# 



Senior Design Seminar

***Project 3 – RDPY User Manual***

***Author:*** Group 3

***Date:*** April 30, 2015

***Version*:** 1

**Group 3:**

* Thomas Larsen
* Nicole Mercer
* Spencer Nuttall
* Ramya Shimoga Prakash

**Document Change Log**

|  |  |  |  |
| --- | --- | --- | --- |
| ***No.*** | ***Version*** | ***Date*** | ***Description*** |
| 1. | 1.0 | Apr-30-2015 | Initial User Manual |
|  |  |  |  |
|  |  |  |  |

# About this manual:

The project was intended to improve and extend the functionality of the existing RDPY package to send mouse and key events to a remote machine. The outcome is our software package based on the RDPY modules that can perform automated testing of a simple graphical user interface (GUI) on a remote machine.

This primary purpose of this user manual is to guide a user with the installation of the software and teach the basic functionality to achieve automated testing of a GUI on a remote machine. The manual provides the user with a plethora of information on various aspects of the software such as:

* Information related to installation of all required libraries/packages on the server and client machine.
* Getting started with the Graphical User Interface to be tested.
* Initiating the Remote Desktop Server to listen to client commands.
* How to begin connection to the remote server from a remote client and initiate automated testing.
* Troubleshooting basic issues.

## Requirements:

The software is designed and found compatible to work in a Windows environment only. To achieve the automated remote desktop testing functionality, the software requires two Windows machine connected to the same local network.

**Version details**:

* Windows – Windows 8 (Windows 7 should be compatible too)
* Python – Python Version 2.7.9
* Pycharm – Pycharm Community Edition 4.0.5

## Keywords:

* RDPY - Remote Desktop Protocol in twisted python (https://github.com/citronneur/rdpy)
* GUI – Graphical User Interface
* Remote Server – A machine that hosts server code & GUI and listens to commands from remote client.
* Remote Client – A machine that hosts client code and sends out commands to remote server.
* In the command Format anything between these symbols ‘< >’means the data is to be replaced with system specific values in the command line. E.g. <port> needs to be replaced with 3389 (no symbols) in command line.

Table of Contents

About this manual: 3

Requirements: 3

Keywords: 4

Overview: 6

Process/ Workflow 6

Installation Process 6

 Adding Python to Windows PATH: 6

 PIP 7

 Remote Server Machine: 7

 Remote Client Machine: 8

Execution Process 9

 Remote Server 9

 Remote Client 12

 Automated GUI Testing 13

Assumptions: 16

FAQs: 17

Team Member Responsibilities 19

 Team Member # 1: Develop GUI Test Application - Nicole Mercer 19

 Team Member # 2: Interpret User Text File Instructions - Spencer Nuttall 19

 Team Member # 3: Interpret Command File Instructions –Thomas Larsen 19

 Team Member # 4: Command execution at Remote Desktop –Ramya Shimoga Prakash 19

Reference: 20

# Overview:

The software package contains python programs that extend the functionality of the GitHub RDPY package to automate the testing of a GUI through a remote connection. Two machines are connected with each other remotely over the local network using the RDPY package. One system acts as the remote server that listens to client commands and executes accordingly. The other acts as the remote client and connects using the server’s IP address and port and sends commands to be executed.

A test GUI application has been designed to check the remote automated testing of a user interface. This GUI application will be hosted in the remote server and is operated by the commands from the remote client.

The installation and step-by-step execution process to test the designed functionality are specified in the following sections.

# Process/ Workflow

## Installation Process

It is assumed that Python has already been installed on the system being used for testing. If not please download and install from the link <https://www.python.org/downloads/>

### Adding Python to Windows PATH:

Ensure that the python and python scripts are already part of your system environment variable PATH, i.e. for python version 2.7, PATH should contain “C:\Python27;C:\Python27\Scripts” directories in it. If not, please follow the below steps to add Python to Windows PATH.

* Right-Click on My Computer / This PC and choose Properties.
* Click on Advanced system settings, under Advanced click on Environment Variables…
* Under System variables, scroll down and choose Path
* Click on Edit… and add “;C:\Python27;C:\Python27\Scripts” if using Python 2.7, and click OK
* Reopen command prompt to make it effective.

