PROGRAM BOOK FOR

SEMESTER INTERNSHIP

Name of the student: - Reddy cherala Gurucharan varma

Name of the College: - Kakaraparti Bhavanarayana College

Registration Number: - K2001543

Period of Internship: From: - 09/01/2023 To: - 10/05/2023

Name & Address of Internship: - ANVHAYA Technical Solutions Private Limited, Plot No.8.9.10. 4th floor Puppalaguda,Rajendranager,Manikonda,Hyderabad-500089.

KAKARAPARTI BHAVANARAYANA COLLEGE (AUTONOMOUS)

Affiliated to Krishna University

YEAR: 2020 - 2023

An Internship Report on

Social Media Comment Classification

Submitted in accordance with the requirement for the Degree of

B.Sc. Data Science

Under the Faculty Guide ship of

Mr. P. Ravindra

Department of Computer Science & Applications

Kakaraparti Bhavanarayana College (Autonomous)

Submitted by: -

Reddy cherala Gurucharan varma

Reg. No: - **K2001543**

Department of Computer Science & Applications

Kakaraparti Bhavanarayana College (Autonomous)

Instructions to Students

Please read the detailed Guidelines on Internship hosted on the website of AP State Council of Higher Education **https://apsche.ap.gov.in**

1. It is mandatory for all the students to complete 2 months (180 hours) of short- term internship either physically or virtually.
2. Every student should identify the organization for internship in consultation with the College Principal/the authorized person nominated by the principal.
3. Report to the intern organization as per the schedule given by the College. You must make your own arrangements for transportation to reach the organization.
4. You should maintain punctuality in attending the internship. Daily attendance is compulsory.
5. You are expected to learn about the organization, policies, procedures, and processes by interacting with the people working in the organization and by consulting the supervisor attached to the interns.
6. While you are attending the internship, follow the rules and regulations of the intern organization.
7. While in the intern organization, always wear your College Identity Card.
8. If your college has a prescribed dress as uniform, wear the uniform daily, as you attend to your assigned duties.
9. You will be assigned a Faculty Guide from your College. He/ She will be creating a What'sapp group with your fellow interns. Post your daily activity done and/or any difficulty you encounter during the internship.
10. Identify five or more learning objectives in consultation with your Faculty Guide. These learning objectives can address:
    a. Data and Information you are expected to collect about the organization and/or industry.
    b. Job Skills you are expected to acquire.
    c. Development of professional competencies that lead to future career success.
11. Practice professional communication skills with team members, co-interns, and your supervisor. This includes expressing thoughts and ideas effectively through oral, written, and non-verbal communication, and utilizing listening skills.

12. Be aware of the communication culture in your work environment. Follow up and communicate regularly with your supervisor to provide updates on your progress with work assignments.

13. Never be hesitant to ask questions to make sure you fully understand what you need to do your work and to contribute to the organization.

14. Be regular in filling up your Program Book. It shall be filled up in your own handwriting. Add additional sheets wherever necessary.

15. At the end of internship, you shall be evaluated by your Supervisor of the intern organization.

16. There shall also be evaluation at the end of the internship by the Faculty Guide and the Principal.

17. Do not meddle with the instruments/equipment you work with.

18. Ensure that you do not cause any disturbance to the regular activities of the intern organization.

19. Be cordial but not too intimate with the employees of the intern organization and your fellow interns.

20. You should understand that during the internship program me, you are the ambassador of your college, and your behavior during the internship program me is of utmost importance.

21. If you are involved in any discipline related issues, you will be withdrawn from the internship program me immediately and disciplinary action shall be initiated.

22. Do not forget to keep up your family pride and prestige of your college.

Student's Declaration

I, **Reddy cherala Gurucharan varma** of a student of B.Sc. Data Science Program, Reg. No. **K2001543** of the Department of Computer Science & Applications, Kakaraparti Bhavanarayana College (Autonomous) do hereby declared that I have completed the mandatory internship from 09/01/2023 to 10/05/2023 in **Anvhaya Technical Solutions Private Limited,** under the Faculty Guide ship of Assistances Professor **Mr.P. Ravindra,** Department of Computer Science & Applications, Kakaraparti Bhavanarayana College (Autonomous),VIJAYAWADA.

(Signature & Date)

Official Certification

This is to certify that **Reddy cherala Gurucharan varma** Reg. No. **K2001543** has completed the his/her internship in **Anvhaya Technical Solutions Private Limited** on **Social Media Comment Classification** under my supervision as a part of partial fulfillment of the requirement for the Degree of **B.Sc. Data Science** in the Department of Computer Science & Applications, Kakaraparti Bhavanarayana College (Autonomous) Vijayawada.

This is accepted for evaluation.

(Signature with Date and Seal)

Endorsements

Faculty Guide

Head of the Department

Principal

Certificate from Intern Organization

This is to certify that **Reddy cherala Gurucharan varma** Reg. No. **K2001543** of Kakaraparti Bhavanarayana College (Autonomous) underwent internship in **Anvhaya Technical Solutions Private Limited** from 09/01/2023 to 10/05/2023.

The overall performance of the intern during his/her internship is found to be Satisfactory (Satisfactory/Not Satisfactory).

SATISH
Head of Human Resource
Anvhaya Technical solutions Pvt Ltd

(Authorized Signatory with Date and Seal)

Acknowledgement's

This project work is a golden opportunity for learning and self-development. We consider our self very lucky to have so many people lead us through in completion of this project.

We express our sincere thanks to the Management and our beloved Principal **Dr. V. NARAYANA RAO** for providing such wonderful facilities required encouragement in completion of this entire project work.

We place a Record, Our sincere gratitude to **Sri. P. RAVINDRA**, HOD of Computer Applications for his constant encouragement. He monitored the progress and arranged all facilities to make this project easier. We owe our profound gratitude to Project Guide **Mr.P. RAVINDRA** who took keen interest on this project work and guided us all along and whose patience we have portablyused to the limit. She was always so involved in the entire process, shared her knowledge, and encouraged to think.

We would like to thank all the other faculty members, technical staff and supporting staff who have provided their contribution in the completion of this project work.

Name	**Roll.No.**	**Reg.No.**
Reddy cherala Gurucharan varma	205243	K2001543

Contents

CHAPTER 1: EXECUTIVE SUMMARY

Social Media Comment Classification:

The social media comment classification project aims to develop a system that can automatically classify comments posted on social media platforms. With the widespread use of social media, there is an enormous volume of user-generated content, including comments, which can be challenging to moderate and analyze manually. By implementing a comment classification system, we can effectively categorize comments based on their content, sentiment, or other relevant attributes, enabling more efficient moderation, content filtering, and data analysis.

Learning Objectives:

- **Understand social media comment classification: Gain a comprehensive understanding of the principles and techniques involved in classifying social media comments. Explore various approaches such as natural language processing (NLP), machine learning, and deep learning for comment analysis.**

- **Data preprocessing and feature extraction: Learn how to preprocess social media comment data, including tasks such as text normalization, tokenization, and handling of special characters or emojis. Extract meaningful features from comments that can be used as input for classification algorithms.**

- **Explore classification algorithms: Familiarize yourself with a range of classification algorithms suitable for comment classification tasks. Study traditional algorithms such as Naive Bayes, Support Vector Machines (SVM), and decision trees, as well as more advanced techniques like recurrent neural networks (RNNs) and transformer models.**

- **Sentiment analysis: Develop skills in sentiment analysis to determine the sentiment expressed in social media comments. Learn how to identify positive, negative, or neutral sentiments, and explore techniques such as lexicon-based approaches, machine learning models, and deep learning architectures for sentiment classification.**

- **Multi-class classification: Understand how to handle multi-class classification scenarios when comments need to be categorized into multiple predefined classes or categories. Learn about techniques like one-vs-all classification, softmax regression, and neural network architectures suitable for multi-class classification.**

- **Evaluation metrics: Learn how to evaluate the performance of a comment classification model. Explore metrics such as accuracy, precision, recall, F1 score, and confusion matrix to assess the effectiveness of the classification system and optimize its performance.**

- **Model deployment and integration: Gain insights into deploying the comment classification model in a real-world setting. Explore techniques for model integration with social media**

CHAPTER 2: OVERVIEW OF THE ORGANIZATION

ANVHAYA Technical Solutions Private Limited provides ultimate software training. Our group technical savvier who are dedicated to bring difference in the way people are trained in software tools & technologies. We focus on practical approach with well-defined process & framework that perfectly transforms the academics to professionals. We believe in Practical Approach than Theoretical Approach.

In ANVHAYA Technical Solutions Private Limited we change each penny of yours to figure up. Accompanied by a motive to furnish extreme branch of subject knowledge to every individual trainee of ours, ANVHAYA Technical Solutions Private Limited, a pioneer in the IT training sector higher up than a decade experienced in training and placements. Our team follows excessive dedication and motivation, we persuade quality of excellence and accomplished fulfillment in our training programs. ANVHAYA Technical Solutions Private Limited is the gateway to a blazing future in IT sector. We market leaders in IT training space and are pioneer's in introducing brand new courses, products and services.

Accompanied specialization in Data Science, numerous programs in future. We are always maintaining our ascendance by being the first to introduce brand new products and services in the industry. We are mainly specializing in career training in the area off software quality assurance. We make the trainee dream into reality. A N V H A Y A T e c h n i c a l S o l u t i o n s P r i v a t e L i m i t e d educational and professional development organization that has been founded to advance the software test and quality assurance profession by promoting and recognizing professionalism through education, consulting Services At ANVHAYA Technical Solutions

Private Limited you get:

- Experience Training of Excellence from our bureaucrats.
- Modernize persistently in Learning New Skills which presents You Beneficial Utility.
- Accessibility of a unique training in magnificent ambience.
- Obtainability of our faculty members for perpetual timing for assistance.
- Premier direction and prop up by our HR-Team members in placing our trainees in foremost organizations. Skill Mission Statement is to furnish a training quality of excellence, career orientation and to attain the goal of bagging opportunities in IT sector. ANVHAYA Technical Solutions Private Limited fulfills its Mission through a disciplined approach. It provides only real time services to its customers through its real time faculty. As Part of Its Mission and Vision ANVHAYA Technical Solutions Private Limited: - Vision is to bring new insights in your individual and organizational search & transform your career. Our mission is to provide the best services to all with value addition and cutting edge in your job search, thereby saving your precious time.
- It Promotes and Provides education of software testing professionals around the world.
- It creates a pool of qualified software testing professionals to meet the needs of testing organizations.
- It Provides assistance and guidance to members, both corporate and individuals, performing testing of all types of software systems.
- It provides a framework for assessing organizational testing practices and procedures.
- It Provides services, methods, and tools supporting the discipline of software testing.
- It Promotes and Provides education of software testing professionals around the world. It creates a pool of qualified software testing professionals to meet the needs of testing organizations. It Provides assistance and guidance to members, both corporate and individuals, performing testing of all types of software systems. It provides a framework for assessing organizational testing practices and procedures. It Provides services, methods, and tools supporting the discipline of software testing. Provide an open forum for discussing different testing issues.

CHAPTER 3: INTERNSHIP PART

PYTHON

Python is a general-purpose interpreted, interactive, object-oriented, and high-level programming language. An interpreted language, Python has a design philosophy that emphasizes code readability (notably using whitespace indentation to delimit code blocks rather than curly brackets or keywords), and a syntax that allows programmers to express concepts in fewer lines of code than might be used in languages such as C++or Java. It provides constructs that enable clear programming on both small and large scales. Python interpreters are available for many operating systems. Python, the reference implementation of Python, is open-source software and has a community-based development model, as do nearly all its variant implementations. Python is managed by the non-profit Python Software Foundation. Python features a dynamic type of system and automatic memory management. It supports multiple programming paradigms, including object-oriented, imperative, functional and procedural, and has a large and comprehensive standard library.

Python is a multi-paradigm programming language. Object-oriented programming and structured programming are fully supported, and many of its features support functional programming and aspect-oriented programming (including by Meta programming and meta-objects (magic methods)). Many other paradigms are supported via extensions, including design by contract and logic programming.

Python uses dynamic typing, and a combination of reference counting and a cycle-detecting garbage collector for memory management. It also features dynamic name resolution (late binding), which binds method and variable names during program execution.

ACTIVITY LOG FOR THE FIRST WEEK

Day & Date	Brief description of the daily activity	Learning Outcome	Person In-Charge Signature
Day – 1 09-Jan-2023	Introduction to HTML Basic structure of HTML HTML Editors	HTML basic structure	
Day – 2 10-Jan-2023	HTML Tags Paragraphs, Heading and Text	Inserting tags, text, headings in HTML	
Day – 3 11-Jan-2023	Formatting Tags HTML Lists	Using Lists in HTML	
Day – 4 12-Jan-2023	HTML example programs	Creating a table using HTML	
Day – 5 13-Jan-2023	HTML Forms HTML Media	Using of form tag and audio, video tag in HTML	
Day –6 17-Jan-2023	HTML format programming	Creating a basic front page using HTML	

WEEKLY REPORT

Week -1 (From: 10/01/23 To: 17/01/23)

This week, I focused on learning the basics of HTML, including tags, attributes, and document structure. Here are the key topics I covered:

Introduction to HTML: I started by learning what HTML is and how it's used to create web pages. I also learned about the basic structure of an HTML document, including the HTML, head, and body tags.

HTML Tags: I learned about the different types of HTML tags, including heading, paragraph, anchor, image, and list tags. I also learned how to use attributes to add additional information to tags.

Document Structure: I learned how to structure an HTML document properly, including using the doctype declaration, head section, and body section. I also learned how to add comments to HTML code.

Styling with CSS: I briefly touched on the concept of Cascading Style Sheets (CSS) and how it can be used to style HTML documents.

Overall, this week was a great introduction to HTML basics. I feel comfortable creating simple HTML pages with basic tags and attributes, and I look forward to continuing my learning journey with more advanced topics in the coming weeks. I plan to continue practicing by creating simple web pages and experimenting with different tags and attributes.

Objective of the Activity Done: HTML (Hyper Text Markup Language)

Detailed Report:

HTML: HYPER TEXT MARKUP LANGUAGE

LANGUAGES:

1. Programming Languages : To generate software's [we use a single programming language]

Examples : C,Cpp,Java.Net... etc

2. Scripting Languages : To create webpages [We use multiple scripting languages]

Examples : Html,Css,Javascript

Types of Webpages:

1. Static webpage : Content is not changed always [facebook or gmail registration form]

2. Dynamic webpage : Content is changed [After login gmail or facebook]

3. Server webpage : if the data is stored in database or loaded from database

Note:

A normal lang contains variables and functions where as html contains only tags

MARKUP LANGUAGE :

The Language which contains only tags

Tag is represented as angular brackets < >

Ex:

<x> : x tag

<upendra> : upendra tag

<aaigen> : aaigen tag

Types of markup languages:

1. HTML[Hyper Text Markup Language] : To create webpages[websites]
2. XML[Extensible Markup Language] : To store the data and forward

HTML :

To create webpages and web applications

It is tag based language [Internet language] or browser under stable language

Every tag contains opening and closing tag

Html was introduced by Tim berner's lee in 1991 and published in 1995.

