self.input_local_stor.setMinimumSize(QtCore.QSize(637, 31))
        self.input_local_stor.setText("")
        self.input_local_stor.setObjectName("input_local_stor")
        self.formLayout_2.setWidget(1, QtWidgets.QFormLayout.FieldRole, self.input_local_stor)
        self.label_rem_stor = QtWidgets.QLabel(self.groupBox_user)
        self.label_rem_stor.setMinimumSize(QtCore.QSize(110, 0))
        self.label_rem_stor.setObjectName("label_rem_stor")
        self.formLayout_2.setWidget(2, QtWidgets.QFormLayout.LabelRole, self.label_rem_stor)
        self.input_rem_stor = QtWidgets.QLineEdit(self.groupBox_user)
        self.input_rem_stor.setMinimumSize(QtCore.QSize(637, 31))
        self.input_rem_stor.setInputMask("")
        self.input_rem_stor.setText("")
        self.input_rem_stor.setObjectName("input_rem_stor")
        self.formLayout_2.setWidget(2, QtWidgets.QFormLayout.FieldRole, self.input_rem_stor)
        self.verticalLayout.addWidget(self.groupBox_user)
        self.groupBox_local = QtWidgets.QGroupBox(Settings)
        sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Preferred,
QtWidgets.QSizePolicy.MinimumExpanding)
        sizePolicy.setHorizontalStretch(0)
        sizePolicy.setVerticalStretch(0)
        sizePolicy.setHeightForWidth(self.groupBox_local.sizePolicy().hasHeightForWidth())
        self.groupBox_local.setSizePolicy(sizePolicy)
        self.groupBox_local.setMinimumSize(QtCore.QSize(0, 145))
        self.groupBox_local.setMaximumSize(QtCore.QSize(16777215, 300))
        self.groupBox_local.setFlat(False)
        self.groupBox_local.setCheckable(False)
        self.groupBox_local.setObjectName("groupBox_local")
        self.gridLayout = QtWidgets.QGridLayout(self.groupBox_local)
        self.gridLayout.setObjectName("gridLayout")
        self.spin_year = QtWidgets.QSpinBox(self.groupBox_local)
        self.spin_year.setEnabled(False)
        self.spin_year.setMinimumSize(QtCore.QSize(110, 31))
        self.spin_year.setMaximumSize(QtCore.QSize(110, 16777215))
        self.spin_year.setWrapping(True)
        self.spin_year.setReadOnly(False)
        self.spin_year.setButtonSymbols(QtWidgets.QAbstractSpinBox.PlusMinus)
        self.spin_year.setAccelerated(True)
        self.spin_year.setProperty("showGroupSeparator", False)
        self.spin_year.setMinimum(2012)
        self.spin_year.setMaximum(2026)
        self.spin_year.setProperty("value", 2018)
        self.spin_year.setObjectName("spin_year")
        self.gridLayout.addWidget(self.spin_year, 0, 1, 1, 1)
        self.label_grad_year = QtWidgets.QLabel(self.groupBox_local)
        sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Preferred, QtWidgets.QSizePolicy.Fixed)
        sizePolicy.setHorizontalStretch(0)
        sizePolicy.setVerticalStretch(0)
        sizePolicy.setHeightForWidth(self.label_grad_year.sizePolicy().hasHeightForWidth())
        self.label_grad_year.setSizePolicy(sizePolicy)
        self.label_grad_year.setMinimumSize(QtCore.QSize(110, 31))
        self.label_grad_year.setMaximumSize(QtCore.QSize(110, 16777215))
        self.label_grad_year.setObjectName("label_grad_year")
        self.gridLayout.addWidget(self.label_grad_year, 0, 0, 1, 1)
        self.input_grader_name = QtWidgets.QLineEdit(self.groupBox_local)
        self.input_grader_name.setEnabled(False)
        self.input_grader_name.setMinimumSize(QtCore.QSize(110, 31))
        self.input_grader_name.setMaximumSize(QtCore.QSize(110, 16777215))
        self.input_grader_name.setObjectName("input_grader_name")
        self.gridLayout.addWidget(self.input_grader_name, 2, 1, 1, 1)
        self.label_semester = QtWidgets.QLabel(self.groupBox_local)
        sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Preferred, QtWidgets.QSizePolicy.Fixed)
        sizePolicy.setHorizontalStretch(0)
        sizePolicy.setVerticalStretch(0)
        sizePolicy.setHeightForWidth(self.label_semester.sizePolicy().hasHeightForWidth())
        self.label_semester.setSizePolicy(sizePolicy)
        self.label_semester.setMinimumSize(QtCore.QSize(110, 31))
        self.label_semester.setMaximumSize(QtCore.QSize(110, 16777215))
        self.label_semester.setObjectName("label_semester")
        self.gridLayout.addWidget(self.label_semester, 0, 3, 1, 1)
        self.label_style = QtWidgets.QLabel(self.groupBox_local)
        self.label_style.setMinimumSize(QtCore.QSize(110, 31))
        self.label_style.setObjectName("label_style")
        self.gridLayout.addWidget(self.label_style, 1, 0, 1, 1)
        self.label_sync_comm = QtWidgets.QLabel(self.groupBox_local)
        self.label_sync_comm.setObjectName("label_sync_comm")
        self.gridLayout.addWidget(self.label_sync_comm, 2, 3, 1, 1)
        self.label_grader_name = QtWidgets.QLabel(self.groupBox_local)
        sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Preferred, QtWidgets.QSizePolicy.Fixed)
        sizePolicy.setHorizontalStretch(0)
        sizePolicy.setVerticalStretch(0)
        sizePolicy.setHeightForWidth(self.label_grader_name.sizePolicy().hasHeightForWidth())
        self.label_grader_name.setSizePolicy(sizePolicy)
        self.label_grader_name.setMinimumSize(QtCore.QSize(110, 31))
        self.label_grader_name.setObjectName("label_grader_name")
        self.gridLayout.addWidget(self.label_grader_name, 2, 0, 1, 1)
        self.style_checkBox = QtWidgets.QCheckBox(self.groupBox_local)
        self.style_checkBox.setEnabled(False)
        sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.MinimumExpanding,
QtWidgets.QSizePolicy.Fixed)
        sizePolicy.setHorizontalStretch(0)
        sizePolicy.setVerticalStretch(0)
        sizePolicy.setHeightForWidth(self.style_checkBox.sizePolicy().hasHeightForWidth())
        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_stuents_btn

```
settings.Ui_Settings.import_stuents_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
```

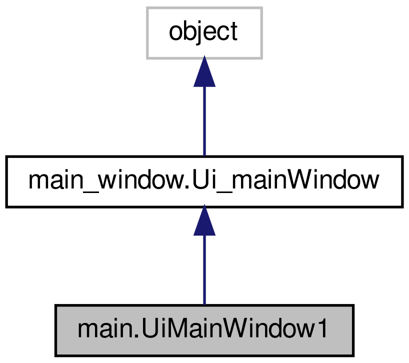
Definition at line 21 of file settings.py.

