=

MAPLESTORY M
AUTO QUEST BOT

COMPLETE GUIDE

Contents

1. INTRODUCTION

Doing quests in Maplestory M is the fastest way to level up, at least until you've reached a triple digit level. With the auto quest feature, it is now more hands off than ever. However, having to stay on the phone to click buttons for the next quest is a chore, plus it builds the very unhealthy habit of checking the phone for that little dopamine rush ;) It feels like a path toward addiction. I decided to create a maple auto quest bot so I can leave the game running and be more productive with life.

If it is your first time implementing a bot or coding, fret not. This guide will outline the exact steps to implement the auto quest bot. On the plus side, you'll also learn how to set up a coding environment, and understand how the code works. These are useful skills, and hopefully this inspires some people to pick up coding.

A few caveats before we start:

- Its a quest bot coded in python. I've tried to make it as comprehensive as possible for people who have not coded before, but it has a long set-up time if you do not have any experience.
- I've only tested this on Windows. I think for Mac it has to be changed because of the way Pyautogui works. Workarounds for Mac can be found here.

2. INSTALLING THE NECESSARY PROGRAMS

Bluestacks

Download bluestacks here, an android PC emulator. Latest bluestacks version: 3N

In order to get Maplestory M working on your computer, you have to install an emulator. I've tested the a few andriod emulators, and found bluestacks to be the best plug and play . If you have an IOS phone, fret not. You can just sign in to your google account and run andriod on your PC for Maplestory M.

Python

1. Download some version of python if you dont have one. For this tutorial, I'll be advocating downloading Anaconda.

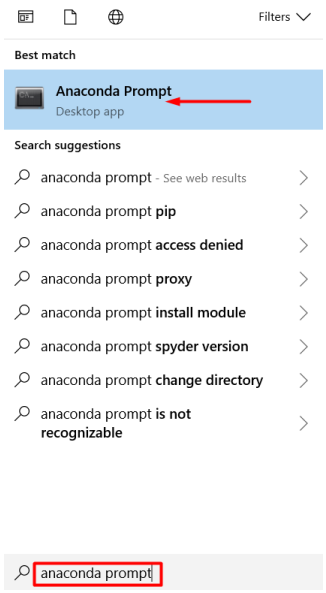
- What is Anaconda? To keep it simple, it is an integrated package that contains python, and Jupyter Notebook.
- What is Jupyter Notebook? Jupyter Notebook is a web-application which you can use to run python code. It is great for testing if each button of our code works because we can run sections our code in different cells.

2. Open the anaconda command prompt (Instructions below). Install ***Pyautogui*** package by copy and pasting in the following command:

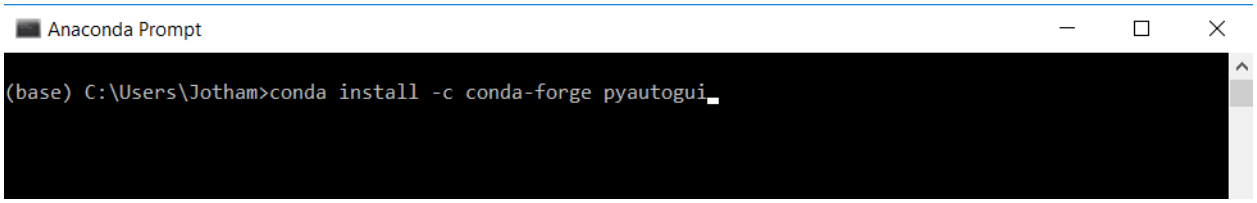
```
conda install -c conda-forge pyautogui
```

Pyautogui is a python library that can control your mouse / keyboard, perfect for our needs. Its documentation can be found here.

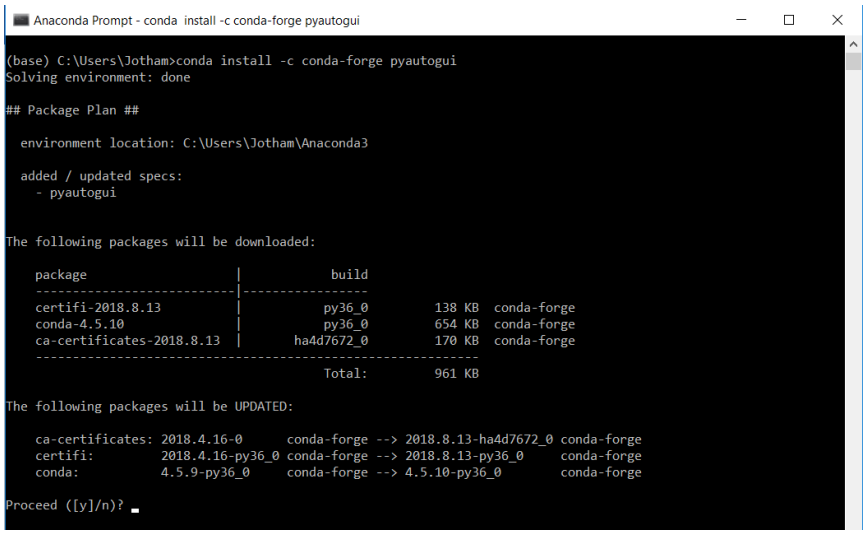
- How to open the anaconda prompt: Search for "*Anaconda Prompt*" using your windows search function



- Paste in the above command in anaconda prompt to install ***Pyautogui*** and *press enter*



- They will ask you if you're sure, type in "y"



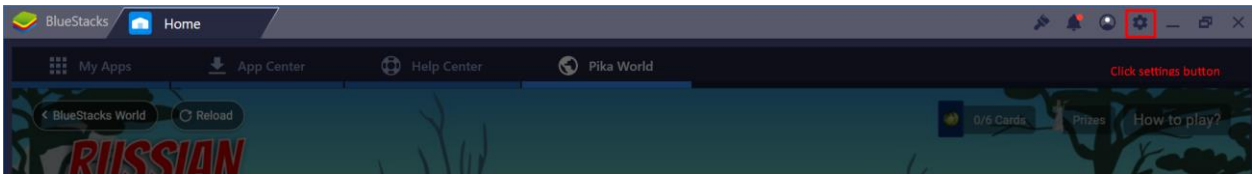
After installing the Bluestacks emulator, Anaconda, and Pyautogui in Anaconda, you're all set! Now let's see what's next.

3. SETTING UP BLUESTACKS AND PYTHON

We now must make sure all of our installed programs work and configure them.

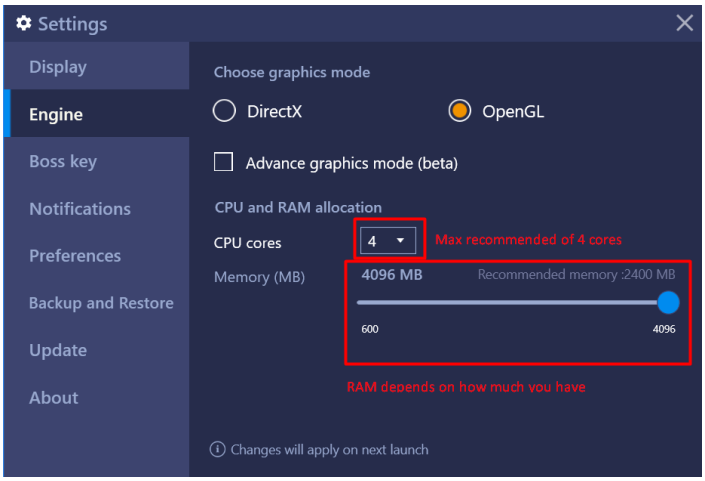
Bluestacks

a. Open Bluestacks, and click the settings button at the top right

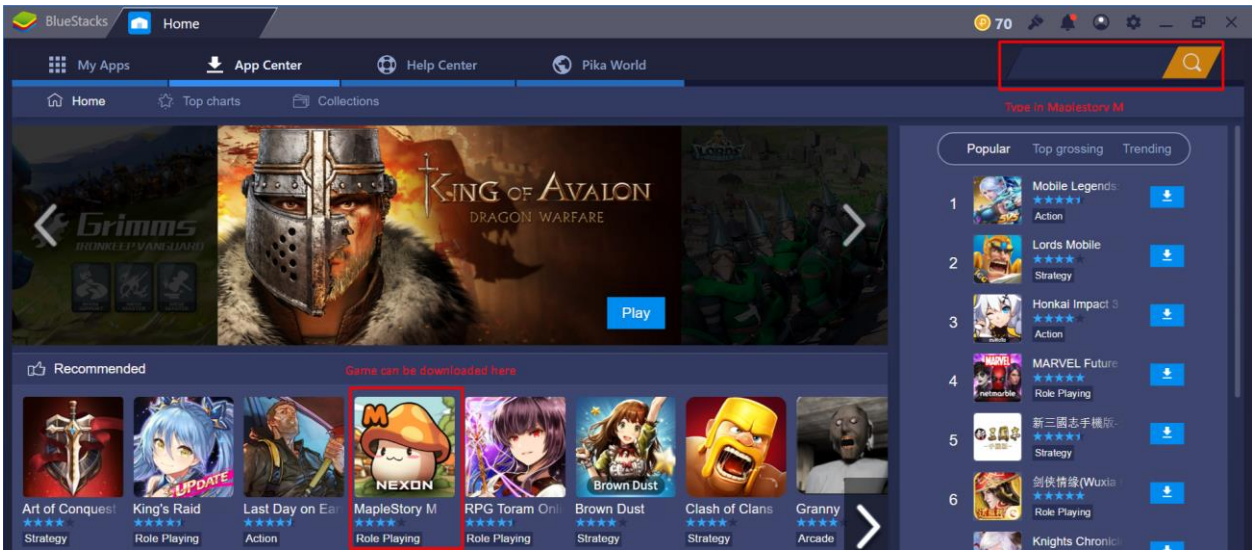


b. Next, adjust the Aspect ratio and DPI so Bluestacks won't take up so much of our computing power.

c. If you want, you can also adjust CPU cores and Memory. I recommend a max of 4 cores. RAM will depend on how much you have, I have 16GB and so allocating 4GB to Bluestacks is fine.



d. Next, we want to download Maplestory M on Bluestacks. Download and install the necessary patches, and you can login to your account.

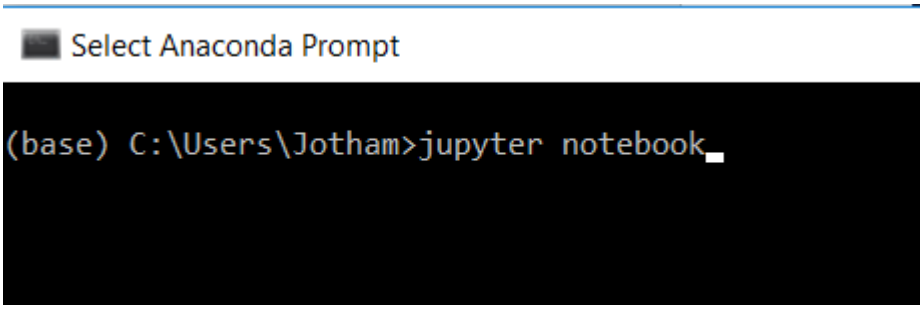


Downloading Maplestory M

You should be able to login to your maplestory account / create a new account. Let's move on to setting up our coding environment.