Hyper text means it is a text which navigates to another document

Types of tags:

1. Container tag/pair tags : Having both opening and closing tags <html> </html>
2. Empty tag/single tags : Having only opening tags
 <hr> <img>

Structure Of Html Program:

```
<!DOCTYPE html> #standard tag
<html>
  <head>
     Title information, CSS, SCRIPT TAGS..etc
  </head>
  <body>  Body of the webpage  </body>
</html>
```

Heading Level Tags:

1. HTML headings are titles or subtitles that you want to display on a webpage.
2. headings are defined with the <h1> to <h6> tags
3. <h1> defines the most important heading. <h6> defines the least important heading.
4. It is important to use headings to show the document structure.

5. <h1> headings should be used for main headings, followed by <h2> headings, then the less important <h3>, and so on.

Headings.html

```
<!DOCTYPE html>
<html>
<body>
<h1>Heading 1</h1>
<h2>Heading 2</h2>
<h3>Heading 3</h3>
<h4>Heading 4</h4>
<h5>Heading 5</h5>
<h6>Heading 6</h6>
</body>
</html>
```

Bigger Headings:

Each HTML heading has a default size. However, you can specify the size for any heading with the style attribute, using the CSS font-size property:

```
<h1 style="font-size:60px;">Heading 1</h1>
```

Paragraphs:

- The HTML <p> element defines a paragraph.
- A paragraph always starts on a new line, and browsers automatically add some white space (a margin) before and after a paragraph.

paragraph1.html

```
<html>
 <head>  <title>Paragraph Tag</title>  </head>
 <body>
    <p>This is first paragraph</p>
   <p>This is second paragraph</p>
```

```
    <p>This is third paragraph</p>
 </body>
</html>
```

Line Breaks:

The HTML
 element defines a line break.

Use
 if you want a line break (a new line) without starting a new paragraph:

Breaktag.html:

```
<p>This is<br>a paragraph<br>with line breaks.</p>
```

Note: The
 tag is an empty tag, which means that it has no end tag.

Text Formatting tags:

HTML contains several elements for defining text with a special meaning.

Tag Description:

<b> Defines bold text

<i> Defines a part of text in an alternate voice or mood

<sub> Defines subscripted text

<sup> Defines superscripted text

<u> Under Line the text

HTML - Hypertext Reference (href) or Hyperlinks or Anchors

HTML Anchor

- The HTML anchor tag defines a hyperlink that links one page to another page. The "href" attribute is the most important attribute of the HTML a tag.

href attribute of HTML anchor tag

- The href attribute is used to define the address of the file to be linked. In other words, it points out the destination page

The syntax of HTML anchor tag is given below.

- Link Text
- **Note:** Hypertext references can be **Internal, Local, or Global.**
- **Internal** - Links to anchors on the current page

- **Local** - Links to other pages within your domain
- **Global** - Links to other domains outside of your site

Local.html

```
<a href="first.html" target="_blank">click here</a><br>
<a href="first.html" target="_self">click here</a>
```

External.html

```
<body>
<a href="https://www.google.com/" target="_blank">click here</a>
</body>
```

Images:

The <img> tag is used to embed an image in an HTML page.

- The <img> tag has two required attributes:
- src - Specifies the path to the image
- alt - Specifies an alternate text for the image, if the image for some reason cannot be displayed

Note: Also, always specify the width and height of an image. If width and height are not specified, the page might flicker while the image loads.

Images1.html:

```
<img src="smiley.gif" alt="Smiley face" width="42" height="42" style="border:5px solid black">
```

Images2. html:

```
<img src="smiley.gif" alt="Smileyface" width="42" height="42" style="vertical-align:bottom">
<img src="smiley.gif" alt="Smileyface" width="42" height="42" style="vertical-align:middle">
<img src="smiley.gif" alt="Smileyface" width="42" height="42" style="vertical-align:top">
<img src="smiley.gif" alt="Smileyface" width="42" height="42" style="float:right">
<img src="smiley.gif" alt="Smiley face" width="42" height="42" style="float:left">
```

HTML Lists:

HTML Lists are used to specify lists of information. All lists may contain one or more list elements.

HTML lists allow web developers to group a set of related items in lists

HTML provides three types of lists**:** Ordered List or Numbered List (ol)

Unordered List or Bulleted List (ul)

Description List or Definition List (dl)

Ordered list (or)Numbered List:

In **Ordered list, all** the list items are marked with numbers.

It is known as numbered list also.

The ordered list starts with **
** tag and ends with **
** and the list items start with **
** tag and **
**

A list can number from any value. The starting value is given by the "**start**" attribute.

**Syntax: <ol [type="1" | "a" | "A" | "I" | "i"] [start = "number"] > … **

Unordered list (or) Bulleted List:

- In Unordered list, all the list items are marked with bullets.
- It is also known as bulleted list also.
- The Unordered list starts with <ul> tag and ends with
- List items start with the <li> tag and .

Syntax:

**<ul[type="disc"|"square"|"circle"]>… **

Tables: HTML tables allow web developers to arrange data into rows and columns.

Define an HTML Table:

- The <table> tag defines an HTML table.
- Each table row is defined with a <tr> tag. Each table header is defined with a <th>tag. Each table data/cell is defined with a <td> tag
- By default, the text in <th> elements are bold and centered.
- By default, the text in <td> elements are regular and left-aligned.

Forms

An HTML form is used to collect user input. The user input is most often sent to a server for processing.

The <form> Element

The HTML <form> element is used to create an HTML form for user input: <form>

form elements

</form>

The <form> element is a container for different types of input elements, such as: text fields, checkboxes, radio buttons, submit buttons, etc.

Attribute describes the way to send it.

Following are **attributes of <form>:**

- **Name:**

The name attribute specifies the name of a form which is used to reference elements in a JavaScript. <form name="Form name">

- **Action:**

The required action attribute specifies where to send the form-data when a form is submitted. ***<form action="URL"> Value: URL***

- **Method:**

The method attribute specifies how to send form-data (the form-data is sent to the page specified in the action attribute). <form method="get|post">

Available Controls:

Controls are created using **<INPUT>** tag & **TYPE** attribute.

- **Form:**

<form> … </form> creates an HTML form, used to enclose HTML controls. The attributes are: Action: Gives the URL which will handle the form data. Method: Indicates a method or protocol for sending data to the target URL. The Get method is the default.

- **Text Fields:**

<input type="text"> creates a text field that the user can enter or edit text inside it. The attributes are:

  - size: Sets the size of the control.
  - value: The value entered in the text field.
  - maxlength: sets the maximum number of characters

enter in the text field.

- **Password:**

<input type="password"> creates a password text field, which shows asterisks on the text field. The attributes are: maxlength: sets the maximum length of the data in the control.

Name: gives the name of the control.

Value: represents the value entered in the text field.

- **Checkbox:**

<input type="checkbox"> creates a checkbox in a form. The attributes are:

Checked: Indicates if the checkbox should appear checked initially or not.

Size: sets the size of the checkbox.

Value: represents the result of the checkbox when clicked, which is passed to the forms action URL.

- **Radio buttons:**

  <input type="radio"> creates a radio button in the form. The attributes are:

Checked: Indicates if the checkbox should appear checked initially or not. Value: represents the result of the checkbox when clicked, which is passed to the forms action URL.

- **Selection lists:**

  <select> … </select> Creates a drop-down lists, where the user can select choice from the list of elements. And **<option>…..</option>** tag is used to provide values . The attributes are:

  - name: represents the name of the control.
  - value: the value of selected item.

- **Submit button:**

  <input type="submit"> creates a submit button that the user can click to send data in the form back to web server.

  The attributes are:

  Name: sets the name of the control.

  Value: Text to be displayed on the button.

- **Reset button:**

  <input type="reset"> creates a reset button in a form that resets all fields to their original values. The attributes are:

- Name: sets the name of the control.
- Value: Text to be displayed on the button.

9. **<Textarea> tag**

It is used to specify a text are or multi line text box. In a text area you can write an unlimited characters. It is mostly use in users feedback, home address etc.

Example:

<textarea cols="5 rows="5" name="Feedback" >

- cols: It is an attribute which specifies the number of columns in text area
- rows: It is an attribute which specifies the number of rows in text area
- name: It Specifies unique name for the input element

ACTIVITY LOG FOR THE SECOND WEEK

Day & Date	Brief description of the daily activity	Learning Outcome	Person In-Charge Signature
Day – 1 18-Jan-2023	Introduction to CSS Types of CSS	CSS basic structure	
Day – 2 19-Jan-2023	CSS Properties, Selectors and Values Applying CSS to HTML	Inserting CSS into HTML	
Day – 3 20-Jan-2023	CSS Colors CSS Box model, Margins	Creating a margins and color patterns	
Day – 4 23-Jan-2023	CSS Padding, Borders CSS Text & Font formats	Types of font and text formats, borders	
Day – 5 24-Jan-2023	CSS Advanced topics(Effects, Animations, Shadows)	Creating animations using CSS	
Day –6 25-Jan-2023	CSS format programming	Creating a basic front page using HTML & CSS	

WEEKLY REPORT

Week -2 (From: 18/01/23 To: 25/01/23)

This week, I focused on learning the basics of Cascading Style Sheets (CSS) and how it can be used to style HTML documents. Here are the key topics I covered:

Introduction to CSS: I started by learning what CSS is and how it's used to add style and design to HTML pages. I also learned about the basic syntax of CSS, including selectors, properties, and values.

CSS Selectors: I learned about the different types of CSS selectors, including element, class, and ID selectors. I also learned how to use combination selectors to target specific elements on a page.

CSS Properties: I learned about the different types of CSS properties, including color, font, text, background, and border properties. I also learned how to use shorthand properties to simplify code.

Layout and Positioning: I learned how to use CSS to control the layout and positioning of elements on a page, including using float, clear, and positioning properties.

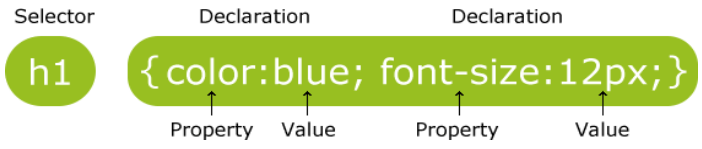
Overall, this week was a great introduction to CSS basics. I feel comfortable creating simple styles for HTML pages and understand how to use selectors and properties to target specific elements. I plan to continue practicing by creating more complex layouts and experimenting with different CSS techniques.

CSS

- CSS stands for Cascading Style Sheets
- Styles define how to display HTML elements
- CSS adds new look to html tags
- External Style Sheets can save a lot of work
- Styles are normally saved in external .css files.

CSS Syntax

A CSS rule has two main parts: a selector, and one or more declarations:



Types of CSS

- Inline CSS - Style Attribute
- Internal/Embedded CSS - <STYLE> tag
- External CSS - <LINK> tag

Inline CSS:

- An inline style can be used to apply a unique style to one single occurrence

of an element. And the style can be defined within the basic HTML tag.

- To use inline styles, use the **style** attribute in the relevant tag. The style attribute can contain any CSS property.

Internal/Embedded CSS

- This type of CSS is only for Single Page.
- When using internal CSS, we must add a new tag, <style>, inside the <head> tag.

External Style Sheet

- When using CSS it is preferable to keep the CSS separate from your HTML.
- Placing CSS in a separate file allows the web designer to completely differentiate between content (HTML) and design (CSS).
- External CSS is a file that contains only CSS code and is saved with a ".css" file extension.
- This CSS file is then referenced in your HTML using the <link> instead of <style>.

Division tag:

tag:

The tag defines a division or a section in an HTML document.

The tag is used as a container for HTML elements - which is then styled with CSS or manipulated with JavaScript.

The tag is easily styled by using the class or id attribute. Any sort of content can be put inside the tag!

ACTIVITY LOG FOR THE THIRD WEEK

Day & Date	Brief description of the daily activity	Learning Outcome	Person In-Charge Signature
Day – 1 26-Jan-2023	HOLIDAY		
Day – 2 27-Jan-2023	Displaying Output in JavaScript Understanding JavaScript Syntax	Output displaying	
Day – 3 28-Jan-2023	Variables & Datatypes Operators	Types of operators and datatypes	
Day – 4 30-Jan-2023	Math and String Manipulations Conditional and looping Statements	Types of statements	
Day – 5 31-Jan-2023	Functions Validations	Importance of funtions and validations	
Day –6 01-Feb-2023	JavaScript programming	Creating a webpage using HTML, CSS, JavaScript	

WEEKLY REPORT

Week -3 (From: 26/01/23 To: 01/02/23)

This week, I focused on learning the basics of JavaScript, including variables, data types, functions, and control structures. Here are the key topics I covered:

Introduction to JavaScript: I started by learning what JavaScript is and how it's used to add interactivity and functionality to web pages. I also learned about the basic syntax of JavaScript, including variables and data types.

Functions: I learned how to define and call functions in JavaScript, including how to pass parameters and return values. I also learned about function scope and closures.

Control Structures: I learned about the different types of control structures in JavaScript, including conditional statements (if/else) and loops (for/while).

Arrays: I learned how to create and manipulate arrays in JavaScript, including how to access and modify array elements, and how to use array methods like push, pop, and splice.

Objects: I learned about objects in JavaScript and how to create and manipulate object properties and methods.

Overall, this week was a great introduction to JavaScript basics. I feel comfortable working with variables, functions, control structures, and data structures like arrays and objects. I plan to continue practicing by creating more complex JavaScript programs and exploring different libraries and frameworks.

Javascript:

JavaScript (js) is a light-weight object-oriented programming language which is used by several websites for scripting the webpages. It is an interpreted, full-fledged programming language that enables dynamic interactivity on websites when applied to an HTML document. It was introduced in the year 1995 for adding programs to the webpages in the Netscape Navigator browser. Since then, it has been adopted by all other graphical web browsers. With JavaScript, users can build modern web applications to interact directly without reloading the page every time. The traditional website uses js to provide several forms of interactivity and simplicity.