### PIP

The libraries can be installed using the pip command. Python 2.7.9 and later included pip by default, if not use the link (<https://pip.pypa.io/en/latest/installing.html>) to install pip on your system.

The easiest way to install libraries from GitHub is to download and extract the zip file, then use pip install command. Example: $ pip install rdpy

### Remote Server Machine:

The following libraries and packages need to be installed on a machine that will act as a Remote Server.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Library** | **Download Link** | **Required** | **Remarks** |
| 1. | PyQt4 | <https://github.com/citronneur/rdpy> |  | Choose the right version for your system from the links given below in RPDY github. |
| 2. | PyWin32 | <https://github.com/citronneur/rdpy> |  | Choose the right version for your system from the links given below in RPDY github. |
| 3. | RDPY | <https://github.com/citronneur/rdpy> | 1 & 2 |  |
| 4. | pyHook | <http://sourceforge.net/projects/pyhook/> |  |  |
| 5. | PyUserInput | <https://github.com/SavinaRoja/PyUserInput> | 2 & 4 |  |

PyQt4, PyWin32 and pyHook can be installed just by executing the downloaded files from the Internet. PIP install command can be used to install rdpy and PyUserInput on the system.

In addition to the above libraries, the system needs the below two programs / files from the software solution package to behave as a Remote Server and host the test GUI application.

* **RDPYServer.py** – Initiates the remote server, listens and executes the client commands.
* **testingGUIWindow.py** – Test GUI application program

### Remote Client Machine:

The following libraries and packages need to be installed on a machine that will act as a Remote Client. Follow instructions mentioned in the Remote Server Machine section above to install the libraries.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Library** | **Link** | **Required** | **Remarks** |
| 1. | PyQt4 | <https://github.com/citronneur/rdpy> |  | Choose the right version for your system from the links given below in RPDY github. |
| 2. | PyWin32 | <https://github.com/citronneur/rdpy> |  | Choose the right version for your system from the links given below in RPDY github. |
| 3. | RDPY | <https://github.com/citronneur/rdpy> | 1 & 2 |  |

In addition to the above libraries, the system needs the below programs/files from the software solution package to behave as a remote client and send commands to the remote server.

* **RDPYClient.py** – Initiates remote connection and sends across the user commands thereby triggering automated testing of the GUI.
* **translator.py** – Reads UserInput.txt and UserInterface.txt files to map the instructions to create a text file (command.txt) containing the consolidated input commands.
* **interpretCommandFiles.py** – Interprets the consolidated input commands to send across mouse and keyboard events accordingly.
* **UserInput.txt** – This text file contains the sequential list of instructions from the user to test the GUI. E.g.: click button1, type Hello world. The user can make changes to this text file to test any other sequence of operation or try different input data. Ensure to include click input and click print commands before and after the type command respectively. This confirms typing of data in the text box and displaying it.
* **UserInterface.txt** – This text file contains the co-ordinates of the buttons on test GUI.

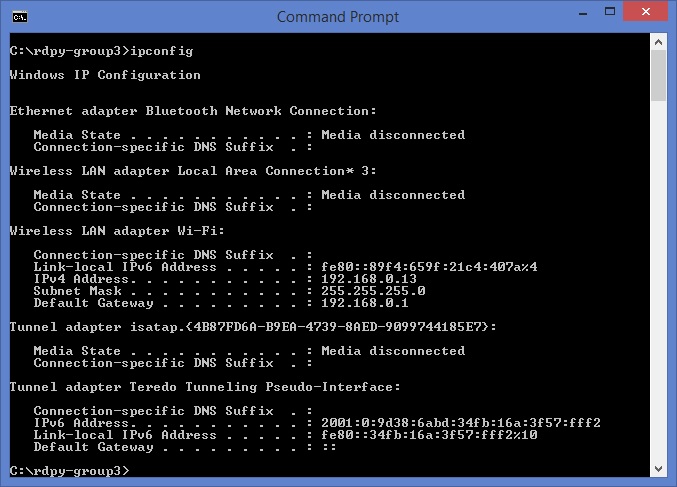
## Execution Process

### Remote Server

Extract the files from the software package “RDPY-Group3.zip” to any directory (e.g. “C:\rdpy-group3”) to be able to access from the command prompt and PyCharm.