Python

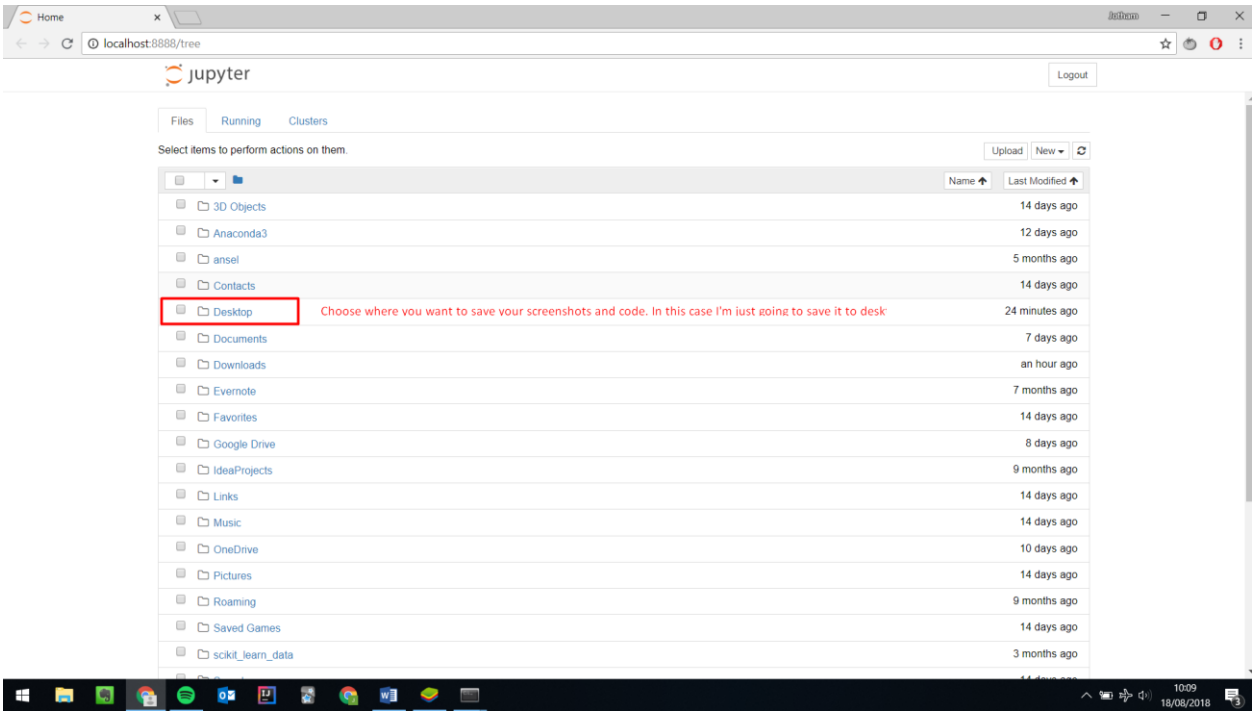
1) Open anaconda prompt again (instructions above here), and type "***jupyter notebook***" in anaconda prompt to run jupyter notebook. Press enter and wait.



Opening Jupyter Notebook in your browser

2) It should open up your browser with the following screen. You have to choose where you want to run your program from. Make sure the python file I sent is already in the folder you want to run. In this case, I saved the file to my desktop, so I click "Desktop".

Caveat: If your python file is in another drive (e.g. G:\), then you have to change the drive before starting jupyter notebook. I recommend just putting it in C:



Choose folder where you saved your *MapleM_bot - Carousell*

3) Click the python file to start the code

Great! We're all set to start editing the code.

4. UNDERSTANDING THE CODE

Overview

The code looks to see if a particular button is on the screen. If the button is present, it will click it. What we have to do is to add each button. There are 2 main types of buttons: orange buttons and the “Start Quest and Skip Buttons”. Below is a brief introduction on each type of button.

Orange Buttons

The code looks at screenshot of buttons (e.g. the "Claim Reward", “Confirm”, “Complete” button) and clicks the button based on the screenshot. This screenshot feature works for all the orange buttons.

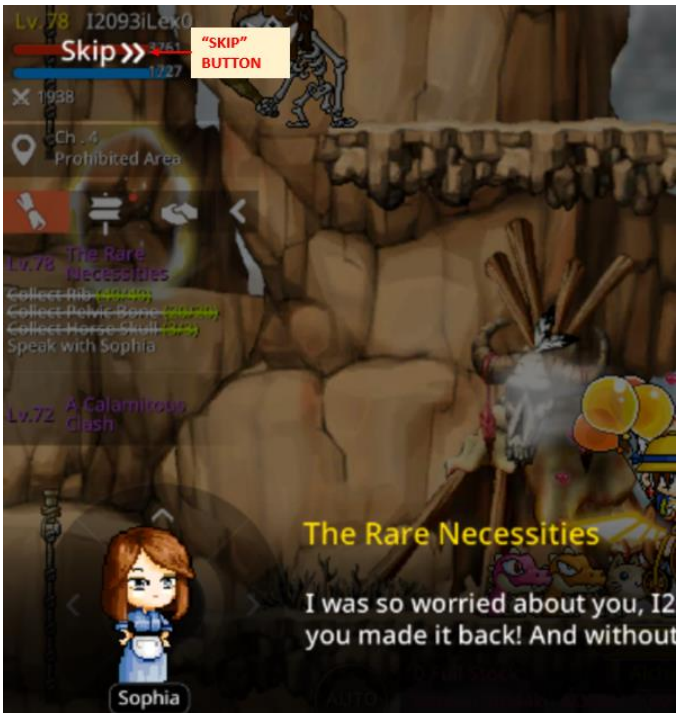
Examples of orange buttons

Section 6: Clicking Orange Buttons will teach you how to edit the code to add these orange buttons. If you do not know how to edit python code in Jupyter Notebook, go to Section 5: Editing Code in Jupyter Notebook

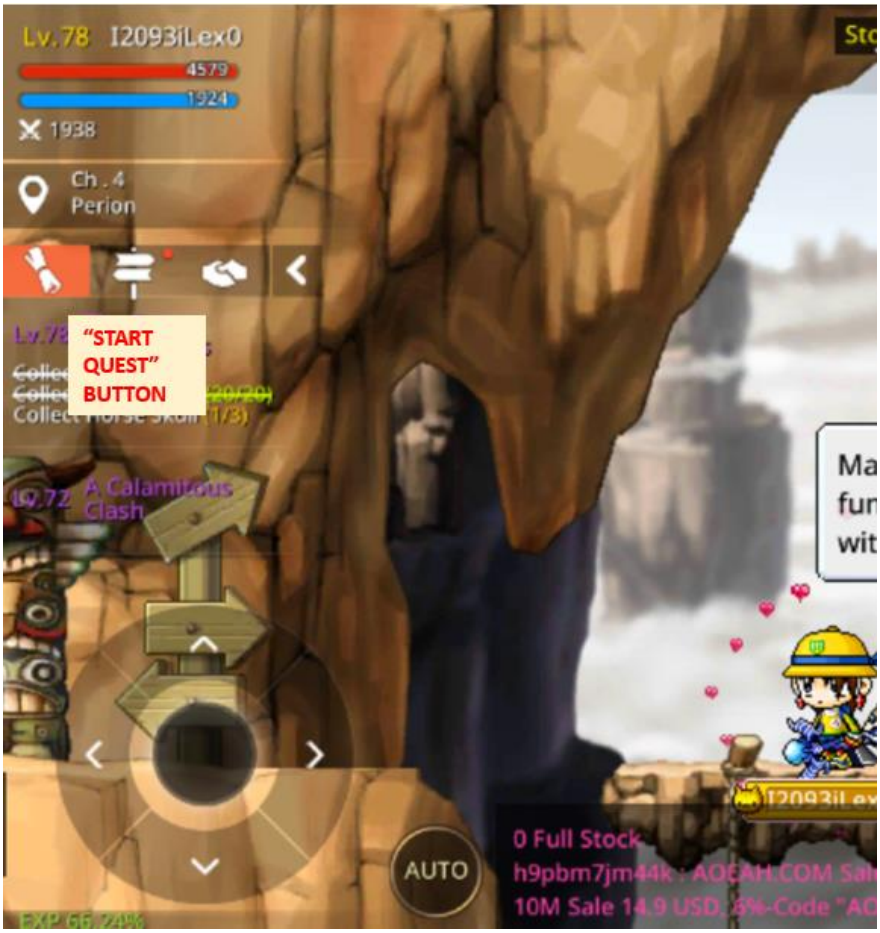
“START QUEST” and “SKIP” Buttons

There are 2 buttons (the "Skip" and the "Start Quest" button) which are not orange, and have a translucent background. As such, the button will look different each time your character moves. Fret not, there are workarounds by looking at pixels.

Section 7: Clicking “Skip” and “Start Quest” Buttons will teach you how to set these buttons up. If you do not know how to edit python code in Jupyter Notebook, go to Section 5: Editing Code in Jupyter Notebook



"Skip" button



"Start quest" button

5. EDITING CODE IN JUPYTER NOTEBOOK

You need to know 2 things to run code in Jupyter Notebook:

a) How to add in code (to add buttons)

There are 2 modes in Jupyter Notebook: “Edit mode” and “Command Mode”.

Edit Mode

When a cell is in edit mode, you can type into the cell, like a normal text editor.

We go into “Edit Mode” of a particular cell by pressing ***Enter***. If you want to exit out of edit mode, press **ESC**.

Command Mode

When you are in command mode, you are able to edit the notebook as a whole, but not type into individual cells. Most importantly, in command mode, the keyboard is mapped to a set of shortcuts that let you perform notebook and cell actions efficiently.

b) How to run code

You can run code cell by cell. To execute a cell, press “**Ctrl” + “Enter”**.

6. CLICKING ORANGE BUTTONS

To Do before adding any button:

a. **Make sure your bluestacks window is in full screen**
b. **Make sure you have configured your bluestacks resolution (See Section 3)**
c. **Make sure you're using the computer you plan to run the bot on**

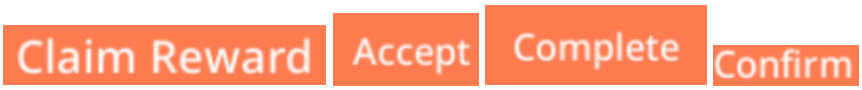
How It Works

Orange buttons are the main buttons for quests. We take screenshots of each button and add them to our code. If our code finds that a button on the screen matches the screenshot that we have taken, our code will click the button.

Commonly Clicked Orange Buttons

Below is a list of the main orange buttons needed to automate quests. You have to take your own screenshots as your screen resolution will be different than the screenshots that I have taken.

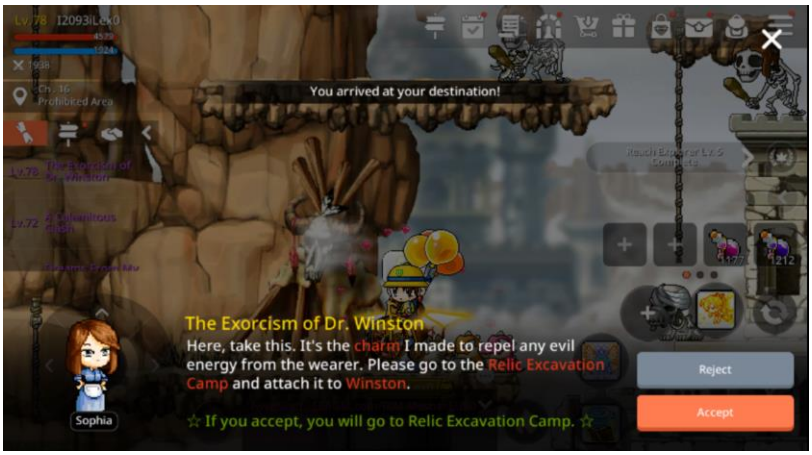
- "Complete" button
- "Accept" button
- "Claim Reward" button
- "Confirm" button