Although, JavaScript has no connectivity with Java programming language. The name was suggested and provided in the times when Java was gaining popularity in the market. In addition to web browsers, databases such as CouchDB and MongoDB uses JavaScript as their scripting and query language.

Features of JavaScript

There are following features of JavaScript:

1. All popular web browsers support JavaScript as they provide built-in execution environments.
2. JavaScript follows the syntax and structure of the C programming language. Thus, it is a structured programming language.
3. JavaScript is a weakly typed language, where certain types are implicitly cast (depending on the operation).

4. JavaScript is an object-oriented programming language that uses prototypes rather than using classes for inheritance.
5. It is a light-weighted and interpreted language.
6. It is a case-sensitive language.
7. JavaScript is supportable in several operating systems including, Windows, macOS, etc.
8. It provides good control to the users over the web browsers.

JavaScript Variable

1. JavaScript variable
2. JavaScript Local variable
3. JavaScript Global variable

A **JavaScript variable** is simply a name of storage location. There are two types of variables in JavaScript : local variable and global variable.
There are some rules while declaring a JavaScript variable (also known as identifiers).

1. Name must start with a letter (a to z or A to Z), underscore(_), or dollar($) sign.
2. After first letter we can use digits (0 to 9), for example value1.
3. JavaScript variables are case sensitive, for example x and X are different variables.

Example of JavaScript variable

```
<script>
var x = 10;
var y = 20;
var z=x+y;
document.write(z);
</script>
```

Java script Data Types

JavaScript provides different **data types** to hold different types of values. There are two types of data types in JavaScript.

1. Primitive data type
2. Non-primitive (reference) data type

JavaScript is a **dynamic type language**, means you don't need to specify type of the variable because it is dynamically used by JavaScript engine. You need to use **var** here to specify the data type. It can hold any type of values such as numbers, strings etc. For example:

```
var a=40;//holding  number
var b="Rahul";//holding string
```

JavaScript primitive data types

There are five types of primitive data types in JavaScript. They are as follows:

Data Type	Description
String	represents sequence of characters e.g. "hello"
Number	represents numeric values e.g. 100
Boolean	represents Boolean value either false or true
Undefined	represents undefined value
Null	represents null i.e. no value at all

JavaScript non-primitive data types

The non-primitive data types are as follows:

Data Type	Description
Object	represents instance through which we can access members
Array	represents group of similar values
Reg.Exp	represents regular expression

JavaScript Operators

JavaScript operators are symbols that are used to perform operations on operands. For example:

```
var sum=10+20;
```

Here, + is the arithmetic operator and = is the assignment operator.

There are following types of operators in JavaScript.

1. Arithmetic Operators
2. Comparison (Relational) Operators
3. Bitwise Operators
4. Logical Operators
5. Assignment Operators
6. Special Operators

JavaScript Arithmetic Operators

Arithmetic operators are used to perform arithmetic operations on the operands. The following operators are known as JavaScript arithmetic operators.

Operator	Description	Example
+	Addition	10+20 = 30
-	Subtraction	20-10 = 10
*	Multiplication	10*20 = 200
/	Division	20/10 = 2
%	Modulus (Remainder)	20%10 = 0
++	Increment	var a=10; a++; Now a = 11
--	Decrement	var a=10; a--; Now a = 9

JavaScript Comparison Operators

The JavaScript comparison operator compares the two operands. The comparison operators are as follows:

Operator	Description	Example
==	Is equal to	10==20 = false
===	Identical (equal and of same type)	10==20 = false
!=	Not equal to	10!=20 = true
!==	Not Identical	20!==20 = false
>	Greater than	20>10 = true
>=	Greater than or equal to	20>=10 = true
<	Less than	20<10 = false
<=	Less than or equal to	20<=10 = false

JavaScript Bitwise Operators

The bitwise operators perform bitwise operations on operands. The bitwise operators are as follows:

Operator	Description	Example
&	Bitwise AND	(10==20 & 20==33) = false
|	Bitwise OR	(10==20 | 20==33) = false
^	Bitwise XOR	(10==20 ^ 20==33) = false
~	Bitwise NOT	(~10) = -10
<<	Bitwise Left Shift	(10<<2) = 40
>>	Bitwise Right Shift	(10>>2) = 2
>>>	Bitwise Right Shift with Zero	(10>>>2) = 2

JavaScript Logical Operators

The following operators are known as JavaScript logical operators.

Operator	Description	Example
&&	Logical AND	(10==20 && 20==33) = false
||	Logical OR	(10==20 || 20==33) = false
!	Logical Not	!(10==20) = true

JavaScript Assignment Operators

The following operators are known as JavaScript assignment operators.

Operator	Description	Example
=	Assign	10+10 = 20
+=	Add and assign	var a=10; a+=20; Now a = 30
-=	Subtract and assign	var a=20; a-=10; Now a = 10
*=	Multiply and assign	var a=10; a*=20; Now a = 200
/=	Divide and assign	var a=10; a/=2; Now a = 5
%=	Modulus and assign	var a=10; a%=2; Now a = 0

JavaScript Special Operators

The following operators are known as JavaScript special operators.

Operator	Description
(?:)	Conditional Operator returns value based on the condition. It is like if-else.
,	Comma Operator allows multiple expressions to be evaluated as single statement.
delete	Delete Operator deletes a property from the object.
in	In Operator checks if object has the given property
instanceof	checks if the object is an instance of given type
new	creates an instance (object)
typeof	checks the type of object.
void	it discards the expression's return value.
yield	checks what is returned in a generator by the generator's iterator.

JavaScript Functions

JavaScript functions are used to perform operations. We can call JavaScript function many times to reuse the code.

Advantage of JavaScript function

There are mainly two advantages of JavaScript functions.

1. **Code reusability**: We can call a function several times so it save coding.
2. **Less coding**: It makes our program compact. We don't need to write many lines of code each time to perform a common task.

JavaScript Function Syntax

The syntax of declaring function is given below.

```
function functionName([arg1, arg2, ...argN]){
 //code to be executed
}
```

ACTIVITY LOG FOR THE FOURTH WEEK

Day & Date	Brief description of the daily activity	Learning Outcome	Person In-Charge Signature
Day – 1 02-Feb-2023	bootstrap introduction and Containers	Basics of bootstrap	
Day – 2 03-Feb-2023	bootstrap setup	Using of bootstrap	
Day – 3 04-Feb-2023	bootstrap button and table styles	Using of button and table styles	
Day – 4 06-Feb-2023	bootstrap navigations and alerts	Creating alert messages in the webpage	
Day – 5 07-Feb-2023	Bootstrap Forms	Creating bootstrap forms	
Day –6 08-Feb-2023	Bootstrap carousel	Creating bootstrap carousel	

WEEKLY REPORT

Week -4 (From: 02/02/23 To: 08/02/23)

This week, I focused on learning the basics of Bootstrap, a popular front-end framework for building responsive and mobile-first websites. Here are the key topics I covered:

Introduction to Bootstrap: I started by learning what Bootstrap is and how it's used to streamline the development of websites. I also learned about the basic components of Bootstrap, including the grid system, typography, and forms.

Grid System: I learned how to use the Bootstrap grid system to create responsive layouts for different screen sizes, including how to define columns and rows and how to use breakpoints.

Typography: I learned how to use the Bootstrap typography classes to style headings, paragraphs, and other text elements on a page.

Forms: I learned how to use the Bootstrap form classes to create input fields, buttons, and other form elements, and how to validate form data using JavaScript.

Navigation: I learned how to use the Bootstrap navigation components to create responsive menus, including the navbar and dropdown menus.

Overall, this week was a great introduction to Bootstrap basics. I feel comfortable using the grid system to create responsive layouts and adding Bootstrap components to my HTML pages. I plan to continue practicing by building more complex websites and exploring different Bootstrap components and features.

Objectives of the Activity Done: Bootstrap

Detailed Report:

Bootstrap is the popular HTML, CSS and JavaScript framework for developing a responsive and mobile friendly website.

Our Bootstrap tutorial includes all topics of Bootstrap such as jumbotron, table, button, grid, form, image, alert, wells, container, carousel, panels, glyphicon, badges, labels, progress bar, pagination, pager, list group, dropdown, collapse, tabs, pills, navbar, inputs, modals, tooltip, popover and scroll spy.

Bootstrap Container

In Bootstrap, container is used to set the content's margins dealing with the responsive behaviours of your layout. It contains the row elements and the row elements are the container of columns (known as grid system).

The **container class** is used to create boxed content.

There are two container classes in Bootstrap:

1. container
2. container-fluid

See the basic layout of a container:

```
<div class="container">
 <div class="row">
  <div class="col-md-xx"></div>
    ...
 </div>
 <div class="row">
  <div class="col-md-xx"></div>
    ...
 </div>
</div>
```

Bootstrap Buttons

There are seven styles to add a button in Bootstrap. Use the following classes to achieve the different button styles:

- .btn-default
- .btn-primary
- .btn-success
- .btn-info
- .btn-warning
- .btn-danger
- .btn-link

Bootstrap Button Example: specifying seven styles

```
<!DOCTYPE html>
<html lang="en">
 <head>
  <title>Job</title>
<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/css/bootstrap.min.css"/>
 </head>
 <body>
  <h1>Button Example!</h1>
<button class="btn btn-default">default</button>
<button class="btn btn-primary">primary</button>
<button class="btn btn-danger">danger</button>
<button class="btn btn-success">success</button>
```

```
<button class="btn btn-info">info</button>
<button class="btn btn-warning">warning</button>
<button class="btn btn-link">Link</button>

<script src="https://ajax.googleapis.com/ajax/libs/jquery/1.11.3/jquery.min.js"></script>
  <script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.5/js/bootstrap.min.js"></script>
  </body>
</html>
```

Bootstrap Tables

We can create different types of Bootstrap tables by using different classes to style them.

Bootstrap Basic Table:

The basic Bootstrap table has a light padding and only horizontal dividers. The **.table class** is used to add basic styling to a table.

Example:

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <title>Job</title>
    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/css/bootstrap.min.cs
s"/>
  </head>
  <body>

<div class="container">
  <h1>Basic Table Example</h1>

<table class="table">
  <tr><th>Id</th><th>Name</th><th>Age</th></tr>
  <tr><td>101</td><td>Rahul</td><td>23</td></tr>
  <tr><td>102</td><td>Umesh</td><td>22</td></tr>
  <tr><td>103</td><td>Max</td><td>29</td></tr>
  <tr><td>104</td><td>Ajeet</td><td>21</td></tr>
</table>
</div>
```

```
<script src="https://ajax.googleapis.com/ajax/libs/jquery/1.11.3/jquery.min.js"></script>
  <script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.5/js/bootstrap.min.js"></script>
  </body>
</html>
```

Bootstrap Forms

In Bootstrap, there are three types of form layouts:

- Vertical form (this is default)
- Horizontal form
- Inline form

Bootstrap Form Rules

There are three standard rules for these 3 form layouts:

- Always use <form role="form"> (helps improve accessibility for people using screen readers)
- Wrap labels and form controls in (needed for optimum spacing)
- Add class .form-control to all textual <input>, <textarea>, and <select> elements

Bootstrap Alerts

Bootstrap Alerts are used to provide an easy way to create predefined alert messages. Alert adds a style to your messages to make it more appealing to the users.

There are four classes that are used within element for alerts.

- .alert-success
- .alert-info
- .alert-warning
- .alert-danger

Bootstrap Alert Example

```
  <!DOCTYPE html>
<html lang="en">
<head>
  <title>Bootstrap Example</title>
  <meta charset="utf-8">
```

```
    <meta name="viewport" content="width=device-width, initial-scale=1">
<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/css/bootstrap.min.css">
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/1.12.0/jquery.min.js"></script>
    <script  src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/js/bootstrap.min.js"></script>
  </head>
  <body>
  <div class="container">
    <h2>Alerts</h2>
    <div class="alert alert-success">
      <strong>Success!</strong> This alert box indicates a successful or positive action.
    </div>
    <div class="alert alert-info">
      <strong>Info!</strong> This alert box indicates a neutral informative change or action.
    </div>
    <div class="alert alert-warning">
      <strong>Warning!</strong> This alert box indicates a warning that might need attention.
    </div>
    <div class="alert alert-danger">
      <strong>Danger!</strong> This alert box indicates a dangerous or potentially negative action.
  </div>
  </div>

  </body>
  </html>
```

ACTIVITY LOG FOR THE FIFTH WEEK

Day & Date	Brief description of the daily activity	Learning Outcome	Person In-Charge Signature
Day – 1 09-Feb-2023	JavaScript Revision for newly joined students	JavaScript revision	
Day – 2 10-Feb-2023	JavaScript Revision for newly joined students	JavaScript revision	
Day – 3 11-Feb-2023	Angular Introduction and Structure of angular	Introduction of Angular JS and structure	
Day – 4 13-Feb-2023	Environment Setup	Angular environment setup	
Day – 5 14-Feb-2023	Installing Angular CLI Directory Structure of Angular	Installation process	
Day –6 15-Feb-2023	Components and Modules in angular	Angular JS modules	

WEEKLY REPORT

Week -5 (From: 09/02/23 To: 15/02/23)

This week, I focused on learning the basics of AngularJS, a JavaScript-based front-end framework for building dynamic and single-page applications. Here are the key topics I covered:

Introduction to AngularJS: I started by learning what AngularJS is and how it's used to simplify the development of complex web applications. I also learned about the basic building blocks of AngularJS, including modules, controllers, and directives.

Data Binding: I learned how to use the two-way data binding feature of AngularJS to synchronize the data between the model and view, and how to use expressions and filters to manipulate data in the view.

Directives: I learned how to use AngularJS directives to add custom behavior and functionality to HTML elements, including how to create custom directives and how to use built-in directives like ng-repeat and ng-if.

Overall, this week was a great introduction to AngularJS basics. I feel comfortable creating simple applications using modules, controllers, and directives, and I understand the basics of data binding, services, and routing. I plan to continue practicing by building more complex applications and exploring different AngularJS features and libraries.

Objectives of the Activity Done: Angular JS

Detailed Report:

Angular JS is an open source JavaScript framework that is used to build web applications. It can be freely used, changed and shared by anyone.

Angular Js is developed by Google.

It is an excellent framework for building single phase applications and line of business applications.

MVC stands for Model View Controller. It is a software design pattern for developing web applications. It is very popular because it isolates the application logic from the user interface layer and supports separation of concerns.

The MVC pattern is made up of the following three parts:

1. **Model:** It is responsible for managing application data. It responds to the requests from view and to the instructions from controller to update itself.
2. **View:** It is responsible for displaying all data or only a portion of data to the users. It also specifies the data in a particular format triggered by the controller's decision to present the data. They are script-based template systems such as JSP, ASP, PHP and very easy to integrate with AJAX technology.
3. **Controller:** It is responsible to control the relation between models and views. It responds to user input and performs interactions on the data model objects. The controller receives input, validates it, and then performs business operations that modify the state of the data model.

Angular JS Module

In Angular JS, a module defines an application. It is a container for the different parts of your application like controller, services, filters, directives etc.

A module is used as a Main() method. Controller always belongs to a module.

How to create a module

The angular object's module() method is used to create a module. It is also called AngularJS function angular.module

