

PROFESSIONAL TRAINING REPORT
at
Sathyabama Institute of Science and Technology
(DEEMED TO BE UNIVERSITY)

Submitted in partial fulfillment of the requirements for the award of
Bachelor of Engineering Degree in
Computer Science and Engineering
By

A D CRIS EVANGELINE
(Reg. No. 40110284)



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SCHOOL OF COMPUTING
SATHYABAMA INSTITUTE OF SCIENCE AND TECHNOLOGY
JEPPIAAR NAGAR, RAJIV GANDHI SALAI,
CHENNAI – 600119. TAMILNADU.

APRIL 2023



SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)

Accredited with Grade “A” by NAAC

(Established under Section 3 of UGC Act, 1956)

JEPPIAAR NAGAR, RAJIV GANDHI SALAI

CHENNAI- 600119

www.sathyabama.ac.in



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of **A D CRIS EVANGELENE (Reg. No. 40110284)** who carried out the project entitled “**LIVE COLOR DETECTION MODEL USING OPENCV**” under my supervision January 2023 to April 2023.

INTERNAL GUIDE

Mrs. V. Asha Judi, M.E.

HEAD OF THE DEPARTMENT

Dr. L. Lakshmanan, M.E., Ph.D.

Submitted for Viva voce Examination held on_____

Internal Examiner

External Examiner

DECLARATION

I, **A D CRIS EVANGELINE**, hereby declare that the project report entitled **LIVE COLOR DETECTION MODEL USING OPENCV** done by me, under the guidance of **Mrs. V. Asha Judi, M.E.** Is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering Degree in Computer Science and Engineering.

DATE:

PLACE:

SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to **the Board of Management of SATHYABAMA** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T. Sasikala M.E., Ph.D., Dean**, School of Computing, **Dr. L. Lakshmanan, M.E., Ph.D., Head of the Department of Computer Science and Engineering** for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **Mrs. V. Asha Judi, M.E.** for her valuable guidance, suggestions, and constant encouragement, which paved the way for the successful completion of my project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the **Department of Computer Science and Engineering** who were helpful in many ways for the completion of the project.

ABSTRACT

We all know of a disability in humans that is very commonly known as “Color Blindness”. Even though most of the people suffering with this disability manage it somehow by compromising that not knowing the colors will not have any adverse impact on their lives, but it still remains as a very common concern that needs to be addressed or taken care of. Because in my view every person gets an equal right to see things and enjoy. This project aims to do so. Even though the person might not view the color with his/her own eyes, using the concept of machine learning we can create a model that helps the person know about those colors easily.

This model can also be used in the self-driving cars. These self-driving cars can identify various colors on the road, and most importantly the traffic lights. They can respond based on the color they identify. If it identifies red in the traffic light, the response will be to stop, if it's green the response will be to go. So all these actions or responses need this color recognition model.

The main idea of this project is to provide ease and efficiency for the people suffering from color blindness to know about the colors around them and also to apply this model in the domain of self-driving cars. All they have to do is to place the object of whose color they want to know in front of the camera. This machine learning model uses OpenCV to detect the colors and displays the color name on the screen.

TABLE OF CONTENTS

CHAPTER No.	TITLE	PAGE NO.
	ABSTRACT	5
	LIST OF FIGURES	8
	LIST OF TABLES	-
	LIST OF ABBREVIATIONS	10
1.	INTRODUCTION	11
2.	AIM AND SCOPE OF THE INVESTIGATION	12
3.	SYSTEM, SOFTWARES AND ALGORITHMS USED	13
	3.1 System Requirements	
	3.2 Python	
	3.3 Installing Python in Windows	
	3.4 Installing Python in MacOS	
	3.5 PyCharm	
	3.6 Installing PyCharm in Windows	
	3.7 Installing PyCharm in MacOS	
	3.8 Installing OpenCV in PyCharm	
	3.9 Algorithm used – OpenCV	
4.	PROCEDURE, PERFORMANCE ANALYSIS AND RESULT	30
	4.1 Procedure	
	4.2 Analysis	
	4.3 Result	
5.	SUMMARY AND CONCLUSION	35

REFERENCES	36
APPENDIX	37
A. Screenshots	37
B. Source Code	40

	LIST OF FIGURES	
FIGURE NO.	FIGURE NAME	PAGE NO.
1	3.1 Python Logo	14
2	3.3.1 Official website of Python	15
3	3.3.2 Locate & Download Python	15
4	3.3.3 Welcome setup of Python	16
5	3.3.4 Selecting optional features	16
6	3.3.5 Advanced options in Python installation	17
7	3.3.6 Setup progress	17
8	3.3.7 Path length prompt	18
9	3.3.8 Installation successful	18
10	3.4.1 Official Python website	19
11	3.4.2 Download Python	19
12	3.4.3 Continue this installation setup	20
13	3.4.4 Read the guidelines and continue	20
14	3.4.5 Agree to terms and conditions	20
15	3.4.6 Select destination	21
16	3.4.7 Installation complete	21
17	3.5.1 PyCharm logo	23
18	3.6.1 PyCharm website	23
19	3.6.2 Download community version	24
20	3.6.3 PyCharm setup	24
21	3.6.4 Installation	25
22	3.6.5 Installation complete	25
23	3.7.1 Directory	25
24	3.7.2 Select custom location	26
25	3.7.3 Accept privacy policy	26
26	3.7.4 Select UI theme	26
27	3.7.5 Plugins	27
28	3.8.1 Python packages in Pycharm	27

	LIST OF FIGURES	
FIGURE NO.	FIGURE NAME	PAGE NO.
29	3.8.2 OpenCV package in PyCharm	27
30	3.8.3 OpenCV package installed	28
31	3.8.4 OpenCV package deletion	28
32	3.9.1 OpenCV logo	29
33	4.2.1 HSV values for red color	31
34	4.2.2 HSV values for blue color	31
35	4.2.3 Color ranges	32
36	4.2.4 Live capturing Frame	32
37	4.2.5 Indicator in the middle	33

LIST OF ABBREVIATIONS

ML	- Machine Learning
HSV	- Hue Saturation Value
CV	- Computer Vision

CHAPTER 1

INTRODUCTION

Machine learning is one of the booming technologies in today's tech-driven world that is used mostly in automation. It is the study of computer algorithms that lets computer programs automatically improve through the experience without human intervention. The advancements in technologies have benefitted mankind in various ways, thus reducing the human intervention in basic day-to-day tasks. The sole purpose of Machine learning is to make computers able to learn automatically. In other words, Machine learning which is a branch of artificial intelligence has a sole purpose that a computer system can learn and adapt itself to new data without any external input or human intervention. In our day-to-day lives, people suffering from “Color Blindness” do not get an equal chance to see things as normal people do.

Everyone person deserves to enjoy their lives and a disability should not be a hindrance to that. Using this software can actually help people suffering with this disability by helping them know about the colors around them.

Any person suffering with this disability of color blindness can use this software to view and know about the different colors that the objects are made of. Even though they might not be able to view all these colors with their own eyes like normal people, using this software will help them find out what colors are around them. This will enable them to learn these colors at an early age.

Hence this software provides a scope to address the problems of people suffering with this disability. This model can be integrated into much more advanced applications that might have better outcomes to offer to the people.

CHAPTER 2

AIM AND SCOPE OF THE INSPECTION

1. To build a machine learning model that detects the color of an object placed in front of the camera.
2. To make the process of learning easy for children, both suffering from color blindness and normal ones.
3. The project is made to reach each and everyone in the society suffering from color blindness so that everyone can benefit from this.
4. To use this color detection model in self-driving cars.

CHAPTER 3

SYSTEM, SOFTWARE AND ALGORITHMS USED

3.1 SYSTEM REQUIREMENTS

OS NAME : Windows
OS VERSION : 11
SYSTEM MODEL : Lenovo IdeaPad 3 15IIL05
MEMORY : 512gb

3.2 PYHTON

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together. Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms and can be freely distributed.

Often, programmers fall in love with Python because of the increased productivity it provides. Since there is no compilation step, the edit-test-debug cycle is incredibly fast. Debugging Python programs is easy: a bug or bad input will never cause a segmentation fault. Instead, when the interpreter discovers an error, it raises an exception.



Fig 3.1 Python Logo

When the program doesn't catch the exception, the interpreter prints a stack trace. A source level debugger allows inspection of local and global variables, evaluation of arbitrary expressions, setting breakpoints, stepping through the code a line at a time, and so on. The debugger is written in Python itself, testifying to Python's introspective power. On the other hand, often the quickest way to debug a program is to add a few print statements to the source: the fast edit-test-debug cycle makes this simple approach very effective.

FEATURES OF PYTHON

- It is free and open source.
- It is easy to read and code.
- Robust standard Libraries
- It is object oriented and procedure oriented.
- It is Dynamically typed.
- It is a high-level language.
- Supports GUI
- It is portable, extensible, and expressive.