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

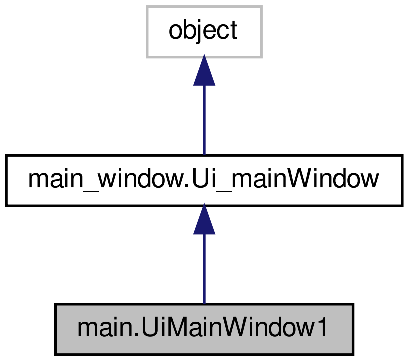
- settings.py

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

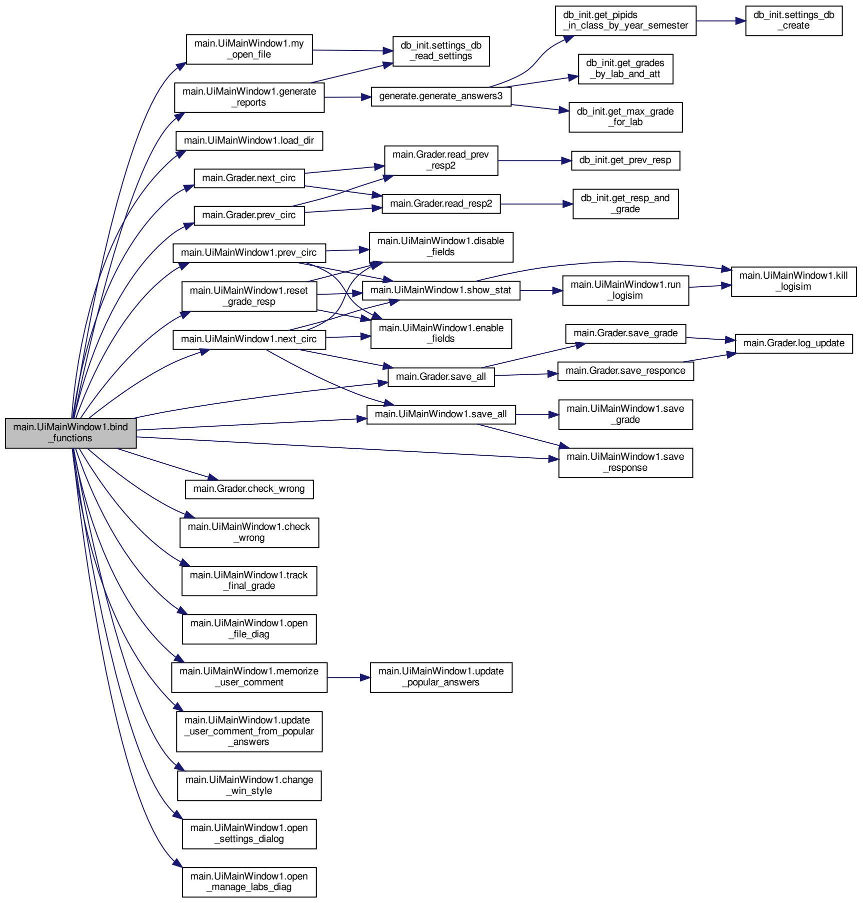
7.17.3.1 bind_functions()

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

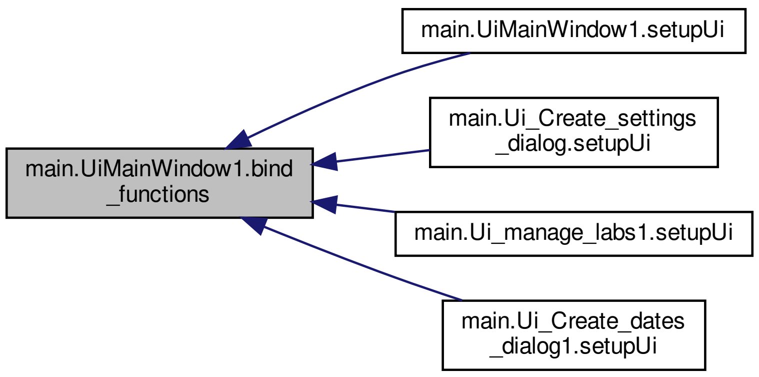
Definition at line 1124 of file main.py.

```
    def bind_functions(self):
        self.but_file_open.clicked.connect(self.my_open_file)
        self.but_begin.clicked.connect(self.load_dir)
        self.but_next.clicked.connect(self.next_circ)
        self.but_prev.clicked.connect(self.prev_circ)
        self.checkB_wrong.clicked.connect(self.check_wrong)
        #  self.but_regrade.clicked.connect(self.regrade)
        self.but_save_all.clicked.connect(self.save_all)
        self.but_save_response.clicked.connect(self.save_response)
        self.input_final_grade.textEdited.connect(self.track_final_grade)
        #  self.but_edit_done.clicked.connect(self.resp_edit_done)
        #  self.popular_answers.activated.connect(self.select_saved_answer)
        # self.but_create_report.setEnabled(True)  # Debug
        self.but_create_report.clicked.connect(self.generate_reports)
        # self.new_window_but.clicked.connect(self.open_dates_dialog)
        #  self.input_response_browser_user.focusInEvent(self, self.memorize_user_comment)
        #  self.custom_but_test.right_clicked[int].connect(self.dummy_d)
        self.input_file_location.dclicked.connect(self.open_file_diag)
        self.input_response_browser_user.focus_lost.connect(self.memorize_user_comment)
        self.popular_answers.currentIndexChanged.connect(self.update_user_comment_from_popular_answers)
        self.set_style_checkbox.stateChanged.connect(self.change_win_style)
        self.but_reset.clicked.connect(self.reset_grade_resp)
        self.settings_but.clicked.connect(self.open_settings_dialog)
        self.manage_labs_but.clicked.connect(self.open_manage_labs_diag)
        # self.sync_but.clicked.connect(self.sync_files)



```

Here is the call graph for this function:


main.UiMainWindow1.bind_functions
main.UiMainWindow1.my_open_file
db_init.settings_db_read_settings
main.UiMainWindow1.generate_reports
generate.generate_answers3
db_init.get_pipids_in_class_by_year_semester
db_init.settings_db_create
db_init.get_grades_by_lab_and_att
db_init.get_max_grade_for_lab
main.UiMainWindow1.load_dir
main.Grader.next_circ
main.Grader.read_prev_resp2
db_init.get_prev_resp
main.Grader.read_resp2
db_init.get_resp_and_grade
main.Grader.prev_circ
main.UiMainWindow1.prev_circ
main.UiMainWindow1.disable_fields
main.UiMainWindow1.reset_grade_resp
main.UiMainWindow1.show_stat
main.UiMainWindow1.run_logisim
main.UiMainWindow1.kill_logisim
main.UiMainWindow1.enable_fields
main.UiMainWindow1.next_circ
main.Grader.save_grade
main.Grader.log_update
main.Grader.save_all
main.Grader.save_responce
main.UiMainWindow1.save_all
main.UiMainWindow1.save_grade
main.UiMainWindow1.save_response
main.Grader.check_wrong
main.UiMainWindow1.check_wrong
main.UiMainWindow1.track_final_grade
main.UiMainWindow1.open_file_diag
main.UiMainWindow1.memorize_user_comment
main.UiMainWindow1.update_popular_answers
main.UiMainWindow1.update_user_comment_from_popular_answers
main.UiMainWindow1.change_win_style
main.UiMainWindow1.open_settings_dialog
main.UiMainWindow1.open_manage_labs_diag

Here is the caller graph for this function:



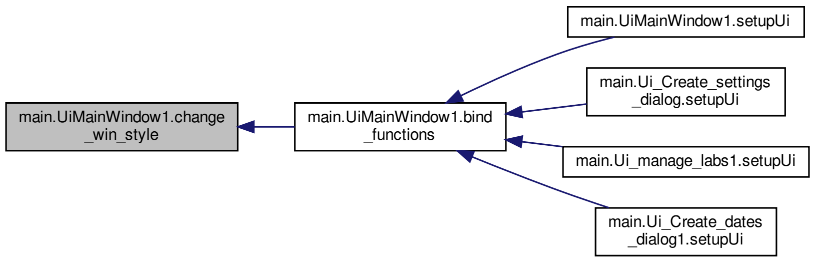
7.17.3.2 change_win_style()

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

Definition at line 1157 of file main.py.

```
    def change_win_style(self):
        if self.set_style_checkbox.isChecked():
            self.progressBar.setStyleSheet(styleData)
        else:
            self.progressBar.setStyleSheet('')

```

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.

```
    def check_file(self):
        self.input_subtract.setText(str(self.grader_ref.subtract))
        self.input_final_grade.setText(str(self.grader_ref.final_grade))

        self.input_log_browser.setText(self.grader_ref.global_log)
        # self.input_log_browser.append(self.grader_ref.global_log)

        if self.grader_ref.input_correct:
            self.checkB_input_pin_status.setChecked(True)
        if self.grader_ref.output_correct:
            self.checkB_output_pin_status.setChecked(True)

        # self.but_save_response.setDisabled(True)
        # self.but_save_all.setDisabled(True)

        # self.but_edit_done.setDisabled(True)
        try:
            # self.grader_ref.generate_response()  #TODO this overwrites File not found.
            self.input_response_browser.setPlainText(self.grader_ref.resp_text)
            # self.but_edit_done.setEnabled(True)
            # self.but_save_response.setEnabled(True)
            # self.but_save_all.setEnabled(True)
        except Exception as e:
            print('Error in generate response:', e)

```

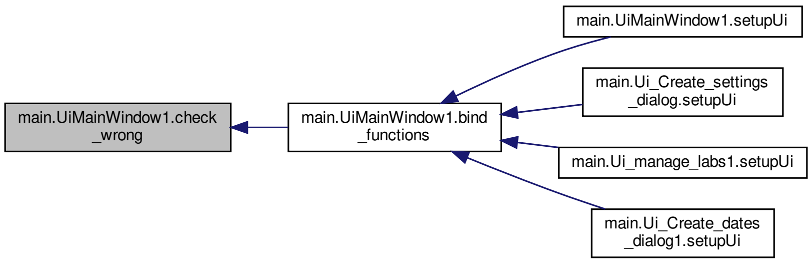
7.17.3.4 check_wrong()

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

Definition at line 976 of file main.py.