#### Initiating the Remote Server

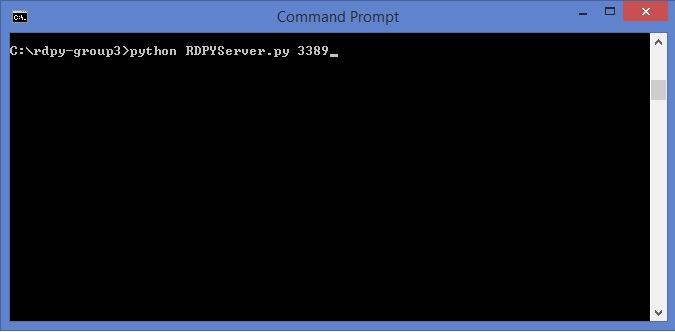
1. Open command prompt and type the command “ipconfig” to obtain the IP address (IPv4 Address) of the machine. **Command:** ipconfig



1. Now change directory structure to directory containing the extracted files. (e.g. “C:\rdpy-group3)
2. Initiate the Remote Server for listening by executing the command below with the port number. It is mandatory to mention port number to get the server up and running. Generally the user can specify the TCP port number 3389 meant for remote desktop protocol. In case if the port is already in use, specify any other open port. The command (netstat -aon | more) can be used on command prompt to see active port connections.
3. The client machine connects to the server using the Remote Server machine IP address and this port number. Type the command below on the command prompt and press enter.