3.3 INSTALLING PYTHON IN WINDOWS

1. Go to the official website of Python which is <https://www.python.org/>.

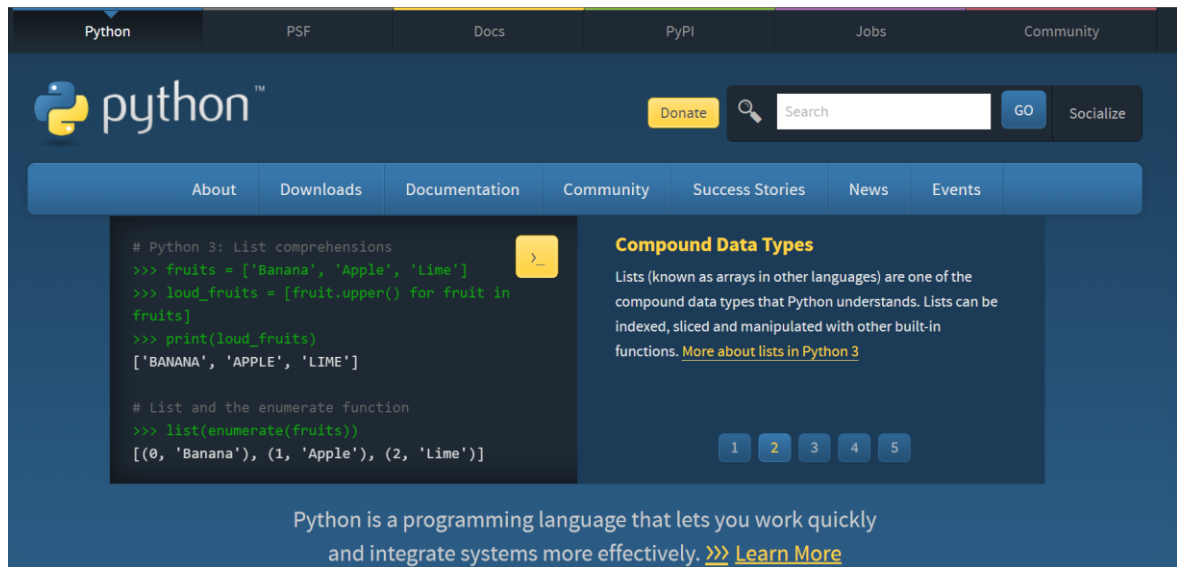


Fig 3.3.1. Official website of Python

2. Click on Downloads section

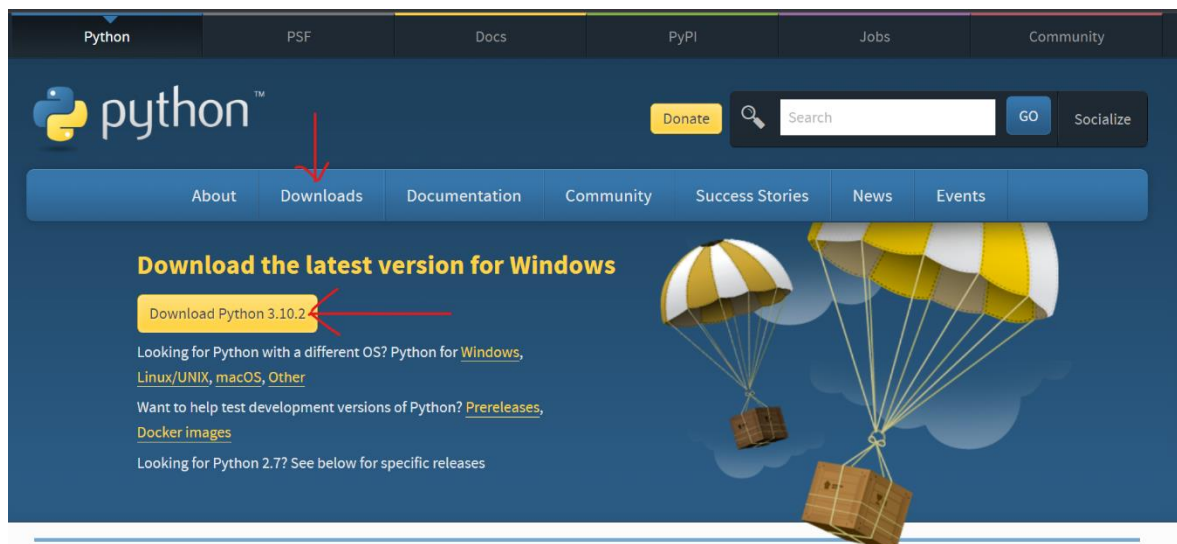


Fig 3.3.2. Locate & Download Python

Here you will get the latest version. Simply download the version that it shows. Keep in mind that here it will always show you the latest version of Python at the top.

3. Double click on the executable file (.exe file) and the installation wizard will open.

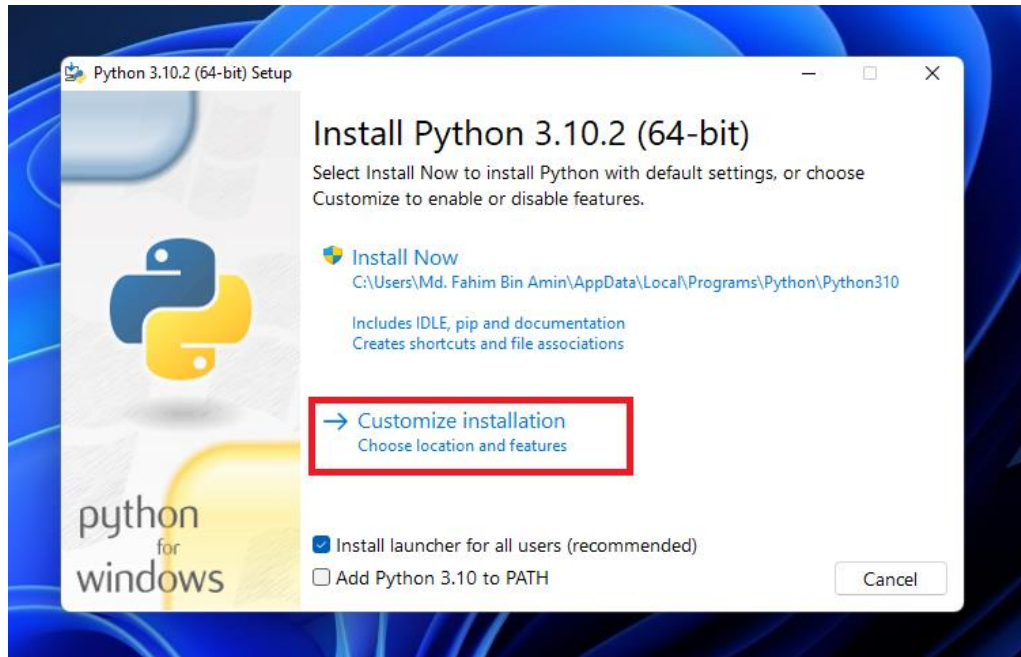


Fig 3.3.3 Welcome setup of Python

4. Click on customize installation.

5. Make sure to check all the boxes and click next.

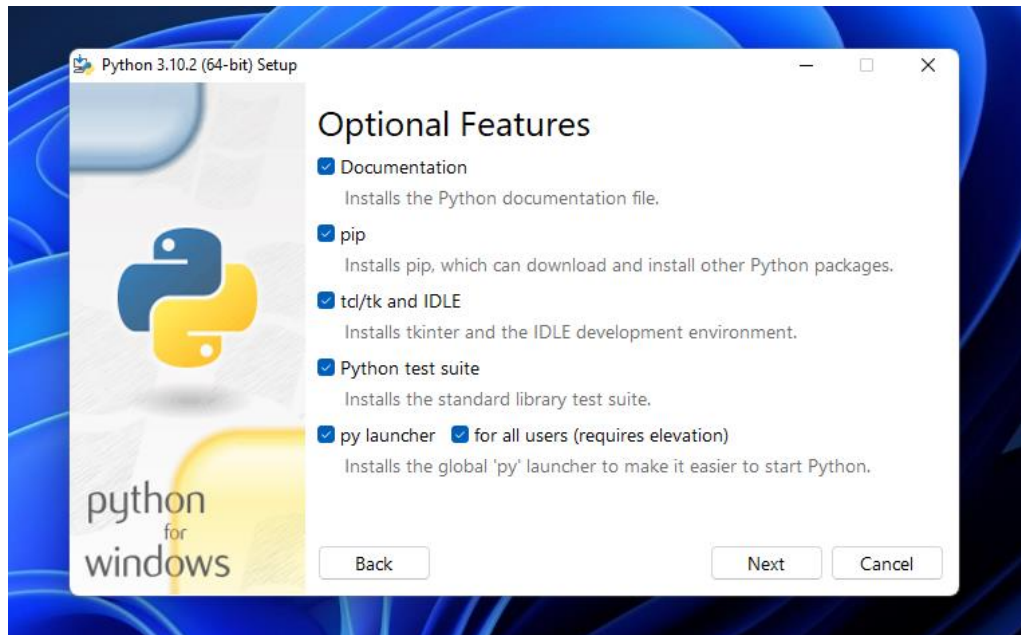


Fig 3.3.4 Selecting optional features.