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

```

Here is the caller graph for this function:


main.UiMainWindow1.check_wrong
main.UiMainWindow1.bind_functions
main.UiMainWindow1.setupUi
main.Ui_Create_settings_dialog.setupUi
main.Ui_manage_labs1.setupUi
main.Ui_Create_dates_dialog1.setupUi

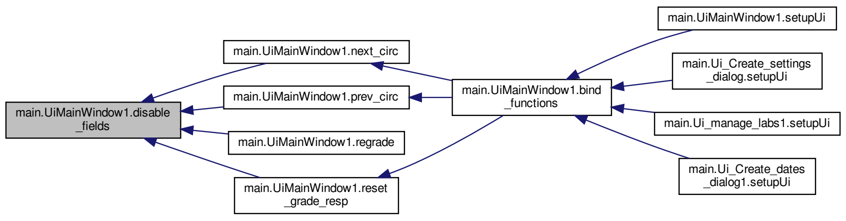
7.17.3.5 disable_fields()

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

Definition at line 733 of file main.py.

```
    def disable_fields(self):
        self.checkB_input_pin_status.setDisabled(True)
        self.checkB_output_pin_status.setDisabled(True)
        # self.input_response_browser.setDisabled(True)
        self.checkB_wrong.setDisabled(True)

        # self.input_subtract.setDisabled(True)
        self.but_regrade.setDisabled(True)
        self.popular_answers.setDisabled(True)
        self.input_final_grade.setDisabled(True)
        self.checkB_wrong.setChecked(False)
        self.check_autosave.setDisabled(True)
        self.input_current_id.setText('')

```

Here is the caller graph for this function:


main.UiMainWindow1.setupUi
main.UiMainWindow1.next_circ
main.Ui_Create_settings _dialog.setupUi
main.UiMainWindow1.bind _functions
main.UiMainWindow1.prev_circ
main.UiMainWindow1.disable _fields
main.Ui_manage_labs1.setupUi
main.UiMainWindow1.regrade
main.Ui_Create_dates _dialog1.setupUi
main.UiMainWindow1.reset _grade_resp

7.17.3.6 dummy_d_1()

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

Definition at line 1164 of file main.py.

```
    def dummy_d_1(self):
        print('dummy_1 activated')

```

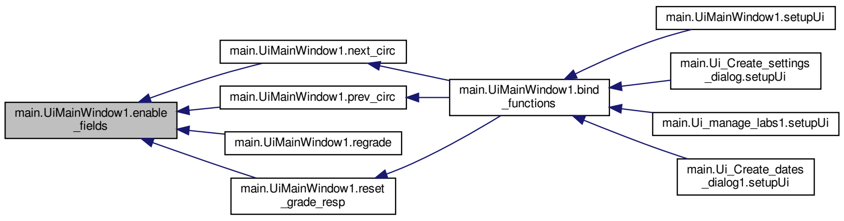
7.17.3.7 enable_fields()

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

Definition at line 751 of file main.py.

```
    def enable_fields(self):
        self.checkB_input_pin_status.setEnabled(True)
        self.checkB_output_pin_status.setEnabled(True)
        # self.input_response_browser.setEnabled(True)
        self.checkB_wrong.setEnabled(True)
        self.input_final_grade.setEnabled(True)
        self.check_autosave.setEnabled(True)

        # self.input_subtract.setEnabled(True)
        # self.but_regrade.setEnabled(True)
        self.popular_answers.setEnabled(True)

```

Here is the caller graph for this function:


main.UiMainWindow1.enable_fields
main.UiMainWindow1.next_circ
main.UiMainWindow1.prev_circ
main.UiMainWindow1.regrade
main.UiMainWindow1.reset_grade_resp
main.UiMainWindow1.bind_functions
main.UiMainWindow1.setupUi
main.Ui_Create_settings_dialog.setupUi
main.Ui_manage_labs1.setupUi
main.Ui_Create_dates_dialog1.setupUi

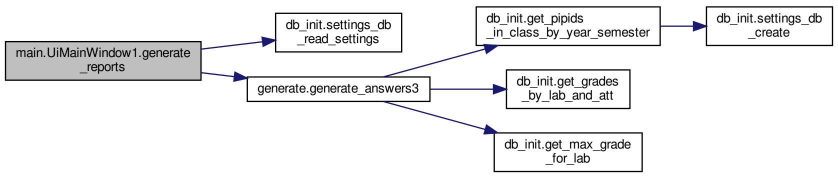
7.17.3.8 generate_reports()

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

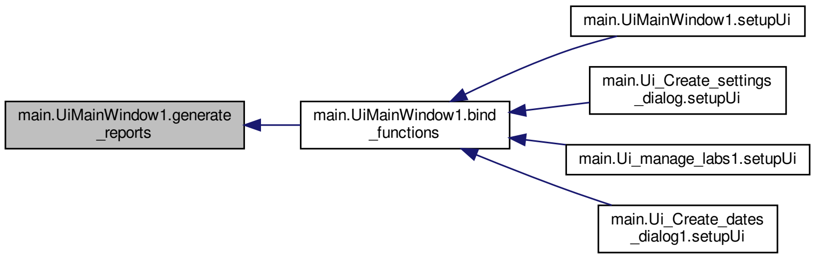
Definition at line 1241 of file main.py.

```
    def generate_reports(self):
        self.but_create_report.setDisabled(True)
        self.but_create_report.setText('Generating..')
        self.but_create_report.repaint()
        # from generate import generate_answers
        # (resubmit_num, dir_name, lab_type, lab_num)
        if hasattr(self, 'grader_ref'):
            loc_settings = settings_db_read_settings()[1]
            generate_answers3(self.grader_ref.lid, self.grader_ref.attempt, self.
      grader_ref.year, self.grader_ref.semester)
            # 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)
            # 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)
            self.but_create_report.setEnabled(True)
            self.but_create_report.setText('Create reports')



```

Here is the call graph for this function:



Here is the caller graph for this function:



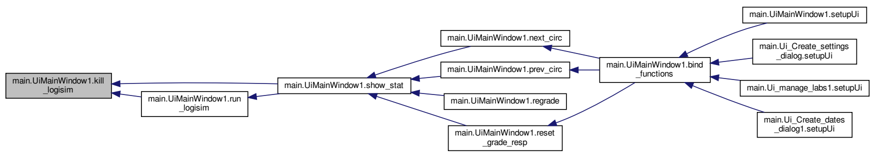
7.17.3.9 kill_logisim()

```
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:



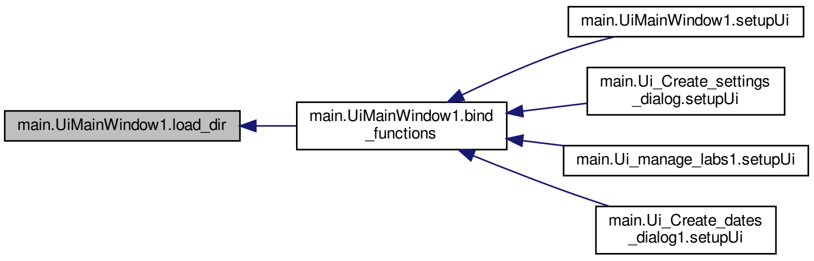
7.17.3.10 load_dir()

