AutoGrader GUI Application - Setup Instructions

A graphical user interface for the AutoGrader system that allows students to check their code against assignment requirements and automatically emails results to the instructor.

Features

- ✅ Student-friendly GUI interface
- ✅ Multiple assignment support via Excel configuration
- ✅ Real-time code checking with detailed feedback
- ✅ Automatic email submission to instructor
- ✅ Student code and results attached to email
- ✅ Persistent student information (no need to re-enter name)
- ✅ Color-coded results (green for pass, red for fail)

Installation

1. Required Python Packages

bash

```
pip install pandas openpyxl matplotlib numpy tkinter
```

Note: `tkinter` usually comes pre-installed with Python. If not:

- **Windows/Mac**: Reinstall Python with tkinter enabled
- **Linux**: `sudo apt-get install python3-tk`

2. Required Files

Place these files in the same directory:

- `autograder.py` - The core AutoGrader class
- `autograder_gui.py` - The GUI application
- `config.json` - Email configuration
- `assignments.xlsx` - Assignment tests configuration

Setup Instructions

Step 1: Create Configuration File

Create a file named `config.json` with your email settings:

json

```json
{
  "email": {
    "smtp_server": "smtp.gmail.com",
    "smtp_port": 587,
    "sender_email": "your_email@gmail.com",
    "sender_password": "your_app_password",
    "instructor_email": "instructor@university.edu"
  }
}
```

Gmail Setup (Recommended)

If using Gmail:

1. **Enable 2-Factor Authentication** on your Google account
2. **Create an App Password**:
   - Go to: https://myaccount.google.com/apppasswords
   - Select "Mail" and your device
   - Generate password
   - Copy the 16-character password (remove spaces)
3. **Use the App Password** in `config.json` (NOT your regular Gmail password)

Other Email Providers

- **Outlook/Hotmail**:
  - SMTP Server: `smtp-mail.outlook.com`
  - Port: `587`
- **Yahoo Mail**:
  - SMTP Server: `smtp.mail.yahoo.com`
  - Port: `587`
  - Enable "Allow apps that use less secure sign in"
- **Custom SMTP**: Contact your email administrator for settings

Step 2: Create Assignments Excel File

Run the provided script to generate the example Excel file:

bash

```bash
python create_assignments_excel.py
```

This creates `assignments.xlsx` with 7 example assignments.

Excel File Structure

Each **sheet (tab)** represents one assignment. Each **row** is a test.

Required Columns:

- `test_type`: The type of test to run (see supported types below)
- Additional columns depend on the test type

Supported Test Types:

Test Type	Required Columns	Optional Columns
variable_value	variable_name, expected_value	tolerance
variable_type	variable_name, expected_value	-
function_exists	function_name	-
function_called	function_name	-
for_loop_used	-	-
while_loop_used	-	-
if_statement_used	-	-
operator_used	operator	-
code_contains	phrase	case_sensitive
plot_created	-	-
plot_properties	-	title, xlabel, ylabel, has_legend, has_grid
plot_data_length	-	min_length, max_length, exact_length
loop_iterations	loop_variable	expected_count

Example Excel Row:

test_type	variable_name	expected_value	tolerance	description
variable_value	total	100	0.0	Variable total should equal 100

Step 3: Generate Example Student Submissions (Optional)

For testing purposes, generate example student submissions:

bash

```
python example_student_submissions.py
```

This creates sample `.py` files for each assignment.

Running the Application

Start the GUI

bash

```
python autograder_gui.py
```

Using the Application

1. **Enter Student Information**
   - Name (required)
   - Email (required)
   - This information persists between submissions
2. **Select Assignment**
   - Choose from the dropdown (populated from Excel sheets)
3. **Select File**
   - Click "Browse..." to select the student's Python file
4. **Check Code**
   - Click "Check Code" button
   - Results appear in the text area below
   - Email is automatically sent to instructor
5. **Submit Another Assignment**
   - Select different assignment
   - Select different file
   - Click "Check Code" again
   - No need to re-enter name/email

Creating Custom Assignments

Method 1: Using Excel (Recommended)

1. Open `assignments.xlsx`
2. Create a new sheet (right-click tabs → Insert Sheet)
3. Name the sheet (e.g., "Assignment 8 - Lists")
4. Add test rows with appropriate columns
5. Save the file