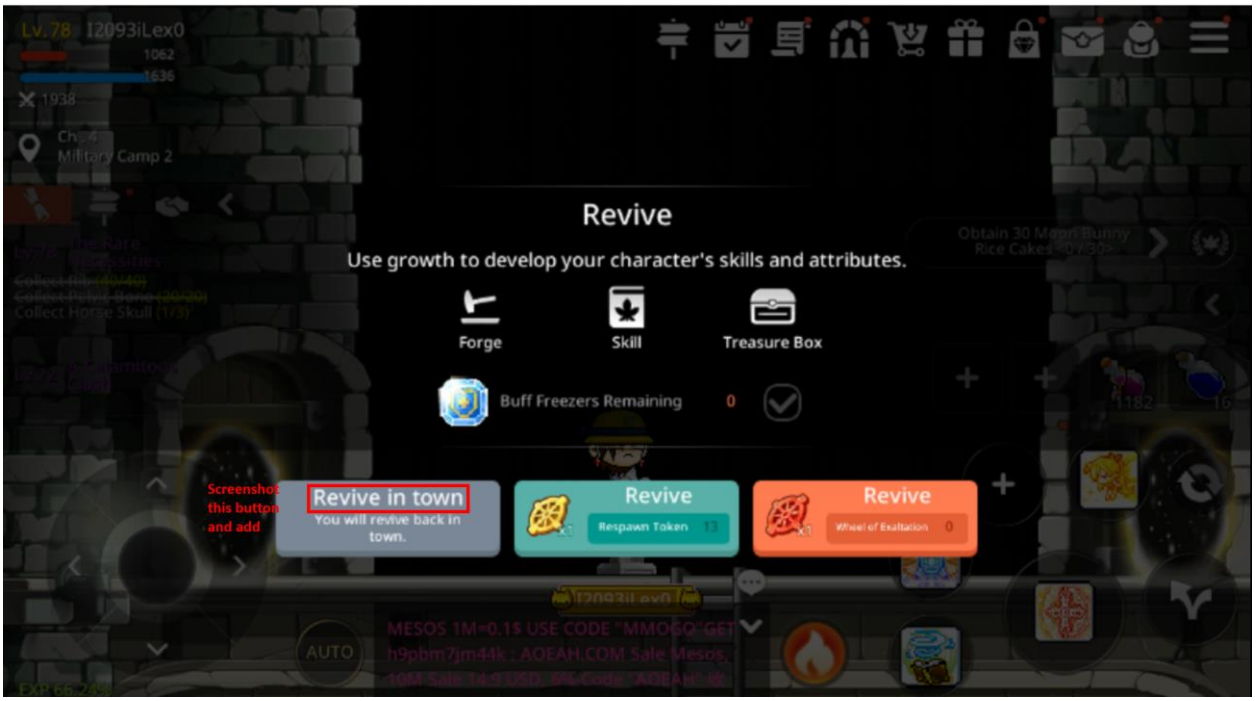
Less Commonly Used Buttons

Below are some buttons which are less commonly encountered. The first time you encounter such a button, the bot will stop. However, just add the button into the code so that the program can recognize the button in future.

Do note that some buttons may look the same, but are in fact different sizes. So your best bet is to add the button into the program should your code freeze. Detailed steps on how to do this is found below.



Size of this "Accept" button different from the normal "Accept" button. The first time you encounter this button, the game will stop. Add this button into our code the first time you encounter it, and you won't encounter this problem after



The first time you die, add the "Revive in town" button, and the next time you die, the quest you auto start

Steps to add any "*Orange*" button (Applicable to any button you can take a screenshot on)

1) Open *bluestacks* and launch Maplestory M, keeping in mind to full screen the *bluestacks* window.

2) Open up *Jupyter Notebook* and the MapleM_bot - Carousell (instructions above here)

3) Execute the first cell (by pressing "***Cltr***" + "***Enter***"). Doing so imports the *pyautogui* and *time* modules into your Jupyter Notebook. With this we can run all other cells in the notebook.

```
In [ ]: import pyautogui
        import time
```

Execute this cell

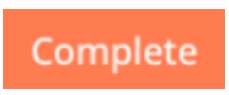
4) Go into Maplestory M game and start a quest. Take a screenshot of a commonly used orange button. I am going to use the "Complete" button here to illustrate the steps. But the following steps will be applicable to all other orange buttons.

For easy screenshots, I recommend downloading this free program, Lightshot.

Complete button will be used as an illustration. Take a screenshot of this button

5) Save the screenshot as a *png* file. In this case, I am going to save it such that the filename is "*complete_button.png*"

You have to make sure you save the screenshot in the same folder as the MapleM_bot. In this case, as my MapleM bot is on my desktop, I will just save the screenshot on my desktop as well.



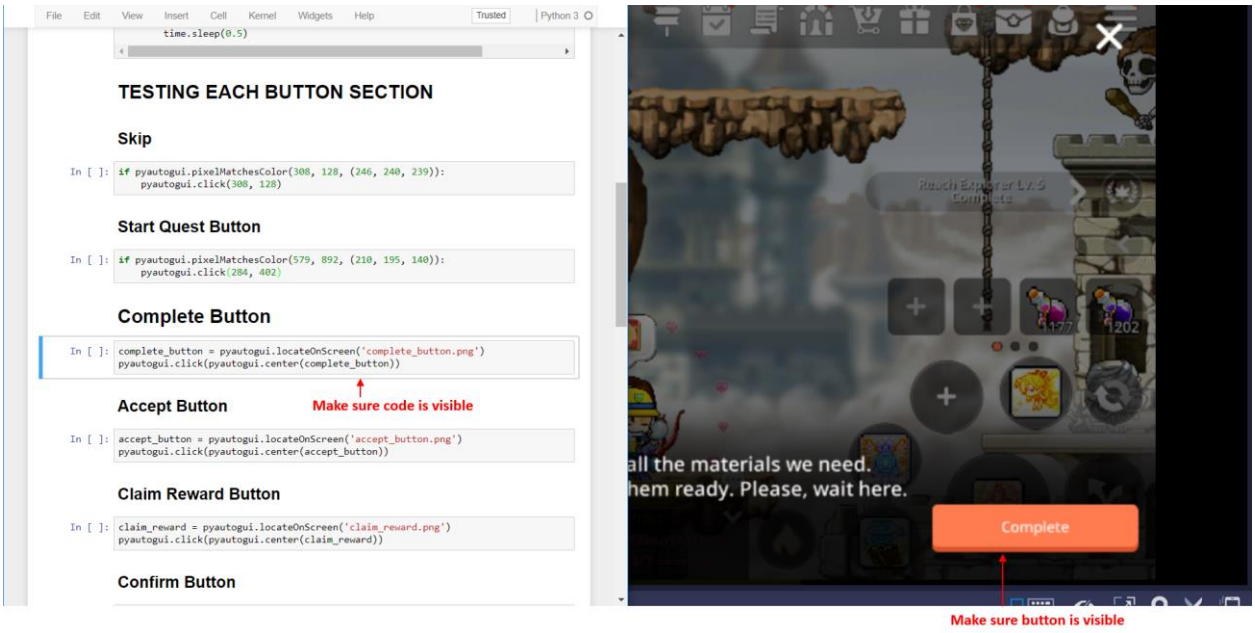
Screenshot of "complete" button

6) Under the "**TESTING EACH BUTTON SECTION**" in Jupyter Notebook, find the orange button you are clicking. In this case, I will look for the "*Complete*" button. I will rename the file to what the screenshot was saved under, as per the diagram below.

Look for the button under the "TESTING EACH BUTTON SECTION" and rename the file

7) Next, arrange your screen such that your Jupyter Notebook as well as the button you are clicking on Bluestacks is visible. You still have to make sure your Bluestacks window is fullscreen at all times.

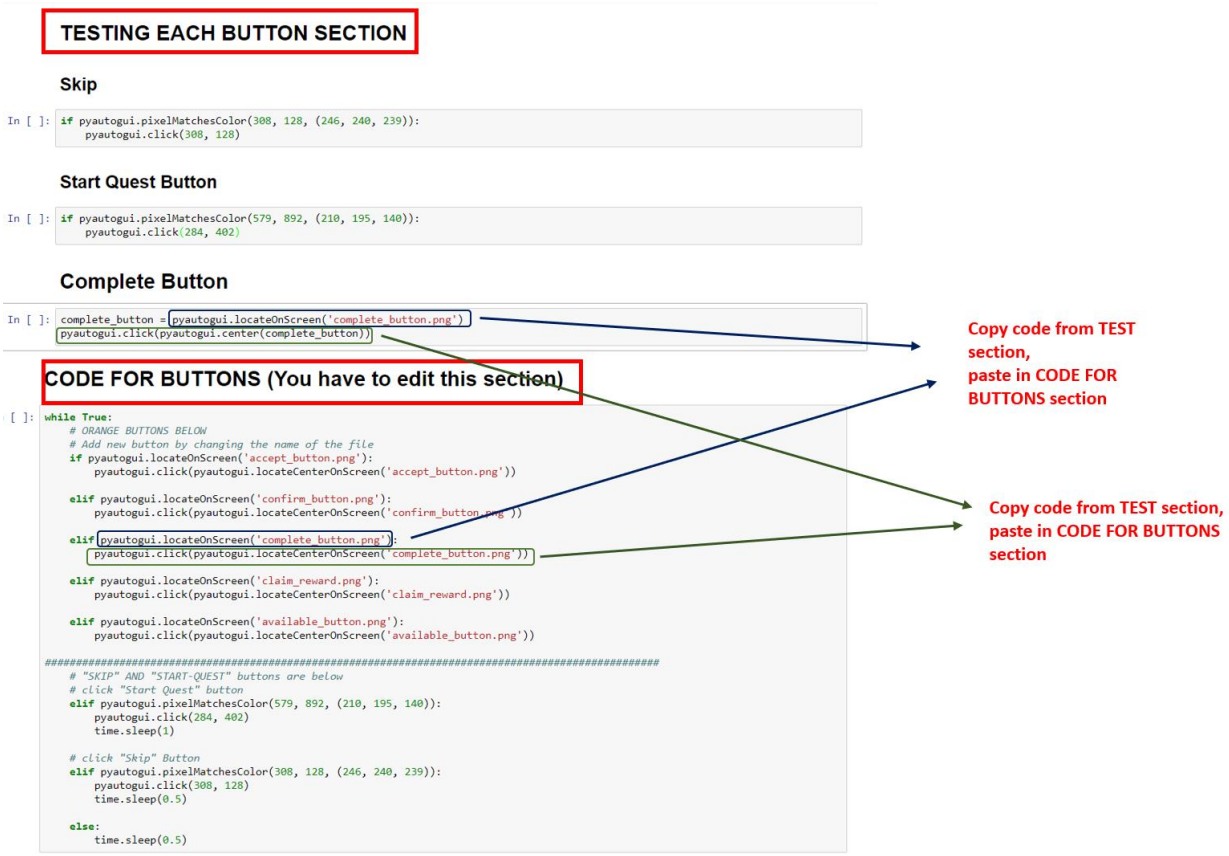
Since I am clicking the "Complete" button here, I will arrange my screen as such.



Making sure both the button and Jupyter Notebook is visible

8) Execute the cell in Jupyter Notebook by pressing "**Cltr**" + "**Enter**". It should click the complete button.

9) We now have to add the code to our program. Copy and paste the code from the "**TESTING EACH BUTTON SECTION**" into the "**CODE FOR BUTTONS**". The diagram below illustrates this.



Copy the code that worked in the "TEST" section into the "CODE FOR BUTTONS" section

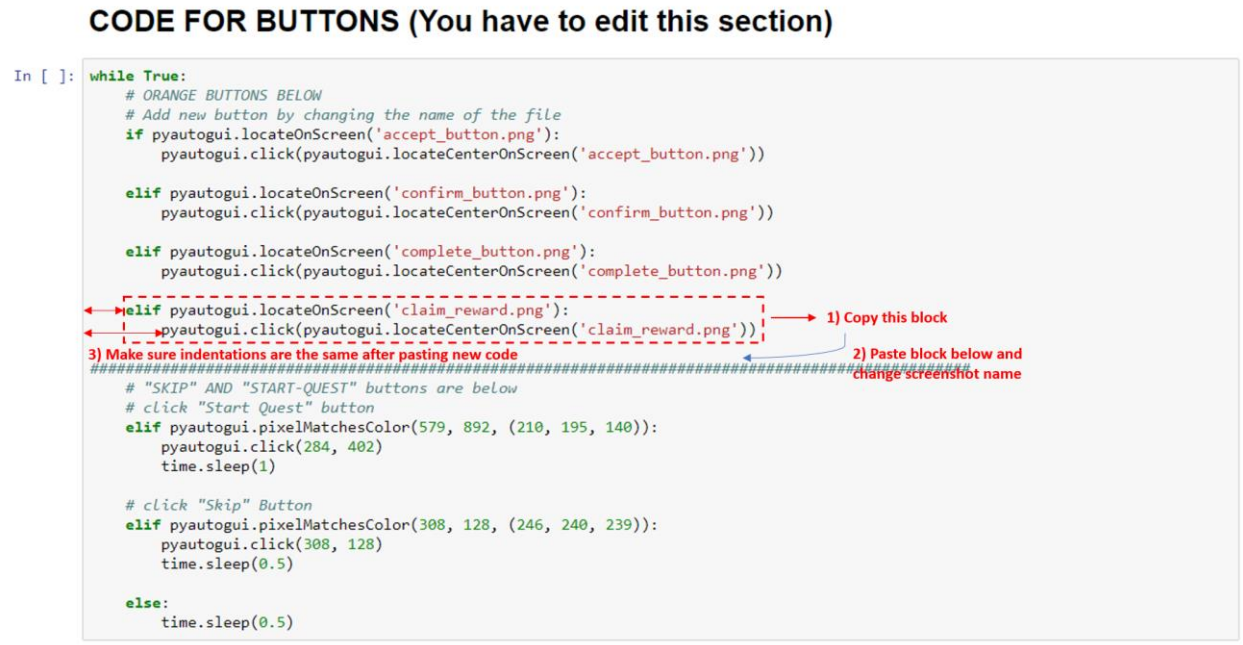
10) Repeat steps 4 – 9 for all orange buttons you want to add. For a start, I recommend adding the following buttons into the "**CODE FOR BUTTONS**" section:

- "Complete" button
- "Accept" button
- "Claim Reward" button
- "Confirm" button

11) Make sure that you have tested all your screenshots individually before adding them to the "**CODE FOR BUTTONS**" section. Also, make sure you have added all the screenshots listed in the "**CODE FOR BUTTONS**" section to your desktop (as per the diagram below). If there is any missing, an error message will pop up.