```
<div ng-app="myApp">...</div>
<script>
var app = angular.module("myApp", []);
</script>
```

Here, "myApp" specifies an HTML element in which the application will run.

Now we can add controllers, directives, filters, and more, to AngularJS application.

How to add controller to a module

If you want to add a controller to your application refer to the controller with the ng-controller directive.

See this example:

```
<!DOCTYPE html>
<html>
<script src="http://ajax.googleapis.com/ajax/libs/angularjs/1.4.8/angular.min.js"></script>
<body>
<div ng-app="myApp" ng-controller="myCtrl">
{{ firstName + " " + lastName }}
</div>
<script>
var app = angular.module("myApp", []);
app.controller("myCtrl", function($scope) {
    $scope.firstName = "Ajeet";
    $scope.lastName = "Maurya";
});
</script>
</body>
</html>
```

Angular Components

Components are the key features of Angular. The whole application is built by using different components.

The core idea behind Angular is to build components. They make your complex application into reusable parts which you can reuse very easily.

How to create a new component?

Open WebStorm>> Go to your project source folder>> Expand the app directory and create a new directory named "server".

Now, create the component within server directory. Right click on the server directory and create a new file named as "server.component.ts". It is the newly created component.

Components are used to build webpages in Angular but they require modules to bundle them together. Now, you have to register our new components in module.

Creating component with CLI

Syntax

1. ng generate component component_name
2. Or
3. ng g c component_name

Let's see how to create a new component by using command line.

Open Command prompt and stop **ng serve** command if it is running on the browser.

Type **ng generate component server2** to create a new component named server2.

You can also use a shortcut **ng g c server2** to do the same task.

ACTIVITY LOG FOR THE SIXTH WEEK

Day & Date	Brief description of the daily activity	Learning Outcome	Person In-Charge Signature
Day – 1 16-Feb-2023	Data Binding and Types [Interpolation,Property binding]	Types of binding techniques	
Day – 2 17-Feb-2023	Angular JS Expressions	Introduction of Angular JS	
Day – 3 18-Feb-2023	HOLIDAY		
Day – 4 20-Feb-2023	Class, style ,event and two way binding	Explanation of binding types	
Day – 5 21-Feb-2023	Directives and types in angular	Angular JS directives and its types	
Day –6 22-feb-2023	Pipes and Different types of pipes	Pipes and its types	

WEEKLY REPORT

Week -6 (From: 16/02/23 To: 22/02/23)

This week, I focused on learning the basics of AngularJS, a JavaScript-based front-end framework for building dynamic and single-page applications. About bindings:

Objectives of the Activity Done:

Detailed Report:

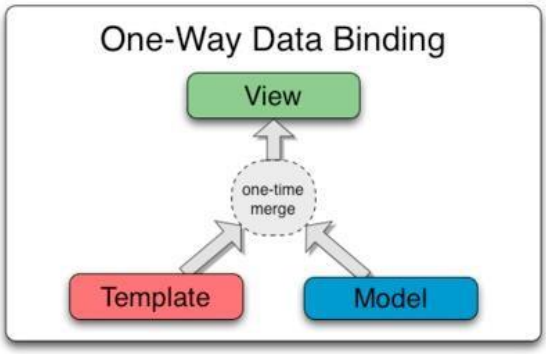
Data Binding

Data binding is a very useful and powerful feature used in software development technologies. It acts as a bridge between the view and business logic of the application.

Angular JS follows Two-Way data binding model.

One-Way Data Binding

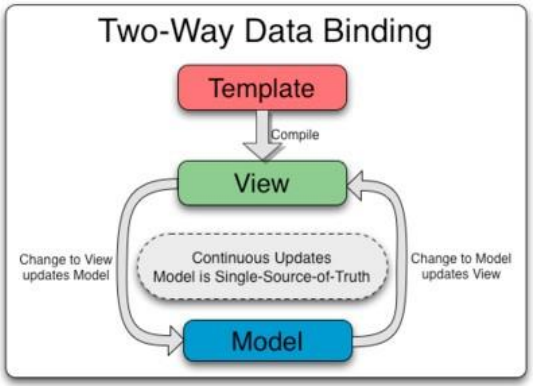
The one-way data binding is an approach where a value is taken from the data model and inserted into an HTML element. There is no way to update model from view. It is used in classical template systems. These systems bind data in only one direction.



Two-Way Data Binding

Data-binding in Angular apps is the automatic synchronization of data between the model and view components.

Data binding lets you treat the model as the single-source-of-truth in your application. The view is a projection of the model at all times. If the model is changed, the view reflects the change and vice versa.



```
<!DOCTYPE html>
<html>
<script src="http://ajax.googleapis.com/ajax/libs/angularjs/1.4.8/angular.min.js"></script>
<body>
<div ng-app="" ng-init="firstName='Ajeet'">
<p>Input something in the input box:</p>
<p>Name: <input type="text" ng-model="firstName"></p>
<p>You wrote: {{ firstName }}</p>
</div>
</body>
</html>
```

Angular JS Directives

Angular JS facilitates you to extend HTML with new attributes. These attributes are called directives.

There is a set of built-in directive in Angular JS which offers functionality to your applications. You can also define your own directives.

Directives are special attributes starting with ng- prefix. Following are the most common directives:

- ng-app: This directive starts an Angular JS Application.
- ng-init: This directive initializes application data.
- ng-model: This directive defines the model that is variable to be used in Angular JS.
- ng-repeat: This directive repeats html elements for each item in a collection.

ng-app directive

ng-app directive defines the root element. It starts an AngularJS Application and automatically initializes or bootstraps the application when web page containing AngularJS Application is loaded. It is also used to load various AngularJS modules in AngularJS Application.

See this example:

In following example, we've defined a default AngularJS application using ng-app attribute of a div element.

```
<div ng-app = "">
…….
</div>
```

ng-init directive

ng-init directive initializes an AngularJS Application data. It defines the initial values for an AngularJS application.

In following example, we'll initialize an array of countries. We're using JSON syntax to define array of countries.

```
<div ng-app = "" ng-init = "countries = [{locale:'en-IND',name:'India'}, {locale:'en-
PAK',name:'Pakistan'}, {locale:'en-AUS',name:'Australia'}]">
   ...
</div>
```

ng-model directive:

ng-model directive defines the model/variable to be used in AngularJS Application.

In following example, we've defined a model named "name".

```
<div ng-app = "">
   ...
   <p>Enter your Name: <input type = "text" ng-model = "name"></p>
</div>
```

ng-repeat directive

ng-repeat directive repeats html elements for each item in a collection. In following example, we've iterated over array of countries.

```
<div ng-app = "">
...
<p>List of Countries with locale:</p>

<ol>
<li ng-repeat = "country in countries">
{{ 'Country: ' + country.name + ', Locale: ' + country.locale }}
</li>
</ol>
```

AngularJS directives Example

Let's take an example to use all the above discussed directives:

```
<!DOCTYPE html>
```

```
<html>
<head>
    <title>AngularJS Directives</title>
</head>
<body>
    <h1>Sample Application</h1>

    <div ng-app = "" ng-init = "countries = [{locale:'en-IND',name:'India'}, {locale:'en-
PAK',name:'Pakistan'}, {locale:'en-AUS',name:'Australia'}]">
      <p>Enter your Name: <input type = "text" ng-model = "name"></p>
      <p>Hello <span ng-bind = "name"></span>!</p>
      <p>List of Countries with locale:</p>

      <ol>
        <li ng-repeat = "country in countries">
          {{ 'Country: ' + country.name + ', Locale: ' + country.locale }}
        </li>
      </ol>
    </div>
<script src = "http://ajax.googleapis.com/ajax/libs/angularjs/1.3.14/angular.min.js"></script>
</body>
</html>
```

Angular Pipes

In Angular 1, filters are used which are later called Pipes onwards Angular2. In Angular 7, it is known as pipe and used to transform data. It is denoted by symbol |

Syntax:

1. {{title | uppercase}}

Pipe takes integers, strings, arrays, and date as input separated with |. It transforms the data in the format as required and displays the same in the browser.

Let's see an example using pipes. Here, we display the title text in upper and lower case by using pipes.

Example:

Define a variable named "title" in component.ts file.

```
import { Component } from '@angular/core';
```

```
@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {
  title = 'my-first-app';
}
```

Use the pipe symbol in component.html file:

```
<h1>
   {{ title | uppercase }} <br/></h1>
<h1>
  {{ title | lowercase }} <br/></h1>
```

Output:

Run ng serve and see the result. You will see the following result.

Here, you can see that pipes have changed the tittle in upper and lowercase.

Angular 7 Built-in Pipes

Angular 7 provides some built-in pipes:

- Lowercasepipe
- Uppercasepipe
- Datepipe
- Currencypipe
- Jsonpipe
- Percentpipe
- Decimalpipe
- Slicepipe

ACTIVITY LOG FOR THE SEVENTH WEEK

Day & Date	Brief description of the daily activity	Learning Outcome	Person In-Charge Signature
Day – 1 23-Feb-2023	Forms Introduction and Types of Forms[Template driven forms and project class 1]	Introduction of forms in angular	
Day – 2 24-Feb-2023	Routings in angular	Routing techniques	
Day – 3 25-Feb-2023	Python Introduction	Introduction of Python	
Day – 4 27-Feb-2023	Python Variables, Python Operators	Introduction of Python variables, Types of operators and its functions	
Day – 5 28-Feb-2023	HOLIDAY		
Day –6 01-Mar-2023	HOLIDAY		

WEEKLY REPORT

Week -7 (From: 23/02/23 To: 01/03/23)

Objectives of the Activity Done:

This week, I focused on learning the basics of AngularJS Forms, a powerful feature of the AngularJS framework for handling user input and validation. Here are the key topics I covered:

Introduction to AngularJS Forms: I started by learning what AngularJS Forms are and how they differ from traditional HTML forms. I also learned about the basic components of AngularJS Forms, including the ng-model directive and form controls.

Form Controls: I learned how to use different types of form controls in AngularJS, including text input, radio buttons, checkboxes, select boxes, and more. I also learned about form control attributes, such as ng-required and ng-minlength.

Form Validation: I learned how to use AngularJS Form validation to ensure that user input is valid and meets certain criteria, including how to use built-in validation directives like ng-required, ng-minlength, and ng-pattern.

Custom Validation: I learned how to create custom validation functions in AngularJS Forms, including how to use the $validators and $asyncValidators APIs.

Form Submission: I learned how to handle form submission in AngularJS, including how to use the ng-submit directive and how to handle form submission errors.

Overall, this week was a great introduction to AngularJS Forms basics. I feel comfortable creating and validating forms using AngularJS, and I understand the basics of form submission and error handling. I plan to continue practicing by building more complex forms and exploring different AngularJS features and libraries.

Detailed Report:

AngularJS Forms

AngularJS facilitates you to create a form enriches with data binding and validation of input controls.

Input controls are ways for a user to enter data. A form is a collection of controls for the purpose of grouping related controls together.

Following are the input controls used in AngularJS forms:

- input elements
- select elements
- button elements
- textarea elements

AngularJS provides multiple events that can be associated with the HTML controls. These events are associated with the different HTML input elements.

Following is a list of events supported in AngularJS:

- ng-click
- ng-dbl-click

- ng-mousedown
- ng-mouseup
- ng-mouseenter
- ng-mouseleave
- ng-mousemove
- ng-mouseover
- ng-keydown
- ng-keyup
- ng-keypress
- ng-change

Angular Routing

Navigation is an important aspect of web applications. A single-page application (**SPA**) does not have multiple-page concepts, and it moves from one view (expense list) to another view.

It provides clear and understandable navigation elements decide the success of an application.

Angular provides a comprehensive set of navigation features to accommodate simple scenarios in a complex environment.

The process of defining the navigation element and associated view is called the routing in Angular. Angular provides a separate module, the Router module, for setting up navigation in an Angular application.

Configure Routing

Angular CLI provides full support for setting up routing during the application build process and working on an application. Let's create a new application with router enabled using the command below -

```
ng new routing-app
```

Angular CLI generate a new module, AppRoutingModule for routing purpose. The code is below -

```
import { NgModule } from '@angular/core';
import { Routes, RouterModule } from '@angular/router';
const routes: Routes = [];
@NgModule({
  imports: [RouterModule.forRoot(routes)],
  exports: [RouterModule]
})
export class AppRoutingModule { }
```

Router module and route **@angular/router package.**

RouterModel provides functionality to configure and perform routing in applications.

The route is the type used to set navigation rules.

- Routes are local variables (of type Route) used to configure the actual navigation rules of the application.
- The RouterMoudle.forRoot() method will set up the navigation rules configured in the route variable.

In Angular CLI, the AppComponent includes the generated AppRouting module as mention below -

```
import { BrowserModule } from '@angular/platform browser;
import { NgModule } from '@angular/core';
import { AppRoutingModule } from './app-routing.module';
import { AppComponent } from './app.component';
@NgModule({
  declarations: [
    AppComponent
  ],
  imports: [
    BrowserModule,
    AppRoutingModule
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

Here,

The **AppRoutingModule** imports the module using the AppComponent import metadata.

Angular CLI provides an option to set routing in the existing application.

1. ng generate module my-module --routing

It will generate a new module with routing features enabled. To enable routing feature in an existing module (AppModule), we need to include an additional option as mentioned below -

1. ng generate module app-routing --module app --flat

Here,

-module app configures the routing module in the AppModule module.

Open a command prompt.

1. cd /go/to/expense-manager
2. Generate routing module using below command -
3. ng generate module app-routing --module app --flat

Output:

```
CREATE src/app/app-routing.module.ts (196 bytes)
UPDATE src/app/app.module.ts (785 bytes)
```

Here,

CLI generate **AppRoutingModule** and configures it in **AppModule.**

Creating Routes

The information to create a route is below -

```
const routes: Routes = [
  { path: 'about', component: AboutComponent },
];
```

Here,

Routes are the variable in the **AppRoutingModule.**

When a user requests http://localhost:4200/about url, the Path matches the about rule, and then **AboutComponent** will be called.

