**Autograder**

**Shane Brandl**

**Bryce Bjorkman**

**David Edwardson**

**Eli Kolb**

**John Snyder**

**Nolan Forehand**

**Tyler Reski**

**Zach Moldenhaur**

**Table of Contents**

[**Installation** 3](#_Toc26196857)

[pyInstall.py 3](#_Toc26196858)

[pyenvsetup.py 3](#_Toc26196859)

[pyreqs.txt 3](#_Toc26196860)

[**File Structure** 4](#_Toc26196861)

[autograder.py 4](#_Toc26196862)

[guiUtils.py 4](#_Toc26196863)

[dynamic\_analysis\_template.py 4](#_Toc26196864)

[tester.py 4](#_Toc26196865)

[commentSummary.py 4](#_Toc26196866)

[**Tutorial** 5](#_Toc26196867)

[**Interacting with Autograder** 6](#_Toc26196868)

[Program Configuration Toggles 6](#_Toc26196869)

[**Team Member Contributions** 8](#_Toc26196870)

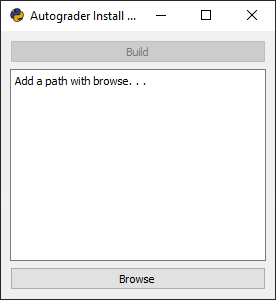
**General**

This project will be developed to assist in the generation of grades for lab assignments in the intro to programming course with Python. This program allows a professor to pass in zip files, directories, and individual files and determines a grade for the submission. This program allows professors to customize grading criteria through a graphical user interface. Student submissions are limited to one script.

# **Installation**

## pyInstall.py

This script will allow for the installation of the Autograder desktop program. Running this script will open a window to select an install directory.



## pyenvsetup.py

This is the environment setup script. It will make sure that the correct version of PyQt5 and other required modules are installed. This script is run by the pyInstall.py script when installing Autograder. If you just wish to run Autograder.py to interact with Autograder, you should run this script before running Autograder.py.

## pyreqs.txt

The following modules are required to run Autograder and are automatically downloaded with the pyenvsetup.py script.

Qt5 version 5.13.1

PyQt5-sip version 4.19.19

python-editor version 1.0.4

pyshortcuts version 1.7

winshell version 0.6

pypiwin32 version 223

pyinstaller version 3.5

# **File Structure**

## autograder.py

Autograder is the GUI that the professor will interact with. It displays options customizing what goes into calculating a student grade. It takes in a professor’s input for script locations and sends each script through dynamic analysis and comment summary.

## strtools.py

This script is a utility class that allows for passing information between the interface and the driver.