Make sure screenshots of all these buttons are on the desktop, if not there will be an error

12) Adding new buttons: Should you want to add new buttons, copy and paste the block of code as demonstrated in the below diagram. Change the screenshot name to the screenshot filename, and the program will be able to click the new button when it appears on the screen.



How to add new buttons to the program

7. CLICKING "SKIP" AND "START QUEST" BUTTONS

To Do before adding any button:

a. **Make sure your bluestacks window is in full screen**

b. **Make sure you have configured your bluestacks resolution (See Section 3)**

c. **Make sure you're using the computer you plan to run the bot on**

"Skip" Button

As the "*Skip*" button is located on a translucent background, it is not possible to take a screenshot of the button. The screenshot will look different when your character moves. We thus have to look at the pixels of the "*Skip*" button, and ***make sure the exact pixels match to the exact colour***. This is why you have to make sure the bluestacks window is in full screen, so that the pixels will be in the same position every time.

The diagram below better illustrates how the "*Skip*" button is clicked.

How we click the skip button: By making sure that specific pixels are white in color

Finding the X and Y Coordinates and pixel colour of the Skip button

The X-Y coordinates of a pixel are represented in code by 2 numbers e.g. (305, 450). Pixel colour is represented by 3 numbers e.g. (245, 255, 254).

Now that we know the logic of how the "Skip" button is clicked, let's implement this in code. I will list out below the exact steps you have to follow to implement this.

Steps to implement the "*Skip*" button

1) Open *Bluestacks* and launch Maplestory M, keeping in mind to full screen the *Bluestacks* window.

2) Open up *Jupyter Notebook* and the code (instructions above here)

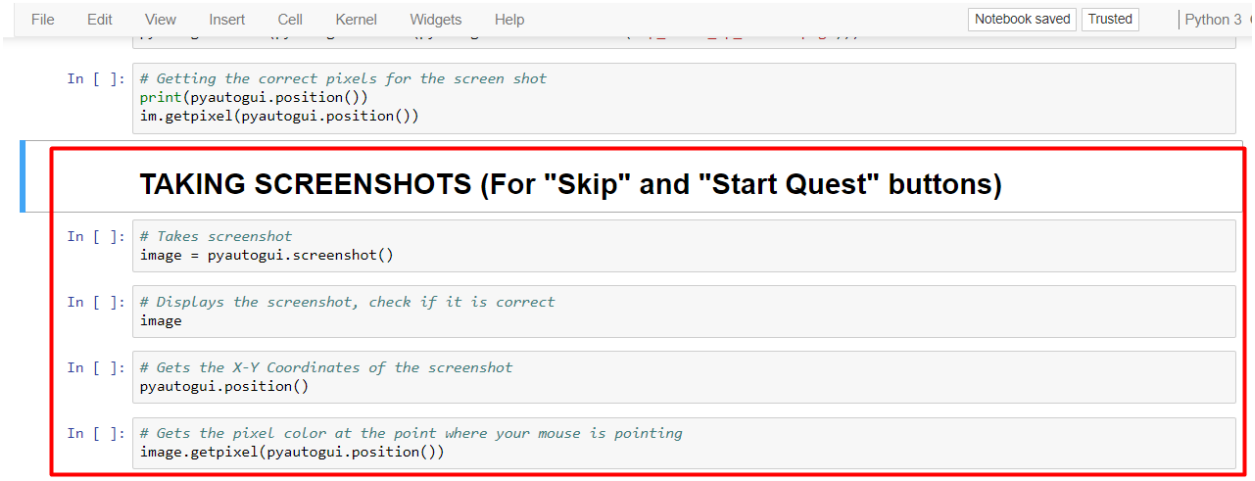
3) Execute the first cell (by pressing "***Cltr***" + "***Enter***"). Doing so imports the *pyautogui* and *time* modules into your Jupyter Notebook. With this we can run all other cells in the notebook.

```
In [ ]: import pyautogui
        import time
```

Execute this cell

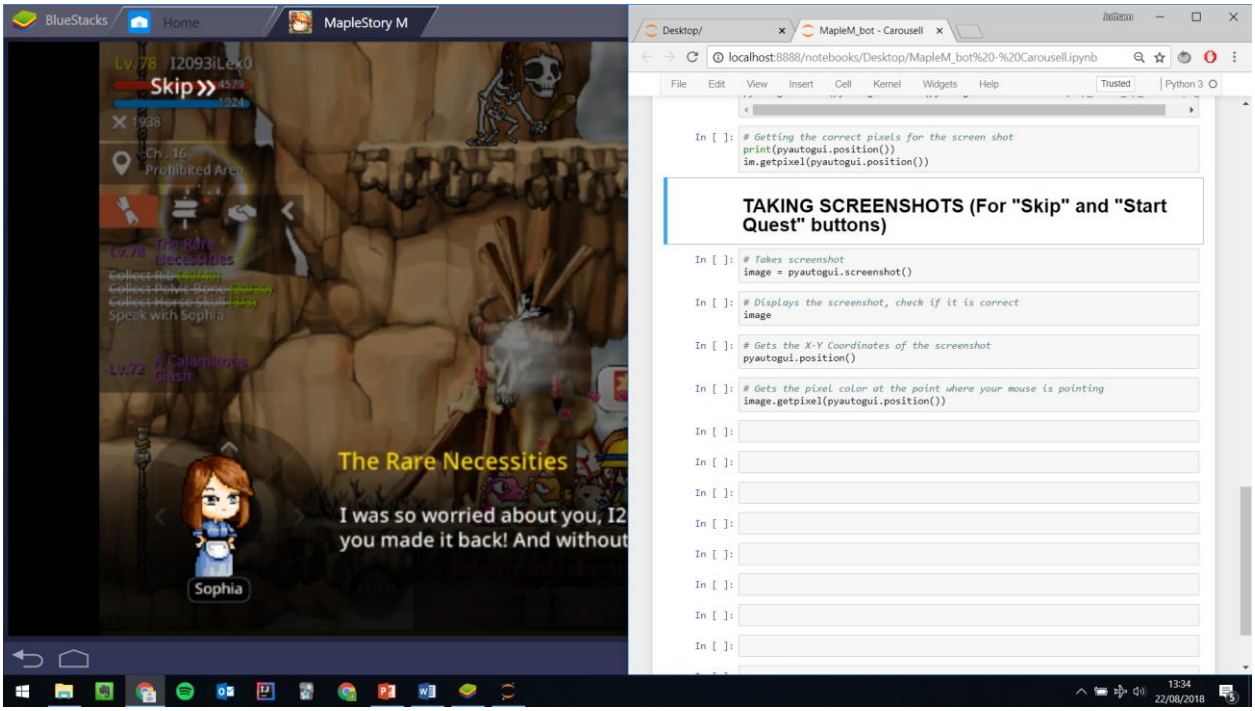
4) In Jupyter Notebook, scroll down to the "**TAKING SCREENSHOTS (For "Skip" and "Start Quest" buttons)**" section.

- This is the section where we can get all the information to click the "Skip" button
- The code in this section can give us 2 types of information we need:
    - **X-Y coordinates** of the pixel of the skip button
    - **Colour of the pixel** at that particular X-Y coordinate



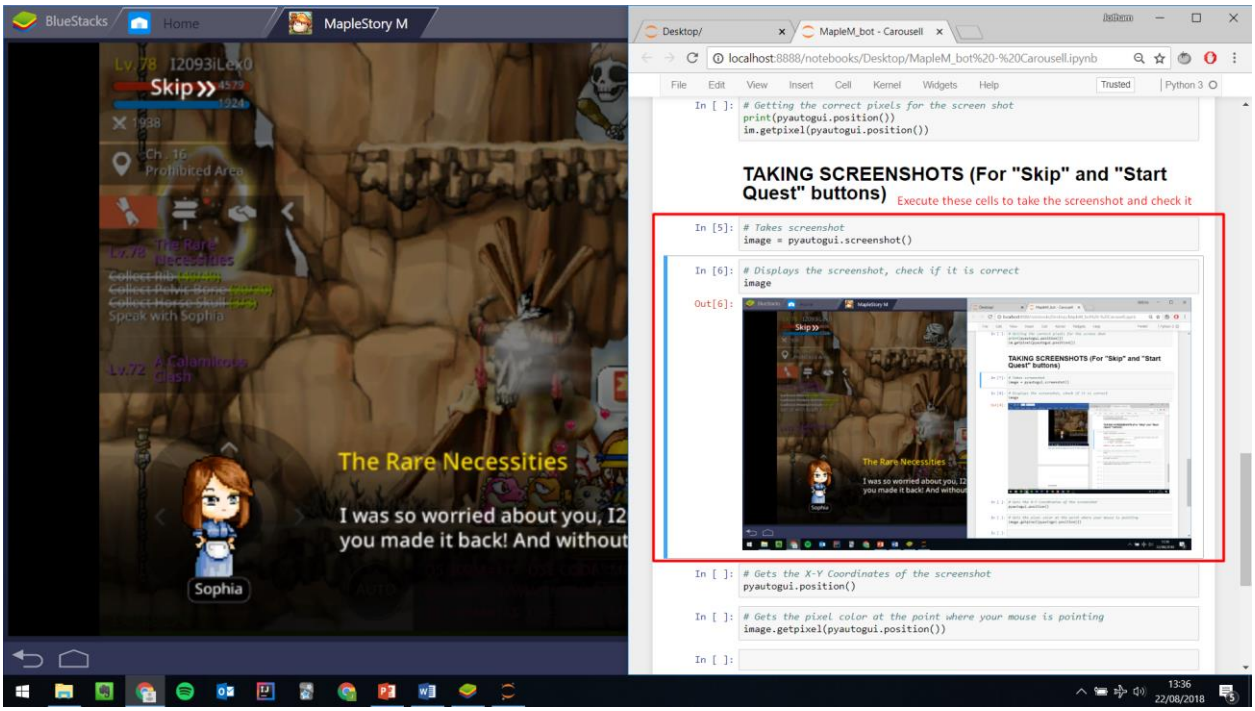
Scroll down to this section

5) Rearrange your Jupyter notebook, and make sure that the "Skip" button is visible. An example is as per the image below. This will enable you to take the right screenshot. Note: Your Bluestacks window always has to be full screen.



