Device Automation and Manipulation

MacKenzie Pennington
West Chester University of Pennsylvania, West Chester, PA, 19380, United States

This project was designed for an independent study at West Chester University of Pennsylvania, to be conducted during the Spring 2019 semester. The independent study was an effort to further learn how to code using Python while utilizing concepts of automation learned in previous job experience. The goal of the project was to be able to conduct full automation of both the mouse and keyboard movements. This includes recording the movement and input and replaying it when the user specifies. Fifteen weeks were given to complete the program. With more time, more complex features could have been added. However, the program meets the original goal and intentions stated at the beginning of the project.

I. Introduction

The independent study program at West Chester University of Pennsylvania is designed to allow students to pursue a study that is not typically accessible through the regular curriculum. The original goal of pursuing an independent study was to further my own education with coding concepts in Python, recommended by future employers at Lockheed Martin. After planning the logistics, Dr. Ngo was able to design the study around coding in Python and past experience in writing automated tests using Java.

The project was inspired by artificial intelligence winning a video game tournament. In August 2018, author Vlad Savov of *The Verge* reported a fully automated team of AI bots winning a tournament in *Dota 2*, an online multiplayer battle arena game created by Valve Corporation. The bots designed by OpenAI use machine learning along with keyboard and mouse automation in order to control the characters and gain knowledge on how to win matches against human players. This inspired the idea to use mouse and keyboard movements to automate processes for more academic purposes, such as gathering research materials. While there would not be time to incorporate a machine learning aspect to the study, automating and recording keyboard and mouse interaction was decided to be a more realistic goal for the fifteen week timeline. The following educational goals were set during the study:

1. Create a process that can capture and replay mouse and keyboard movement
2. Take a screenshot and analyze the on-screen data
3. Gain an understanding of the coding concepts and conventions of Python
4. Further current experience in programming automated processes

II. Process

While planning began during the previous semester on Friday, December 7th, the actual project began on Thursday, January 24th, and continued with weekly check-ins every Thursday afternoon. The first step was to conduct research on other, similar programs, and to lay out what versions of different services and libraries to use.

During the research process, two big programs were found that allowed for controlling and recording mouse and keyboard input. Both were found when searching the GitHub databases under the keywords 'mouse keyboard automation'. The program that allowed for both was PyAutoGui, developed mainly by Al Sweigart, the author of *Automate the Boring Stuff with Python*. PyAutoGui is an innovative cross-platform GUI Python module allowing automatic control over the mouse and keyboard. However, there is minimal functionality for recording and replaying, as the main purpose is to simply control the devices. While PyAutoGui is a great tool, its libraries were going to be too complex for a Python beginner and therefore was not going to be much help with the project.

The other program found was called Clicker, by GitHub user niekvdbos. Clicker seemed to do exactly what the independent study is supposed to do. It records and replays mouse and keyboard movement with Python, along with a few other functions. The program also introduced Pandas and Pynput, two essential libraries used in this project.

The research done was extremely helpful in advancing the project. The next step was to create a GitHub repository to house the project. The repository is titled "Pennington-IndependentStudy" with different Python files, a Journal file, and a Project Plan file. Next was to lay out the GUI of the program in order to visualize what functionality was necessary. The original plan was to create a window that displayed a table of step-by-step mouse and keyboard actions with columns labeled "Device", "Coordinates", and "Event". The window would also include buttons to record

actions, play actions, clear the table, and save the series of actions into a reloadable file. A library called Tkinter was chosen to implement this step first. A basic tablet was created, but there was difficulty in embedding button widgets within table widgets, and the learning curve from Java to Python was becoming an obstacle. It was then decided to try PyQt, a library for Python UI development. Though some programmers opt to use PyQt4 still, further research indicated that PyQt5 was going to be best for this process.

