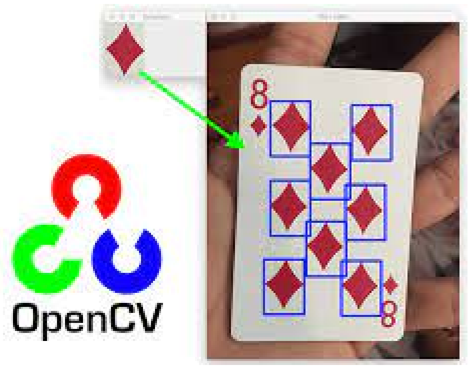
1. What Is Template Matching?

Template matching is the concept in digital image processing used to extract a location of a pattern which is present in the image which is close to a convolution.



2. Advantage:

```
- It is very useful to extract small features in an image.
```

3. Disadvantage:

```
- More the number of template matching performed slower will be the
process

- since convolution is a slow process performing on a large pixeled image
can reduce the performance of the code

- slows down your computer on consecutive running and hence not
consistent

- It depends on your processor speed greater it is the better
```

4. Template matching is not the best when comes to game automation doesnt mean it lies in the bottom end. It can be used to automate games with less patterns

Karate Chop Kick 🎮🎮

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

5. Before trying `Kick Chop` game we tried all these games and they were not compatible with Template matching (or We werent good enough 🙂)

- Karate chop kick
- Tower Boxer
- Parkour climb
- Cookie clicker

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

We created a Bot which at its best scored upto `2620` in the game of Karate chop kick.

1. Installing Libraries

In [ ]:

```
!pip install opencv-python
!pip install numpy
!pip install keyboard
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