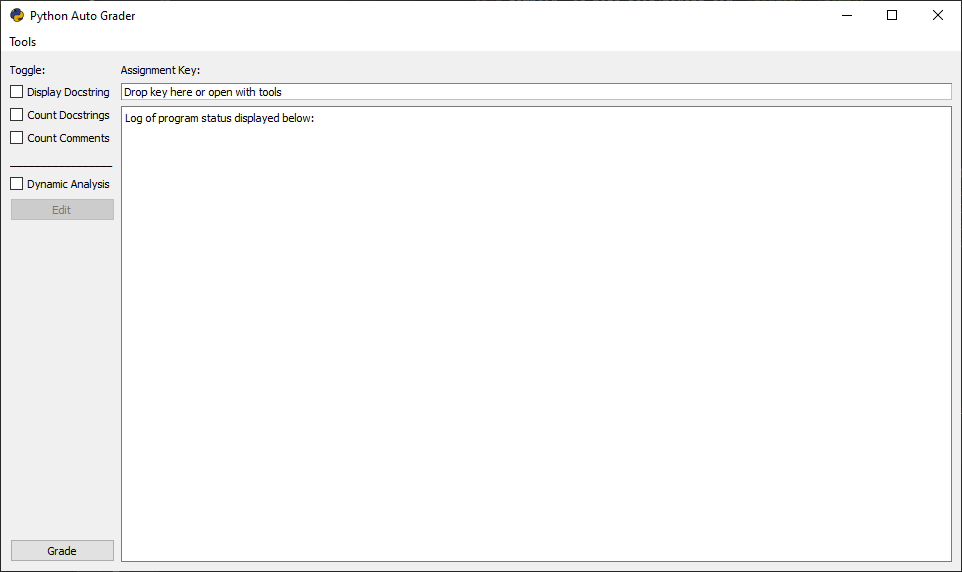
## tester.py

Dynamic analysis defaults to running student scripts as main, though does allow the user to edit the testing framework generated after adding a grading key. The generated testing framework is a typical Python unit test suite with template variables that are used to load student scripts during execution, and allows the user to edit a dictionary of global variables for running student scripts as main, as well as edit method-specific test stubs that pass by default.

## commentSummary.py

Comment summary finds all docString and inline comments in a script.

# **Tutorial**



Step 1 – Select a Key

The first step to grading is inputting a professor’s key. The key is the correct script for a given assignment that student submissions will be compared to. Inputting a key can be done two different ways. The first way is by dragging and dropping the file into the field that says, “Drop key here or open with tools.” The second way is by selecting “Tools” in the upper left corner and then “Open Key.”

Step 2 (Optional) – Unzip a Zipped Directory

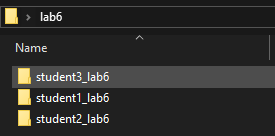
Autograder requires files to be unzipped prior to pressing the grade button. Files can be unzipped with a zip folder utility such as 7-zip, but Autograder also provides this functionality through the tools menu. Press the “Tools” dropdown and press “Import Zip(s).” Select the desired zipped folders to run against the grading key, and then select the location of where the unzipped version of those folders should be stored. By default, this directory is named “studentWork.”

Step 3 – Toggle Desired Grading Criteria

Select all boxes for comment analysis and dynamic analysis for the complete Autograder experience. If a professor does not wish to have the display area cluttered with comment information or simply doesn’t care about comments at all, they should only check the “Dynamic Analysis” field. This field will run every student submission and capture its output.

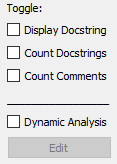
Step 4 – Select Grade

Press the “Grade” button. A new window should pop up that allows you to select a directory. The selected directory should contain subdirectories separated by student. For the following scenario, the “lab6” folder should be selected.



# **Interacting with Autograder**

## Program Configuration Toggles



Autograder’s configuration toggles will allow a professor to customize the output. Checking additional toggles will add more information to the display area.

|  |  |
| --- | --- |
| Prints all docString style comments in a file |  |
| Displays the number of docString style comments in a file |  |
| Displays the number of inline comments plus the number of docString comments. |  |
| Runs student scripts and captures output. If a script does not run, it will print “fails to run.” Script output is broken up into tokens for grading. If a script is expected to output “hello world” and it actually outputs “hello bob,” the grade will be 1/2. |  |
| Selecting “Edit” will open up an editor window. This allows a professor to set global variables by putting them inside “GLOBALS = {}” |  |

**Known Bugs**

# **Team Member Contributions**

Shane Brandl (Project Manager)

Arrange Meeting times/Manage Scrum Board

Manage GIT

Front end design

GUI Design and Implementation

Debugging

Bryce Bjorkman

Dynamic Analysis

Py environment setup

Core grading functions

Key editor generator

MOST Back end code design

David Edwardson

SCRUM board creation/management

Scope definer

SRS/User Manual

sampleHomework examples

Eli Kolb

Front end GUI design

Pyinstaller executable

Debugging

Key editor link up

Testing

John Snyder

File Menu – Import directories, zips and keys

File Dialogs

Grade Button

GUI Design and Implementation

Nolan Forehand

Comment Summary

User Manual

Code Reviews

Program Icon

Debugging

Tyler Reski

Comment Summary

Separate platform tester (Surface)

Drag & Drop key functionality

Front end GUI design

Zach Moldenhaur

Pyinstaller executable

GUI resizing

GUI Design and Implementation

Debugging