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

Definition at line 767 of file main.py.

```
    def load_dir(self):
        # activate elements
        cur_year, cur_sem = self.grader_ref.working_dir.split('/')[-3].split('_')
        self.class_id_to_id = get_ids_in_class_by_year_semester(cur_year,
    cur_sem)[1]
        self.but_begin.setDisabled(True)
        self.but_begin.repaint()
        self.progressBar.setEnabled(True)

        self.disable_fields()

        self.grader_ref.tot_elem = len(self.grader_ref.lab_paths)
        if self.grader_ref.tot_elem > 1:
            self.but_next.setEnabled(True)

        self.progressBar.setMaximum(self.grader_ref.tot_elem)
        self.progressBar.setValue(0)
        self.popular_answers.clear()

        # self.grader_ref.check_file(0)
        # self.grader_ref.stud_id = self.grader_ref.stud_ids[self.grader_ref.cur_idx]
        self.grader_ref.cur_idx = -1
        # graded = self.grader_ref.read_resp2()
        # if graded:
        #     self.grader_ref.read_prev_resp2()
        self.next_circ()
        # self.grader_ref.read_resp()
        # self.grader_ref.read_prev_resp()
        # self.show_stat()
        # self.check_file()
        # self.input_current_id.setPlainText(self.grader_ref.get_stud_id())

        self.enable_fields()
        self.input_response_browser_user.setEnabled(True)
        self.but_regrade.setText('GRADE')
        self.but_save_all.setEnabled(True)
        self.but_save_response.setEnabled(True)
        self.check_autosave.setEnabled(True)
        self.but_reset.setEnabled(True)

```

Here is the caller graph for this function:


main.UiMainWindow1.load_dir
main.UiMainWindow1.bind_functions
main.UiMainWindow1.setupUi
main.Ui_Create_settings_dialog.setupUi
main.Ui_manage_labs1.setupUi
main.Ui_Create_dates_dialog1.setupUi

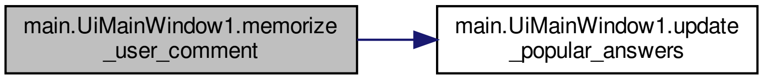
7.17.3.11 memorize_user_comment()

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

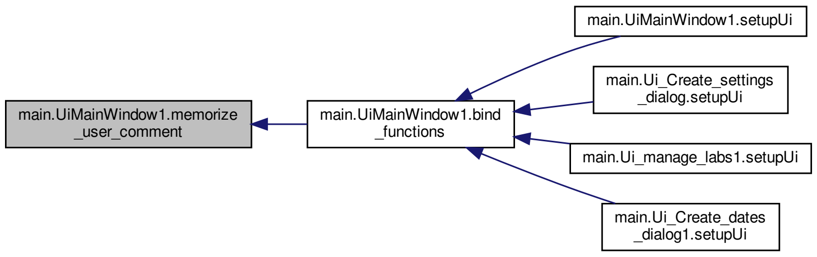
Definition at line 1191 of file main.py.

```
    def memorize_user_comment(self):
        typed = self.input_response_browser_user.toPlainText()
        if hasattr(self, 'grader_ref') and typed:
            try:
                index = self.popular_answers.findText(self.input_response_browser_user.toPlainText(),
                                                      QtCore.Qt.MatchFixedString)
                if index >= 0:
                    self.popular_answers.setCurrentIndex(index)
                else:
                    self.grader_ref.add_to_common_answers(typed)
                    self.update_popular_answers()
                    index = self.popular_answers.findText(self.input_response_browser_user.toPlainText(),
                                                          QtCore.Qt.MatchFixedString)
                    try:
                        self.popular_answers.setCurrentIndex(index)
                    except Exception as e:
                        print('Failed to select proper index: ', e)
                        raise
            except Exception as e:
                print('failed to add popular answer: ', e)

```

Here is the call graph for this function:


main.UiMainWindow1.memorize_user_comment
main.UiMainWindow1.update_popular_answers

Here is the caller graph for this function:


main.UiMainWindow1.memorize_user_comment
main.UiMainWindow1.bind_functions
main.UiMainWindow1.setupUi
main.Ui_Create_settings_dialog.setupUi
main.Ui_manage_labs1.setupUi
main.Ui_Create_dates_dialog1.setupUi

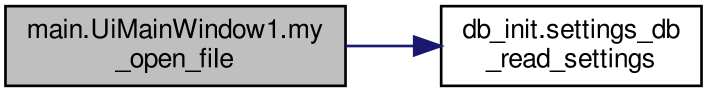
7.17.3.12 my_open_file()

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

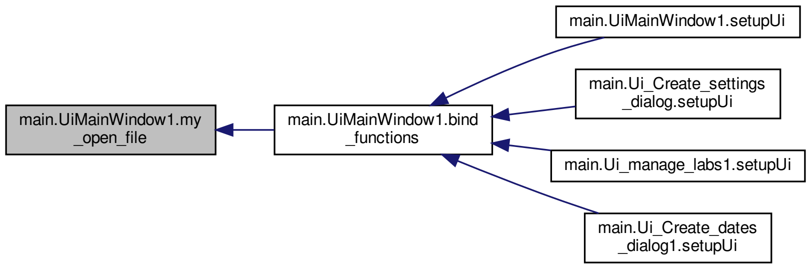
Definition at line 812 of file main.py.

```
    def my_open_file(self):
        working_dir = self.input_file_location.text()
        # self.input_response_browser.clear()
        # self.input_response_browser_user.clear()
        self.input_response_browser.setPlainText('I did not find any errors. Good job!')
        grader_name = settings_db_read_settings()[1][0]
        self.current_tz = QDateTime.currentDateTime().timeZoneAbbreviation()

        try:
            my_grader = Grader(working_dir, grader_name)
            my_grader.open_dir()

            self.grader_ref = my_grader

            self.input_max_pos_grade.setText(str(my_grader.lab_max_grade))
            self.input_attempt.setText(str(my_grader.attempt))
            self.dateTimeEdit_from.setDateTime(my_grader.time_from_qt)
            self.dateTimeEdit_to.setDateTime(my_grader.time_to_qt)
            self.grader_ref.add_to_common_answers('')  # helps to remove all text in user comment section
            # QDateTime.currentDateTime().timeZone()
            # global MAIN_FILE_NAME, MAIN_FILE_NAME_OVERRIDE

            # MAIN_FILE_NAME = get_lab_filename(my_grader.lab_id)[0]
            # if not MAIN_FILE_NAME:
            #     # Old way, I was determining filename as the most common submitted file.
            #     if not MAIN_FILE_NAME_OVERRIDE:
            #         a = []
            #         for root, dirs, files in os.walk(working_dir):
            #             for file in files:
            #                 if file.endswith(".circ"):
            #                     a.append(file)
            #         a = np.array(a)
            #
            #         MAIN_FILE_NAME = Counter(a.flat).most_common(1)[0][0]
            #     else:
            #         MAIN_FILE_NAME = MAIN_FILE_NAME_OVERRIDE
            #     # Now I can just read it from DB

            # self.grader_ref.circ_file_name = MAIN_FILE_NAME
            self.filename_lineEdit.setText(self.grader_ref.circ_file_name.split('.')[0])
            # self.reset_grade_resp()
            self.but_save_all.setChecked(False)

            self.but_create_report.setEnabled(True)
            self.but_begin.setEnabled(True)

        except Exception as e:  # TODO add log error
            print('Error in open_file : ', e)
            print(sys.exc_info()[0])

```

Here is the call graph for this function:


main.UiMainWindow1.my_open_file
db_init.settings_db_read_settings

Here is the caller graph for this function:


main.UiMainWindow1.my_open_file
main.UiMainWindow1.bind_functions
main.UiMainWindow1.setupUi
main.Ui_Create_settings_dialog.setupUi
main.Ui_manage_labs1.setupUi
main.Ui_Create_dates_dialog1.setupUi