7. Then you will see the below screen. Check the boxes if you wish to enable the advanced options.

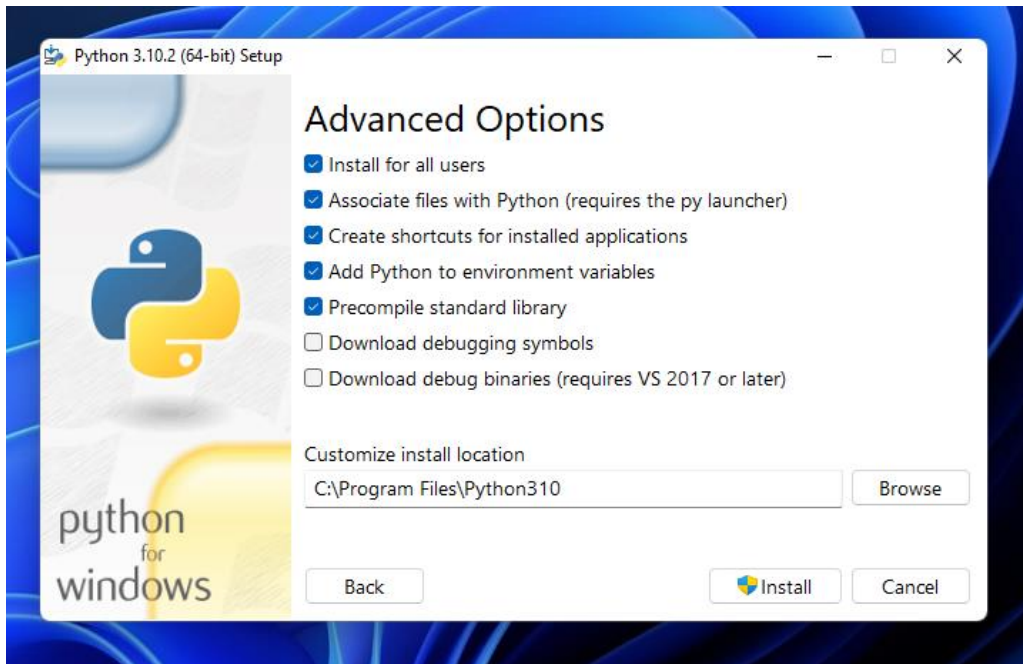


Fig 3.3.5 Advanced options in Python installation

8. Click on Install and the setup progress will be shown.

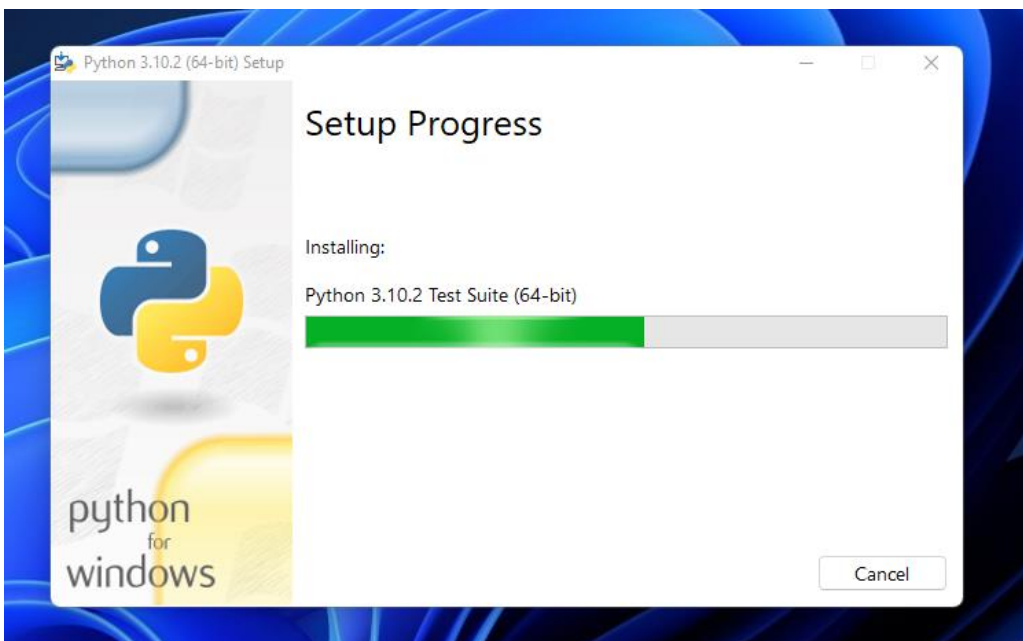


Fig 3.3.6 Setup progress

9. If you get this type of prompt to disable the path length limit, then simply click on that box. It disables the path length limit by removing the limitation on the MAX_PATH variable.

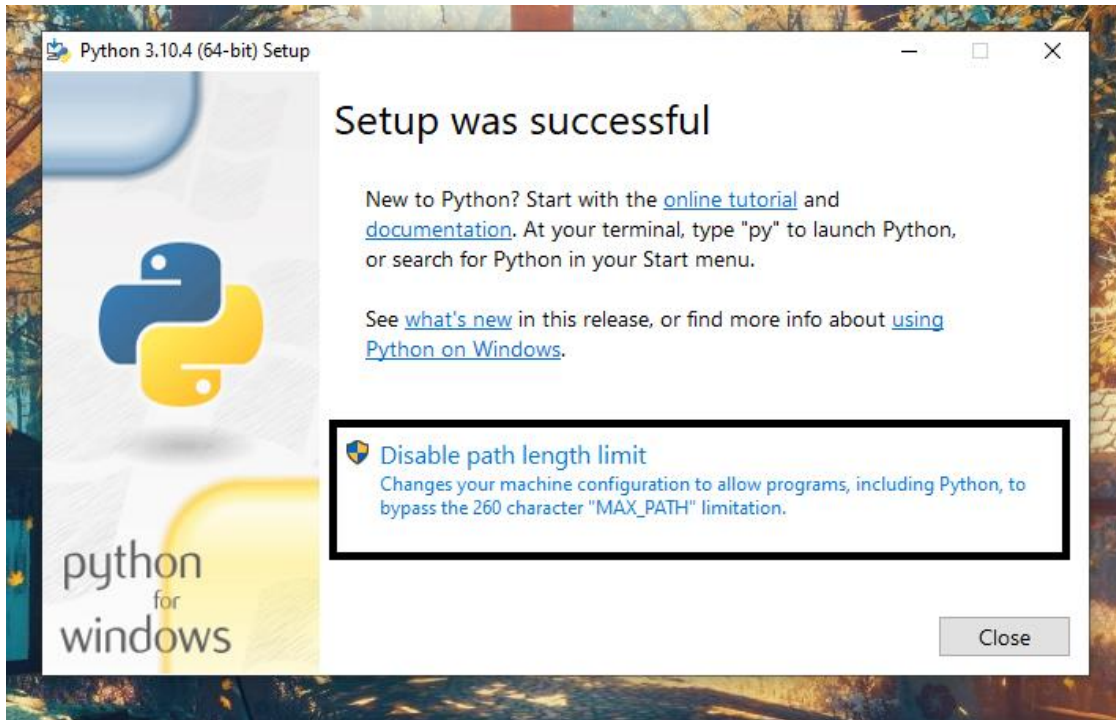


Fig 3.3.7 Path Length prompt

10. The installation has been finished successfully.

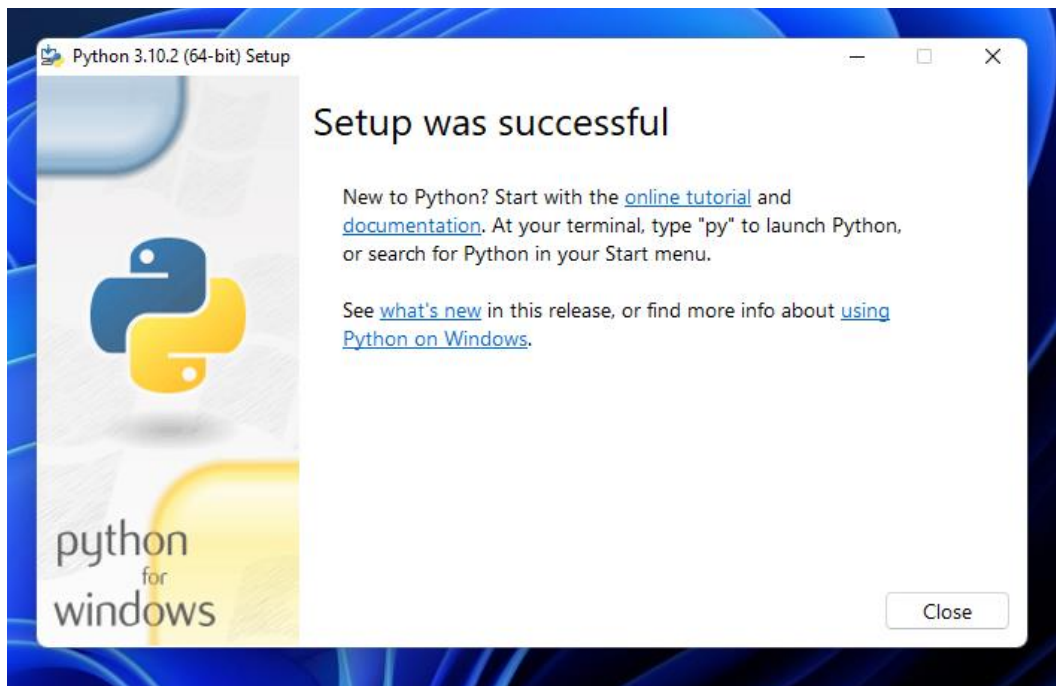


Fig 3.3.8 Installation successful

3.4 INSTALLING PYTHON ON MacOS

1. Go to the official Python website which is <https://www.python.org/>.

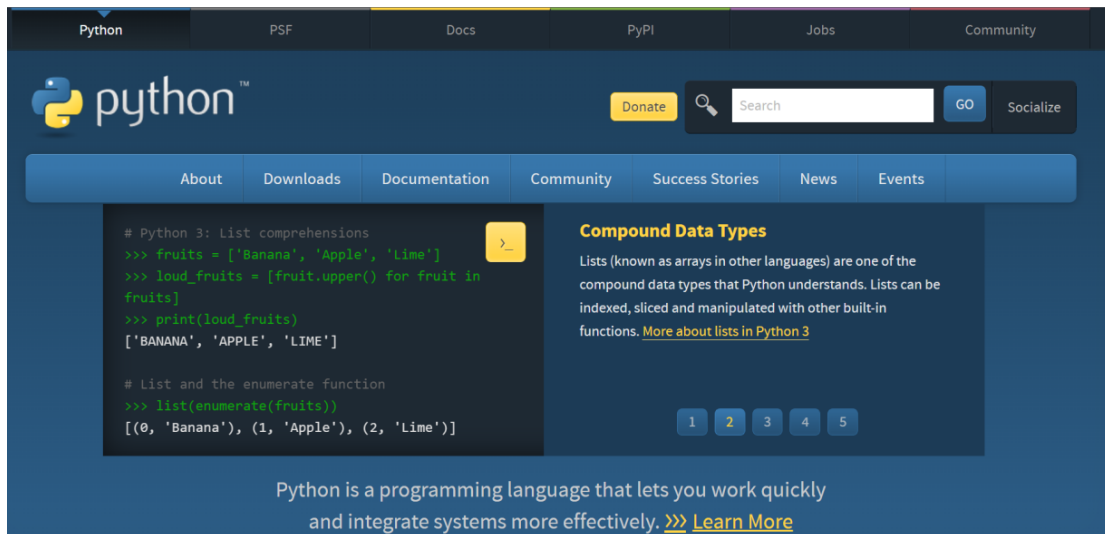


Fig 3.4.1 Official Python website

2. Click on Downloads section

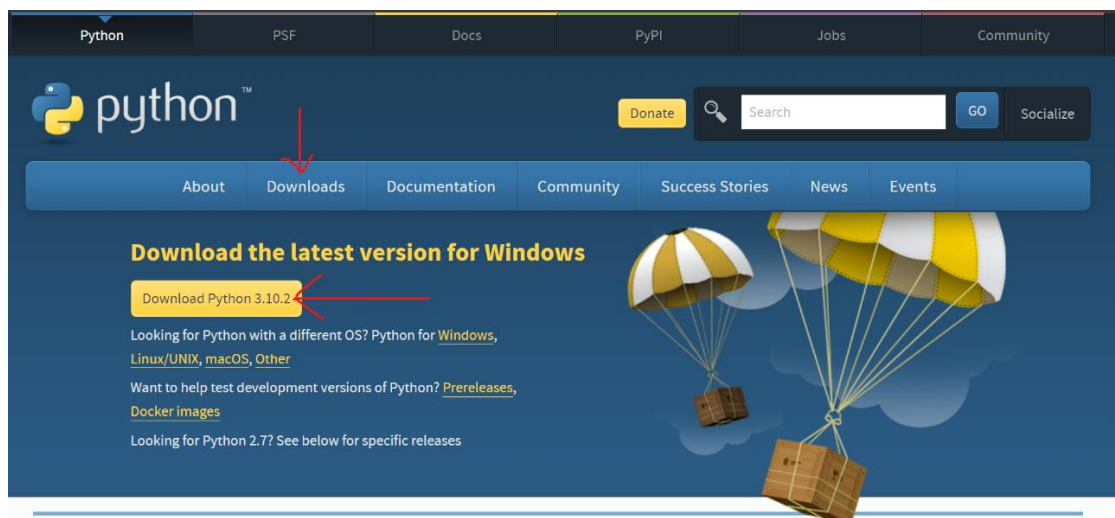


Fig 3.4.2 Download Python

3. Wait for the download to be completed. Once it's finished, double-click the package to start the installation process. You can follow the on-screen instructions in the Python installer for this step.