```
!pip install pyautogui
!pip install pynput
!pip install matplotlib
```

```
Requirement already satisfied: opencv-python in c:\users\hp\anaconda3\lib\site-packages
(4.5.5.62)
Requirement already satisfied: numpy>=1.19.3 in c:\users\hp\anaconda3\lib\site-packages
(from opencv-python) (1.20.3)
Requirement already satisfied: numpy in c:\users\hp\anaconda3\lib\site-packages (1.20.3)
Requirement already satisfied: keyboard in c:\users\hp\anaconda3\lib\site-packages (0.1
3.5)
Requirement already satisfied: pyautogui in c:\users\hp\anaconda3\lib\site-packages (0.
9.53)
Requirement already satisfied: pygetwindow>=0.0.5 in c:\users\hp\anaconda3\lib\site-pack
ages (from pyautogui) (0.0.9)
Requirement already satisfied: mouseinfo in c:\users\hp\anaconda3\lib\site-packages (fro
m pyautogui) (0.1.3)
Requirement already satisfied: pymsgbox in c:\users\hp\anaconda3\lib\site-packages (from
pyautogui) (1.0.9)
Requirement already satisfied: pyscreeze>=0.1.21 in c:\users\hp\anaconda3\lib\site-packa
ges (from pyautogui) (0.1.28)
Requirement already satisfied: PyTweening>=1.0.1 in c:\users\hp\anaconda3\lib\site-packa
ges (from pyautogui) (1.0.4)
Requirement already satisfied: pyrect in c:\users\hp\anaconda3\lib\site-packages (from p
ygetwindow>=0.0.5->pyautogui) (0.1.4)
Requirement already satisfied: pyperclip in c:\users\hp\anaconda3\lib\site-packages (fro
m mouseinfo->pyautogui) (1.8.2)
Requirement already satisfied: pynput in c:\users\hp\anaconda3\lib\site-packages (1.7.6)
Requirement already satisfied: six in c:\users\hp\anaconda3\lib\site-packages (from pynp
ut) (1.16.0)
Requirement already satisfied: matplotlib in c:\users\hp\anaconda3\lib\site-packages (3.
4.3)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\users\hp\anaconda3\lib\site-packa
ges (from matplotlib) (1.3.1)
Requirement already satisfied: numpy>=1.16 in c:\users\hp\anaconda3\lib\site-packages (f
rom matplotlib) (1.20.3)
Requirement already satisfied: pyparsing>=2.2.1 in c:\users\hp\anaconda3\lib\site-packag
es (from matplotlib) (3.0.4)
Requirement already satisfied: cycler>=0.10 in c:\users\hp\anaconda3\lib\site-packages
(from matplotlib) (0.10.0)
Requirement already satisfied: pillow>=6.2.0 in c:\users\hp\anaconda3\lib\site-packages
(from matplotlib) (8.4.0)
Requirement already satisfied: python-dateutil>=2.7 in c:\users\hp\anaconda3\lib\site-pa
ckages (from matplotlib) (2.8.2)
Requirement already satisfied: six in c:\users\hp\anaconda3\lib\site-packages (from cycl
er>=0.10->matplotlib) (1.16.0)
```

2. Importing Libraries

In [ ]:

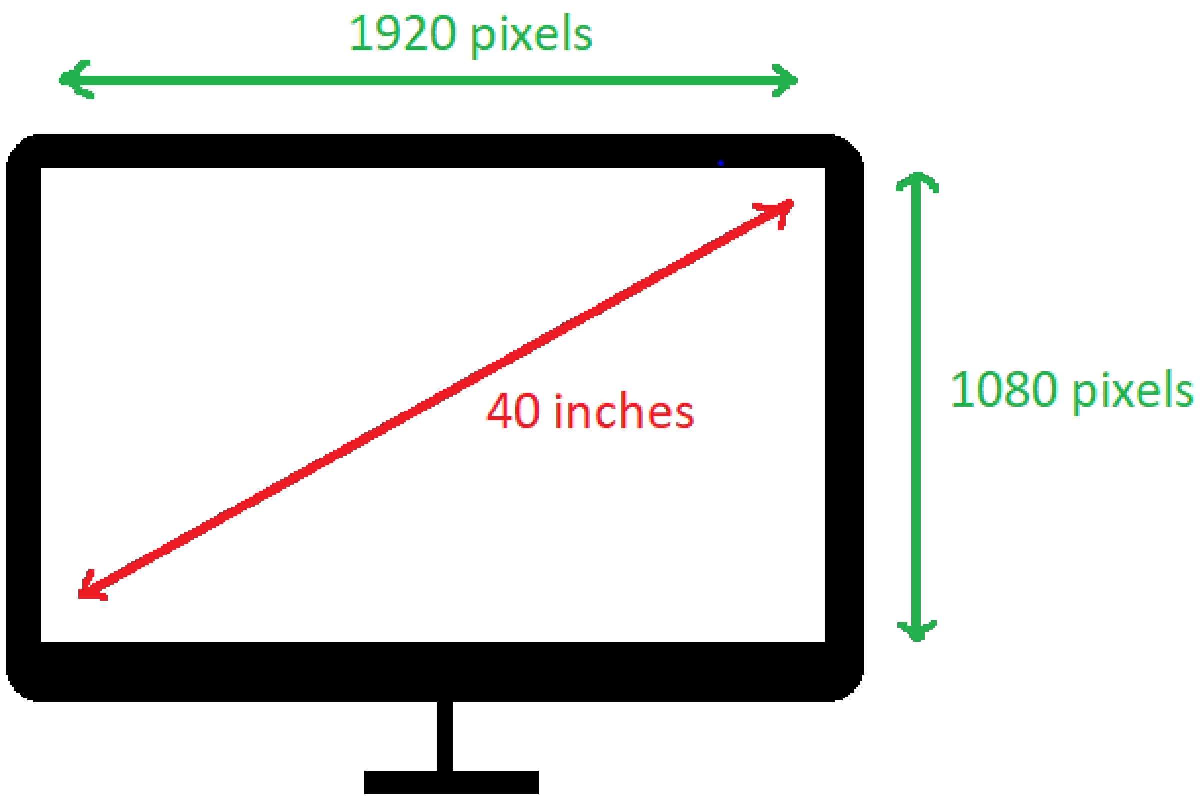
```
import keyboard # For keyboard input

import cv2 # For image processing

import numpy as np # For numpy mathematiacl array operations

from time import sleep # For time delay

import pyautogui # For mouse control
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

```python
from pynput.mouse import Button,Controller # For mouse control

import matplotlib.pyplot as plt # For plotting
```

2. Usage and Tips:

- This code is built to work on a `1920 x 1080` screen .
- Make sure to fullscreen the game before running.
- Keyboard is used to recieve the input `s` and `q`
- `s` is used to start the game & `q` is used to quit the game.
- `pyautogui.pause` is to prevent break interval for pyautogui library



3. Game ? - Karate Chop Kick 🎮🎮

4. GAME DESCRIPTION + INSTRUCTIONS

- Kick Ya Chop is a funny clicker game with ninjas!
- There is a ninja in the woods named Ryu and there is nothing more he likes doing than chopping trees!

- Click right and left while avoiding branches to get the best possible score on this online game.

- The faster you go the more time you will have.
- Monitor the bar at the top to make sure you are not running out of time.
- Go as fast as you can without making any errors.
- If you hit a branch, you lose. This is the kind of game you can play again and again though.
- Fight against yourself and try to beat your best score every time you play this online game.