**Command Format:** python RDPYServer.py <port>

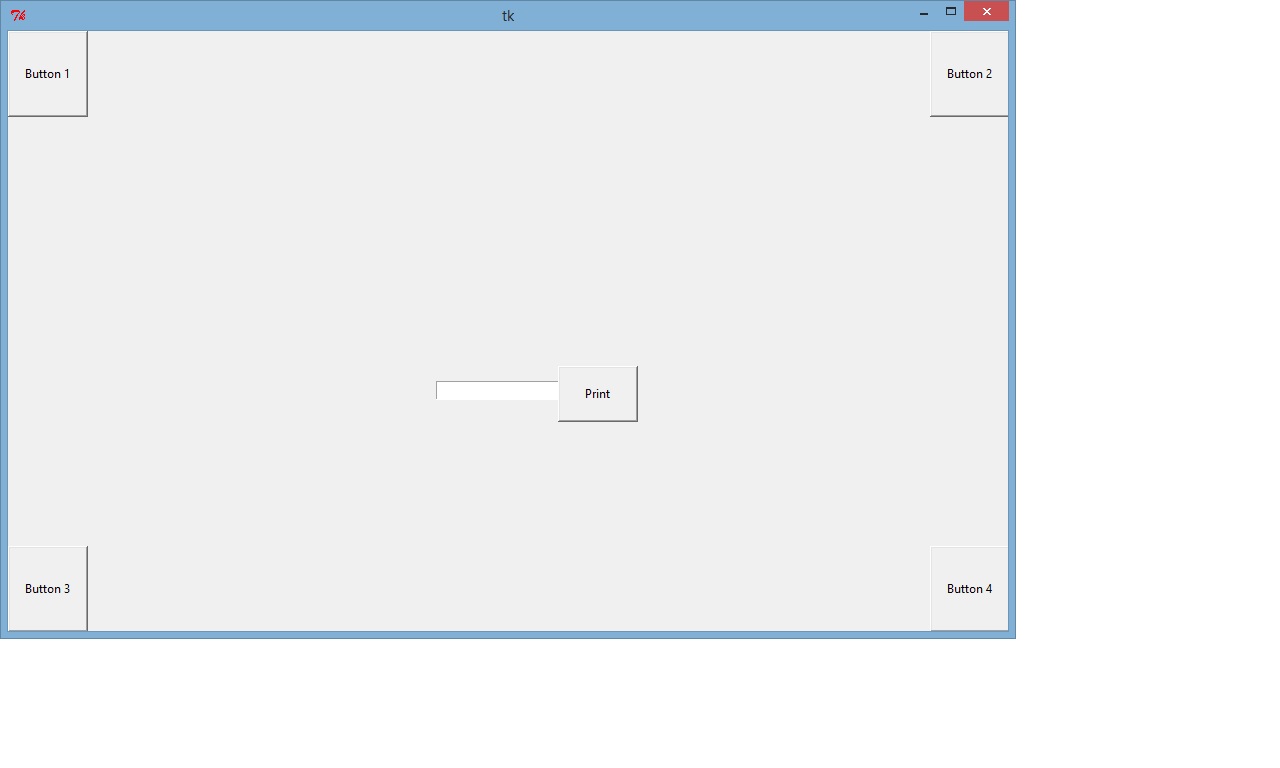
**E.**g. python RDPYServer.py 3389



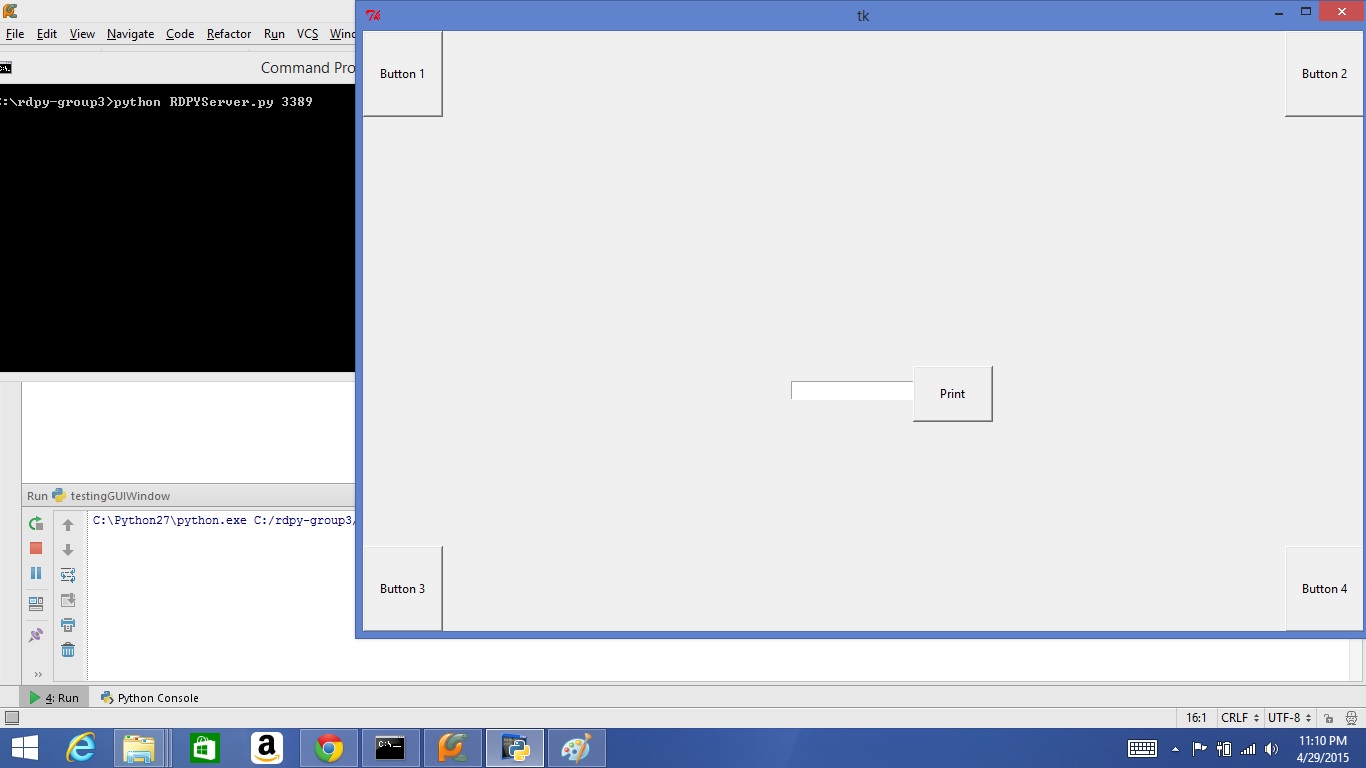
1. The Remote Server keeps listening and then displays the log information once it starts receiving commands from the Remote Client machine.