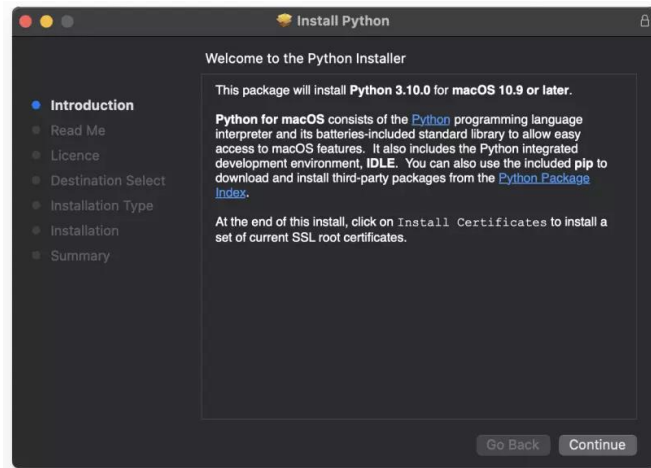


Fig 3.4.3 Continue the installation setup

4. Click on Continue

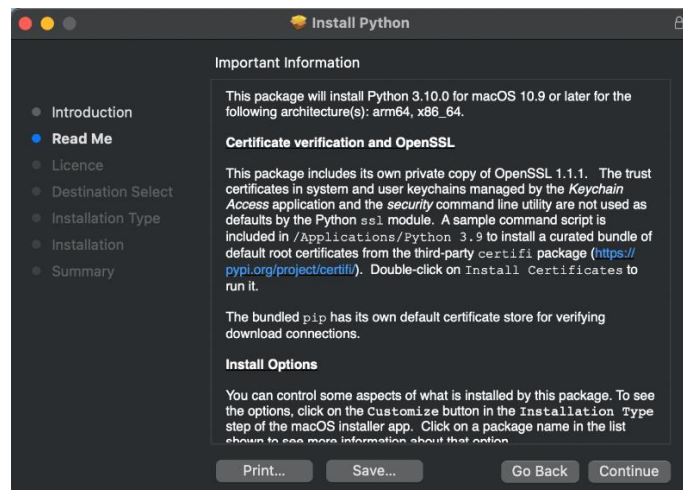


Fig 3.4.4 Read the guidelines and continue

5. Select agree if you agree with the agreement and conditions

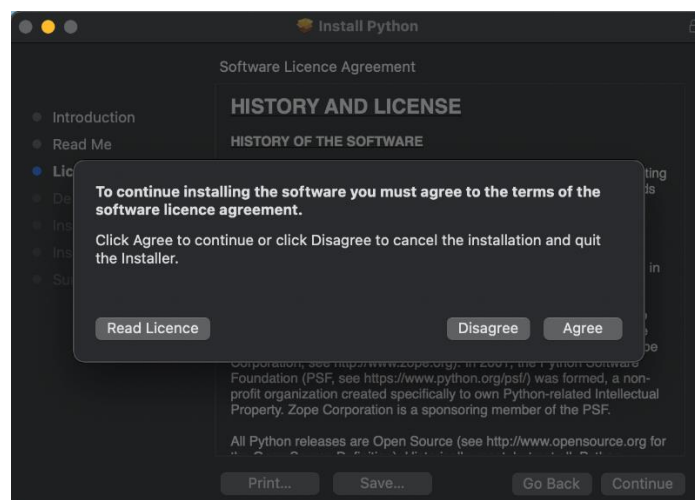


Fig 3.4.5 Agree to terms and conditions

6. Select the destination where you would like to install your files. Recommendation is not to change the default location that this installer has selected.

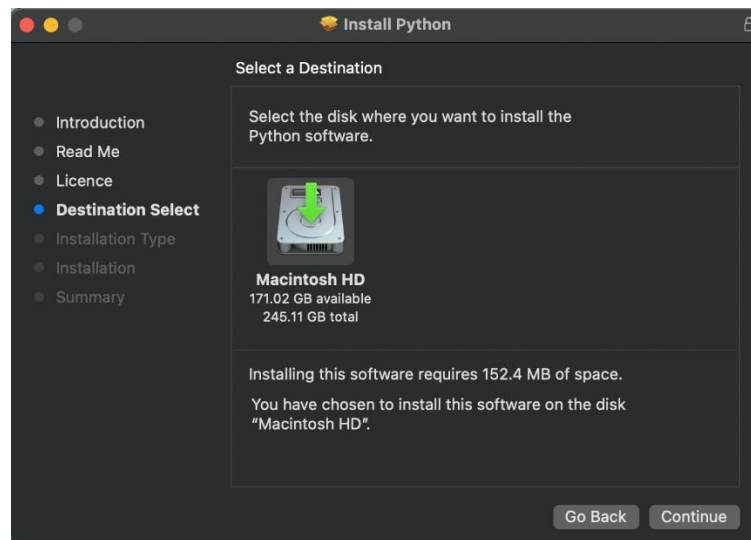


Fig 3.4.6 Select destination

7. Installation will start, and you'll get this message once done. You may close this window as Python 3.0 has now been installed on your machine.

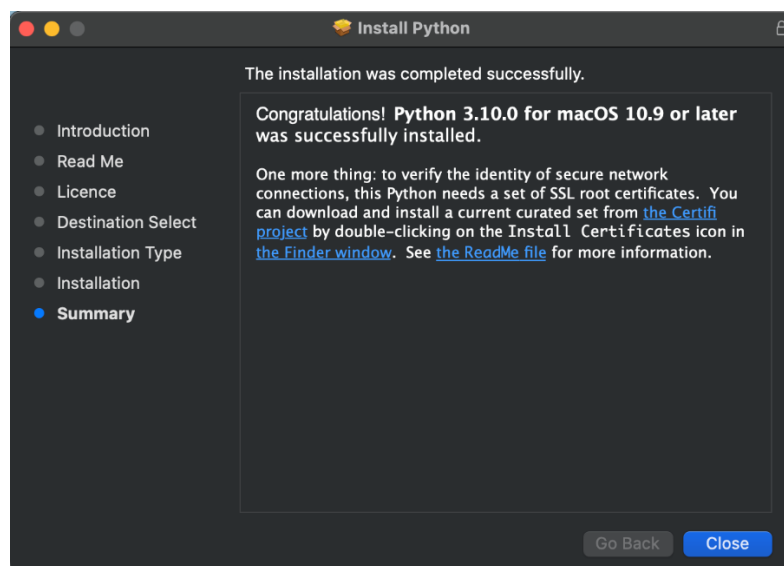


Fig 3.4.7 Installation complete

3.5 PYCHARM

PyCharm is a dedicated Python Integrated Development Environment (IDE) developed by JetBrains providing a wide range of essential tools for Python developers, tightly integrated to create a convenient environment for productive Python, web, and data science development. We can run PyCharm on Windows, Linux, or Mac OS. Additionally, it contains modules and packages that help programmers develop software using Python in less time and with minimal effort. Further, it can also be customized according to the requirements of developers. It supports two versions: v2.x and v3.x.

PyCharm is one of the most integrated Python IDEs, offering a range of modules and tools which make coding a lot faster and easier for programmers. One of the reasons for its popularity is the credentials of its developer, JetBrains, a Czech who is renowned for creating some of the most popular Java and JavaScript IDEs. It offers support for both Python 2 (2.7) and Python 3 (3.5 and above) versions and can be used on a multitude of platforms including Windows, Linux, and macOS.

PyCharm can be used for code analysis, debugging, and testing, among other things. It is particularly useful for web creation using web application frameworks like Django and Flask. Python plugins can be built by programmers using various APIs. It also allows programmers to access a range of databases without integrating with other tools. While designed specifically for programming with Python, it can also be used to create HTML, CSS, and JavaScript files. It also comes with a great user interface that can be modified based on applications using plugins.



Fig 3.5.1 PyCharm logo

3.6 INSTALLING PYCHARM IN WINDOWS

1. To download PyCharm, visit the official website of JetBrains:
<https://www.jetbrains.com/pycharm/>



Fig 3.6.1 PyCharm Website

2. Click on the “download” button.
3. After that, you will see the below window with two options, Professional and Community.
4. Download the Community version.

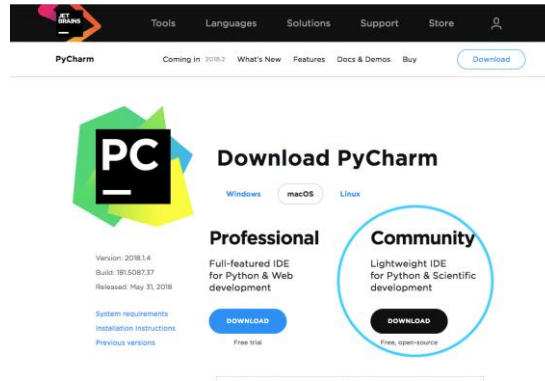


Fig 3.6.2 Download Community version

5. After downloading the file, click on it
6. When the following window appears, click on Next and the installation process will start

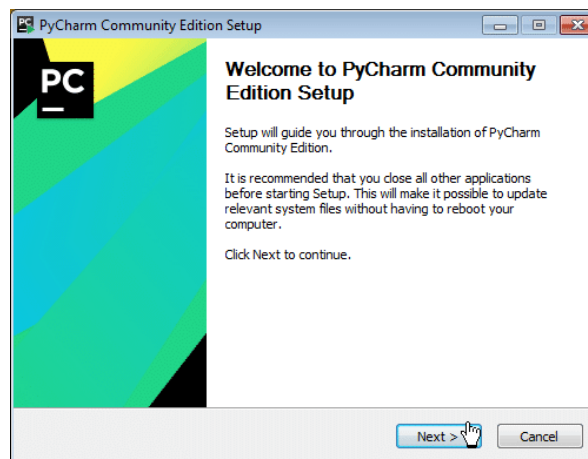


Fig 3.6.3 PyCharm Setup

7. After clicking on Next, first, a window for setting up the installation location will appear.
8. In the next step, you can set the Installation Options as per requirements, and then, click on the Next button to proceed.
9. Now, you have to select the Start Menu folder, or you can leave it as default.
10. After these steps, click on the Install button as above to start the installation process.