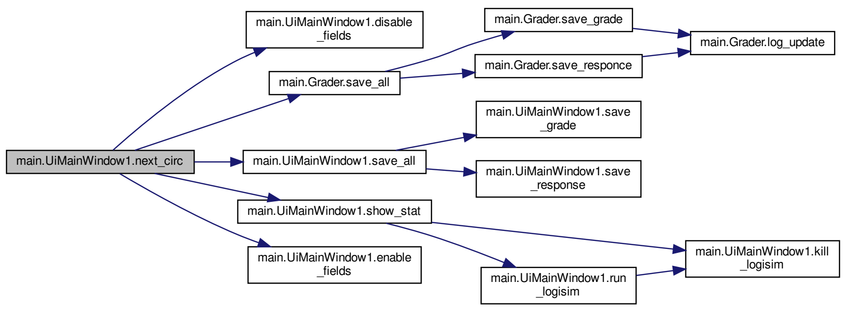
7.17.3.13 next_circ()

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

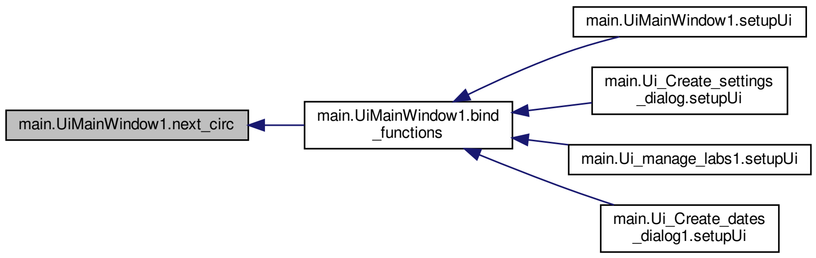
Definition at line 932 of file main.py.

```
    def next_circ(self):
        self.disable_fields()
        self.but_regrade.setText('GRADE')
        if self.check_autosave.isChecked() and self.grader_ref.cur_idx >= 0:
            self.save_all()
        # else:
        #     self.check_autosave.setDisabled(True)
        next_idx = self.grader_ref.next_circ()
        # self.check_file()
        self.show_stat()
        if next_idx >= self.grader_ref.tot_elem-1:
            self.but_next.setDisabled(True)
        if next_idx == 1:
            self.but_prev.setEnabled(True)

        self.progressBar.setValue(next_idx)
        self.enable_fields()

```

Here is the call graph for this function:


main.UiMainWindow1.next_circ
main.UiMainWindow1.disable_fields
main.Grader.save_all
main.UiMainWindow1.save_all
main.UiMainWindow1.show_stat
main.UiMainWindow1.enable_fields
main.Grader.save_grade
main.Grader.save_responce
main.UiMainWindow1.save_grade
main.UiMainWindow1.save_response
main.UiMainWindow1.run_logisim
main.Grader.log_update
main.UiMainWindow1.kill_logisim

Here is the caller graph for this function:


main.UiMainWindow1.next_circ
main.UiMainWindow1.bind_functions
main.UiMainWindow1.setupUi
main.Ui_Create_settings_dialog.setupUi
main.Ui_manage_labs1.setupUi
main.Ui_Create_dates_dialog1.setupUi

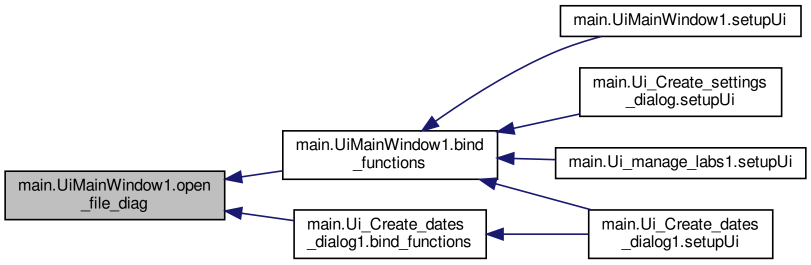
7.17.3.14 open_file_diag()

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

Definition at line 1180 of file main.py.

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

```

Here is the caller graph for this function:


main.UiMainWindow1.open_file_diag
main.UiMainWindow1.bind_functions
main.Ui_Create_dates_dialog1.bind_functions
main.UiMainWindow1.setupUi
main.Ui_Create_settings_dialog.setupUi
main.Ui_manage_labs1.setupUi
main.Ui_Create_dates_dialog1.setupUi

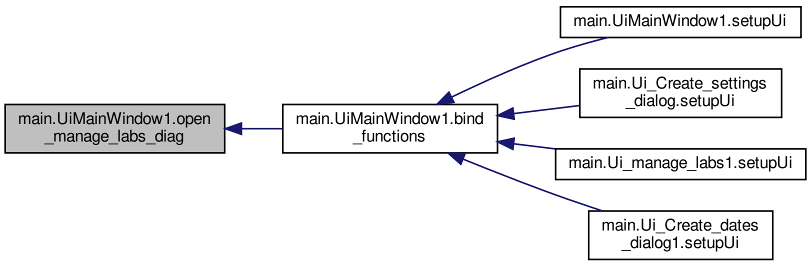
7.17.3.15 open_manage_labs_diag()

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

Definition at line 1315 of file main.py.

```
    def open_manage_labs_diag(self):
        self.manage_labs_but.setDisabled(True)
        self.manage_labs_but.repaint()
        self.centralwidget.setDisabled(True)
        self.centralwidget.repaint()
        self.manage_labs_window = QtWidgets.QDialog()
        dui = Ui_manage_labs1()
        dui.setupUi(self.manage_labs_window)

        self.manage_labs_window.show()
        self.manage_labs_window.exec_()

        self.centralwidget.setEnabled(True)
        self.manage_labs_but.setEnabled(True)

        if not self.but_file_open.isEnabled():
            paths, local = settings_db_read_settings()
            # if there are some labs in server sync directory:
            if len(os.walk(get_full_path(paths, local) + "/server_sync/").__next__()[1]) > 0:
                self.but_file_open.setEnabled(True)
                self.input_file_location.setEnabled(True)


```

Here is the caller graph for this function:


main.UiMainWindow1.open_manage_labs_diag
main.UiMainWindow1.bind_functions
main.UiMainWindow1.setupUi
main.Ui_Create_settings_dialog.setupUi
main.Ui_manage_labs1.setupUi
main.Ui_Create_dates_dialog1.setupUi

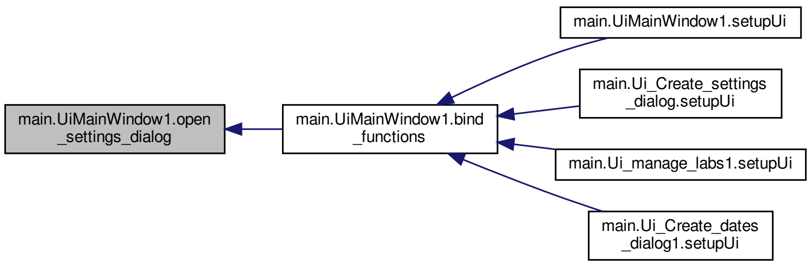
7.17.3.16 open_settings_dialog()

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

Definition at line 1285 of file main.py.