#### Running the test graphical user interface (GUI)

1. Open Pycharm application and then open the folder where the extracted files from the software package are saved on the system.
2. Now open the file “testingGUIWindow.py” in Pycharm and right-click on ‘testingGUIWindow.py’ and choose Run ‘testingGUIWindow’, you will see a new window open showing the test graphical user interface. Alternatively the program can be executed from a new command prompt using the command “python testingGUIWindow.py”.

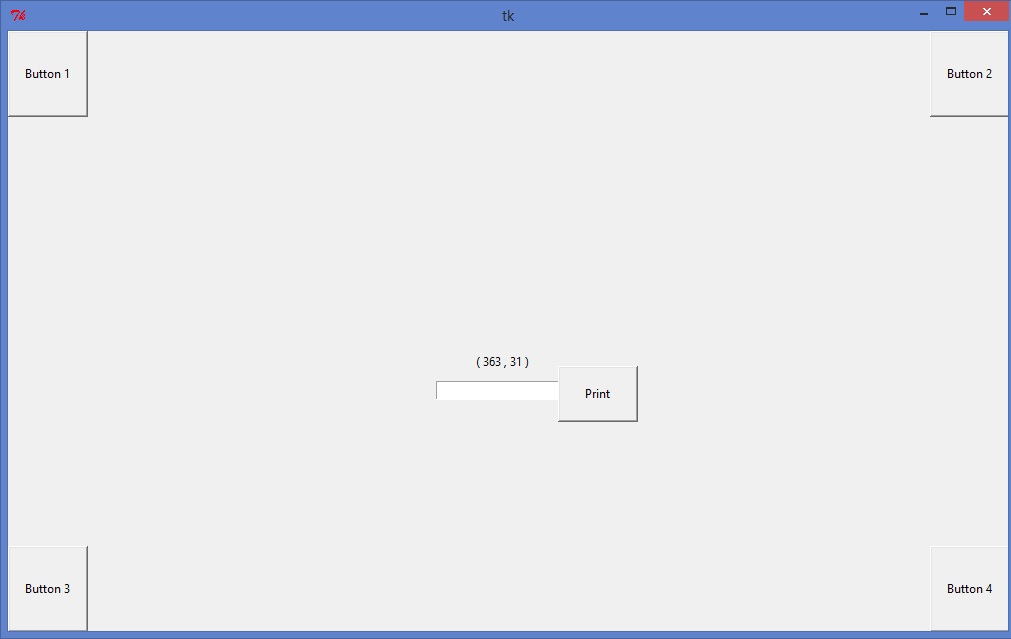


1. Once the GUI is on the screen, position it at a fixed spot on the screen such that all the four corner buttons are clearly visible and the GUI screen is at the foreground with no other screen on top of it.



1. The offset of this GUI is obtained by clicking on the “Button 1” at the top left corner. The offset is displayed above the text box at the center of the GUI. Make a note of this offset to be able to initiate Remote Client. NOTE: Please do not reposition the GUI once the remote client is initiated using the offset obtained. If you reposition the window before initiating the client, obtain the new offset by re-clicking on the “Button 1”.

E.g.: 363, 31 as seen on the screen is the offset



### Remote Client

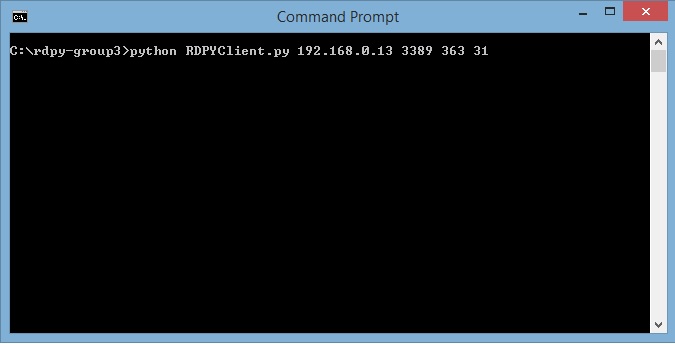
Extract the files from the software package “RDPY-Group3.zip” to any directory (e.g. “C:\rdpy-group3”) to be able to access from the command prompt.