How you should arrange your screen to take a proper screenshot

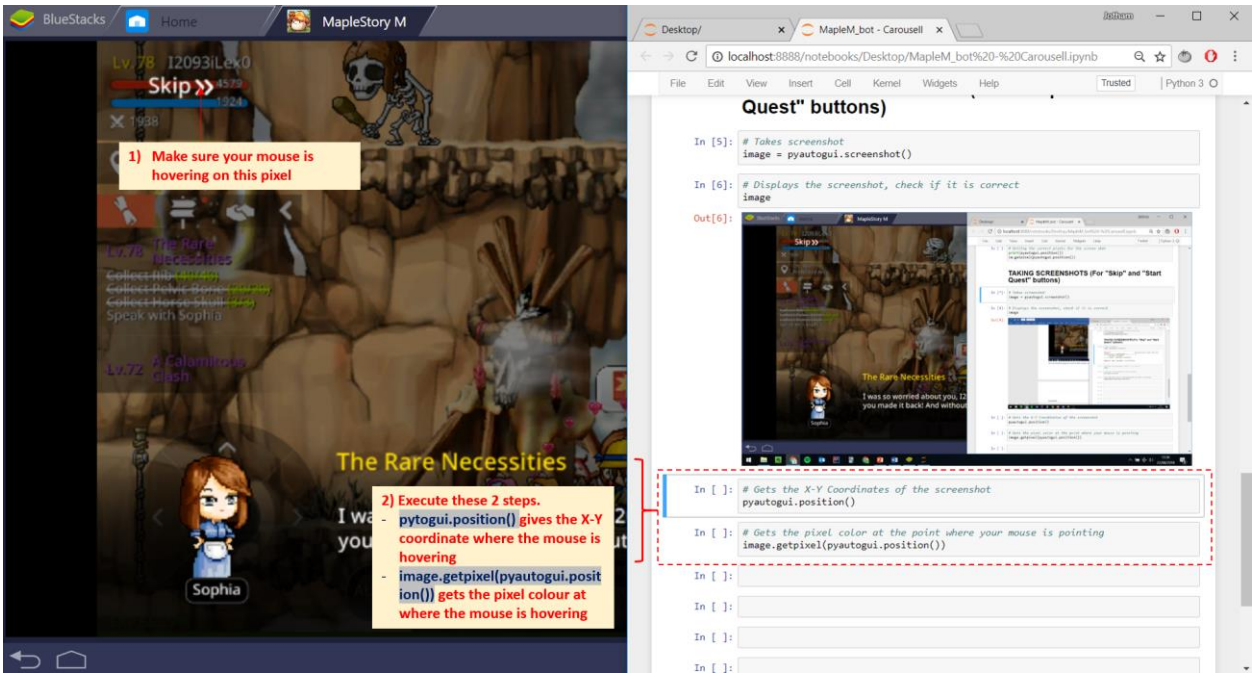
6) After you do that, execute the cells in the image below. This takes a screenshot and displays the screenshot. Make sure the screenshot is displayed correctly in Jupyter Notebook.



Execute these 2 highlighted cells above to take the screenshot

7) After you have taken the screenshot, hover your mouse on the pixel on the "Skip" button in *Bluestacks*. This is illustrated in Step 1 of the below diagram.

Next, execute the 2 cells in Step 2 as per the diagram below. This will get you the X-Y coordinates and Colour of the "Skip" button.



Follow the steps in the diagram above to get the X-Y coordinates and Colour

8) You should see the X-Y coordinates and the colour after you execute the cells.

```
In [15]: # Gets the X-Y Coordinates of the screenshot
         pyautogui.position()
Out[15]: (308, 128)   You should see X-Y coordinates here

In [18]: # Gets the pixel color at the point where your mouse is pointing
         image.getpixel(pyautogui.position())
Out[18]: (246, 240, 239)   You should see the colour here
```

X-Y coordinates and colour output after execution

9) In Jupyter Notebook, go to the "**CODE FOR BUTTONS (You have to edit this section)**" section. This is the cell which code you are to edit. Find the "*Skip*" button, and input the X-Y coordinates and colour output. If you are confused, refer to the diagram below.

CODE FOR BUTTONS (You have to edit this section)

```
In [ ]: while True:
            # ORANGE BUTTONS BELOW
            ########################################################
            # ADD ACCEPT BUTTON HERE BY CHANGING THE NAME OF
            elif pyautogui.locateOnScreen('accept_button.png'):
                pyautogui.click(pyautogui.locateCenterOnScreen('accept_button.png'))

            elif pyautogui.locateOnScreen('confirm_button.png'):
                pyautogui.click(pyautogui.locateCenterOnScreen('confirm_button.png'))

            elif pyautogui.locateOnScreen('complete_button3.png'):
                pyautogui.click(pyautogui.locateCenterOnScreen('complete_button3.png'))

            elif pyautogui.locateOnScreen('claim_reward.png'):
                pyautogui.click(pyautogui.locateCenterOnScreen('claim_reward.png'))

            elif pyautogui.locateOnScreen('available_button.png'):
                pyautogui.click(pyautogui.locateCenterOnScreen('available_button.png'))

            elif pyautogui.locateOnScreen('revive_button.png'):
                pyautogui.click(pyautogui.locateCenterOnScreen('revive_button.png'))

        ##############################################################################################
            # "SKIP" AND "START-QUEST" buttons are below
            # click "Start Quest" button
            elif pyautogui.pixelMatchesColor(544, 901, (192, 178, 127)) & pyautogui.pixelMatchesColor(579, 892, (229, 213, 152)):
                pyautogui.click(275,388)
                time.sleep(1)

            # click "Skip" Button
            elif pyautogui.pixelMatchesColor(308, 128, (246, 240, 239)):
                pyautogui.click(308, 128)
                time.sleep(0.5)

            else:
                count += 1
                time.sleep(0.5)
```

"Skip" button section: to edit this

Replace with your X-Y coordinates

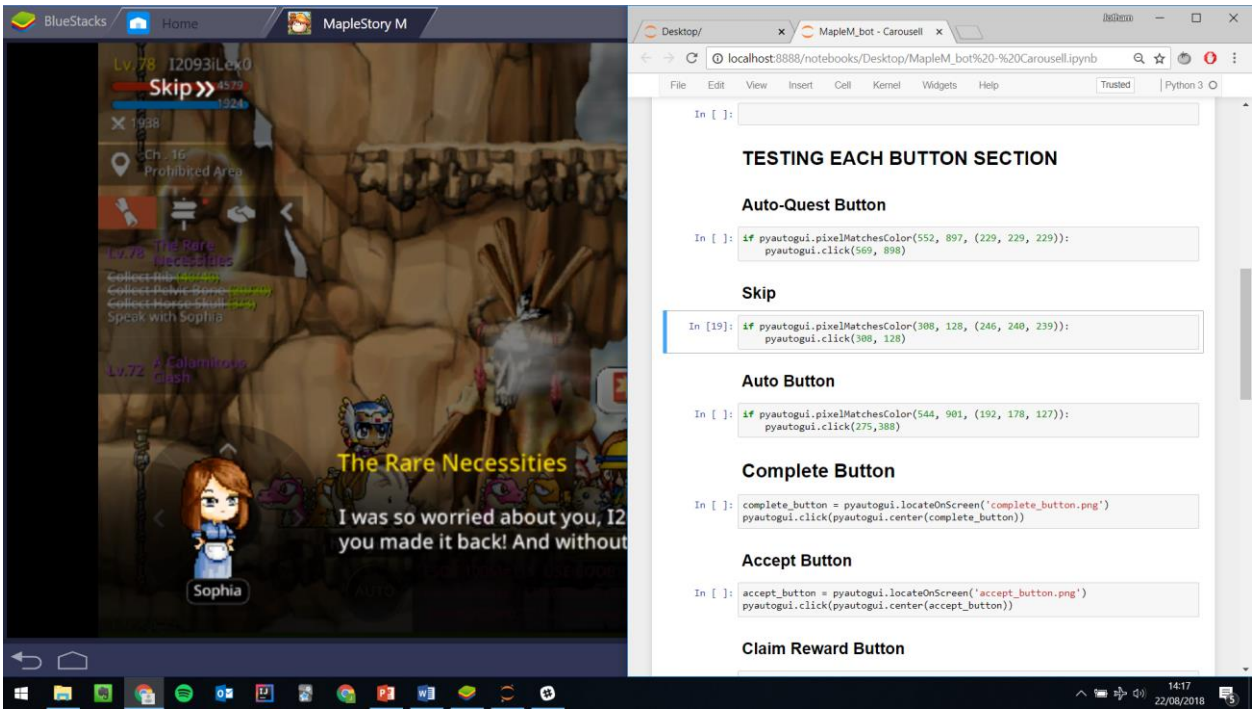
Replace with your colour

How to input the X-Y coordinates and Colour output into the code

10) Before we move on, let's test this button to make sure it works. Go to the "**TESTING EACH BUTTON SECTION**" and find the "**Skip**" button. Again, input your X-Y coordinates and Colour.

Input X-Y coordinates and colour to test the "Skip" button

11) Arrange the screen as per the screenshot below.



12) Execute the "Skip" button cell. It should click the button in your Maplestory M window.

If it doesn't click, try finding a new pixel and colour on the "Skip" button, and input the new coordinates as per the above steps.

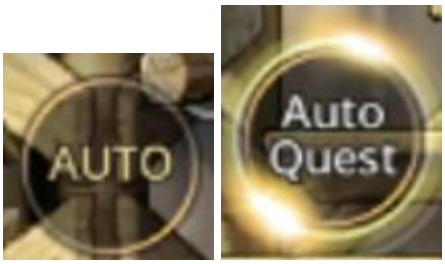
Make sure to update the "**CODE FOR BUTTONS (You have to edit this section)**" every time you make any changes to the code in the "**TEST**" section.

13) Congratulations! We are done for the "Skip" button.

“Start Quest” Button

This is the most complicated button to configure. We have to click the “Start Quest” button in order for a quest to start. However, as the region is translucent, it changes every time your character moves. Taking a screenshot is thus not possible.

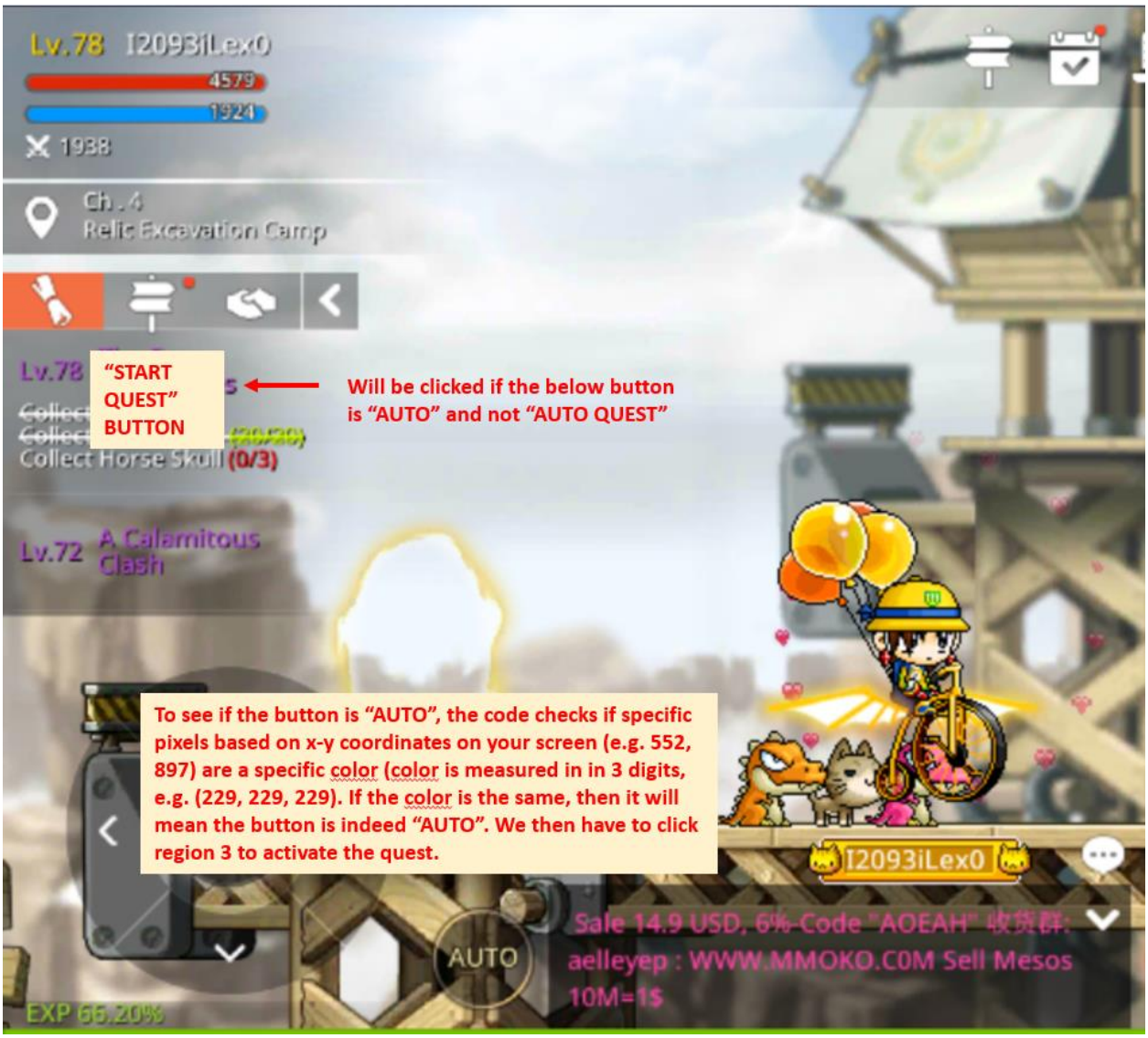
We will thus have to check if our character is on a quest or not. How do we do this? By looking at differences in this button:



We click “Start Quest” button if the above button is “AUTO”. If the above button is “Auto Quest”, we do not take any action because we already are on a quest.