5. Code

In [ ]:

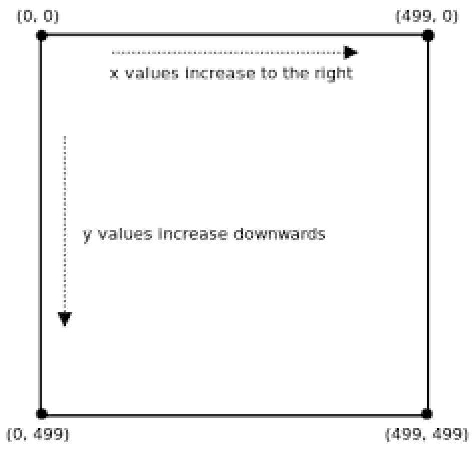
```
pyautogui.PAUSE=0 #pyautogui.PAUSE is to prevent break interval for pyautogui library

print('Press s to start playing:')
print('Press q to quit playing:')
# keyboard.wait('s')
```

```
Press s to start playing:
Press q to quit playing:
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

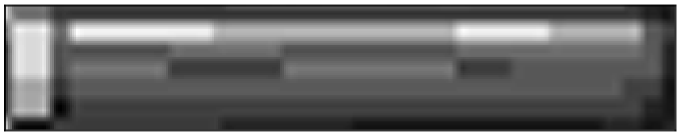
6. Getting screen coordinates

- Using Ms paint we figured out the left log and right log.
- It is better to use a `gray scale images` to get a higher probability of template matching.

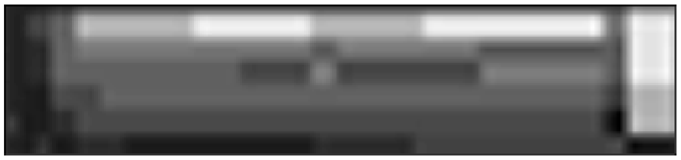


In [ ]:
```python
leftside=(810,500,200,200) # Left log coordinates - (x,y,width,height)

rightside=(1000,500,200,200) # Right log coordinates - (x,y,width,height)

#left wood block image
leftwood=cv2.imread('left.png',0)

#right wood block image
rightwood=cv2.imread('right.png',0)
```

In [ ]:
```python
plt.imshow(leftwood,cmap='gray') #plot left wood block image
plt.xticks([]), plt.yticks([])  # to hide tick values on X and Y axis
plt.show()
```



In [ ]:
```python
plt.imshow(rightwood,cmap='gray') #plot right wood block image
plt.xticks([]), plt.yticks([])  # to hide tick values on X and Y axis
plt.show()
```



7. Analysing Mouse Clicking Library

7.1. pyautogui / pynput / mss

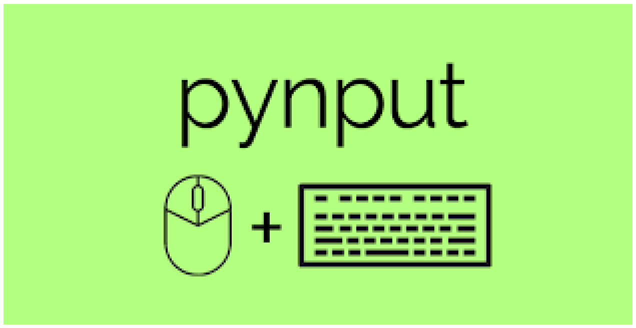
Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

1. `pyautogui` was the slowest among all gives a result very close to `2100` at max

2. **pynput**: is another mouse controller module which is slightly slower than `mouse` module but is efficient when comes to working with our platform

3. `mouse` module is faster by 0.03ms but it didnt support the platform which was being used

So `pynput` wins 🏆🏆



In [ ]:
```
mouse = Controller() # Mouse controller object (pynput.mouse module)

flag=1
```

- `flag` = 1 -> left log just above his head
- `flag` = 0 -> right log just above his head

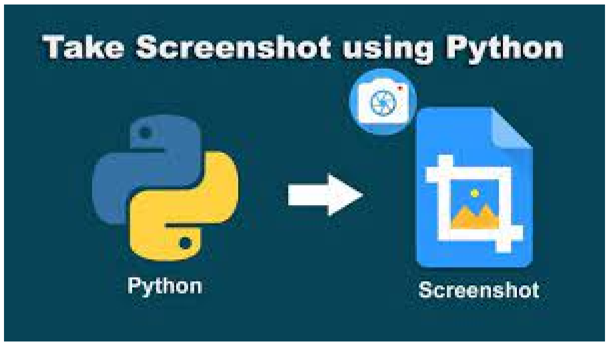
8. Screenshoting Libraries: 💻

4 options were available

- `pywin32` - Really fast , but not for Linux
- `mss` - Fast , but not for Linux
- `imagegrab` not in linux but good in mac and windows
  - (Alternatives)[https://stackoverflow.com/questions/43520757/imagegrab-alternative-in-linux]
- `pyautogui` screenshot : didnt want to use it but is the last resort

so we used `pyautogui`

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js



This is the class which describes the protaganist the template matching is done

In [ ]:
```
def matchobstacle(flag):