```
    def open_settings_dialog(self):
        self.settings_but.setDisabled(True)
        self.settings_but.repaint()
        self.settings_window = QtWidgets.QDialog()
        dui = Ui_Create_settings_dialog()
        dui.setupUi(self.settings_window)

        self.centralwidget.setDisabled(True)
        self.centralwidget.repaint()

        self.settings_window.show()
        self.settings_window.exec_()

        self.sync_params_to_settings()
        self.centralwidget.setEnabled(True)

        self.settings_but.setEnabled(True)

        if not self.manage_labs_but.isEnabled():
            from pathlib import Path
            settings_location = str(Path(os.path.expandvars(os.path.expanduser('./settings.sqlite3'))).
      absolute())
            if os.path.isfile(settings_location):
                self.manage_labs_but.setEnabled(True)


```

Here is the caller graph for this function:


main.UiMainWindow1.open_settings_dialog
main.UiMainWindow1.bind_functions
main.UiMainWindow1.setupUi
main.Ui_Create_settings_dialog.setupUi
main.Ui_manage_labs1.setupUi
main.Ui_Create_dates_dialog1.setupUi

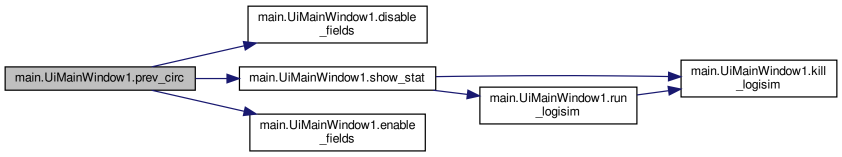
7.17.3.17 prev_circ()

def main.UiMainWindow1.prev_circ (
    self)

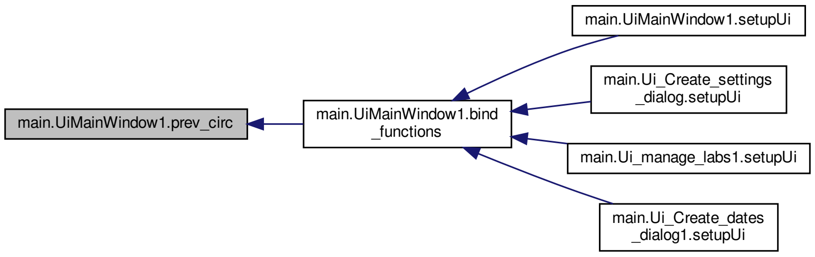
Definition at line 957 of file main.py.

```
    def prev_circ(self):
        self.disable_fields()
        self.but_regrade.setText('GRADE')
        next_idx = self.grader_ref.prev_circ()
        # self.check_file()
        self.show_stat()
        if next_idx <= self.grader_ref.tot_elem-1:
            self.but_next.setEnabled(True)
        if next_idx == 0:
            self.but_prev.setDisabled(True)

        self.progressBar.setValue(next_idx)
        self.enable_fields()

```

Here is the call graph for this function:


main.UiMainWindow1.prev_circ
main.UiMainWindow1.disable_fields
main.UiMainWindow1.show_stat
main.UiMainWindow1.enable_fields
main.UiMainWindow1.run_logisim
main.UiMainWindow1.kill_logisim

Here is the caller graph for this function:


main.UiMainWindow1.prev_circ
main.UiMainWindow1.bind_functions
main.UiMainWindow1.setupUi
main.Ui_Create_settings_dialog.setupUi
main.Ui_manage_labs1.setupUi
main.Ui_Create_dates_dialog1.setupUi

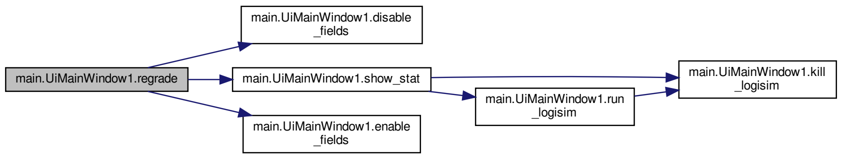
7.17.3.18 regrade()

```
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:


main.UiMainWindow1.regrade
main.UiMainWindow1.disable_fields
main.UiMainWindow1.show_stat
main.UiMainWindow1.enable_fields
main.UiMainWindow1.run_logisim
main.UiMainWindow1.kill_logisim

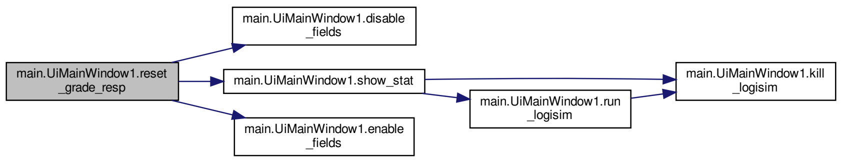
7.17.3.19 reset_grade_resp()

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

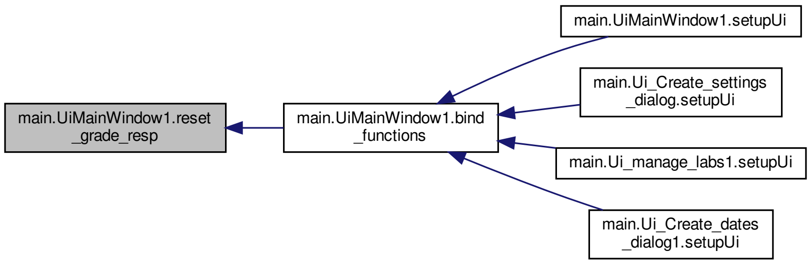
Definition at line 1004 of file main.py.

```
    def reset_grade_resp(self):
        self.disable_fields()
        self.show_stat()
        # self.grader_ref.check_file()
        # if self.grader_ref.check_circ_exist():
        if self.grader_ref.lab_num > 8 and self.grader_ref.lab_type == 'Closed':
           self.grader_ref.final_grade, report = self.grader_ref.precheck_PLDs(self.grader_ref.cur_idx)
           self.input_response_browser.setPlainText(report)
        else:
            self.grader_ref.final_grade = self.grader_ref.lab_max_grade
            self.input_response_browser.setPlainText('I did not find any errors. Good job!')

        self.input_final_grade.setText(str(self.grader_ref.final_grade))
        self.enable_fields()

```

Here is the call graph for this function:


main.UiMainWindow1.reset_grade_resp
main.UiMainWindow1.disable_fields
main.UiMainWindow1.show_stat
main.UiMainWindow1.enable_fields
main.UiMainWindow1.run_logisim
main.UiMainWindow1.kill_logisim

Here is the caller graph for this function:



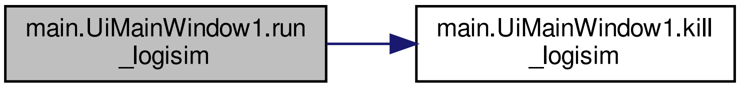
7.17.3.20 run_logisim()

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

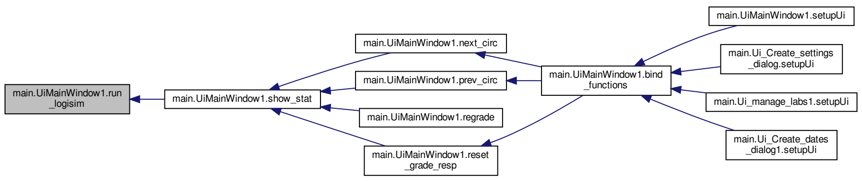
Definition at line 1228 of file main.py.

```
    def run_logisim(self, filename):

        command = 'java -jar ' + self.logisim_path + 'logisim-generic-2.7.1.jar {}'.format(filename)
        # command_with_file = command + os.path.join(self.grader_ref.file_list[self.grader_ref.cur_idx],
MAIN_FILE_NAME)
        #  if self.grader_ref.logisim_pid.pid > 0:
        self.kill_logisim()
        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:



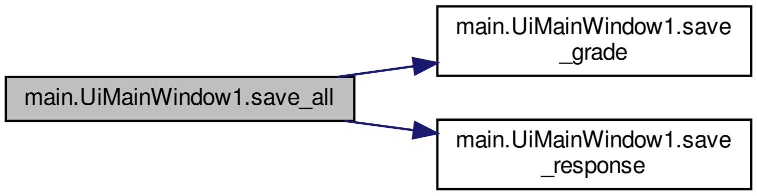
7.17.3.21 save_all()

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

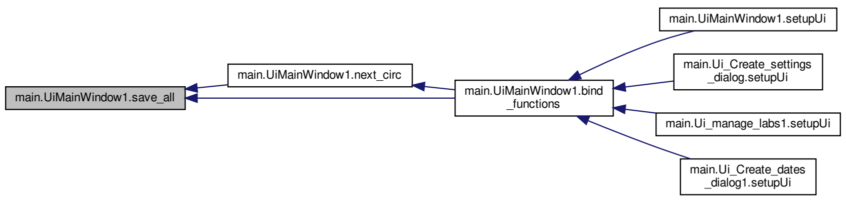
Definition at line 1053 of file main.py.