See the screenshot below for a better explanation on how this works.

The next problem is that the "AUTO" button is translucent as well. How do we click it? Simple, by looking at pixels, just like what we did with the "Skip" button. Look at the below screenshot to see how the code checks if button is "***AUTO***" or "***AUTO QUEST***".



Now that we know the logic on how the "Start Quest" button is clicked, let's implement this in code. I will list out below the exact steps you have to follow to implement this. Many of the steps in section will be very similar to that of the "Skip" button above, with a few variations.

Steps to implement the "*Start Quest*" button

1) Open *Bluestacks* and launch Maplestory M, keeping in mind to full screen the *Bluestacks* window.

2) Open up *Jupyter Notebook* and the code (instructions above here)

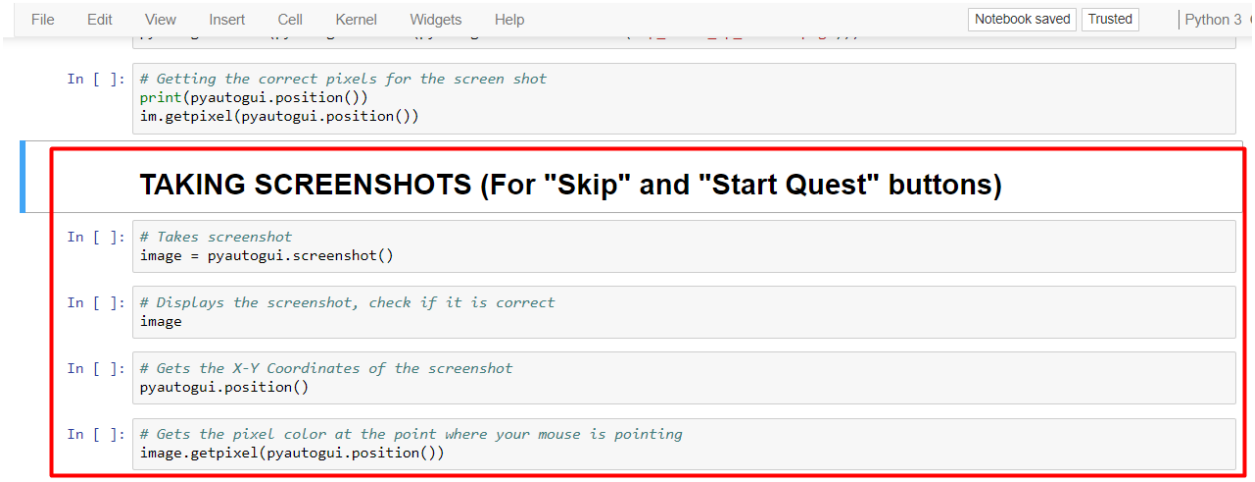
3) Execute the first cell (by pressing "***Cltr***" + "***Enter***"). Doing so imports the *pyautogui* and *time* modules into your Jupyter Notebook. With this we can run all other cells in the notebook.

```
In [ ]: import pyautogui
        import time
```

Execute this cell

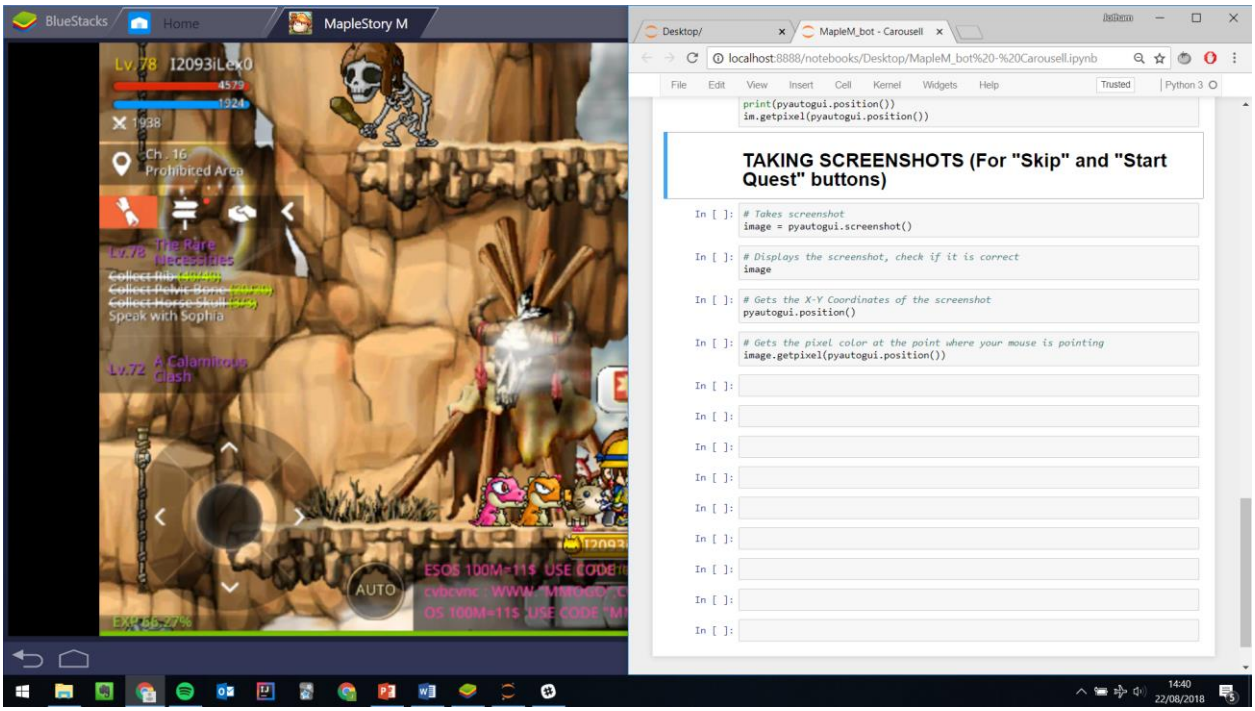
4) In Jupyter Notebook, scroll down to the "**TAKING SCREENSHOTS (For "Skip" and "Start Quest" buttons)**" section.

- This is the section where we can get all the information to click the "*Start Quest*" button
- The code in this section can give us 2 types of information we need:
    - **X-Y coordinates** of the pixel of the "*Start Quest*" button
    - **Colour of the pixel** at that X-Y coordinate



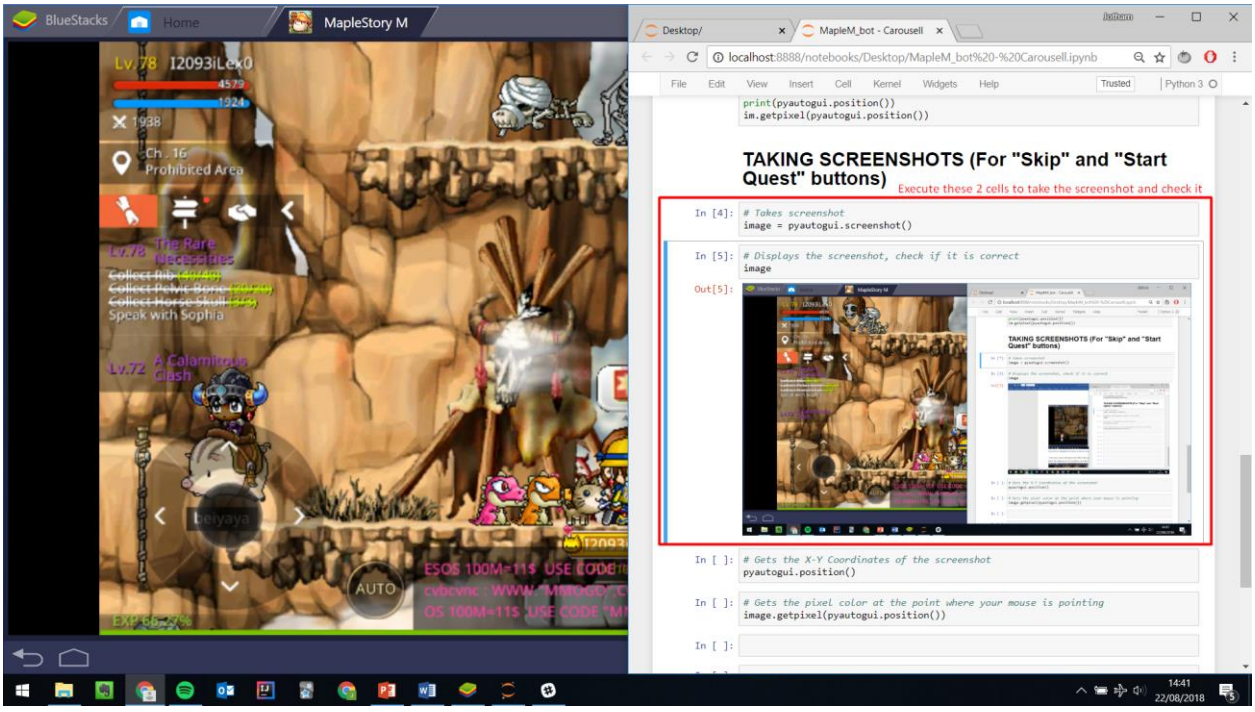
Scroll down to this section

5) Rearrange your Jupyter notebook, and **make sure that the "AUTO" button is visible**. An example is as per the image below. This will enable you to take the right screenshot. Note: Your Bluestacks window always has to be full screen.



How you should arrange your screen to take a proper screenshot

6) After you do that, execute the two cells in the image below. This takes a screenshot and displays the screenshot. Make sure the screenshot is displayed correctly in Jupyter Notebook.