#### Initiating the Remote Client:

1. Open command prompt and change directory structure using the “cd” command to directory containing the extracted files. (E.g. “C:\rdpy-group3”)
2. Initiate the Remote Client by executing the below command with the IP address, port number and the offset co-ordinates. It is mandatory to mention all these four arguments to be able to initiate client and in turn initiate automated testing of the GUI.

**Command Format:** python RDPYClient.py <IP address> <port> <offset\_x> <offset\_y>

**E.**g. python RDPYClient.py 192.168.0.13 3389 363 31



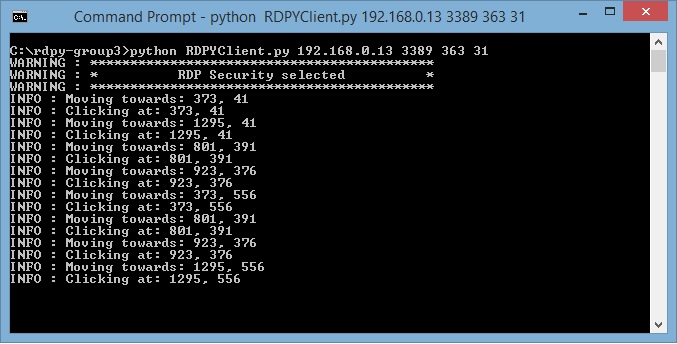
### Automated GUI Testing

Once the server is up and running and the GUI is positioned conveniently, the user can initiate the remote client that in turn triggers the automated testing of the GUI. User input and User interface text files containing the user commands and the co-ordinates of the GUI are read by the appropriate methods to create a file containing the commands that map the user instructions with the GUI co-ordinates. The commands are then interpreted and sent across sequentially to the Remote server to be executed there.

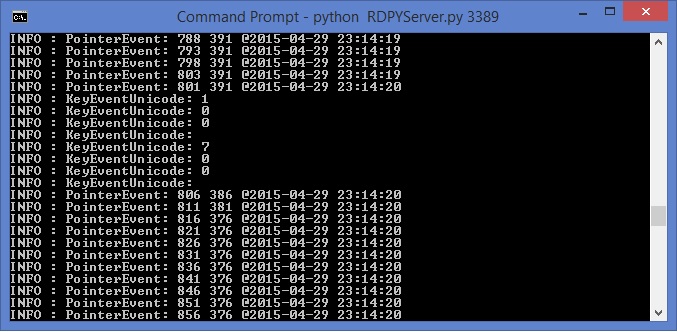
As soon as you initiate the remote client and the connection is successful, you can see that the mouse pointer starts moving across the GUI and executes the user’s instructions sequentially by clicking the corresponding buttons and also typing the text on the input text box. The co-ordinates of the buttons being clicked can be seen above the input text box. And it clicks on print button to display the contents of the input text box.

The following screen shots show the status of the different systems during the execution.