Python

Python is a general purpose, dynamic, high-level, and interpreted programming language. It supports Object Oriented programming approach to develop applications. It is simple and easy to learn and provides lots of high-level data structures.

Python is easy to learn yet powerful and versatile scripting language, which makes it attractive for Application Development.

Python's syntax and dynamic typing with its interpreted nature make it an ideal language for scripting and rapid application development.

Python supports multiple programming pattern, including object-oriented, imperative, and functional or procedural programming styles.

Python is not intended to work in a particular area, such as web programming. That is why it is known as multipurpose programming language because it can be used with web, enterprise, 3D CAD, etc.

Python makes the development and debugging *fast* because there is no compilation step included in Python development, and edit-test-debug cycle is very fast.

Python Variables

Variable is a name that is used to refer to memory location. Python variable is also known as an identifier and used to hold value.

In Python, we don't need to specify the type of variable because Python is a infer language and smart enough to get variable type.

Variable names can be a group of both the letters and digits, but they have to begin with a letter or an underscore.

It is recommended to use lowercase letters for the variable name. Rahul and rahul both are two different variables.

Identifier Naming

Variables are the example of identifiers. An Identifier is used to identify the literals used in the program. The rules to name an identifier are given below.

- The first character of the variable must be an alphabet or underscore (_).
- All the characters except the first character may be an alphabet of lower-case(a-z), upper-case (A-Z), underscore, or digit (0-9).
- Identifier name must not contain any white-space, or special character (!, @, #, %, ^, &, *).
- Identifier name must not be similar to any keyword defined in the language.
- Identifier names are case sensitive; for example, my name, and MyName is not the same.
- Examples of valid identifiers: a123, _n, n_9, etc.
- Examples of invalid identifiers: 1a, n%4, n 9, etc.

Declaring Variable and Assigning Values

Python does not bind us to declare a variable before using it in the application. It allows us to create a variable at the required time.

We don't need to declare explicitly variable in Python. When we assign any value to the variable, that variable is declared automatically.

The equal (=) operator is used to assign value to a variable.

Object Identity

In Python, every created object identifies uniquely in Python. Python provides the guaranteed that no two objects will have the same identifier. The built-in **id()** function, is used to identify the object identifier. Consider the following example.

```
a = 50
b = a
print(id(a))
print(id(b))
# Reassigned variable a
a = 500
print(id(a))
```

Output:

```
140734982691168
140734982691168
2822056960944
```

We assigned the **b = a, a** and **b** both point to the same object. When we checked by the **id()** function it returned the same number. We reassign **a** to 500; then it referred to the new object identifier.

Variable Names

We have already discussed how to declare the valid variable. Variable names can be any length can have uppercase, lowercase (A to Z, a to z), the digit (0-9), and underscore character(_). Consider the following example of valid variables names.

```
name = "Devansh"
age = 20
marks = 80.50

print(name)
print(age)
print(marks)
```

Output:

```
Devansh
20
80.5
```

Consider the following valid variables name.

```
name = "A"
Name = "B"
naMe = "C"
NAME = "D"
n_a_m_e = "E"
_name = "F"
name_ = "G"
_name_ = "H"
na56me = "I"

print(name,Name,naMe,NAME,n_a_m_e, NAME, n_a_m_e, _name, name_,_name, na56me)
```

Output:

```
A B C D E D E F G F I
```

In the above example, we have declared a few valid variable names such as name, _name_ , etc. But it is not recommended because when we try to read code, it may create confusion. The variable name should be descriptive to make code more readable.

Python Variable Types

There are two types of variables in Python - Local variable and Global variable. Let's understand the following variables.

Local Variable

Local variables are the variables that declared inside the function and have scope within the function. Let's understand the following example.

Example -

```
# Declaring a function
def add():
    # Defining local variables. They has scope only within a function
    a = 20
    b = 30
    c = a + b
    print("The sum is:", c)

# Calling a function
add()
```

Output:

```
The sum is: 50
```

Explanation:

In the above code, we declared a function named **add()** and assigned a few variables within the function. These variables will be referred to as the **local variables** which have scope only inside the function. If we try to use them outside the function, we get a following error.

```
add()
# Accessing local variable outside the function
print(a)
```

Output:

```
The sum is: 50
  print(a)
NameError: name 'a' is not defined
```

We tried to use local variable outside their scope; it threw the **NameError.**

Global Variables

Global variables can be used throughout the program, and its scope is in the entire program. We can use global variables inside or outside the function.

A variable declared outside the function is the global variable by default. Python provides the **global** keyword to use global variable inside the function. If we don't use the **global** keyword, the function treats it as a local variable. Let's understand the following example.

Example -

```
# Declare a variable and initialize it
x = 101
# Global variable in function
def mainFunction():
   # printing a global variable
   global x
   print(x)
   # modifying a global variable
   x = 'Welcome To Javatpoint'
   print(x)
 mainFunction()
print(x)
```

Output:

```
101
Welcome To Javatpoint
Welcome To Javatpoint
```

Explanation:

In the above code, we declare a global variable **x** and assign a value to it. Next, we defined a function and accessed the declared variable using the **global** keyword inside the function. Now we can modify its value. Then, we assigned a new string value to the variable x.

Now, we called the function and proceeded to print **x**. It printed the as newly assigned value of x.

Python Operators

The operator is a symbol that performs a certain operation between two operands, according to one definition. In a particular programming language, operators serve as the foundation upon which logic is constructed in a programme. The different operators that Python offers are listed here.

- Arithmetic operators
- Comparison operators
- Assignment Operators
- Logical Operators
- Bitwise Operators
- Membership Operators
- Identity Operators
- Arithmetic Operators

Conditional Expressions in Python

Python's conditional statements carry out various calculations or operations according to whether a particular Boolean condition is evaluated as true or false. In Python, IF statements deal with conditional statements.

We'll learn how to use conditional statements in Python in this tutorial.

What is Python If Statement?

To make decisions, utilize the if statement in Python. It has a body of instructions that only executes whenever the if statement's condition is met. The additional else statement, which includes some instructions for the else statement, runs if the if condition is false.

Python's if-else statement is used when you wish to satisfy one statement while the other is false.

Python Syntax of the if Statement:

if <conditional expression>

Statement

else

Statement

Code

```
a, b = 6, 5
if a > b:
    code = "a is greater than b"
    print(code)
```

Output:

```
a is greater than b
```

How to Use the else Condition?

The "else condition" is usually used when judging one statement based on another. If the condition mentioned in the if code block is wrong, then the interpreter will execute the else code block.

Code

```
# Python program to execute if-else statement

a, b = 6, 5
if a < b:
    code = "a is less than b"
    print(code)
else:
    print("a is greater than b")
```

Output:

```
a is greater than b
```

How to use the elif Condition?

We can employ the "elif" clause to fix the issue caused by the "else condition" made earlier. You can instruct the software to print the third condition or alternative when the first two conditions fail or are erroneous by using the "elif" condition.

Code

```
a, b = 9, 9
```

```
if a < b:
    code = "a is less than b"
elif a == b:
    code = "a is equal to b"
else:
    code = "a is greater than b"
print(code)
```

Output:

```
a is equal to b
```

Python Nested if Statement

The following example demonstrates nested if Statement Python

Code

```
A = 100
B = 200
C = 300
  if B > A:
    if B > C:
        print("B is the largest number")
    else:
        if A > B:
            if A > C:
  print("A is the largest number")
        elif C > A:
            if C > B:

print("C is the largest number")
        else:
print("All numbers are equal")

if B % C == 0:
    if A % C == 0:
        print("C is a common factor of A and B")
```

Output:

```
C is the largest number
```

ACTIVITY LOG FOR THE EIGHTH WEEK

Day & Date	Brief description of the daily activity	Learning Outcome	Person In- Charge Signature
Day – 1 02-Mar-2023	List and Tuples	Basics of lists and tuples in python	
Day – 2 03-Mar-2023	Set and Dictionary	Basics of set and dictionary in python	
Day – 3 04-Mar-2023	Functions introduction and categories of functions	Introduction of functions and its categories	
Day – 4 06-Mar-2023	Modules and Packages	Introdution of modules like tKinter module and packages like numpy, pandas etc	
Day – 5 07-Mar-2023	Introduction of OOPS [Class & Object]	Explanation of class and object	
Day –6 08-Mar-2023	HOLIDAY		

WEEKLY REPORT

Week -8 (From: 02/03/23 To: 08/03/23)

Objectives of the Activity Done:

This week, I focused on learning the basics of Python Lists, one of the most important and commonly used data structures in Python. Here are the key topics I covered:

Introduction to Python Lists: I started by learning what Python Lists are and how they're used to store collections of data. I also learned about the basic syntax of Lists, including how to create and access List elements.

List Methods: I learned how to use different List methods in Python, including how to add and remove elements from a List, how to sort a List, and how to manipulate List elements using slicing and concatenation.

Nested Lists: I learned how to create and work with nested Lists in Python, including how to access and manipulate nested List elements.

Overall, this week was a great introduction to Python Lists basics. I feel comfortable creating and manipulating Lists in Python, and I understand how to use List methods and List comprehension. I plan to continue practicing by building more complex Python programs and exploring different data structures and libraries.

Detailed Report:

Python List

A list in Python is used to store the sequence of various types of data. A list can be defined as a collection of values or items of different types. Python lists are mutable type which implies that we may modify its element after it has been formed. The items in the list are separated with the comma (,) and enclosed with the square brackets [].

Although Python has six data types that may hold sequences, the list is the most popular and dependable form. The collection of data is stored in a list, a sequence data type. Similar sequence data formats are Tuples and String.

Python lists are identical to dynamically scaled arrays that are specified in other languages, such as Java's ArrayList and C++'s vector. A list is a group of items that are denoted by the symbol [] and subdivided by commas.

List Declaration

Code

```
# a simple list
list1 = [1, 2, "Python", "Program", 15.9]
list2 = ["Amy", "Ryan", "Henry", "Emma"]
print(list1)
print(list2)
```

```
print(type(list1))
print(type(list2))
```

Output:

```
[1, 2, 'Python', 'Program', 15.9]
['Amy', 'Ryan', 'Henry', 'Emma']
< class ' list ' >
< class ' list ' >
```

Characteristics of Lists

The list has the following characteristics:

- The lists are ordered.
- The element of the list can access by index.
- The lists are the mutable type.
- The lists are mutable types.
- A list can store the number of various elements.

Ordered List Checking

Code

```
# example
a = [ 1, 2, "Ram", 3.50, "Rahul", 5, 6 ]
b = [ 1, 2, 5, "Ram", 3.50, "Rahul", 6 ]
a == b
```

Output:

```
False
```

The identical elements were included in both lists, but the second list modified the index position of the fifth element, which is against the lists' intended order. When the two lists are compared, false is returned.

Code

```
a = [ 1, 2, "Ram", 3.50, "Rahul", 5, 6]
b = [ 1, 2, "Ram", 3.50, "Rahul", 5, 6]
a == b
```

Output:

```
True
```

Code

```
# list example in detail
emp = [ "John", 102, "USA"]
Dep1 = [ "CS",10]
Dep2 = [ "IT",11]
HOD_CS = [ 10,"Mr. Holding"]
HOD_IT = [11, "Mr. Bewon"]
print("printing employee data ...")
print(" Name : %s, ID: %d, Country: %s" %(emp[0], emp[1], emp[2]))
print("printing departments ...")
print("Department 1:\nName: %s, ID: %d\n Department 2:\n Name: %s, ID: %s"%( Dep1[0], Dep2[1
], Dep2[0], Dep2[1]))
print("HOD Details ....")
print("CS HOD Name: %s, Id: %d" %(HOD_CS[1], HOD_CS[0]))
print("IT HOD Name: %s, Id: %d" %(HOD_IT[1], HOD_IT[0]))
print(type(emp), type(Dep1), type(Dep2), type(HOD_CS), type(HOD_IT))
```

Output:

```
printing employee data...
Name : John, ID: 102, Country: USA
printing departments...
Department 1:
Name: CS, ID: 11
Department 2:
Name: IT, ID: 11
HOD Details ....
CS HOD Name: Mr. Holding, Id: 10
IT HOD Name: Mr. Bewon, Id: 11
<class ' list '> <class ' list '> <class ' list '> <class ' list '> <class ' lis
```

Python Tuples

A Python Tuple is a group of items that are separated by commas. The indexing, nested objects, and repetitions of a tuple are somewhat like those of a list, however unlike a list, a tuple is immutable.

The distinction between the two is that while we can edit the contents of a list, we cannot alter the elements of a tuple once they have been assigned.

Example

1. ("Suzuki", "Audi", "BMW"," Skoda ") is a tuple.

Features of Python Tuple

- Tuples are an immutable data type, which means that once they have been generated, their elements cannot be changed.
- Since tuples are ordered sequences, each element has a specific order that will never change.

Creating of Tuple:

To create a tuple, all the objects (or "elements") must be enclosed in parenthesis (), each one separated by a comma. Although it is not necessary to include parentheses, doing so is advised.

A tuple can contain any number of items, including ones with different data types (dictionary, string, float, list, etc.).

Code:

```
empty_tuple = ()
print("Empty tuple: ", empty_tuple)
int_tuple = (4, 6, 8, 10, 12, 14)
print("Tuple with integers: ", int_tuple)
mixed_tuple = (4, "Python", 9.3)
print("Tuple with different data types: ", mixed_tuple)
nested_tuple = ("Python", {4: 5, 6: 2, 8:2}, (5, 3, 5, 6))
print("A nested tuple: ", nested_tuple)
```

Output:

```
Empty tuple:  ()
Tuple with integers:  (4, 6, 8, 10, 12, 14)
Tuple with different data types:  (4, 'Python', 9.3)
A nested tuple:  ('Python', {4: 5, 6: 2, 8: 2}, (5, 3, 5, 6))
```

Python Set

A Python set is the collection of the unordered items. Each element in the set must be unique, immutable, and the sets remove the duplicate elements. Sets are mutable which means we can modify it after its creation.

Unlike other collections in Python, there is no index attached to the elements of the set, i.e., we cannot directly access any element of the set by the index. However, we can print them all together, or we can get the list of elements by looping through the set.

Creating a set

The set can be created by enclosing the comma-separated immutable items with the curly braces {}. Python also provides the set() method, which can be used to create the set by the passed sequence.

Example 1: Using curly braces

```
Days = {"Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday", "Sunday"}
print(Days)
print(type(Days))
print("looping through the set elements ... ")
for i in Days:
   print(i)
```

Output:

```
{'Friday', 'Tuesday', 'Monday', 'Saturday', 'Thursday', 'Sunday', 'Wednesday'}
<class 'set'>
looping through the set elements ...
Friday
Tuesday
Monday
Saturday
Thursday
Sunday
Wednesday
```

Python Dictionary

An effective data structure for storing data in Python is dictionaries, in which can simulate the real-life data arrangement where some specific value exists for some particular key.