```
    def save_all(self):
        self.grader_ref.save_grade()
        # self.grader_ref.save_responce()
        self.save_response()
        self.grader_ref.save_all2()

```

Here is the call graph for this function:



Here is the caller graph for this function:


main.UiMainWindow1.save_all
main.UiMainWindow1.next_circ
main.UiMainWindow1.bind
_functions
main.UiMainWindow1.setupUi
main.Ui_Create_settings
_dialog.setupUi
main.Ui_manage_labs1.setupUi
main.Ui_Create_dates
_dialog1.setupUi

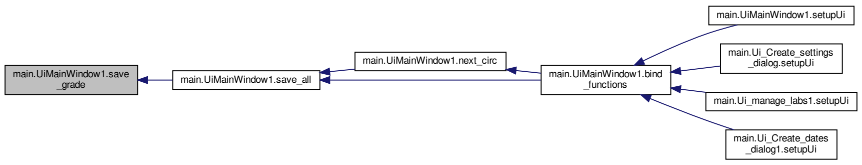
7.17.3.22 save_grade()

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

Definition at line 1035 of file main.py.

```
    def save_grade(self):
        self.grader_ref.save_grade()

```

Here is the caller graph for this function:


main.UiMainWindow1.save_grade
main.UiMainWindow1.save_all
main.UiMainWindow1.next_circ
main.UiMainWindow1.bind_functions
main.UiMainWindow1.setupUi
main.Ui_Create_settings_dialog.setupUi
main.Ui_manage_labs1.setupUi
main.Ui_Create_dates_dialog1.setupUi

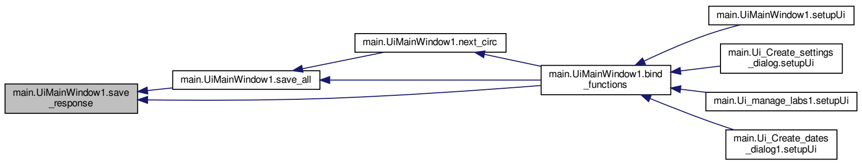
7.17.3.23 save_response()

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

Definition at line 1043 of file main.py.

```
    def save_response(self):
        self.grader_ref.resp_text = self.input_response_browser.toPlainText()
        self.grader_ref.user_comment = self.input_response_browser_user.toPlainText()
        self.grader_ref.save_responce()

```

Here is the caller graph for this function:


main.UiMainWindow1.save_response
main.UiMainWindow1.save_all
main.UiMainWindow1.next_circ
main.UiMainWindow1.bind_functions
main.UiMainWindow1.setupUi
main.Ui_Create_settings_dialog.setupUi
main.Ui_manage_labs1.setupUi
main.Ui_Create_dates_dialog1.setupUi

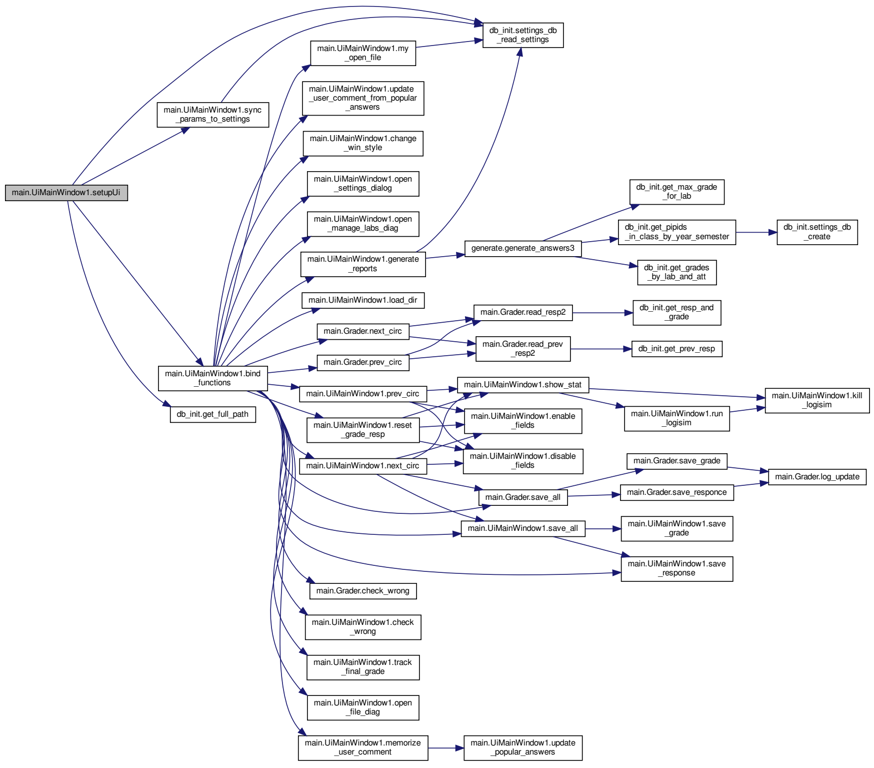
7.17.3.24 setupUi()

```
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         if os.path.isfile(settings_location):
1085             paths, local = settings_db_read_settings()
1086             try:
1087                 if len(os.walk(get_full_path(paths, local) + "/server_sync/").__next__()[1]) >
       0:
1088                     if not self.manage_labs_but.isEnabled():
1089                         self.manage_labs_but.setEnabled(True)
1090                     if not self.but_file_open.isEnabled():
1091                         self.but_file_open.setEnabled(True)
1092                         self.input_file_location.setEnabled(True)
1093             except Exception as e:
1094                 print("Most likely you did not fill all the settings: ", e)
1095 
1096 
```

Here is the call graph for this function:


main.UiMainWindow1.setupUi
main.UiMainWindow1.sync_params_to_settings
db_init.settings_db_read_settings
main.UiMainWindow1.my_open_file
main.UiMainWindow1.update_user_comment_from_popular_answers
main.UiMainWindow1.change_win_style
main.UiMainWindow1.open_settings_dialog
main.UiMainWindow1.open_manage_labs_diag
main.UiMainWindow1.generate_reports
generate.generate_answers3
db_init.get_max_grade_for_lab
db_init.get_pipids_in_class_by_year_semester
db_init.settings_db_create
db_init.get_grades_by_lab_and_att
main.UiMainWindow1.load_dir
main.Grader.read_resp2
db_init.get_resp_and_grade
main.Grader.next_circ
main.Grader.read_prev_resp2
db_init.get_prev_resp
main.Grader.prev_circ
main.UiMainWindow1.bind_functions
db_init.get_full_path
main.UiMainWindow1.prev_circ
main.UiMainWindow1.show_stat
main.UiMainWindow1.kill_logisim
main.UiMainWindow1.run_logisim
main.UiMainWindow1.reset_grade_resp
main.UiMainWindow1.enable_fields
main.UiMainWindow1.disable_fields
main.UiMainWindow1.next_circ
main.Grader.save_grade
main.Grader.log_update
main.Grader.save_all
main.Grader.save_responce
main.UiMainWindow1.save_all
main.UiMainWindow1.save_grade
main.UiMainWindow1.save_response
main.Grader.check_wrong
main.UiMainWindow1.check_wrong
main.UiMainWindow1.track_final_grade
main.UiMainWindow1.open_file_diag
main.UiMainWindow1.memorize_user_comment
main.UiMainWindow1.update_popular_answers

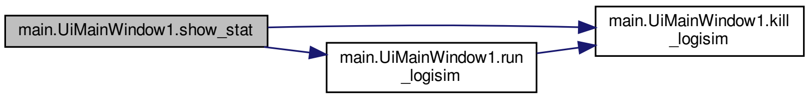
7.17.3.25 show_stat()

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

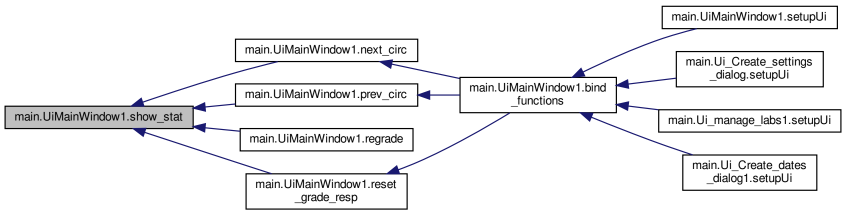
Definition at line 867 of file main.py.