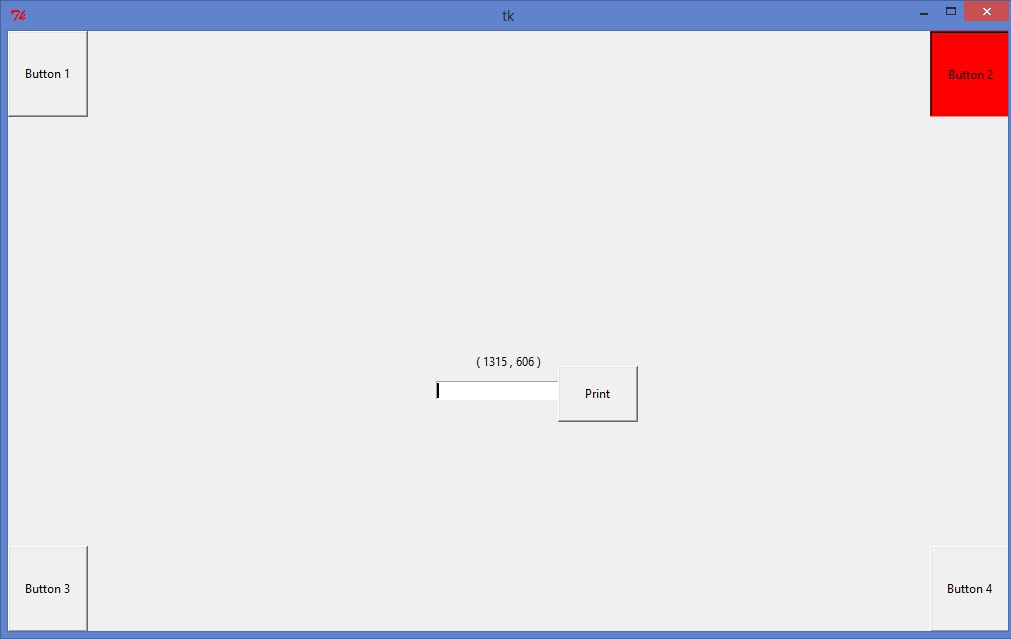
* Remote Client showing the details of mouse pointer events being sent to the server machine.



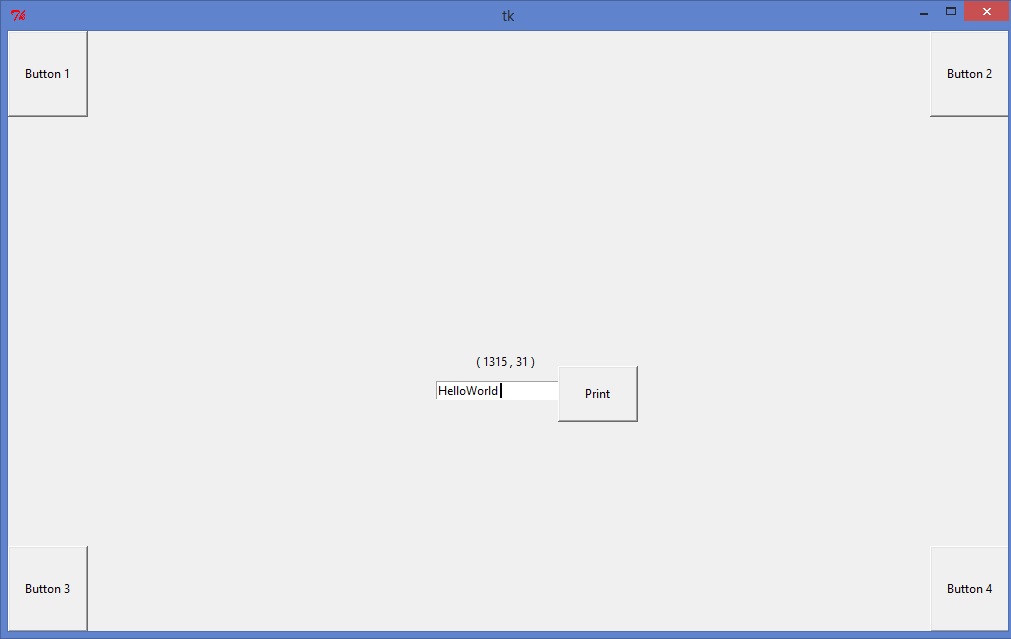
* Remote Server showing the various pointer and keyboard events received from the client.



* The GUI during the execution of the click command at “Button 2”. The button background color changes to red upon clicking.



* The GUI during the execution of a type command. Typing “HelloWorld” in the input text box.