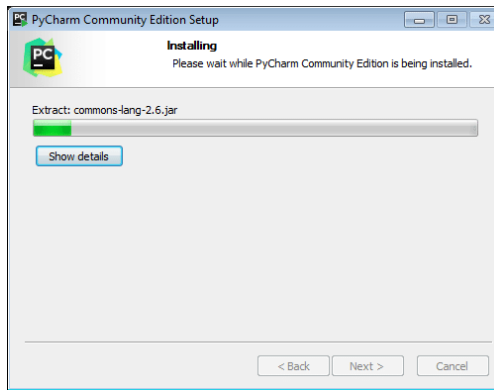


Fig 3.6.4 Installation

11. When you click on the Finish button, your PyCharm installation completes.

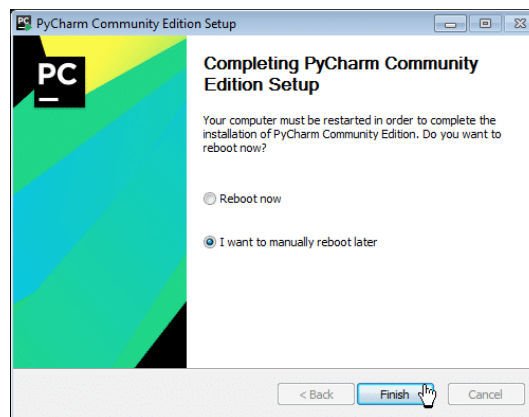


Fig 3.6.5 Installation complete

3.7 INSTALLING PYCHARM IN MacOS

Step 1: Download PyCharm from the official website of

Step 2: Mount the downloaded file to the volumes directory using the following command: `$ hdiutil mount Downloads/pycharm-community-2017.3.1.dmg`

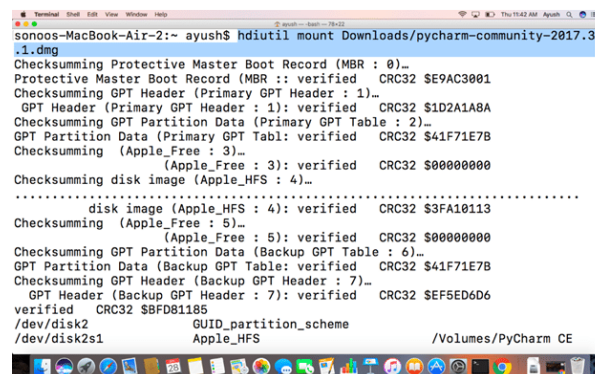


Fig 3.7.1 Directory

Step 3: To initiate PyCharm installation, write the below command:

```
$sudo /Volumes/PyCharm\ CE/PyCharm\ CE.app/Contents/MacOS/pycharm
```

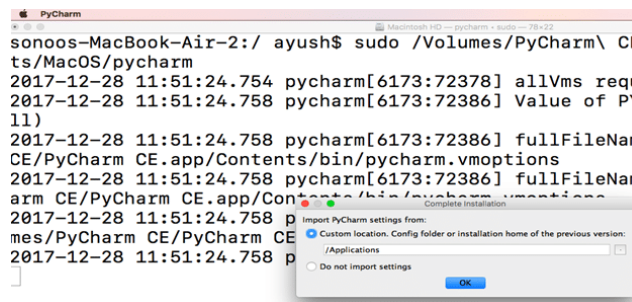


Fig 3.7.2 Select custom location

Step 4: Accept JetBrains privacy policy

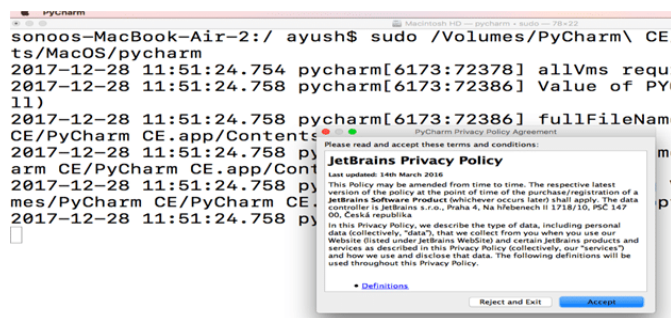


Fig 3.7.3 Accpet privacy policy

Step 5: After that, you will get an option to select a UI theme. Select it and click on Next.

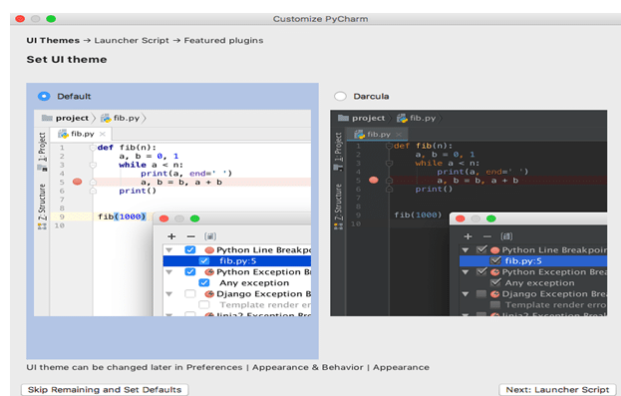


Fig 3.7.4 Select UI theme

Step 6: After selecting the UI theme, set the path

Step 7: Now, you can add featured plugins for your editor. After this step, press on Start using PyCharm to get started with PyCharm

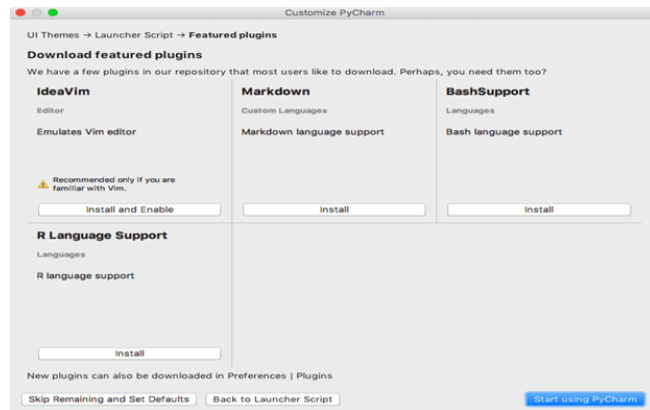


Fig 3.7.5 Plugins

3.8 INSTALLING OPENCV IN PYCHARM

1. Go to the Python Packages option at the bottom of the IDE window.

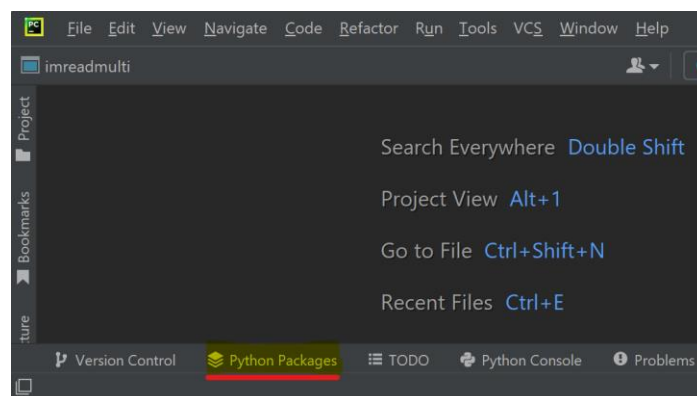


Fig 3.8.1 Python packages in PyCharm

2. Search for “opencv-python” and select the option from PyPI. Click on Install Package.

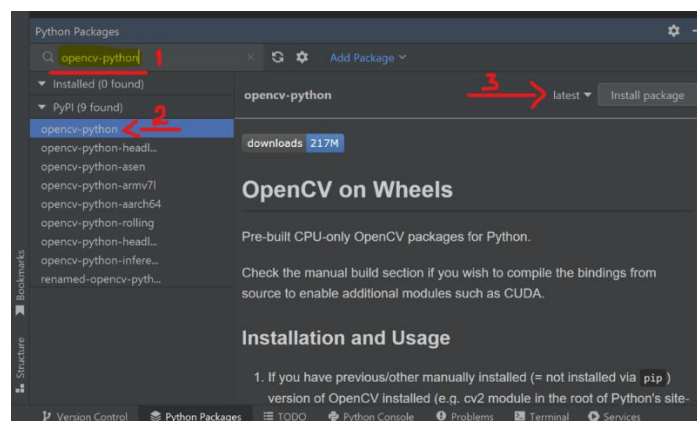


Fig 3.8.2 OpenCV package in PyCharm

3. The package is installed and is visible in the “Installed” Tab.

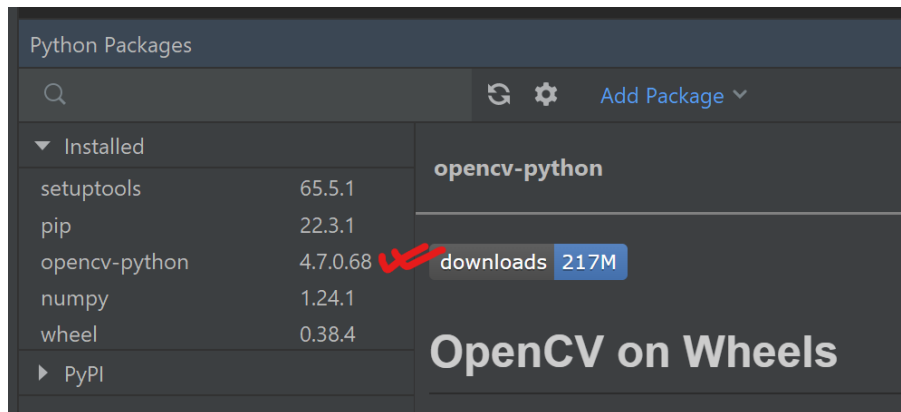


Fig 3.8.3 OpenCV package installed

4. To remove the package, click on the package name (here : opencv-python) from the Installed. Click three dots, select “Delete Package”.