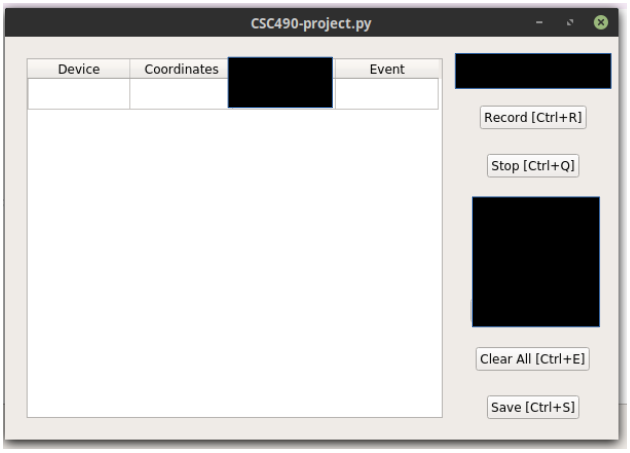


Figure 1. The current GUI window with functions blocked out according to what progress was done at this point in the process

Over the course of February, a basic GUI interface was completed for the project and development for the functionality of all of the buttons began. Online tutorials became very helpful throughout this process. During this time, basic functionality for “Clear”, “Record”, and “Stop” was laid out. Record was the least developed, only populating the table with values before actually recording anything. This was resolved by the end of February, and basic mouse movement was able to be tracked. Using Pynput functions, a listener was created for mouse movement to be recorded. After some discrepancies with populating the table with the recorded actions and timing of the actual recording were resolved, a mouse controller was also created

through Pynput. This allowed for step-by-step replay of everything the mouse listener captured during recording.

In the month of March, the focus was to debug all issues with recording and replaying. A new branch was created in the GitHub repository titled “mouse-recording”, where most of the following work was performed in. Mouse actions became more differentiated, so that instead of just movement, the listener could track movement direction, left-versus-right-clicking, and scrolling direction. The recorded data was converted into Pandas dataframes in order to improve data manipulation. Hot keys were assigned to each button action, so that users could be able to start and stop recording by typing “Ctrl-R” and “Ctrl-Q”, respectively, and clear the recorded actions using “Ctrl-C”. There was also discussion as to whether screen size and resolution would have to be taken into consideration for each device the program was run on, but it was then decided that this would not matter in the long run due to the nature of the recording functions.

By the end of March, the project was in good enough standing to begin incorporating keyboard recording and replaying. A new branch was opened in the GitHub repository titled “keyboard-recording”. During this time, a keyboard listener and controller were both added to the program. A new column was also added to the dataframe, titled “Key”, which would leave the “Coordinates” column blank and indicate the recorded keystrokes. There was many issues in allowing both the keyboard and mouse to listen at the same time as well as control at the same time. These problems were later resolved in April, however the focus for the remainder of March was to make the mouse recording and replaying as seamless as possible.

Mouse recording began to be tested against the National Science Foundation’s websites with attempts to click and download at least one award. This was done by reading mouse actions row by row and implementing each individually recorded action. The goal was to be able to navigate

to the Awards page, and click and download one of the listed awards. Once this became successful, the branch “mouse-recording” was merged into the master branch of the repository.

At the beginning of April, the mouse and keyboard listeners and controllers were finally developed to be able to function simultaneously. The solution was to begin listening and controlling separately rather than forcing the functions to work together. There was also an error in the code that kept changing the keyboard mapping and typing different keys than what was recorded, but it was found to be a simple and fixable typing inconsistency when reading in the objects from the dataframe. Once fixed, the “keyboard-recording” branch was merged into the GitHub repository’s master branch.

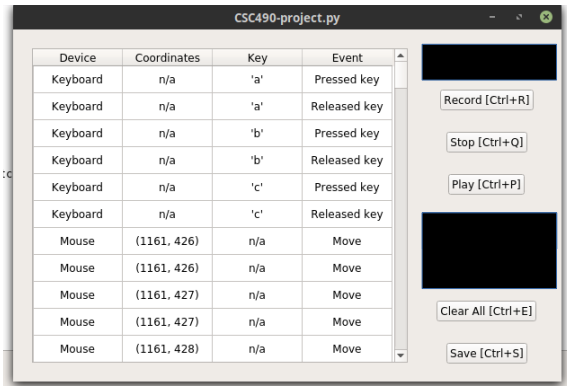
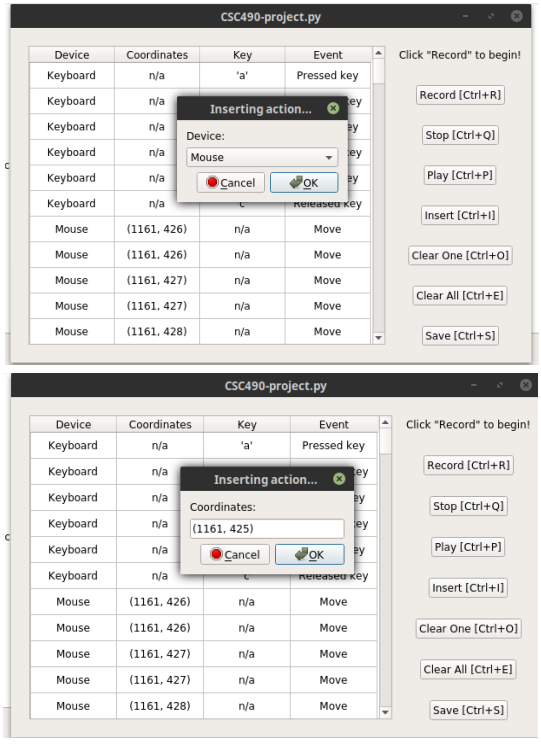