- Python Dictionary is used to store the data in a key-value pair format.
- It is the mutable data-structure.
- The elements Keys and values is employed to create the dictionary.
- Keys must consist of just one element.
- Value can be any type such as list, tuple, integer, etc.

In other words, we can say that a dictionary is the collection of key-value pairs where the value can be of any Python object. In contrast, the keys are the immutable Python object, i.e., Numbers, string, or tuple. Dictionary entries are ordered as of Python version 3.7. In Python 3.6 and before, dictionaries are generally unordered.

Creating the Dictionary

The simplest approach to create a Python dictionary is by using curly brackets {}, but there are other methods as well. The dictionary can be created by using multiple key-value pairs enclosed with the curly brackets {}, and each key is separated from its value by the colon (:). The syntax to define the dictionary is given below.

Syntax:

```
Dict = {"Name": "Chris", "Age": 20}
```

In the above dictionary **Dict**, The keys **Name** and **Age** are the strings which comes under the category of an immutable object.

Let's see an example to create a dictionary and print its content.

Code

```
Employee = {"Name": "John", "Age": 29, "salary":25000,"Company":"GOOGLE"}
print(type(Employee))
print("printing Employee data..... ")
print(Employee)
```

Output

```
<class 'dict'>
printing Employee data ....
{'Name': 'John', 'Age': 29, 'salary': 25000, 'Company': 'GOOGLE'}
```

Python Functions

The fundamentals of Python functions, including what they are, their syntax, their primary parts, return keywords, and major types, will be covered in this tutorial. Additionally, we'll examine several instances of Python function definitions.

What are Python Functions?

A function is a collection of related assertions that performs a mathematical, analytical, or evaluative operation. A collection of statements called Python Functions returns the particular task. Python functions are simple to define and essential to intermediate-level programming. The exact criteria hold to function names as they do to variable names. The goal is to group up certain often performed actions and define a function. We may call the function and reuse the code contained within it with different variables rather than repeatedly creating the same code block for different input variables.

User-defined and built-in functions are the two main categories of functions in Python. It helps maintain the programme concise, unique, and well-structured.

Advantages of Functions in Python

Python functions have the following Perks.

- By including functions, we can prevent repeating the same code block repeatedly in a program.
- Python functions, once defined, can be called many times and from anywhere in a program.

- If our Python program is large, it can be separated into numerous functions which is simple to track.
- The key accomplishment of Python functions is we can return as many outputs as we want with different arguments.

However, calling functions has always been overhead in a Python program.

Syntax of Python Function

```
1. #  An example Python Function
2. def function_name( parameters ):
3.     # code block
```

The following elements make up to define a function, as seen above.

- The beginning of a function header is indicated by a keyword called def.
- function_name is the function's name that we can use to separate it from others. We will use this name to call the function later in the program. In Python, name functions must follow the same rules as naming variables.
- We pass arguments to the defined function using parameters. However, they are optional.
- The function header is terminated by a colon (:).
- We can use a documentation string called docstring in the short form to explain the purpose of the function.
- We can use a return expression to return a value from a defined function.

Example of a User-Defined Function

We will define a function that when called will return the square of the number passed to it as an argument.

Code

```
def square( num ):
   """     This function computes the square of the number.      """
   return num**2
object_ = square(6)
print( "The square of the given number is: ", object_ )
```

Output:

```
The square of the given number is:  36
```

Python Modules

This tutorial will explain how to construct and import custom Python modules. Additionally, we may import or integrate Python's built-in modules via various methods.

What is Modular Programming?

Modular programming is the practice of segmenting a single, complicated coding task into multiple, simpler, easier-to-manage sub-tasks. We call these subtasks modules. Therefore, we can build a bigger program by assembling different modules that act like building blocks.

Modularizing our code in a big application has a lot of benefits.

Simplification: A module often concentrates on one comparatively small area of the overall problem instead of the full task. We will have a more manageable design problem to think about if we are only concentrating on one module. Program development is now simpler and much less vulnerable to mistakes.

Flexibility: Modules are frequently used to establish conceptual separations between various problem areas. It is less likely that changes to one module would influence other portions of the program if modules are constructed in a fashion that reduces interconnectedness. (We might even be capable of editing a module despite being familiar with the program beyond it.) It increases the likelihood that a group of numerous developers will be able to collaborate on a big project.

Reusability: Functions created in a particular module may be readily accessed by different sections of the assignment (through a suitably established api). As a result, duplicate code is no longer necessary.

Scope: Modules often declare a distinct namespace to prevent identifier clashes in various parts of a program.

In Python, modularization of the code is encouraged through the use of functions, modules, and packages.

What are Modules in Python?

A document with definitions of functions and various statements written in Python is called a Python module.

In Python, we can define a module in one of 3 ways:

- Python itself allows for the creation of modules.
- Similar to the re (regular expression) module, a module can be primarily written in C programming language and then dynamically inserted at run-time.
- A built-in module, such as the itertools module, is inherently included in the interpreter.

A module is a file containing Python code, definitions of functions, statements, or classes. An example_module.py file is a module we will create and whose name is example_module.

We employ modules to divide complicated programs into smaller, more understandable pieces. Modules also allow for the reuse of code.

Rather than duplicating their definitions into several applications, we may define our most frequently used functions in a separate module and then import the complete module.

Let's construct a module. Save the file as example_module.py after entering the following.

Code

```
# Python program to show how to create a module.
# defining a function in the module to reuse it
def square( number ):
   """This function will square the number passed to it"""

   result = number ** 2
   return result
```

Here, a module called example_module contains the definition of the function square(). The function returns the square of a given number.

How to Import Modules in Python?

In Python, we may import functions from one module into our program, or as we say into, another module.

For this, we make use of the import Python keyword. In the Python window, we add the next to import keyword, the name of the module we need to import. We will import the module we defined earlier example_module.

Code

```
import example_module
```

The functions that we defined in the example_module are not immediately imported into the present program. Only the name of the module, i.e., example_ module, is imported here.

We may use the dot operator to use the functions using the module name. For instance:

Code

```
result = example_module.square(  4  )
print( "By using the module square of number is: ", result )
```

Output:

```
By using the module square of number is: 16
```

There are several standard modules for Python. The complete list of Python standard modules is available. The list can be seen using the help command.

Python OOPs Concepts

Like other general-purpose programming languages, Python is also an object-oriented language since its beginning. It allows us to develop applications using an Object-Oriented approach. In Python, we can easily create and use classes and objects.

An object-oriented paradigm is to design the program using classes and objects. The object is related to real-word entities such as book, house, pencil, etc. The oops concept focuses on writing the reusable code. It is a widespread technique to solve the problem by creating objects.

Major principles of object-oriented programming system are given below.

- Class
- Object
- Method
- Inheritance
- Polymorphism
- Data Abstraction
- Encapsulation

Class

The class can be defined as a collection of objects. It is a logical entity that has some specific attributes and methods. For example: if you have an employee class, then it should contain an attribute and method, i.e. an email id, name, age, salary, etc.

Syntax

```
class ClassName:
    <statement-1>
    .       .
    <statement-N>
```

Object

The object is an entity that has state and behavior. It may be any real-world object like the mouse, keyboard, chair, table, pen, etc.

Everything in Python is an object, and almost everything has attributes and methods. All functions have a built-in attribute __doc_, which returns the docstring defined in the function source code.

When we define a class, it needs to create an object to allocate the memory. Consider the following example.

ACTIVITY LOG FOR THE NINTH WEEK

Day & Date	Brief description of the daily activity	Learning Outcome	Person In- Charge Signature
Day – 1 09-Mar-2023	Abstraction Encapsulation	Introduction of abstraction and encapsulation	
Day – 2 10-Mar-2023	Inheritance Polymorphism	Introduction of inheritence and its types and polymorphism	
Day – 3 11-Mar-2023	Controls Statements[if,if else and elif and while ,for loop]	Types of control statements	
Day – 4 13-Mar-2023	Method overriding Method overloading	Introduction of method overriding	
Day – 5 14-Mar-2023	Introduction of Exception Handling	Exception handling	
Day –6 15-Mar-2023	Python data types	Types of data types	

WEEKLY REPORT

Week -9 (From: 09/03/23 To: 15/03/23)

Objectives of the Activity Done:

Detailed Report:

This week, I focused on learning the basics of Python Lists, one of the most important and commonly used data structures in Python. Here are the key topics I covered:

Inheritance

Inheritance is the most important aspect of object-oriented programming, which simulates the real-world concept of inheritance. It specifies that the child object acquires all the properties and behaviors of the parent object.

By using inheritance, we can create a class which uses all the properties and behavior of another class. The new class is known as a derived class or child class, and the one whose properties are acquired is known as a base class or parent class.

It provides the re-usability of the code.

Polymorphism

Polymorphism contains two words "poly" and "morphs". Poly means many, and morph means shape. By polymorphism, we understand that one task can be performed in different ways. For example - you have a class animal, and all animals speak. But they speak differently. Here, the "speak" behavior is polymorphic in a sense and depends on the animal. So, the abstract "animal" concept does not actually "speak", but specific animals (like dogs and cats) have a concrete implementation of the action "speak".

Encapsulation

Encapsulation is also an essential aspect of object-oriented programming. It is used to restrict access to methods and variables. In encapsulation, code and data are wrapped together within a single unit from being modified by accident.

Data Abstraction

Data abstraction and encapsulation both are often used as synonyms. Both are nearly synonyms because data abstraction is achieved through encapsulation.

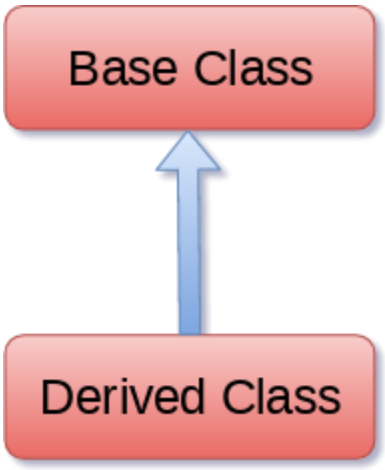
Abstraction is used to hide internal details and show only functionalities. Abstracting something means to give names to things so that the name captures the core of what a function or a whole program does.

Python Inheritance

Inheritance is an important aspect of the object-oriented paradigm. Inheritance provides code reusability to the program because we can use an existing class to create a new class instead of creating it from scratch.

In inheritance, the child class acquires the properties and can access all the data members and functions defined in the parent class. A child class can also provide its specific implementation to the functions of the parent class. In this section of the tutorial, we will discuss inheritance in detail.

In python, a derived class can inherit base class by just mentioning the base in the bracket after the derived class name. Consider the following syntax to inherit a base class into the derived class.



Syntax

class derived-**class**(base **class**):

<**class**-suite>

A class can inherit multiple classes by mentioning all of them inside the bracket. Consider the following syntax.

Syntax

class derive-**class**(<base **class** 1>, <base **class** 2>, ... <base **class** n>):

<**class** - suite>

Example 1

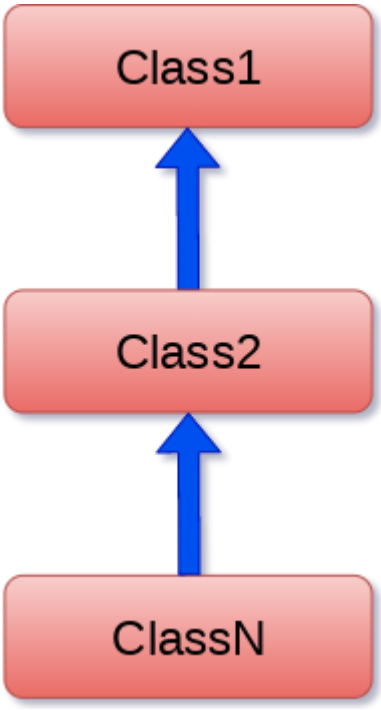
```
class Animal:
    def speak(self):
        print("Animal Speaking")
#child class Dog inherits the base class Animal
class Dog(Animal):
    def bark(self):
        print("dog barking")
d = Dog()
d.bark()
d.speak()
```

Output:

```
dog barking
Animal Speaking
```

Python Multi-Level inheritance

Multi-Level inheritance is possible in python like other object-oriented languages. Multi-level inheritance is archived when a derived class inherits another derived class. There is no limit on the number of levels up to which, the multi-level inheritance is archived in python.



The syntax of multi-level inheritance is given below.

Syntax

```
class class1:
    <class-suite>
class class2(class1):
    <class suite>
class class3(class2):
    <class suite>
.
.
```

Example

```
class Animal:
    def speak(self):
        print("Animal Speaking")
class Dog(Animal):
    def bark(self):
        print("dog barking")
```

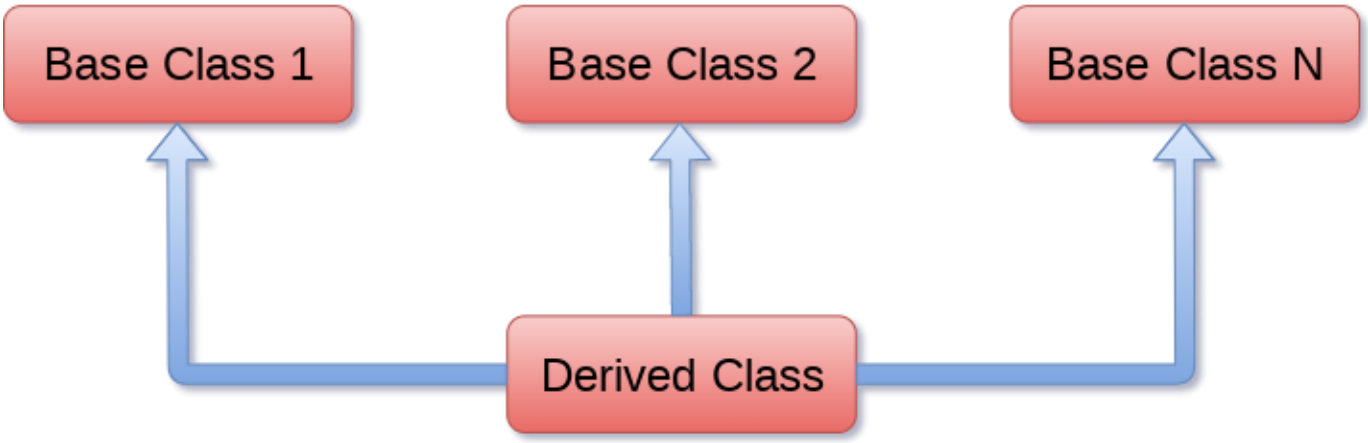
```
class DogChild(Dog):
    def eat(self):
        print("Eating bread...")
d = DogChild()
d.bark()
d.speak()
d.eat()
```

Output:

```
dog barking
Animal Speaking
Eating bread...
```

Python Multiple inheritance

Python provides us the flexibility to inherit multiple base classes in the child class.