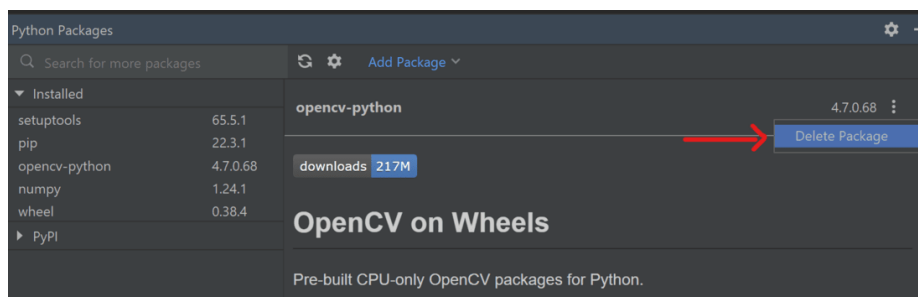


Fig 3.8.3 OpenCV package deletion

3.9 ALGORITHM USED – OPENCV (COMPUTER VISION)

OpenCV (Open-Source Computer Vision Library) is an open source computer vision and machine learning software library. OpenCV was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in commercial products. Being an Apache 2 licensed product, OpenCV makes it easy for businesses to utilize and modify the code.

The library has more than 2500 optimized algorithms, which includes a comprehensive set of both classic and state-of-the-art computer vision and machine learning algorithms. These algorithms can be used to detect and recognize faces, identify objects, classify human actions in videos, track camera movements, track moving objects, extract 3D models of objects, produce 3D point clouds from stereo

cameras, stitch images together to produce a high resolution image of an entire scene, find similar images from an image database, remove red eyes from images taken using flash, follow eye movements, recognize scenery and establish markers to overlay it with augmented reality, etc. OpenCV has more than 47 thousand people in the user community and an estimated number of downloads exceeding 18 million. The library is used extensively in companies, research groups and by governmental bodies.

It has C++, Python, Java, and MATLAB interfaces and supports Windows, Linux, Android, and Mac OS. OpenCV leans mostly towards real-time vision applications and takes advantage of MMX and SSE instructions when available. Full-featured CUDA and OpenCL interfaces are being actively developed right now. There are over 500 algorithms and about 10 times as many functions that compose or support those algorithms. OpenCV is written natively in C++ and has a templated interface that works seamlessly with STL containers.

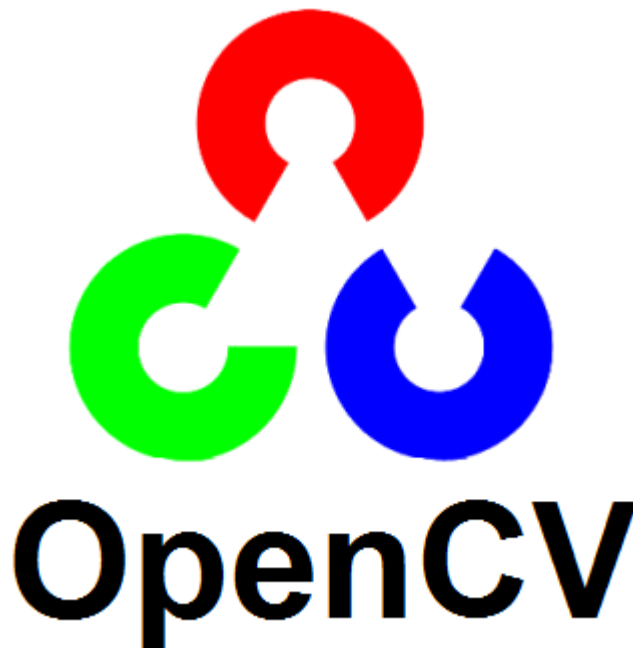


Fig 3.9.1 OpenCV logo

CHAPTER 4

PROCEDURE, PERFORMANCE ANALYSIS AND RESULT

4.1 PROCEDURE

1. Create a color picker to understand the ranges of various colors.
2. Use the HSV concept to make this color picker.
3. Use the color picker to choose the color range.
4. Assign the ranges for hue, saturation, and brightness in the code
5. Identify the range values of hue, saturation, and brightness that needs to be used in order to train the model.
6. Create a frame to capture the live actions of the user
7. Choose a place where you can place the indicator to place the object
8. Use the position of that indicator to capture the pixel values
9. The pixel values identified by the model is then mapped with the range values specified in the code.
10. After the values of that particular pixel fall into any one of the color ranges, the color name is displayed on the screen.

4.2 PERFORMANCE ANALYSIS

1. HSV color Picker

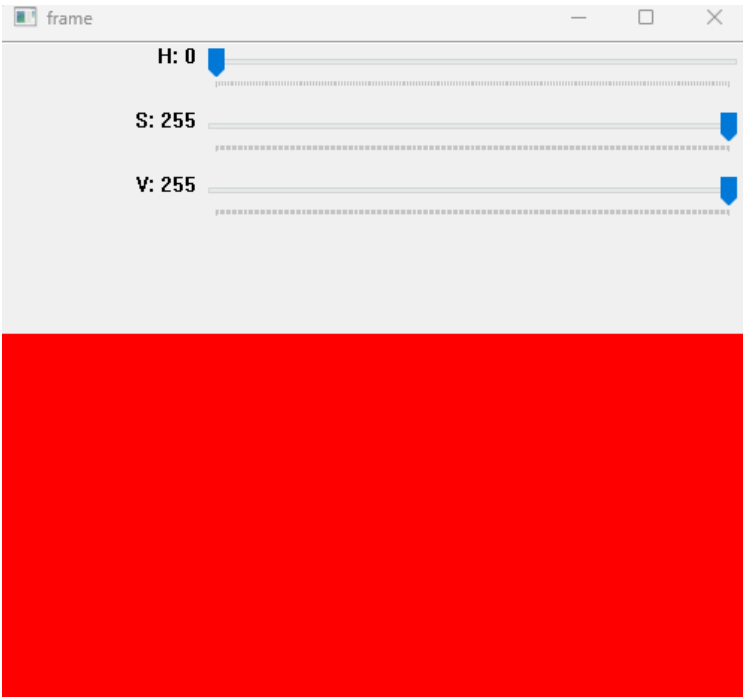


Fig 4.2.1 HSV values for Red color

2. Finding different colors using the HSV color poicker

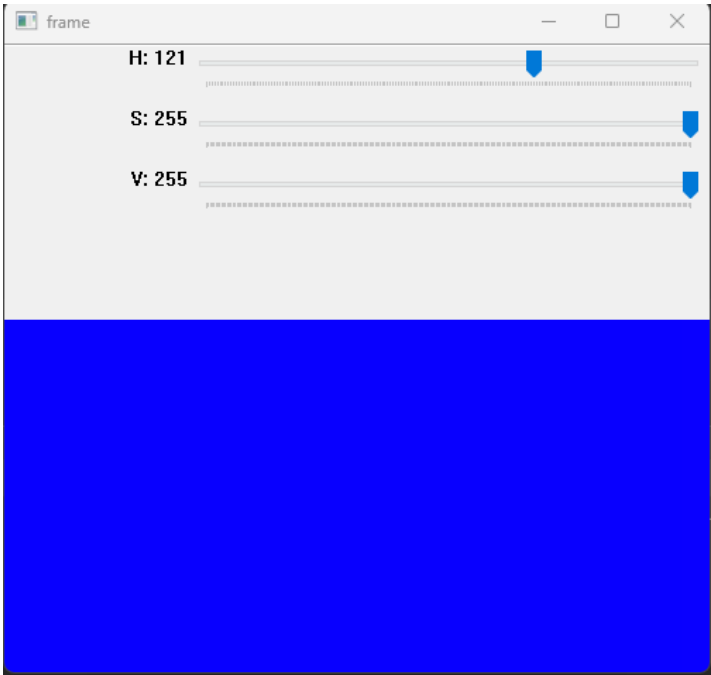


Fig 4.2.2 HSV values for Blue color

In my case, the HSV values for

Red are [0, 255, 255]

Blue are [121, 255, 255]

3. Using the same HSV color picker and by changing the Hue values we can get various colors. Color names are assigned to the respective range using elif statements.

```
if h < 180 and s < 255 and v < 30 and h > 0 and s > 0 and v > 0:  
    color = 'black'  
elif h < 180 and s < 18 and v < 255 and h > 0 and s > 0 and v > 231:  
    color = 'white'  
elif h < 180 and s < 255 and v < 255 and h > 159 and s > 50 and v > 70:  
    color = 'red'  
elif h < 9 and s < 55 and v < 255 and h > 0 and s > 50 and v > 70:  
    color = 'red'  
elif h < 89 and s < 255 and v < 255 and h > 36 and s > 50 and v > 70:  
    color = 'green'  
elif h < 128 and s < 255 and v < 255 and h > 90 and s > 50 and v > 70:  
    color = 'blue'  
elif h < 35 and s < 255 and v < 255 and h > 25 and s > 50 and v > 70:  
    color = 'yellow'  
elif h < 158 and s < 255 and v < 255 and h > 129 and s > 50 and v > 70:  
    color = 'purple'  
elif h < 24 and s < 255 and v < 255 and h > 10 and s > 50 and v > 70:  
    color = 'orange'  
elif h < 180 and s < 18 and v < 230 and h > 0 and s > 0 and v > 40:  
    color = 'gray'
```