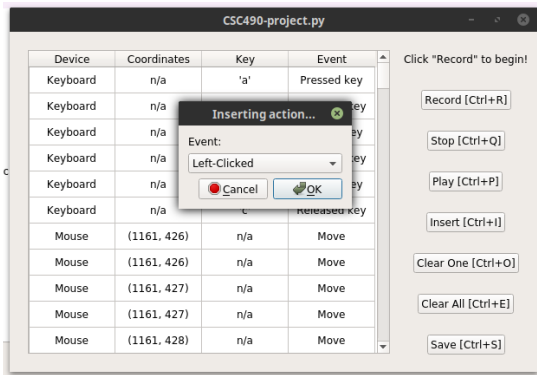
Figure 2. The GUI window with functions blocked out to match the point of progress in the project at the beginning of April

In mid-April, once all device actions were able to be successfully recorded and replayed with little-to-no-error, the main goals of the project were complete. The only one not met was the original goal to take screenshots of the data during recording, but it was decided that adding the functionality would not fit within the scope of the one semester-long project. It was instead encouraged to begin adding smaller features, such as the ability to insert wait-times for windows to load, or to download and rename multiple files on a website at a time. This called for a third branch titled “advanced” to be opened.

Within the “advanced” branch, an “Insert” button was created, and the “Clear” button was split into two: “Clear One” and “Clear All”. A status label was also added above the buttons, instructing users to “Click ‘Record’ to begin!”, however there was difficulties with changing the label along with button presses. The “Clear One” button was able to clear the selected row of actions from the dataframe, whereas the “Clear All” button would wipe the dataframe entirely. The functionality of these two buttons were tested successfully. The “Insert” button was not as successfully developed.

By the end of the study at the end of April, the “Insert” button was able to ask the user for what they wished to insert. A separate window would appear, asking if the user wished to insert a mouse action, a keyboard action, or a wait time. It then asks for the appropriate information for each device, be it keys to enter, coordinates to move or click to, or amount of seconds to wait. Currently, the button can only insert the action to the top or bottom of the dataframe.



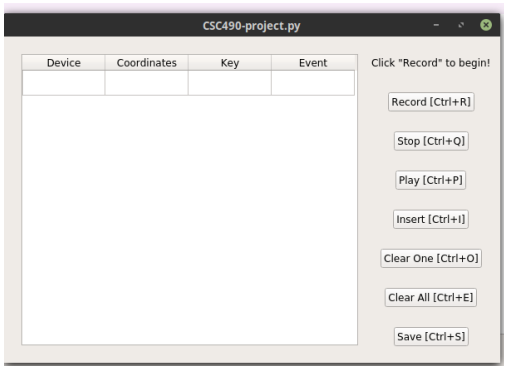


Figures 3-6. Displaying the Insert functionality as a pop-up window asking for user input

Should there have been more time to complete the project, the following goals would have been met:

1. Complete functionality of the “Insert” button, including insertion at a given specific index
2. Insertion of wait-times into the recording process, either through actual recorded wait-times or manually-inputted timing
3. Ability to replay steps between two chosen indexes a predetermined number of times
4. Accurate status labels displayed in the interface window

Figure 7. The interface showing the final product



III. Roadblocks

The following roadblocks were problems that appeared multiple times throughout the project. Most of what is listed was resolved unless otherwise stated.

A. Lack of knowledge and experience coding in Python 3.6.5

Prior to the study, there was minimal experience in coding in Python programming language. Knowledge of the programming language was obtained originally through a Python Boot Camp taught by Dr. Liu Cui on Friday, November 17th, 2017. This experience was then supplemented by online tutorials through Codecademy and Software Carpentry. While learning was the main goal of the entire project, the lack of knowledge became the main obstacle that slowed down the program's development.