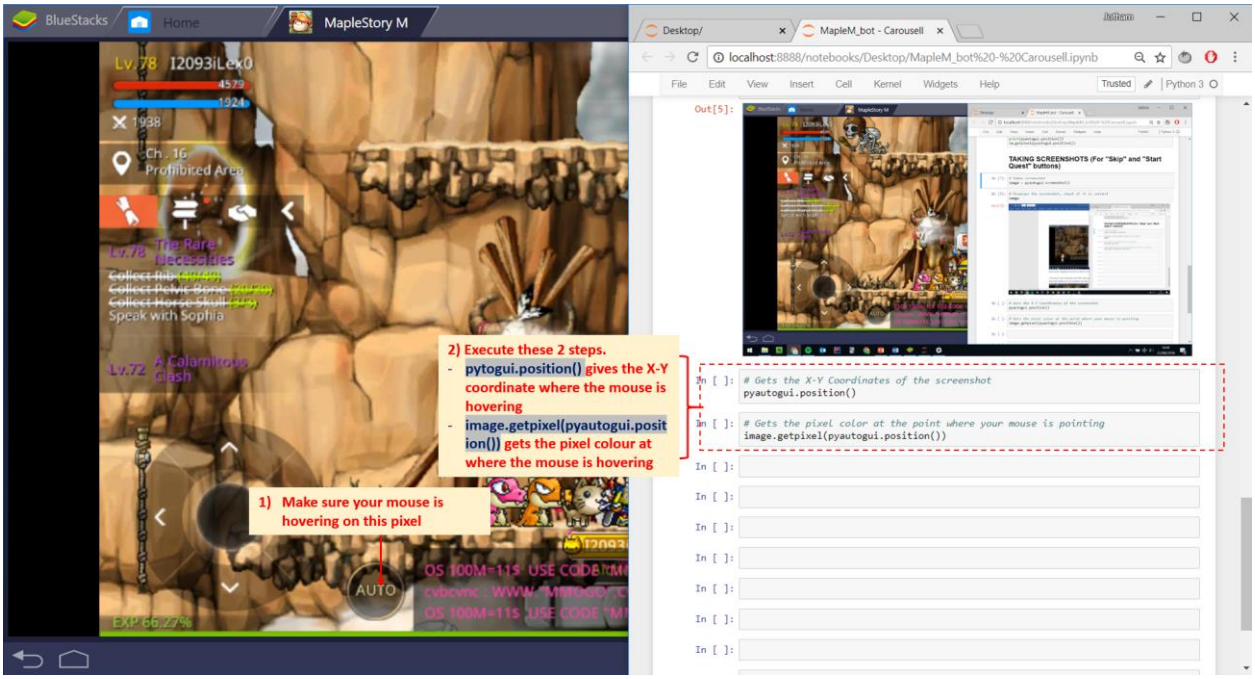


Execute these 2 highlighted cells above to take the screenshot

7)

After you have taken the screenshot, hover your mouse on the pixel on the “AUTO” button in *Bluestacks*. This is illustrated in Step 1 of the below diagram.

Next, execute the 2 cells in Step 2 as per the diagram below. This will get you the X-Y coordinates and Colour of the “AUTO” button.



Follow the steps in the diagram above to get the X-Y coordinates and Colour

8) You should see the X-Y coordinates and the colour of the “AUTO” button after you execute the cells.

```
In [10]: # Gets the X-Y Coordinates of the screenshot
         pyautogui.position()
Out[10]: (579, 893)  You should see X-Y coordinates here

In [11]: # Gets the pixel color at the point where your mouse is pointing
         image.getpixel(pyautogui.position())
Out[11]: (181, 169, 120)  You should see colour here
```

X-Y coordinates and colour of “AUTO” button after execution

9) In Jupyter Notebook, go to the “**CODE FOR BUTTONS (You have to edit this section)**” section. This is the cell whose code you are to edit. Find the “*Start Quest*” button, and input the X-Y coordinates and colour output. If you are confused, refer to the diagram below. Do note that this step is a little different from the “Skip” button.

CODE FOR BUTTONS (You have to edit this section)

In [ ]:
```
while True:
    # ORANGE BUTTONS BELOW
    ######################################################
    # ADD ACCEPT BUTTON HERE BY CHANGING THE NAME OF
    elif pyautogui.locateOnScreen('accept_button.png'):
        pyautogui.click(pyautogui.locateCenterOnScreen('accept_button.png'))

    elif pyautogui.locateOnScreen('confirm_button.png'):
        pyautogui.click(pyautogui.locateCenterOnScreen('confirm_button.png'))

    elif pyautogui.locateOnScreen('complete_button3.png'):
        pyautogui.click(pyautogui.locateCenterOnScreen('complete_button3.png'))

    elif pyautogui.locateOnScreen('claim_reward.png'):
        pyautogui.click(pyautogui.locateCenterOnScreen('claim_reward.png'))

    elif pyautogui.locateOnScreen('available_button.png'):
        pyautogui.click(pyautogui.locateCenterOnScreen('available_button.png'))

    elif pyautogui.locateOnScreen('revive_button.png'):
        pyautogui.click(pyautogui.locateCenterOnScreen('revive_button.png'))

##############################################################################################
    # "SKIP" AND "START-QUEST" buttons are below
    # click "Start Quest" button
    elif pyautogui.pixelMatchesColor(544, 901, (192, 178, 127)):
        pyautogui.click(275,388)
        time.sleep(1)
    # click "Skip" Button
    elif pyautogui.pixelMatchesColor(308, 128, (246, 240, 239)):
        pyautogui.click(308, 128)
        time.sleep(0.5)

    else:
        count += 1
        time.sleep(0.5)
```

"Start Quest" button section: to edit this

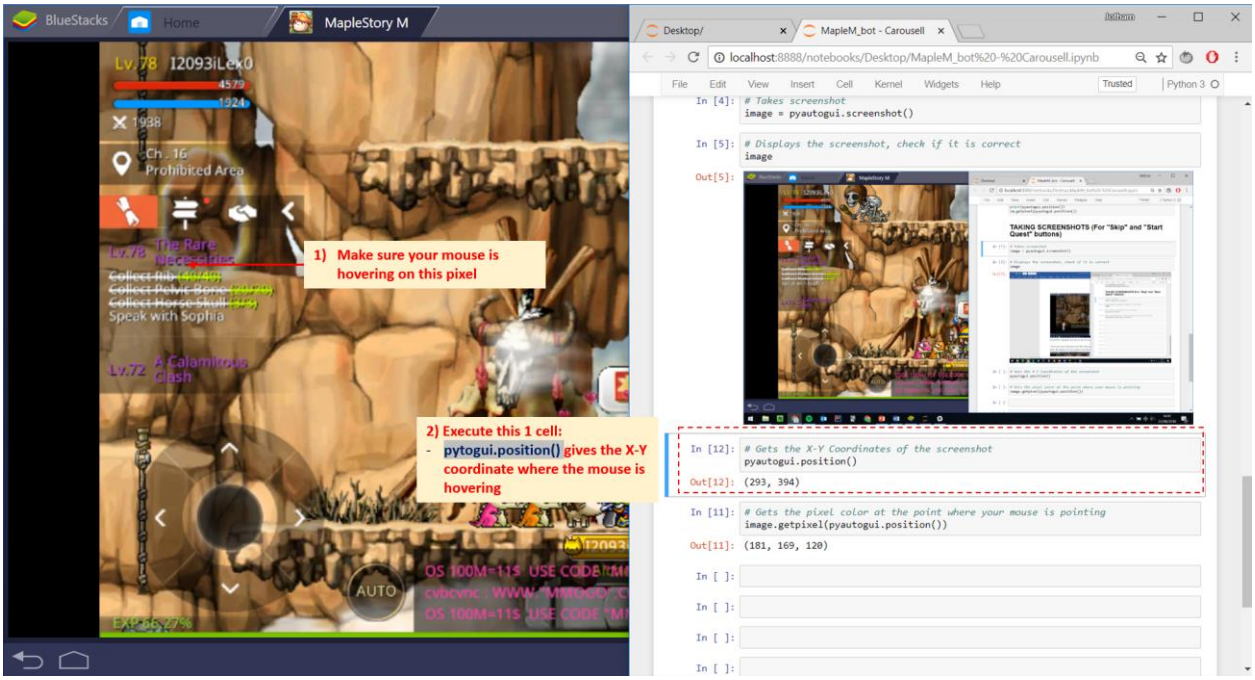
Do not replace this

Replace with your X-Y coordinates

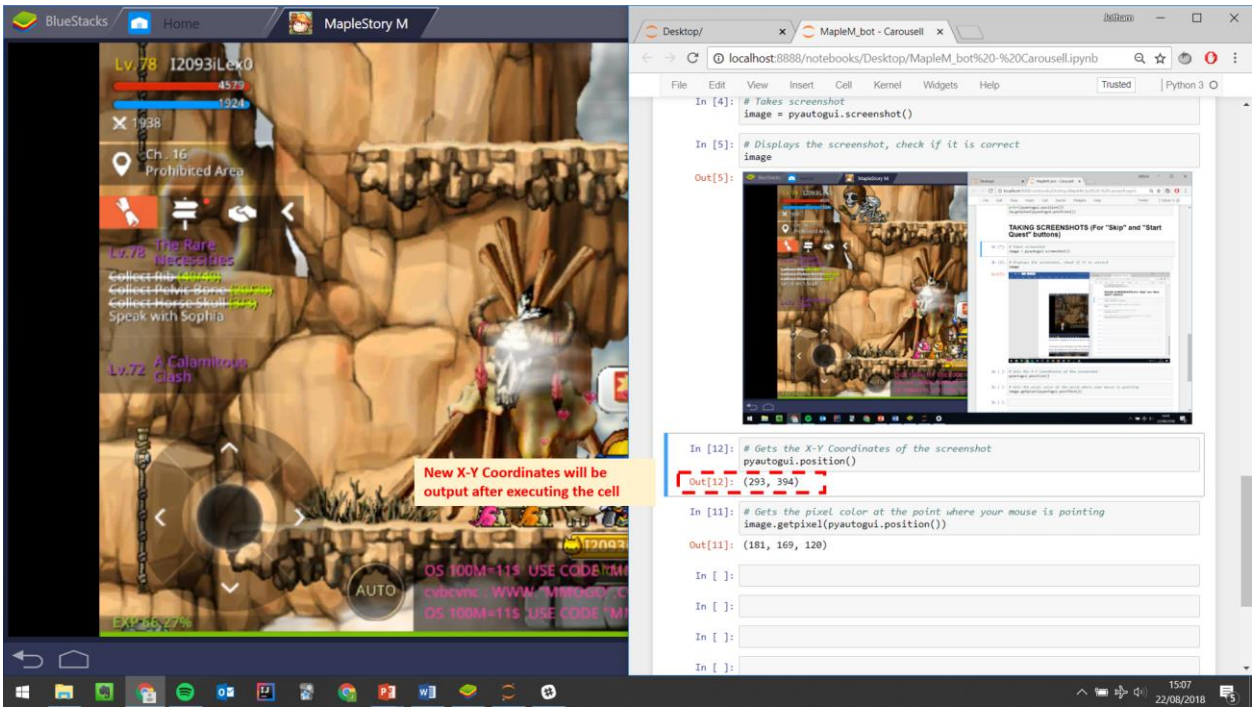
Replace with your colour

How to input the X-Y coordinates and Colour output into the code

10) Next, we have to find the X-Y coordinates to click for the quest to start. Refer to the diagram below and execute the instructions in order.



11) You should get new X-Y coordinates as an output.



12) Go to the "**CODE FOR BUTTONS (You have to edit this section)**" section again. Find the "*Start Quest*" button, and **input the new X-Y coordinates as per the diagram below.**

CODE FOR BUTTONS (You have to edit this section)

In [ ]:
```
while True:
    # ORANGE BUTTONS BELOW
    ##########################################################
    # ADD ACCEPT BUTTON HERE BY CHANGING THE NAME OF
    elif pyautogui.locateOnScreen('accept_button.png'):
        pyautogui.click(pyautogui.locateCenterOnScreen('accept_button.png'))

    elif pyautogui.locateOnScreen('confirm_button.png'):
        pyautogui.click(pyautogui.locateCenterOnScreen('confirm_button.png'))

    elif pyautogui.locateOnScreen('complete_button3.png'):
        pyautogui.click(pyautogui.locateCenterOnScreen('complete_button3.png'))

    elif pyautogui.locateOnScreen('claim_reward.png'):
        pyautogui.click(pyautogui.locateCenterOnScreen('claim_reward.png'))

    elif pyautogui.locateOnScreen('available_button.png'):
        pyautogui.click(pyautogui.locateCenterOnScreen('available_button.png'))

    elif pyautogui.locateOnScreen('revive_button.png'):
        pyautogui.click(pyautogui.locateCenterOnScreen('revive_button.png'))

##########################################################################################################
    # "SKIP" AND "START-QUEST" buttons are below
    # click "Start Quest" button
    elif pyautogui.pixelMatchesColor(544, 901, (192, 178, 127)):
        pyautogui.click(275,388)
        time.sleep(1)

    # click "Skip" Button
    elif pyautogui.pixelMatchesColor(308, 128, (246, 240, 239)):
        pyautogui.click(308, 128)
        time.sleep(0.5)

    else:
        count += 1
        time.sleep(0.5)
```

"Start Quest" button section: to edit this

Do not change these values as these are for the "AUTO" button

Input new X-Y coordinates here.