Fig 4.2.3 Color ranges

4. Live camera Capturing Frame

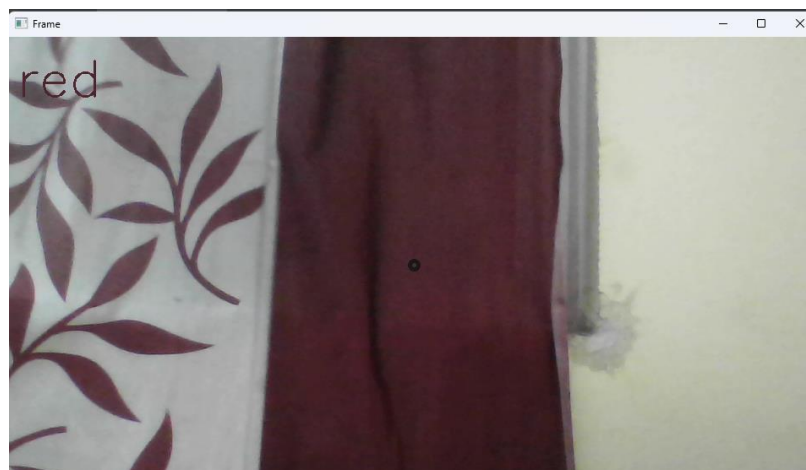


Fig 4.2.4 Live capturing Frame

5. Object placement indicator

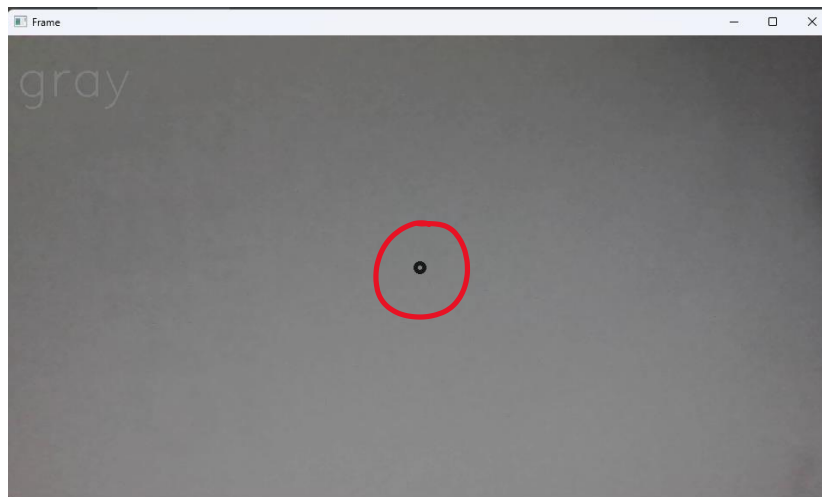


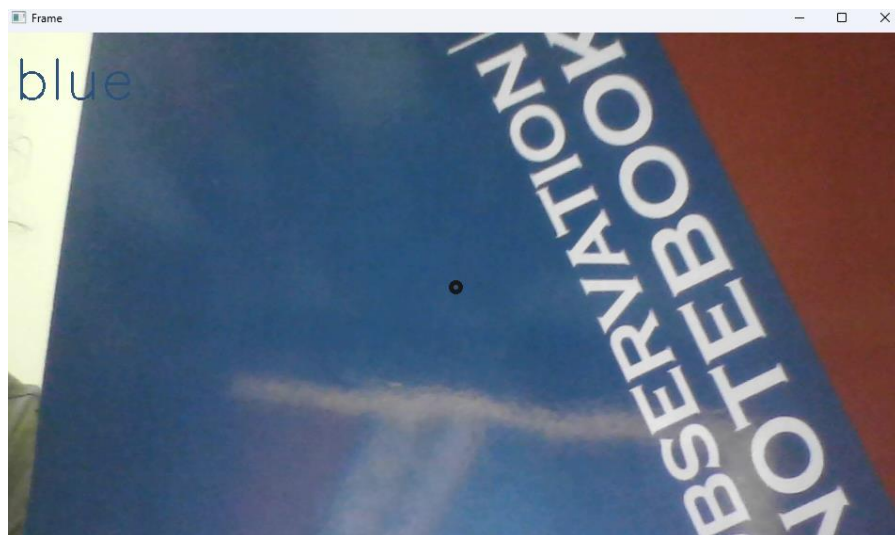
Fig 4.2.5 Indicator in the middle

6. Capturing the object
7. Color detection of object
8. Result is displayed on screen

4.3 RESULT

This report presents my developments pertaining to object detection and color detection. When the live camera captures the objects in front of it, the OpenCV algorithm tries to analyse and detects the color. The variations in the values of Hue, Saturation, brightness gives us various colors that can be detected by a human eye. So, the captured object color is matched with the range of colors that is predefined in the code using elif statements.

When I tested the model using various objects of different colors, out of which I have mentioned the results of 2 colors for reference. These are the results obtained:



CHAPTER 5

SUMMARY AND CONCLUSION

Color detection technology has come a long way and has a long way to go. We see self-drive cars running on roads by themselves following the traffic rules using the same color detection technology. Today, the machines are ready for it. Tesla is a frontrunner in this technology. However, next-generation color detection programs will have more upgradations. The apps in smart environments - where computers and equipment's are like assistants. To achieve this goal computers must be able to reliably identify nearby things and their basic properties like size shape and color in a manner that is naturally consistent within the normal human pattern. They do not require special interactions and should be in line with people's understanding of when recognition goes. This suggests that future intelligent environments should use the same methods as humans and have the same limitations.

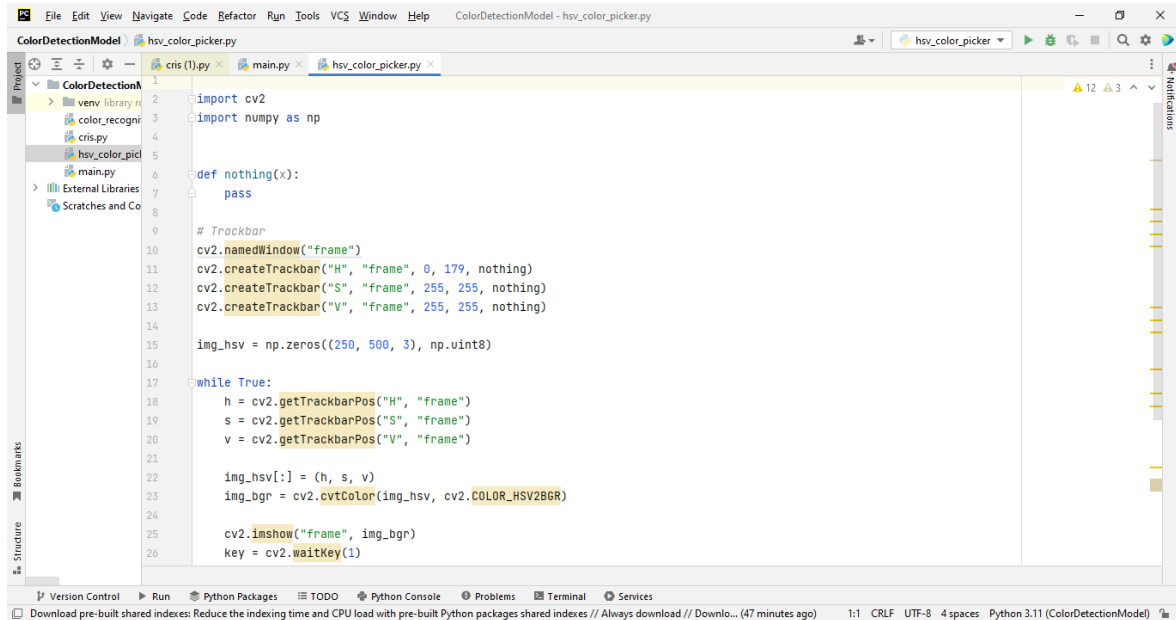
There might be a lot of ways to do this project, but I have used OpenCV which is an open-source computer vision library. OpenCV is used in many real-time applications also. It has some built-in functions to perform Color detection. Using this computer vision library, I was successfully able to train the machine learning model to detect the object color. Which now makes all the goals achievable and justifies the reason to create this project.

REFERENCES

1. Python - <https://www.python.org/doc/essays/blurb/>
2. Featured of Python - <https://www.simplilearn.com/python-features-article#:~:text=Python%20can%20be%20used%20to,in%20a%20more%20appealing%20way.>
3. Python installation in Windows - <https://www.freecodecamp.org/news/how-to-install-python-in-windows-operating-system/>
4. Python installation in MacOS - <https://www.scaler.com/topics/python/how-to-install-python-on-macos/>
5. PyCharm - <https://emeritus.org/blog/coding-what-is-pycharm/>
6. PyCharm installation in Windows - <https://intellipaat.com/blog/tutorial/pycharm-tutorial/pycharm-installation/>
7. PyCharm installation in MacOS - <https://intellipaat.com/blog/tutorial/pycharm-tutorial/pycharm-installation/>
8. OpenCV - <https://towardsdatascience.com/convolutional-neural-networks-explained-9cc5188c4939>

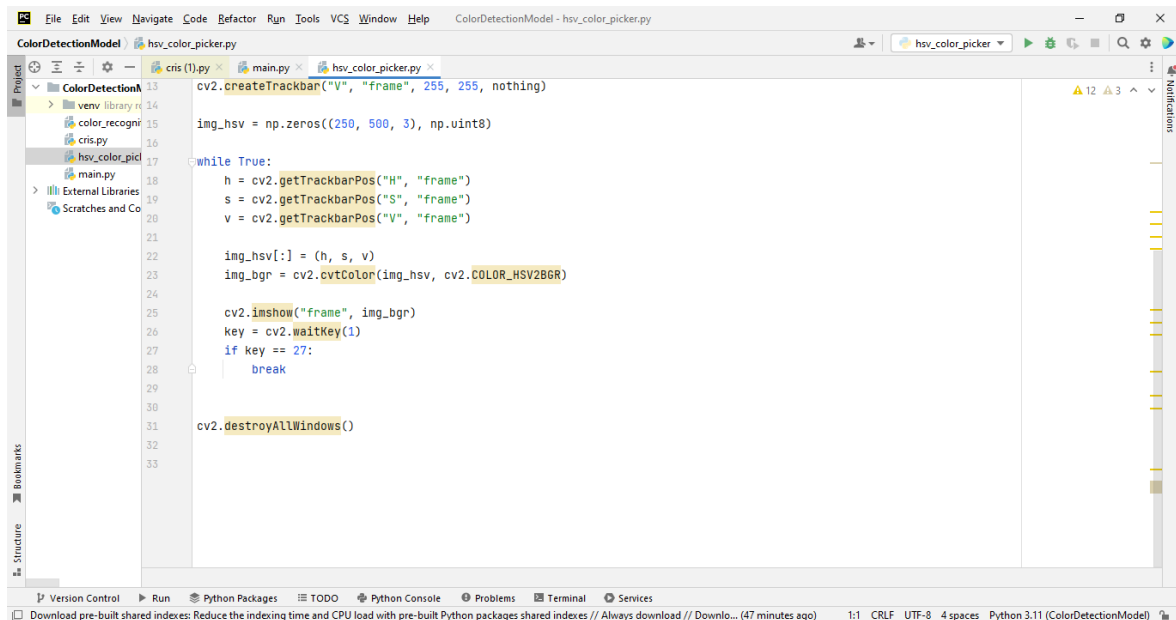
APPENDIX

A. SCREENSHOTS



This screenshot shows the first 26 lines of the `hsv_color_picker.py` script in an IDE. The script imports `cv2` and `numpy`, defines a `nothing(x)` function, and sets up three sliders for Hue, Saturation, and Value. It initializes an `img_hsv` array and enters a `while True` loop where it updates the sliders, converts the HSV values to BGR, and displays the result in a window named "frame".

```
1 import cv2
2 import numpy as np
3
4 def nothing(x):
5     pass
6
7 # Trackbar
8 cv2.namedWindow("frame")
9 cv2.createTrackbar("H", "frame", 0, 179, nothing)
10 cv2.createTrackbar("S", "frame", 255, 255, nothing)
11 cv2.createTrackbar("V", "frame", 255, 255, nothing)
12
13 img_hsv = np.zeros((250, 500, 3), np.uint8)
14
15 while True:
16     h = cv2.getTrackbarPos("H", "frame")
17     s = cv2.getTrackbarPos("S", "frame")
18     v = cv2.getTrackbarPos("V", "frame")
19
20     img_hsv[:] = (h, s, v)
21     img_bgr = cv2.cvtColor(img_hsv, cv2.COLOR_HSV2BGR)
22
23     cv2.imshow("frame", img_bgr)
24     key = cv2.waitKey(1)
```