The syntax to perform multiple inheritance is given below.

Syntax

```
class Base1:
    <class-suite>

class Base2:
    <class-suite>
.
.
.
class BaseN:
    <class-suite>
```

```
class Derived(Base1, Base2, ..... BaseN):
   <class-suite>
```

Example

```
class Calculation1:
   def Summation(self,a,b):
      return a+b;
class Calculation2:
   def Multiplication(self,a,b):
      return a*b;
class Derived(Calculation1,Calculation2):
   def Divide(self,a,b):
      return a/b;
d = Derived()
print(d.Summation(10,20))
print(d.Multiplication(10,20))
print(d.Divide(10,20))
```

Output:

```
30
200
0.5
```

Python Exceptions

When a Python program meets an error, it stops the execution of the rest of the program. An error in Python might be either an error in the syntax of an expression or a Python exception. We will see what an exception is. Also, we will see the difference between a syntax error and an exception in this tutorial. Following that, we will learn about trying and except blocks and how to raise exceptions and make assertions. After that, we will see the Python exceptions list.

What is an Exception?

An exception in Python is an incident that happens while executing a program that causes the regular course of the program's commands to be disrupted. When a Python code comes across a condition it can't handle, it raises an exception. An object in Python that describes an error is called an exception.

When a Python code throws an exception, it has two options: handle the exception immediately or stop and quit.

Exceptions versus Syntax Errors

When the interpreter identifies a statement that has an error, syntax errors occur. Consider the following scenario:

Code

```
string = "Python Exceptions"

    for s in string:
       if (s != o:
          print( s )
```

Output:

```
   if (s != o:
           ^
Syntax Error: invalid syntax
```

The arrow in the output shows where the interpreter encountered a syntactic error. There was one unclosed bracket in this case. Close it and rerun the program:

Code

```
    string = "Python Exceptions"


    for s in string:
       if (s != o):
          print( s )
```

Output:

```
    2 string = "Python Exceptions"
    4 for s in string:
----> 5     if (s != o):
    6         print( s )

Name Error: name 'o' is not defined
```

We encountered an exception error after executing this code. When syntactically valid Python code produces an error, this is the kind of error that arises. The output's last line specified the name of the exception error code encountered. Instead of displaying just "exception error", Python displays information about the sort of exception error that occurred. It was a NameError in this situation. Python includes several built-in exceptions. However, Python offers the facility to construct custom exceptions.

Try and Except Statement - Catching Exceptions

In Python, we catch exceptions and handle them using try and except code blocks. The try clause contains the code that can raise an exception, while the except clause contains the code lines that handle the exception. Let's see if we can access the index from the array, which is more than the array's length, and handle the resulting exception.

Code

```
a = ["Python", "Exceptions", "try and except"]
try:
   #looping through the elements of the array a, choosing a range that goes beyond the length of the ar
ray
    for i in range( 4 ):
      print( "The index and element from the array is", i, a[i] )
except:
   print ("Index out of range")
```

Output:

```
The index and element from the array is 0 Python
The index and element from the array is 1 Exceptions
The index and element from the array is 2 try and except
Index out of range
```

ACTIVITY LOG FOR THE TENTH WEEK

Day & Date	Brief description of the daily activity	Learning Outcome	Person In- Charge Signature
Day – 1 16-Mar-2023	Exception Handling Techniques	Basics and techniques of exception handling	
Day – 2 17-Mar-2023	Introduction of Multithreading	Multithreading process	
Day – 3 18-Mar-2023	HOLIDAY		
Day – 4 20-Mar-2023	Implementation of Life cycle of threads	Life cycle of threads	
Day – 5 21-Mar-2023	Introduction of mysql database	MySql database introduction	
Day –6 22-Mar-2023	HOLIDAY		

WEEKLY REPORT

Week -10 (From: 16/03/23 To: 22/03/23)

Objectives of the Activity Done:

Detailed Report:

This week, I focused on learning the basics of Python Lists, one of the most important and commonly used data structures in Python. Here are the key topics I covered:

Multithreading in Python

A thread is the smallest unit of a program or process executed independently or scheduled by the Operating System. In the computer system, an Operating System achieves multitasking by dividing the process into threads. A thread is a lightweight process that ensures the execution of the process separately on the system. In Python 3, when multiple processors are running on a program, each processor runs simultaneously to execute its tasks separately.

Python Multithreading

Multithreading is a threading technique in Python programming to run multiple threads concurrently by rapidly switching between threads with a CPU help (called context switching). Besides, it allows sharing of its data space with the main threads inside a process that share information and communication with other threads easier than individual processes. Multithreading aims to perform multiple tasks simultaneously, which increases performance, speed and improves the rendering of the application.

Benefits of Multithreading in Python

Following are the benefits to create a multithreaded application in Python, as follows:

1. It ensures effective utilization of computer system resources.
2. Multithreaded applications are more responsive.
3. It shares resources and its state with sub-threads (child) which makes it more economical.
4. It makes the multiprocessor architecture more effective due to similarity.
5. It saves time by executing multiple threads at the same time.
6. The system does not require too much memory to store multiple threads.

When to use Multithreading in Python?

It is a very useful technique for time-saving and improving the performance of an application. Multithreading allows the programmer to divide application **tasks** into sub-tasks and simultaneously run them in a program. It allows threads to communicate and share resources such as files, data, and memory to the same processor. Furthermore, it increases the user's responsiveness to continue running a program even if a part of the application is the length or blocked.

How to achieve multithreading in Python?

There are two main modules of multithreading used to handle threads in Python.

1. The thread module
2. The threading module

Thread modules

It is started with Python 3, designated as obsolete, and can only be accessed with **_thread** that supports backward compatibility.

Syntax:

1. thread.start_new_thread (function_name, args[, kwargs])

To implement the thread module in Python, we need to import a **thread** module and then define a function that performs some action by setting the target with a variable.

Thread.py

```
import thread # import the thread module
import time # import time module
  def cal_sqre(num): # define the cal_sqre function
    print(" Calculate the square root of the given number")
    for n in num:
       time.sleep(0.3) # at each iteration it waits for 0.3 time
       print(' Square is : ', n * n)
  def cal_cube(num): # define the cal_cube() function
    print(" Calculate the cube of the given number")
    for n in num:
       time.sleep(0.3) # at each iteration it waits for 0.3 time
       print(" Cube is : ", n * n *n)
  arr = [4, 5, 6, 7, 2] # given array
  t1 = time.time() # get total time to execute the functions
cal_sqre(arr) # call cal_sqre() function
cal_cube(arr) # call cal_cube() function
print(" Total time taken by threads is :", time.time() - t1)
```

Output:

```
Calculate the square root of the given number
 Square is:  16
```

```
Square is:  25
Square is:  36
Square is:  49
Square is:  4
Calculate the cube of the given number
Cube is: 64
Cube is:  125
Cube is:  216
Cube is:  343
Cube is: 8
Total time taken by threads is: 3.005793809890747
```

Threading Modules

The threading module is a high-level implementation of multithreading used to deploy an application in Python. To use multithreading, we need to import the threading module in Python Program.

Thread Class Methods

Methods	Description
start()	A start() method is used to initiate the activity of a thread. And it calls only once for each thread so that the execution of the thread can begin.
run()	A run() method is used to define a thread's activity and can be overridden by a class that extends the threads class.
join()	A join() method is used to block the execution of another code until the thread terminates.

MySql:

MySQL is currently the most popular database management system software used for managing the relational database. It is open-source database software, which is supported by Oracle Company. It is fast, scalable, and easy to use database management system in comparison with Microsoft SQL Server and Oracle Database. It is commonly used in conjunction with PHP scripts for creating powerful and dynamic server-side or web-based enterprise applications.

It is developed, marketed, and supported by **MySQL AB, a Swedish company**, and written in C programming language and C++ programming language. The official pronunciation of MySQL is not the My Sequel; it is ***My Ess Que Ell***. *However, you can pronounce it in your way.* Many small and big companies use MySQL. MySQL supports many Operating Systems like Windows, Linux, MacOS, etc. with C, C++, and Java languages.

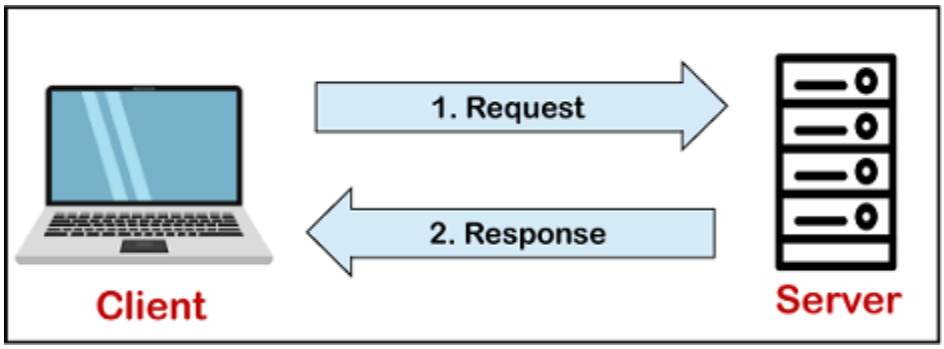
MySQL is a Relational Database Management System (RDBMS) software that provides many things, which are as follows:

- It allows us to implement database operations on tables, rows, columns, and indexes.
- It defines the database relationship in the form of tables (collection of rows and columns), also known as relations.

- It provides the Referential Integrity between rows or columns of various tables.
- It allows us to updates the table indexes automatically.
- It uses many SQL queries and combines useful information from multiple tables for the end-users.

How MySQL Works?

MySQL follows the working of Client-Server Architecture. This model is designed for the end-users called clients to access the resources from a central computer known as a server using network services. Here, the clients make requests through a graphical user interface (GUI), and the server will give the desired output as soon as the instructions are matched. The process of MySQL environment is the same as the client-server model.



The core of the MySQL database is the MySQL Server. This server is available as a separate program and responsible for handling all the database instructions, statements, or commands. The working of MySQL database with MySQL Server are as follows:

1. MySQL creates a database that allows you to build many tables to store and manipulate data and defining the relationship between each table.
2. Clients make requests through the GUI screen or command prompt by using specific SQL expressions on MySQL.
3. Finally, the server application will respond with the requested expressions and produce the desired result on the client-side.

A client can use any MySQL GUI. But, it is making sure that your GUI should be lighter and user-friendly to make your data management activities faster and easier. Some of the most widely used MySQL GUIs are MySQL Workbench, SequelPro, DBVisualizer, and the Navicat DB Admin Tool. Some GUIs are commercial, while some are free with limited functionality, and some are only compatible with MacOS. Thus, you can choose the GUI according to your needs.

SQL Languages

SQL Languages are those languages that allow the database users to read, modify, delete and store data in the database systems.

Following are the four different types of languages or commands which are widely used in SQL queries:

1. TCL (Transaction Control Language)
2. DML (Data Manipulation Language)
3. DCL (Data Control Language)
4. DDL (Data Definition Language)

DDL (Data Definition Language)

Data Definition Languages allow users to create, modify, and destroy the schema of database objects.

We can enter the correct data in the database by applying the constraints in the DDL languages.

The DDL Languages or commands are categorized into five commands which are widely used in the SQL queries:

1. CREATE DDL Command
2. ALTER DDL Command
3. DROP DDL Command
4. TRUNCATE DDL Command
5. RENAME DDL Command

Let's discuss each DDL command with syntax and examples.

CREATE Command

This DDL command allows us to create the new table, function, stored procedure, and other database objects.

Syntax of Create DDL Command to create a new table in the database:

1. **CREATE TABLE** Name_of_Table (Column1 datatype (Length), Column2 datatype (Length) … ….);

Example of Create Command:

The following SQL query creates the new Mobile_Details table using CREATE DDL command:

CREATE TABLE Mobile_Details
(Mobile_Number **INT** NOT NULL,
Mobile_Name **Varchar**(50),
Manufacturing_Year **INT** NOT NULL,
Mobile_Cost **INT**
); ALTER Command

This DDL command allows us to modify the structure of database objects.

Syntax of Alter Command to modify the existing table:

ALTER TABLE Name_of_Table **ADD** Column_Name Datatype (Length **of Column**);

Example of Alter Command:

The following SQL query adds the new column in the Mobile_Details table using ALTER DDL command:

ALTER TABLE Mobile_Details **ADD** Mobile_Color **Varchar** (50);

DROP Command

This DDL command allows us to remove the table definition and data from the SQL systems.

Syntax of Drop Command to remove the existing table:

DROP TABLE Name_of_Table;

ACTIVITY LOG FOR THE ELEVENTH WEEK

Day & Date	Brief description of the daily activity	Learning Outcome	Person In- Charge Signature
Day – 1 23-Mar-2023	database Constraints	Introduction of database constraints	
Day – 2 24-Mar-2023	Creating a connection with database	Database connection	
Day – 3 25-Mar-2023	HOLIDAY		
Day – 4 27-Mar-2023	Introduction of django	Django introduction	
Day – 5 28-Mar-2023	Sql languages	Sql introduction	
Day –6 29-Mar-2023	Django pattern MODEL VIEW TEMPLATE	Template model view pattern of django	

WEEKLY REPORT

Week -11 (From: 23/03/23 To: 29/03/23)

Objectives of the Activity Done:

This week, I focused on learning the basics of SQL, the Structured Query Language used to interact with relational databases. Here are the key topics I covered:

Detailed Report:

Constraints in SQL

Constraints in SQL means we are applying certain conditions or restrictions on the database. This further means that before inserting data into the database, we are checking for some conditions. If the condition we have applied to the database holds true for the data which is to be inserted, then only the data will be inserted into the database tables.

Constraints in SQL can be categorized into two types:

1. **ColumnLevelConstraint:**
   Column Level Constraint is used to apply a constraint on a single column.
2. **TableLevelConstraint:**
   Table Level Constraint is used to apply a constraint on multiple columns.