This lack of knowledge was mainly displayed when trying to understand Tkinter in the very beginning, in debugging type inconsistencies, struggling with populating the data table versus the console, and issues with creating a smooth user-friendly user interface. There was also minor misunderstandings of how Pandas dataframes truly worked, which were later resolved thanks to a second Python Boot Camp ran by Dr. Ngo himself on Friday, April 5th, 2019.

B. Choosing the correct device automation software

From the project start-date on January 24th until February 22nd, the tool chosen to conduct device automation was PyAutoGui. While there is some link to the previous roadblock of not fully understanding Python, it took time to understand that PyAutoGui was not the correct tool to be using. PyAutoGui only allowed for the controlling of the mouse and keyboard without prior recording. This would have only worked if the program was designed to do specific tasks on a specific website. This was not the intention for the program. In the cases outlined at the beginning of the project, the goal was for the program to be used universally across any web

interface. Though trying PyAutoGui was a great learning experience and could have been used if supplemented with another library, it took more time than expected to finally choose Pynput as the main device automation tool.

C. Timing

The obvious roadblock mentioned throughout Section II would be the timing of the project. Only fifteen weeks were allotted to the project itself. With balancing other classes and a full-time work schedule on top of the project, it became difficult to keep up with. Luckily, the original goals set were met, but it is unfortunate that the second set of goals could only be halfway completed. The intention going forward would be to work on this project in the future in whatever free-time can be found over the next few months.

IV. Conclusion and Self-Assessment

Ending programming work on Thursday, April 25th, the program can successfully perform the following:

1. Capture both mouse and keyboard actions at the same time
2. Record the captured data into a dataframe
3. Replay actions sequentially, iterating through the dataframe
4. Clear all actions recorded or one of the selected rows of actions
5. Insert a row of actions at the beginning or end of the dataframe

Given the above, all but one goal of the project has been successfully completed. Further goals have been created and met as well, even though it is not to completion. To conclude the project, I have gone back and added comments above most of the program's components, and cleaned up unnecessary code (i.e. extra import statements, a nested loop that did not have to be nested, more efficient ways of accessing the cells of a Pandas dataframe).

I have always known that I have been a student who learned better by "doing" and by practical application than by sitting in a classroom listening to lectures and writing notes. I do not believe I created a "perfect" program, or a very efficient one at that. There is definitely more that can be done to follow more typical Python coding conventions and "universalize" the code. However I am proud of the work that has been completed and the concepts I have learned along the way. I hope to continue with this project in the future, in fact I have already begun sharing my repository with certain peers to see if any would be able to offer feedback, better solutions, or comments regarding what could be done differently. My hope from this project was to be able to confidently begin my job as a Software Engineering Associate at Lockheed Martin and learn some Python before starting, and I believe I have achieved that much.

Acknowledgments

A great thank you to contributor, mentor, and professor Dr. Ngo (West Chester University of Pennsylvania, West Chester, Pennsylvania, 19380, United States). He imagined the original concept of the project and was able to guide me through the process, however slowly I learned.

Acknowledgment should also be given to the entirety of the West Chester University of Pennsylvania's Computer Science department for giving me the tools to learn new programming concepts and pursue challenges outside of the curriculum.

References

Electronic Publications

National Science Foundation, *The National Science Foundation* [online database], URL: https://www.nsf.gov/ [retrieved 21 March 2019].

Niekvdbos [username], "Clicker," *GitHub* [online repository], URL: https://github.com/niekvdbos/Clicker [retrieved 31 January 2019].

"pandas.Dataframe," *PyData* [online documentation], URL: https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.html [retrieved 5 March 2019].

PyPi, "Pynput 1.4.2," *Python Software Foundation* [online documentation], URL: https://pypi.org/project/pynput/ [retrieved 22 February 2019].

"Python Basic Tutorial," *TutorialsPoint by* SimplyEasyLearning [online article], URL: https://www.tutorialspoint.com/python/index.htm [retrieved 3 February 2019].

Savov, Vlad, "The OpenAI Dota 2 Bots Just Defeated a Team of Former Pros," *The Verge* [online article], URL: https://www.theverge.com/2018/8/6/17655086/dota2-openai-bots-professional-gaming-ai [retrieved 2 May 2019].

Sweigart, Al, "PyAutoGui," *GitHub* [online repository], URL: https://github.com/asweigart/pyautogui [retrieved 31 January 2019].