This screenshot shows the continuation of the `hsv_color_picker.py` script, lines 13 through 33. It completes the `while True` loop by adding a `break` statement when the 'q' key is pressed, and then calls `cv2.destroyAllWindows()` to close all windows.

```
13 cv2.createTrackbar("V", "frame", 255, 255, nothing)
14
15 img_hsv = np.zeros((250, 500, 3), np.uint8)
16
17 while True:
18     h = cv2.getTrackbarPos("H", "frame")
19     s = cv2.getTrackbarPos("S", "frame")
20     v = cv2.getTrackbarPos("V", "frame")
21
22     img_hsv[:] = (h, s, v)
23     img_bgr = cv2.cvtColor(img_hsv, cv2.COLOR_HSV2BGR)
24
25     cv2.imshow("frame", img_bgr)
26     key = cv2.waitKey(1)
27     if key == 27:
28         break
29
30 cv2.destroyAllWindows()
31
32
33
```

The screenshot shows an IDE window titled 'ColorDetectionModel - C:\Users\crise\Downloads\cris (1).py'. The file explorer on the left shows a project named 'ColorDetectionModel' with files: 'venv library', 'color_recogni...', 'cris.py', 'hsv_color_p...', 'main.py', and 'Scratches and Co...'. The main editor displays the following Python code:

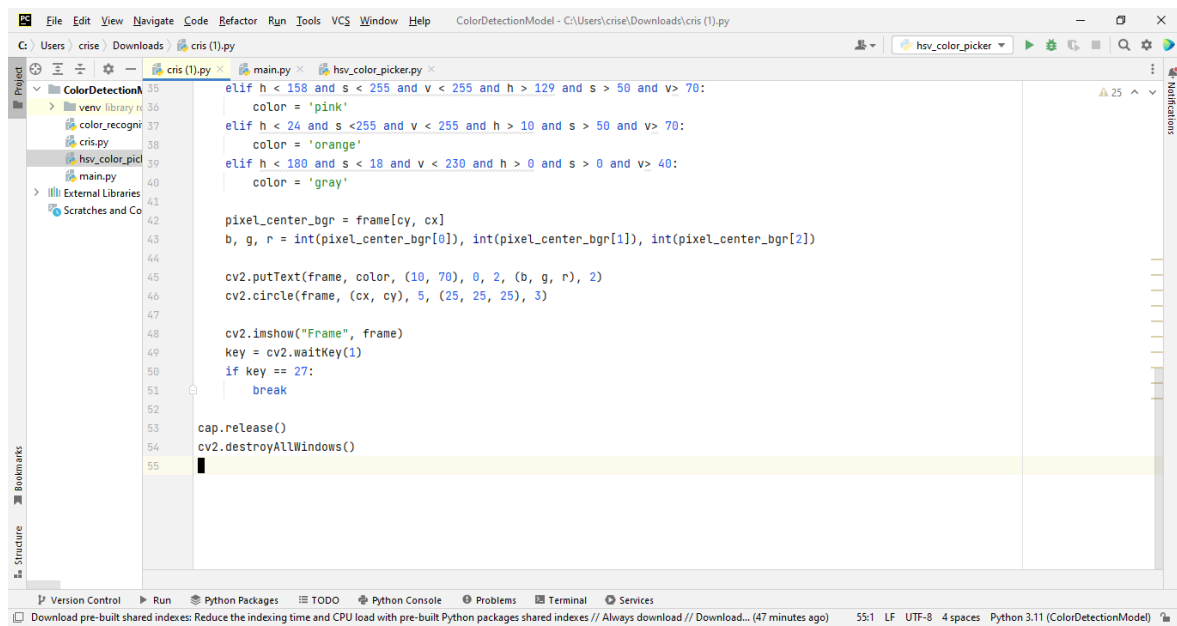
```
1 import cv2
2
3 cap = cv2.VideoCapture(0)
4 cap.set(cv2.CAP_PROP_FRAME_WIDTH, 1150)
5 cap.set(cv2.CAP_PROP_FRAME_HEIGHT, 720)
6
7 while True:
8     _, frame = cap.read()
9     hsv_frame = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)
10    # to get the centre of the frame
11    height, width, _ = frame.shape
12
13    cx = int(width / 2)
14    cy = int(height / 2)
15
16    # Pick pixel value
17    pixel_center = hsv_frame[cy, cx]
18    h, s, v = pixel_center
19
20    color = ""
21    if h < 180 and s < 255 and v < 30 and h > 0 and s > 0 and v > 0:
22        color = 'black'
23    elif h < 180 and s < 18 and v < 255 and h > 0 and s > 0 and v > 231:
24        color = 'white'
25    elif h < 180 and s < 255 and v < 255 and h > 159 and s > 50 and v > 70:
26        color = 'red'
27    while True:
28        if key == 27:
```

The status bar at the bottom indicates: 'Download pre-built shared indexes: Reduce the indexing time and CPU load with pre-built Python packages shared indexes // Always download // Download... (46 minutes ago) 51:14 LF UTF-8 4 spaces Python 3.11 (ColorDetectionModel)'.

The screenshot shows the same IDE window, but the code is scrolled down to show the continuation of the color detection logic and the final display steps:

```
25 elif h < 180 and s < 255 and v < 255 and h > 159 and s > 50 and v > 70:
26     color = 'red'
27 elif h < 9 and s < 55 and v < 255 and h > 0 and s > 50 and v > 70:
28     color = 'red'
29 elif h < 89 and s < 255 and v < 255 and h > 36 and s > 50 and v > 70:
30     color = 'green'
31 elif h < 128 and s < 255 and v < 255 and h > 90 and s > 50 and v > 70:
32     color = 'blue'
33 elif h < 35 and s < 255 and v < 255 and h > 25 and s > 50 and v > 70:
34     color = 'yellow'
35 elif h < 158 and s < 255 and v < 255 and h > 129 and s > 50 and v > 70:
36     color = 'pink'
37 elif h < 24 and s < 255 and v < 255 and h > 10 and s > 50 and v > 70:
38     color = 'orange'
39 elif h < 180 and s < 18 and v < 230 and h > 0 and s > 0 and v > 40:
40     color = 'gray'
41
42 pixel_center_bgr = frame[cy, cx]
43 b, g, r = int(pixel_center_bgr[0]), int(pixel_center_bgr[1]), int(pixel_center_bgr[2])
44
45 cv2.putText(frame, color, (10, 70), 0, 2, (b, g, r), 2)
46 cv2.circle(frame, (cx, cy), 5, (25, 25, 25), 3)
47
48 cv2.imshow("Frame", frame)
49 key = cv2.waitKey(1)
50 if key == 27:
51     while True:
52         if key == 27:
```

The status bar at the bottom indicates: 'Download pre-built shared indexes: Reduce the indexing time and CPU load with pre-built Python packages shared indexes // Always download // Download... (47 minutes ago) 51:14 LF UTF-8 4 spaces Python 3.11 (ColorDetectionModel)'.



B. SOURCE CODE

#1. HSV color picker

```
import cv2
import numpy as np

def nothing(x):
    pass

# Trackbar
cv2.namedWindow("frame")
cv2.createTrackbar("H", "frame", 0, 179, nothing)
cv2.createTrackbar("S", "frame", 255, 255, nothing)
cv2.createTrackbar("V", "frame", 255, 255, nothing)

img_hsv = np.zeros((250, 500, 3), np.uint8)

while True:
    h = cv2.getTrackbarPos("H", "frame")
    s = cv2.getTrackbarPos("S", "frame")
    v = cv2.getTrackbarPos("V", "frame")

    img_hsv[:] = (h, s, v)
    img_bgr = cv2.cvtColor(img_hsv, cv2.COLOR_HSV2BGR)

    cv2.imshow("frame", img_bgr)
    key = cv2.waitKey(1)
    if key == 27:
        break

cv2.destroyAllWindows()
```

#2. Color detection model

```
import cv2

cap = cv2.VideoCapture(0)
cap.set(cv2.CAP_PROP_FRAME_WIDTH, 1150)
cap.set(cv2.CAP_PROP_FRAME_HEIGHT, 720)

while True:
    _, frame = cap.read()
    hsv_frame = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)
    # to get the centre of the frame
    height, width, _ = frame.shape
```



```

cx = int(width / 2)
cy = int(height / 2)

# Pick pixel value
pixel_center = hsv_frame[cy, cx]
h, s, v = pixel_center

color = ""
if h < 180 and s < 255 and v < 30 and h > 0 and s > 0 and v > 0:
    color = 'black'
elif h < 180 and s < 18 and v < 255 and h > 0 and s > 0 and v > 231:
    color = 'white'
elif h < 180 and s < 255 and v < 255 and h > 159 and s > 50 and v > 70:
    color = 'red'
elif h < 9 and s < 55 and v < 255 and h > 0 and s > 50 and v > 70:
    color = 'red'
elif h < 89 and s < 255 and v < 255 and h > 36 and s > 50 and v > 70:
    color = 'green'
elif h < 128 and s < 255 and v < 255 and h > 90 and s > 50 and v > 70:
    color = 'blue'
elif h < 35 and s < 255 and v < 255 and h > 25 and s > 50 and v > 70:
    color = 'yellow'
elif h < 158 and s < 255 and v < 255 and h > 129 and s > 50 and v > 70:
    color = 'pink'
elif h < 24 and s < 255 and v < 255 and h > 10 and s > 50 and v > 70:
    color = 'orange'
elif h < 180 and s < 18 and v < 230 and h > 0 and s > 0 and v > 40:
    color = 'gray'

pixel_center_bgr = frame[cy, cx]
b, g, r = int(pixel_center_bgr[0]), int(pixel_center_bgr[1]), int(pixel_center_bgr[2])

cv2.putText(frame, color, (10, 70), 0, 2, (b, g, r), 2)
cv2.circle(frame, (cx, cy), 5, (25, 25, 25), 3)

cv2.imshow("Frame", frame)
key = cv2.waitKey(1)
if key == 27:
    break

cap.release()
cv2.destroyAllWindows()

```