# Assumptions:

1. Since a few of the libraries/packages used in this project are specifically meant for Windows operating system, the project has been implemented and tested using two Windows machines only.
2. The package is designed to work seamlessly with Python v 2.7.9. It may require changes to be compatible with other versions of Python.
3. The remote server machine and the remote client machine are required to be on the same network to be able to connect with each other.
4. All the packages/libraries specified above must be installed successfully.
5. The Remote Server and GUI must be initiated first to obtain required parameters before initiating the Remote Client.
6. Test GUI application should be open and be in the foreground when the Remote Client is initiated. Do not reposition the GUI once the offset is obtained and you start the client. The automated testing of the user command instructions execution will fail in that case since it assumes the GUI to be static.
7. In rare cases of corrupt data transmission, the server throws an exception and stops the execution of the commands. Re-start the server and the client in such a scenario.
8. To stop the Server connection, just close the command prompt.

# FAQs:

***Q: I am facing installation error when using pip install command?***

A: Please make sure PIP is installed on your system successfully; python & python scripts path are added to the environment variable “PATH”. The directories must be seen when you execute the command “echo $PATH” at the command prompt. Command prompt must be restarted after editing the PATH each time. There should be no blank space in the PATH.

***Q: My remote server throws exceptions and stops as soon as the client initiates connection.***

A: This might be due to corrupt data transfer from the client system; the server program throws exception and exits in such case. Kindly re-run the server and client initiation commands in such case.

***Q: The automated test execution was not as per my instructions. It did not click at right buttons.***

A: The offset must be obtained by clicking Button 1 after positioning the test GUI at a fixed spot. If the test GUI is not at the foreground without any other application window on top it or if the GUI is repositioned after obtaining the offset and initiating the client, the mouse click and type commands will not be at appropriate positions as expected.

***Q: I repositioned the test GUI by mistake after obtaining the offset but have not initiated the client yet; do I need to obtain new offset values?***

A: Yes, the offset values must be as per the current position of the test GUI. Get the new offset and initiate the client program to trigger automated testing.

***Q: How do I close the remote server and remote client?***

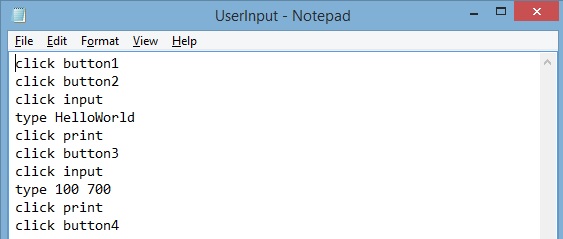
A: Remote client automatically disconnects after all the commands are executed. The Server can be stopped closing the command prompt i.e. by clicking on the red “X” button at the top of the command prompt.

***Q: I initiated the remote client connection and waited for a while, but nothing happens?***

A: It takes a while to establish connection and get things started. So, kindly wait for it to begin execution.

***Q: How do I change the instructions to be executed?***

A: Open the UserInput.txt file from the directory where you extracted all the package files. Type the command “click <button-name>” for mouse click instructions and the command “type <text>” for the keyboard instructions. Hit “Enter” after each command so that there is only one command per line in the text file. Please do not forget to mention “click input” command before the type instruction and “click print” command after the type instruction. Save the UserInput.txt in the same directory as the rest of the files.



# Team Member Responsibilities

### Team Member # 1: Develop GUI Test Application - Nicole Mercer

* + Developed the GUI application program - *testingGUIWindow.py*
  + Created Design document.
  + Review of User Manual & other programs.

### Team Member # 2: Interpret User Text File Instructions - Spencer Nuttall

* + Developed the translator program which creates a consolidated command file by mapping user instructions and the interface details – *translator.py*
  + Created Design document.
  + Review of User Manual & other programs.

### Team Member # 3: Interpret Command File Instructions –Thomas Larsen

* + Developed the interpreter program that parses the command file instructions to mouse and keyboard events correspondingly– *interpretCommandFiles.py*
  + Created Design document.
  + Review of User Manual & other programs.

### Team Member # 4: Command execution at Remote Desktop –Ramya Shimoga Prakash

* + Developed the remote server and client programs that handles the remote connection and executes user instructions and integrated the solution – *RDPYServer.py & RDPYClient.py*
  + Created Design document.
  + Created User Manual & Review of other programs.

# Reference:

<https://github.com/citronneur/rdpy>

<https://docs.python.org/2/tutorial/>

<https://github.com/SavinaRoja/PyUserInput>