Example Assignment Sheet:

test_type	variable_name	expected_value	tolerance	description
variable_value	my_list	[1,2,3,4,5]	-	Check list contents
variable_type	my_list	list	-	Check variable type
for_loop_used	-	-	-	Must use a for loop

Method 2: Programmatically

Add assignment definitions in `create_assignments_excel.py`:

python

```
assignment8_tests = [
    {
        'test_type': 'variable_value',
        'variable_name': 'result',
        'expected_value': 42,
        'tolerance': 0.0,
        'description': 'result should equal 42'
    },
    # Add more tests...
]

# Add to the writer
pd.DataFrame(assignment8_tests).to_excel(
    writer,
    sheet_name='Assignment 8 - My Topic',
    index=False
)
```

Email Functionality

When a student clicks "Check Code":

1. **Email is sent to instructor** containing:
    - Student name and email
    - Assignment name
    - Submission timestamp
    - Full test results
    - Student's code file (attached)
2. **Email NOT sent if**:
    - Email configuration is incomplete
    - SMTP connection fails
    - A message appears in results indicating the issue

Troubleshooting

"config.json not found"

- Create the file in the same directory as `autograder_gui.py`
- Use the template provided above

"assignments.xlsx not found"

- Run `python create_assignments_excel.py`
- Or manually create the Excel file with proper structure

"Email failed to send"

- Verify email credentials in `config.json`
- For Gmail: Use App Password, not regular password
- Check internet connection
- Verify SMTP server and port

"Script execution failed"

- Check student code for syntax errors
- Review the error message in results
- Ensure all required imports are available

"Module 'autograder' not found"

- Ensure `autograder.py` is in the same directory
- Or install it: `pip install -e .` (if packaged)

Colors not showing in results

- This is normal - colors only appear in the GUI
- Emailed results show PASS/FAIL text instead

Timeout errors

- Increase timeout in AutoGrader initialization
- Default is 10 seconds, can increase to 15-30 for complex code
- Edit line in `autograder_gui.py`: `self.grader = AutoGrader(filepath, timeout=30)`

Customization

Change Window Size

Edit in `autograder_gui.py`:

python

```python
self.root.geometry("900x700")  # Width x Height
```

Change Result Colors

Edit in `autograder_gui.py`:

python

```
self.results_text.tag_config('pass', foreground='green')
self.results_text.tag_config('fail', foreground='red')
```

Change Font

Edit in `autograder_gui.py`:

python

```python
self.results_text = scrolledtext.ScrolledText(
    results_frame,
    font=('Courier', 10)  # Font family, size
)
```

Add Custom Test Types

1. Add handling in `run_tests()` method
2. Add corresponding method in `AutoGrader` class
3. Document in Excel structure

File Structure

```
project_directory/
│
├── autograder.py               # Core AutoGrader class
├── autograder_gui.py           # GUI application
├── config.json                 # Email configuration
├── assignments.xlsx            # Assignment tests
│
├── create_assignments_excel.py  # Script to generate Excel
├── example_student_submissions.py  # Generate test files
│
└── student_submissions/        # Student files (optional folder)
    ├── assignment1_submission.py
    ├── assignment2_submission.py
    └── ...
```

Security Notes

1. **config.json contains passwords** - Add to `.gitignore`
2. **Student code is executed** - Use in controlled environment
3. **Not a complete sandbox** - For production, use Docker
4. **Email credentials** - Use app-specific passwords, not main passwords

Best Practices

1. **Test each assignment** with known-good submissions before release
2. **Provide clear instructions** to students on:
   - Required variable names
   - Required function names
   - Expected output format
3. **Set appropriate tolerances** for floating-point comparisons
4. **Use descriptive test descriptions** in Excel
5. **Keep timeout reasonable** (10-30 seconds)
6. **Backup assignments.xlsx** regularly

Support

For issues:

1. Check this README
2. Review error messages in GUI
3. Test with example submissions
4. Verify all configuration files

Future Enhancements

Potential features to add:

- Student login/authentication
- Grade storage and tracking
- Test case weights for partial credit
- Batch grading multiple submissions
- Export results to CSV
- Student submission history

Version: 1.0
Last Updated: 2025