    ''' This function is to match the obstacle with the log '''

    if flag: # If flag is 1, then left log is selected

      scr=pyautogui.screenshot(region=leftside)

      wood=leftwood

    else : # If flag is 0, then right log is selected

      scr=pyautogui.screenshot(region=rightside)

      wood=rightwood

    scr=np.array(scr) # Convert the screenshot to numpy array for image processing

    scr=cv2.cvtColor(scr,cv2.COLOR_BGR2GRAY) # Convert the image to grayscale

    # Gray scale image of the wood block gives better results


    # ! Algo used - TM_CCOEFF_NORMED
    result=cv2.matchTemplate(scr,wood,cv2.TM_CCOEFF_NORMED)

    _,maxval,_,maxloc=cv2.minMaxLoc(result) # Find the maximum value and its location

    Threshold=0.6 # Threshold value for the maximum value

    if maxval>Threshold: # Object Matching
        flag=not flag

    return flag,maxloc
```

9. Template Matching Algorithms

Among the 6 algo we tested all and the best one was `cv2.TM_CCOEFF_NORMED`

1. `cv2.TM_CCOEFF`

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

3. `cv2.TM_CCORR`
4. `cv2.TM_CCORR_NORMED`
5. `cv2.TM_SQDIFF`
6. `cv2.TM_SQDIFF_NORMED`

So We used `cv2.TM_CCOEFF_NORMED`

In [ ]:
```
while 1:

  flag,maxloc=matchobstacle(flag) # Get flag and maxloc values

  if(flag): # If flag is 1, then left log is selected

    mouse.position=(885,700)

  else:

    mouse.position=(1165,700)

  val =0.043

  sleep(val)

  if keyboard.is_pressed('q'):
      break
```

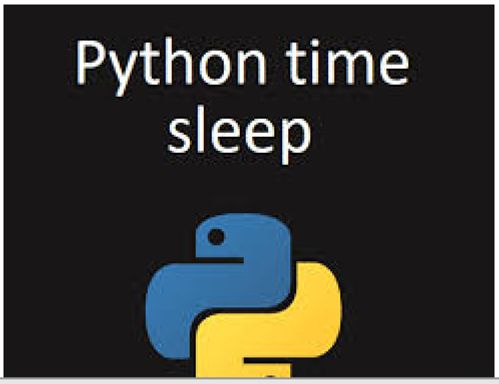
10. Note:

10.1. `time.sleep(val)`

```
If val increases -> the game will be Slower -> Lesser Hiscore
If val decreases -> the game will be faster -> PC stops working
```

After trying a lot of different values of `time.sleep(val)` we found that 0.043 was the best value.

! This value may differ from one computer to another.



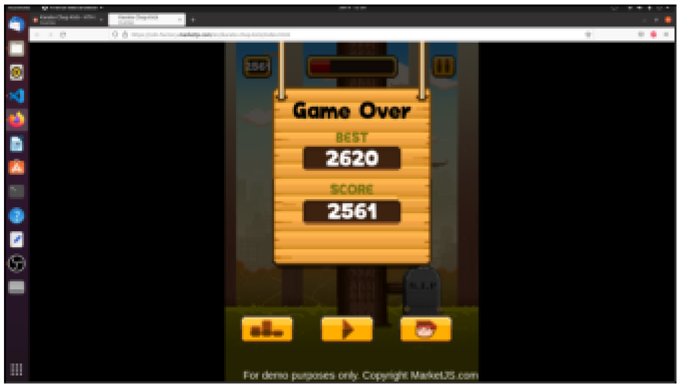
11. Conclusion

We got a high score of `2620` but in out run we got the score as `2561` (The game is very random)

Check out our high score!

In [ ]:
```python
# Image of hiscore
img = cv2.imread('hiscore.png')

# BGR to RGB
img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
```

In [ ]:
```python
# Show image
plt.imshow(img,cmap='gray')
plt.xticks([]), plt.yticks([])  # to hide tick values on X and Y axis
plt.figure(figsize=(30,30))
plt.show();
```



```
<Figure size 2160x2160 with 0 Axes>
```

So that's it from our side , Thanks for reading this. 😄

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js