```
    def show_stat(self):
        self.input_prev_response.setPlainText(self.grader_ref.previous_responses)
        file_path = os.path.join(self.grader_ref.lab_paths[self.grader_ref.cur_idx], self.grader_ref.
circ_file_name)
        if not Path(file_path).is_file():
            self.kill_logisim()
            self.grader_ref.final_grade = 0
            self.input_response_browser.setPlainText('File does not exist.')
            self.grader_ref.final_grade = 0
        else:
            if self.but_regrade.text() == '&GRADE' or self.but_regrade.text() == 'GRADE':
                try:
                    self.run_logisim(file_path)
                except Exception as e:
                    print('Error in run_logisim: ', e)
                    print(sys.exc_info()[0])

        self.input_current_id.setText(self.class_id_to_id[self.grader_ref.get_stud_id()])
        self.dateTimeEdit_submitted.setDateTime(QDateTime.fromSecsSinceEpoch(self.grader_ref.timestamps[
self.grader_ref.cur_idx]))
        self.input_subtract.setText('')
        self.input_final_grade.setText(str(self.grader_ref.final_grade))
        self.input_log_browser.setText(self.grader_ref.global_log)
        self.input_response_browser.setPlainText(self.grader_ref.resp_text)
        self.input_response_browser_user.setPlainText(self.grader_ref.user_comment)
        self.checkB_input_pin_status.setChecked(False)
        self.checkB_output_pin_status.setChecked(False)
        self.popular_answers.setCurrentIndex(-1)
```

Here is the call graph for this function:


main.UiMainWindow1.show_stat
main.UiMainWindow1.kill_logisim
main.UiMainWindow1.run_logisim

Here is the caller graph for this function:


main.UiMainWindow1.show_stat
main.UiMainWindow1.next_circ
main.UiMainWindow1.prev_circ
main.UiMainWindow1.regrade
main.UiMainWindow1.reset_grade_resp
main.UiMainWindow1.bind_functions
main.UiMainWindow1.setupUi
main.Ui_Create_settings_dialog.setupUi
main.Ui_manage_labs1.setupUi
main.Ui_Create_dates_dialog1.setupUi

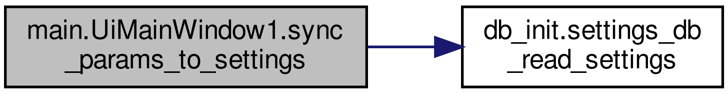
7.17.3.26 sync_params_to_settings()

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

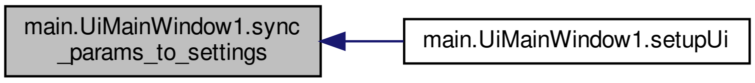
Definition at line 1101 of file main.py.

```
    def sync_params_to_settings(self):
        paths, local = settings_db_read_settings()
        working_dir = ''
        if paths and len(paths) == 4:
            self.logisim_path = paths[0]
            if len(paths[1]) > 0:
                working_dir = paths[1]
            else:
                working_dir = './'
        if local and len(local) >= 4:
            self.grader_name = local[0]
            working_dir += str(local[1])
            working_dir += '_' + local[2] + '/'
            self.set_style_checkbox.setChecked(bool(local[3]))

        if len(working_dir) > 0:
            self.input_file_location.setText(os.path.expanduser(working_dir))


```

Here is the call graph for this function:


main.UiMainWindow1.sync_params_to_settings
db_init.settings_db_read_settings

Here is the caller graph for this function:


main.UiMainWindow1.sync_params_to_settings
main.UiMainWindow1.setupUi

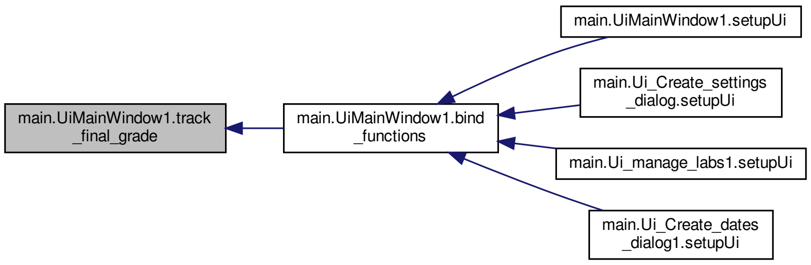
7.17.3.27 track_final_grade()

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

Definition at line 1063 of file main.py.

```
    def track_final_grade(self):
        grade = self.input_final_grade.text()
        self.grader_ref.log_update('Manual grade change from : ' + str(self.grader_ref.final_grade))
        self.input_log_browser.setText(self.grader_ref.global_log)
        self.grader_ref.final_grade = int(grade)
        self.grader_ref.log_update('Manual grade change to: ' + str(grade))
        self.input_log_browser.setText(self.grader_ref.global_log)

```

Here is the caller graph for this function:


main.UiMainWindow1.track_final_grade
main.UiMainWindow1.bind_functions
main.UiMainWindow1.setupUi
main.Ui_Create_settings_dialog.setupUi
main.Ui_manage_labs1.setupUi
main.Ui_Create_dates_dialog1.setupUi

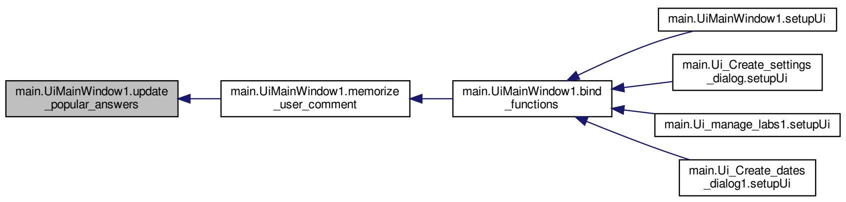
7.17.3.28 update_popular_answers()

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

Definition at line 1024 of file main.py.

```
    def update_popular_answers(self):
        if len(self.popular_answers) != len(self.grader_ref.input_suggestion):
            self.popular_answers.clear()
            self.popular_answers.addItems(self.grader_ref.input_suggestion)
            # for item in self.grader_ref.input_suggestion:

```

Here is the caller graph for this function:


main.UiMainWindow1.update_popular_answers
main.UiMainWindow1.memorize_user_comment
main.UiMainWindow1.bind_functions
main.UiMainWindow1.setupUi
main.Ui_Create_settings_dialog.setupUi
main.Ui_manage_labs1.setupUi
main.Ui_Create_dates_dialog1.setupUi

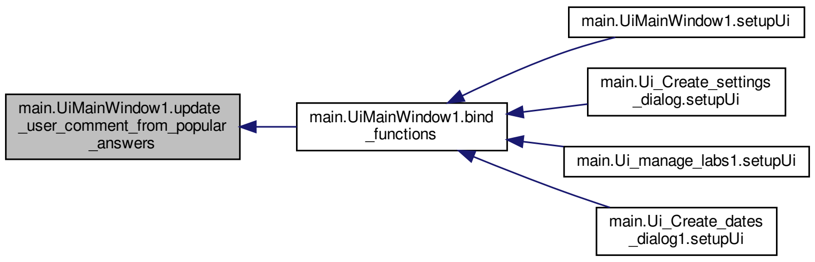
7.17.3.29 update_user_comment_from_popular_answers()

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

Definition at line 1171 of file main.py.

```
    def update_user_comment_from_popular_answers(self):
        if self.popular_answers.hasFocus():
            self.input_response_browser_user.setPlainText(self.popular_answers.currentText())

```

Here is the caller graph for this function:


main.UiMainWindow1.update_user_comment_from_popular_answers
main.UiMainWindow1.bind_functions
main.UiMainWindow1.setupUi
main.Ui_Create_settings_dialog.setupUi
main.Ui_manage_labs1.setupUi
main.Ui_Create_dates_dialog1.setupUi

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