10) Before we move on, let's test this button to make sure it works. Go to the "**TESTING EACH BUTTON SECTION**" and find the "**Start Quest**" button. Copy and paste the values you have just inputted from the "Start-Quest" Button above, and input them to the "TESTING EACH BUTTON SECTION". This is illustrated below.

CODE FOR BUTTONS (You have to edit this section)

```
In [ ]: while True:
            # ORANGE BUTTONS BELOW
            ######################################################
            # ADD ACCEPT BUTTON HERE BY CHANGING THE NAME OF
            elif pyautogui.locateOnScreen('accept_button.png'):
                pyautogui.click(pyautogui.locateCenterOnScreen('accept_button.png'))

            elif pyautogui.locateOnScreen('confirm_button.png'):
                pyautogui.click(pyautogui.locateCenterOnScreen('confirm_button.png'))

            elif pyautogui.locateOnScreen('complete_button3.png'):
                pyautogui.click(pyautogui.locateCenterOnScreen('complete_button3.png'))

            elif pyautogui.locateOnScreen('claim_reward.png'):
                pyautogui.click(pyautogui.locateCenterOnScreen('claim_reward.png'))

            elif pyautogui.locateOnScreen('available_button.png'):
                pyautogui.click(pyautogui.locateCenterOnScreen('available_button.png'))

            elif pyautogui.locateOnScreen('revive_button.png'):
                pyautogui.click(pyautogui.locateCenterOnScreen('revive_button.png'))

        ##########################################################################################
            # "SKIP" AND "START-QUEST" buttons are below
            # click "Start Quest" button
            elif pyautogui.pixelMatchesColor(544, 901, (192, 178, 127)):
                pyautogui.click(275,388)
                time.sleep(1)

            # click "Skip" Button
            elif pyautogui.pixelMatchesColor(308, 128, (246, 240, 239)):
                pyautogui.click(308, 128)
                time.sleep(0.5)

            else:
                count += 1
                time.sleep(0.5)
```

Copy these values

Values to copy

TESTING EACH BUTTON SECTION

Auto-Quest Button

```
In [ ]: if pyautogui.pixelMatchesColor(552, 897, (229, 229, 229)):
            pyautogui.click(569, 898)
```

Skip

```
In [ ]: if pyautogui.pixelMatchesColor(308, 128, (246, 240, 239)):
            pyautogui.click(308, 128)
```

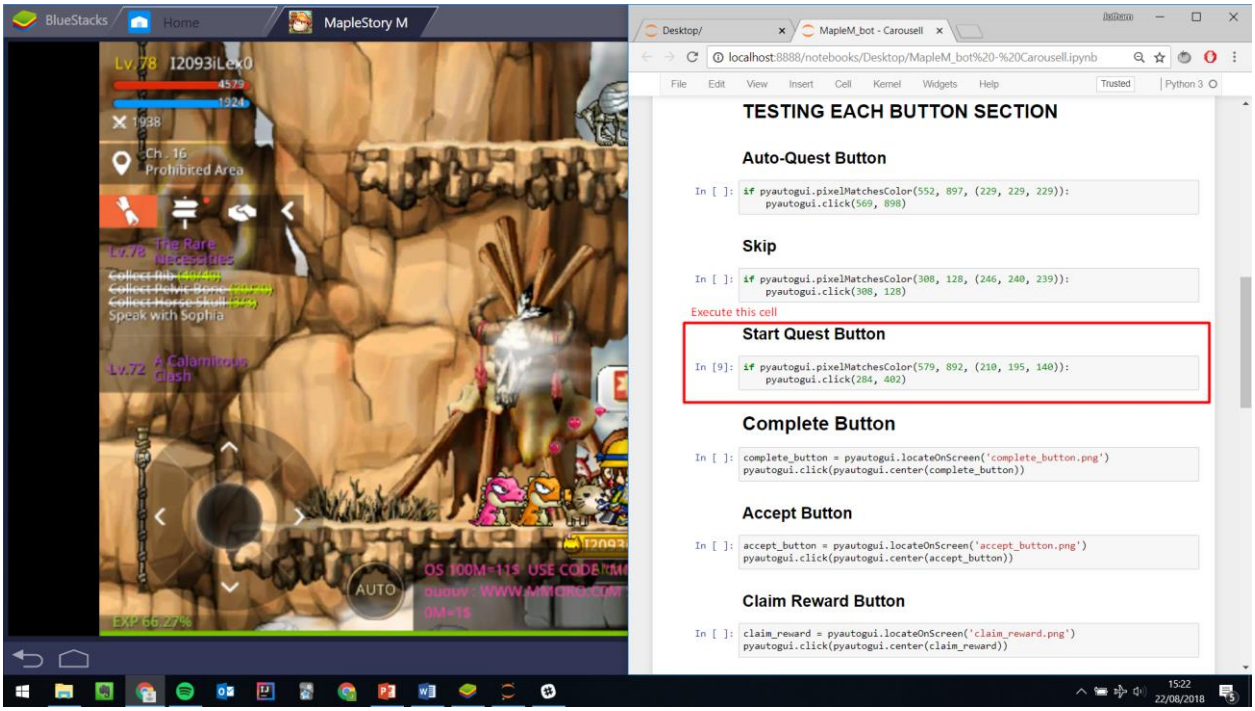
Start Quest Button

```
In [9]: if pyautogui.pixelMatchesColor(579, 892, (210, 195, 140)):
            pyautogui.click(284, 402)
```

Paste Values here

Cell to paste values

11) Next, rearrange your Jupyter notebook, and **make sure that the "AUTO" button and the "Quest" button is visible**. This is illustrated in the below diagram. Note: Your Bluestacks window always has to be full screen.



12) Execute the “Start Quest Button” cell.

It should click the quest button in your Maplestory M window. If it doesn’t click, repeat steps 4 – 11. This will enable you to find a new pixel and colour on the “AUTO” button, and input the new coordinates as per the above steps.

Make sure to update the “**CODE FOR BUTTONS (You have to edit this section)**” every time you make any changes.

13) Congratulations! We are done for the “Start Quest” button.

8) RUNNING THE PROGRAM

Congratulations!!! You have made sure all the buttons were working individually and added these buttons to the "**CODE FOR BUTTONS**" section.

To run the Auto Quest Bot, you just have to execute the first 3 cells (as illustrated in the below diagram). After executing these cells, alt-tab to your Maplestory M game and watch magic happen.

In [ ]:
```
import pyautogui
import time
```

In [ ]:
```
# Pauses for 0.3 seconds after a button is click
pyautogui.PAUSE = 0.3
```

0.3 Second delay whenever a button is clicked

CODE FOR BUTTONS (You have to edit this section)

In [ ]:
```
while True:
    # ORANGE BUTTONS BELOW
    # Add new button by changing the name of the file
    if pyautogui.locateOnScreen('accept_button.png'):
        pyautogui.click(pyautogui.locateCenterOnScreen('accept_button.png'))

    elif pyautogui.locateOnScreen('confirm_button.png'):
        pyautogui.click(pyautogui.locateCenterOnScreen('confirm_button.png'))

    elif pyautogui.locateOnScreen('complete_button.png'):
        pyautogui.click(pyautogui.locateCenterOnScreen('complete_button.png'))

    elif pyautogui.locateOnScreen('claim_reward.png'):
        pyautogui.click(pyautogui.locateCenterOnScreen('claim_reward.png'))

##########################################################################################
    # "SKIP" AND "START-QUEST" buttons are below
    # click "Start Quest" button
    elif pyautogui.pixelMatchesColor(579, 892, (210, 195, 140)):
        pyautogui.click(284, 402)
        time.sleep(1)

    # click "Skip" Button
    elif pyautogui.pixelMatchesColor(308, 128, (246, 240, 239)):
        pyautogui.click(308, 128)
        time.sleep(0.5)

    else:
        time.sleep(0.5)
```

Loops through to find if a button is on the screen. The loop will keep running unless you stop it

Execute these 3 cells to run the program

Error Messages

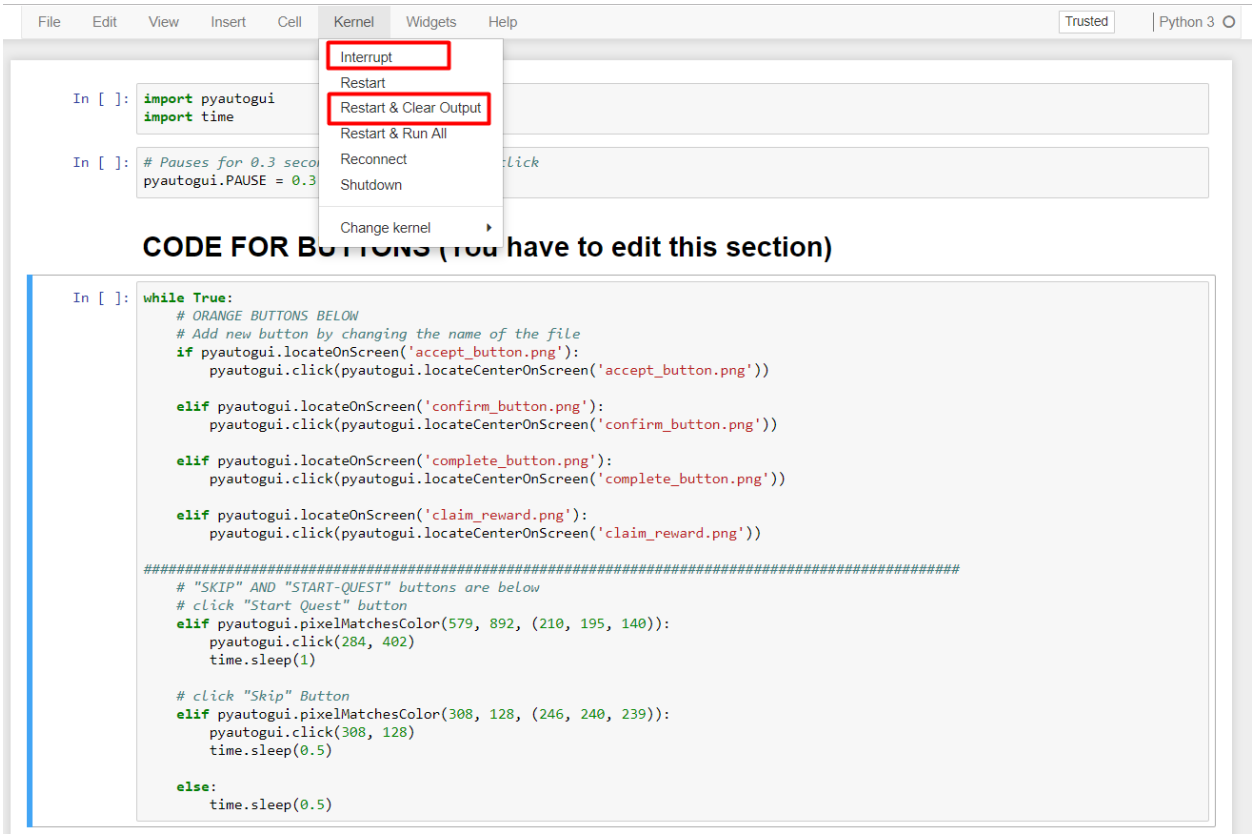
If you get an error message, most probably you have not added all buttons under the "CODE FOR BUTTONS" section. (i.e. some of the screenshots do not exist in the same folder as your MapleBot file)

If your bot gets stuck at some point in the game, the most probably reason is that you have yet to add that button. Follow the steps here to add the button, and your bot will not encounter the same problem again.

If your bot gets stuck at the "Skip" or "Start Quest" button, make sure your Bluestacks window is full screen. Chances are the pixels on your screen are not the same as when you first configured it.

Stopping the Program

Since the program is on a forever loop, it will not stop unless you tell it to. In Jupyter Notebook, click either "Interrupt" or "Restart & Clear Output" to stop the program.



Clicking either of these will stop the program

Thank you for reading through the entire tutorial, I know it was a long drawn process. I hope it was a good experience if this was your first time implementing code, and I hope it will inspire you to pick up python. Also, this guide took a lot of effort to create, so please do not share it around.

If you have any recommendations on how to improve the tutorial, please reach up to me.

THE END