Some of the real-life examples of constraints are as follows:

1. Every person has a unique email id. This is because while creating an email account for any user, the email providing services such as Gmail, Yahoo or any other email providing service will always check for the availability of the email id that the user wants for himself. If some other user already takes the email id that the user wants, then that id cannot be assigned to another user. This simply means that no two users can have the same email ids on the same email providing service. So, here the email id is the constraint on the database of email providing services.
2. Whenever we set a password for any system, there are certain constraints that are to be followed. These constraints may include the following:
   - There must be one uppercase character in the password.
   - Password must be of at least eight characters in length.
   - Password must contain at least one special symbol.

Constraints available in SQL are:

1. NOT NULL
2. UNIQUE
3. PRIMARY KEY
4. FOREIGN KEY

5. CHECK
6. DEFAULT
7. CREATE INDEX

Database Connection

In this section of the tutorial, we will discuss the steps to connect the python application to the database.

There are the following steps to connect a python application to our database.

1. Import mysql.connector module
2. Create the connection object.
3. Create the cursor object
4. Execute the query

Creating the connection

To create a connection between the MySQL database and the python application, the connect() method of mysql.connector module is used.

Pass the database details like HostName, username, and the database password in the method call. The method returns the connection object.

The syntax to use the connect() is given below.

1. Connection-Object= mysql.connector.connect(host = <host-name> , user = <username> , passwd = <password>)

Consider the following example.

Example

```
import mysql.connector

#Create the connection object
myconn = mysql.connector.connect(host = "localhost", user = "root",passwd = "google")

#printing the connection object
print(myconn)
```

Output:

```
<mysql.connector.connection.MySQLConnection object at 0x7fb142edd780>
```

Example

```
import mysql.connector
myconn = mysql.connector.connect(host = "localhost", user = "root",passwd = "google", database = "mydb")
print(myconn)
```

Output:

```
<mysql.connector.connection.MySQLConnection object at 0x7ff64aa3d7b8>
```

Creating a cursor object

The cursor object can be defined as an abstraction specified in the Python DB-API 2.0. It facilitates us to have multiple separate working environments through the same connection to the database. We can create the cursor object by calling the 'cursor' function of the connection object. The cursor object is an important aspect of executing queries to the databases.

The syntax to create the cursor object is given below.

1. <my_cur> = conn.cursor()

Example

```
import mysql.connector
myconn = mysql.connector.connect(host = "localhost", user = "root",passwd = "google", database = "mydb")
print(myconn)
cur = myconn.cursor()
print(cur)
```

Output:

```
<mysql.connector.connection.MySQLConnection object at 0x7faa17a15748>
MySQLCursor: (Nothing executed yet)
```

Django:

Django is a web application framework written in Python programming language. It is based on MVT (Model View Template) design pattern. The Django is very demanding due to its rapid development feature. It takes less time to build application after collecting client requirement.

This framework uses a famous tag line:**The web framework for perfectionists with deadlines.**

By using Django, we can build web applications in very less time. Django is designed in such a manner that it handles much of configure things automatically, so we can focus on application development only.

Django MVT

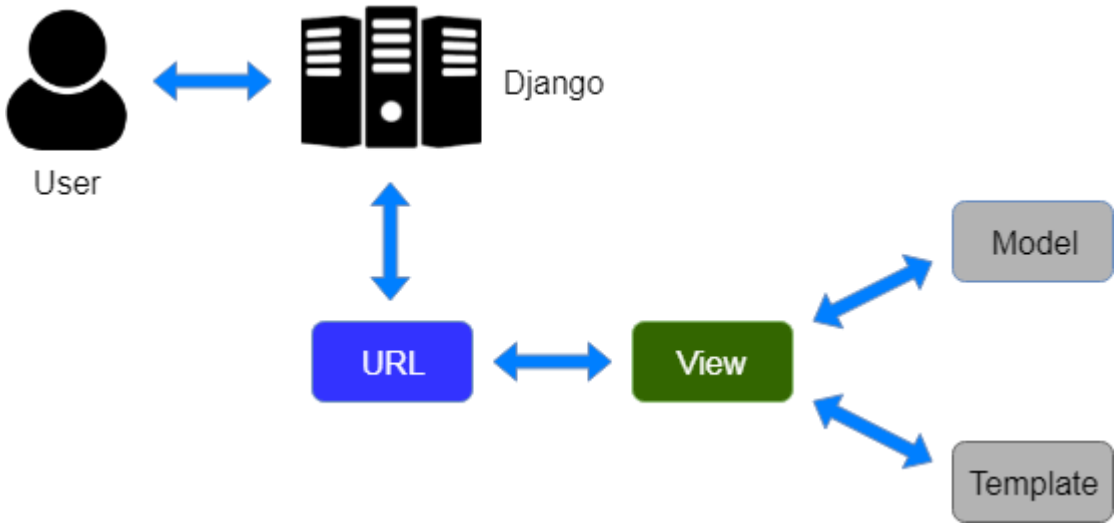
The MVT (Model View Template) is a software design pattern. It is a collection of three important components Model View and Template. The Model helps to handle database. It is a data access layer which handles the data.

The Template is a presentation layer which handles User Interface part completely. The View is used to execute the business logic and interact with a model to carry data and renders a template.

Although Django follows MVC pattern but maintains it?s own conventions. So, control is handled by the framework itself.

There is no separate controller and complete application is based on Model View and Template. That?s why it is called MVT application.

See the following graph that shows the MVT based control flow.



Here, a user **requests** for a resource to the Django, Django works as a controller and check to the available resource in URL.

If URL maps, **a view is called** that interact with model and template, it renders a template.

Django responds back to the user and sends a template as a **response**.

ACTIVITY LOG FOR THE TWELVETH WEEK

Day & Date	Brief description of the daily activity	Learning Outcome	Person In- Charge Signature
Day – 1 30-Mar-2023	HOLIDAY		
Day – 2 31-Mar-2023	Introduction of SQLite Database	Sqlite database introduction	
Day – 3 01-Apr-2023	Connect wth Sqlite Database	Connectivity of sqlite database	
Day – 4 03-Apr-2023	Connect with Mysql Database	Connectivity of mysql database	
Day – 5 04-Apr-2023	Templates in django	Templates creation	
Day –6 05-Apr-2023	HOLIDAY		

WEEKLY REPORT

Week -12 (From: 30/03/23 To: 05/04/23)

Objective of the activity done:

Detailed report:

This week, I focused on learning the basics of building web applications using Python with the Django Framework, one of the most popular and powerful web development frameworks for Python. Here are the key topics I covered:

Introduction to Django: I started by learning what Django is and how it's used to build web applications. I also learned about the basic components of a Django web application, including models, views, templates, and URLs.

Models and Databases: I learned how to define models in Django, which are used to define the structure and behavior of a database. I also learned how to create a database using Django's Object-Relational Mapping (ORM) system.

Views and Templates: I learned how to define views in Django, which are used to handle HTTP requests and generate responses. I also learned how to use Django templates to generate HTML pages based on dynamic data.

Forms: I learned how to create forms in Django, which allow users to input data and submit it to the server. I also learned how to validate form data and handle form submissions.

User Authentication: I learned how to use Django's built-in user authentication system to create user accounts, handle user logins and logouts, and restrict access to certain parts of a web application.

Overall, this week was a great introduction to building web applications using Python with the Django Framework. I feel comfortable creating basic Django applications and working with databases, views, templates, forms, and user authentication. I plan to continue practicing by building more complex Django applications and exploring different Django features and libraries.

Django Templates

Django provides a convenient way to generate dynamic HTML pages by using its template system.

A template consists of static parts of the desired HTML output as well as some special syntax describing how dynamic content will be inserted.

Why Django Template?

In HTML file, we can't write python code because the code is only interpreted by python interpreter not the browser. We know that HTML is a static markup language, while Python is a dynamic programming language.

Django template engine is used to separate the design from the python code and allows us to build dynamic web pages.

Django Template Configuration

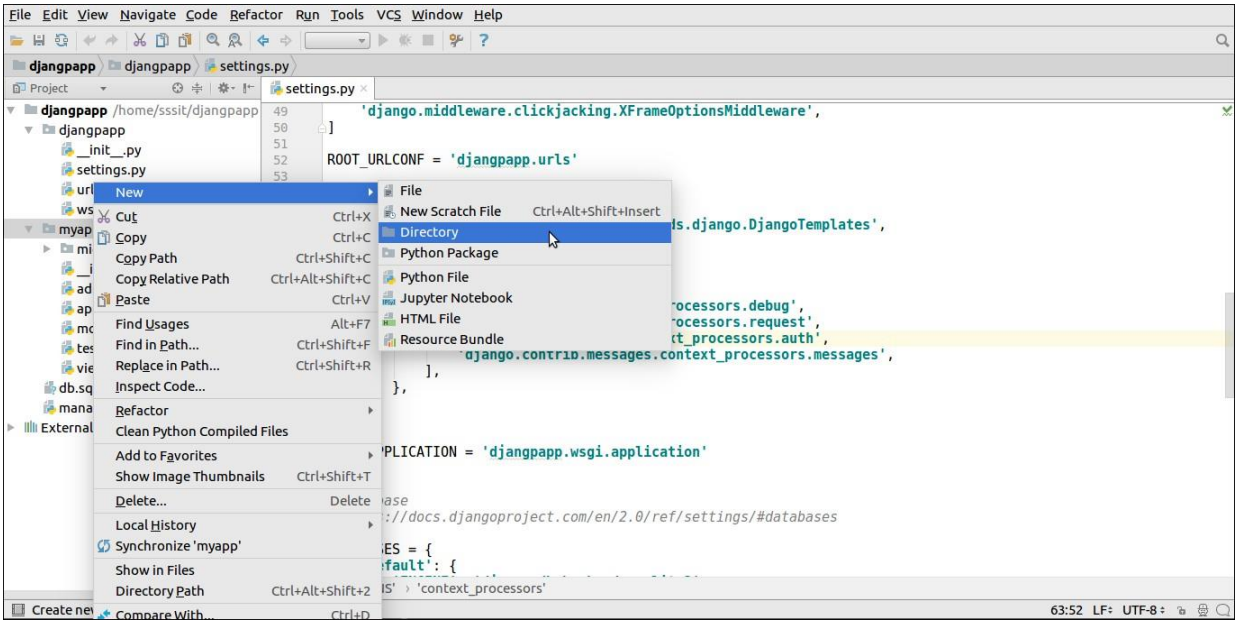
To configure the template system, we have to provide some entries in **settings.py** file.

```
TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [os.path.join(BASE_DIR,'templates')],
        'APP_DIRS': True,
        'OPTIONS': {
            'context_processors': [
                'django.template.context_processors.debug',
                'django.template.context_processors.request',
                'django.contrib.auth.context_processors.auth',
                'django.contrib.messages.context_processors.messages',
            ],
        },
    },
]
```

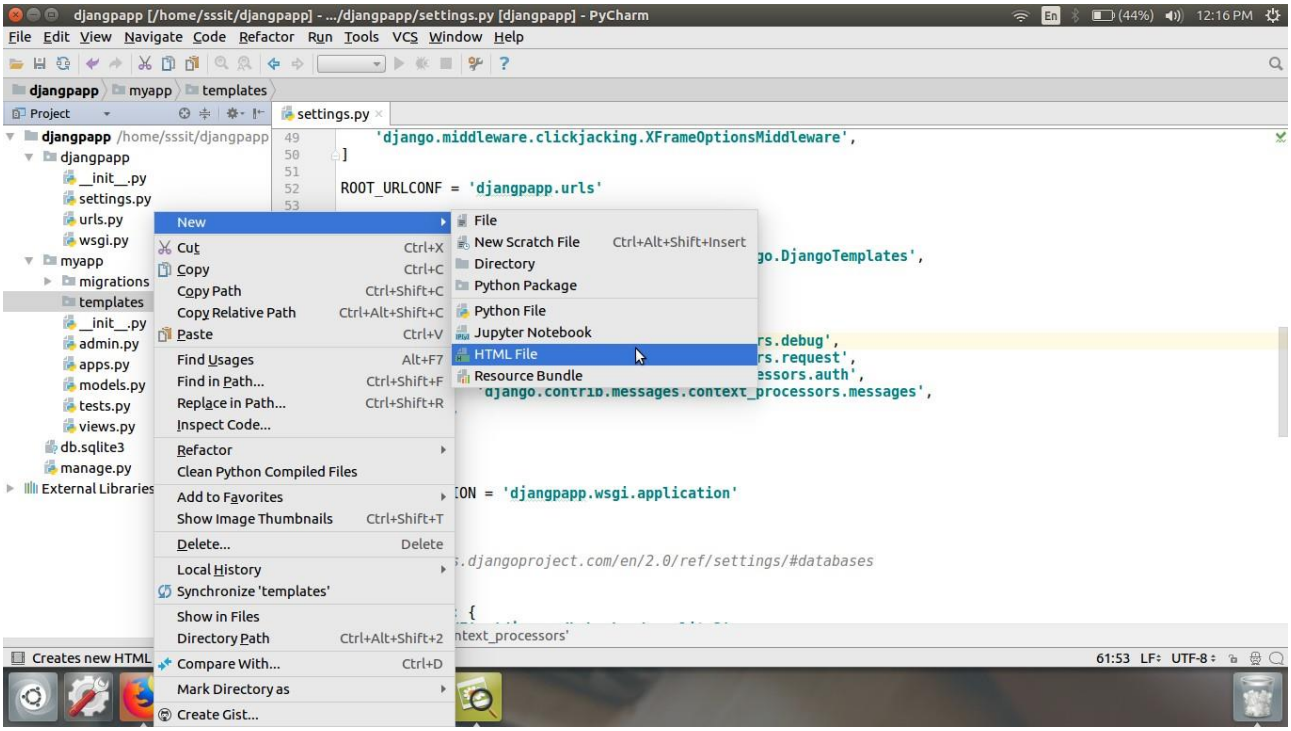
Here, we mentioned that our template directory name is **templates**. By default, DjangoTemplates looks for a **templates** subdirectory in each of the INSTALLED_APPS.

Django Template Simple Example

First, create a directory **templates** inside the project app as we did below.



After that create a template **index.html** inside the created folder.



Our template **index.html** contains the following code.

// **index.html**

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <title>Index</title>
</head>
<body>
<h2>Welcome to Django!!!</h2>
</body>
</html>
```

Loading Template

To load the template, call get_template() method as we did below and pass template name.

//views.py

```
from django.shortcuts import render
#importing loading from django template
from django.template import loader
# Create your views here.
from django.http import HttpResponse
```

```
def index(request):
    template = loader.get_template('index.html') # getting our template
    return HttpResponse(template.render())       # rendering the template in HttpResponse
```

Set a URL to access the template from the browser.

//urls.py

```
    path('index/', views.index),
```

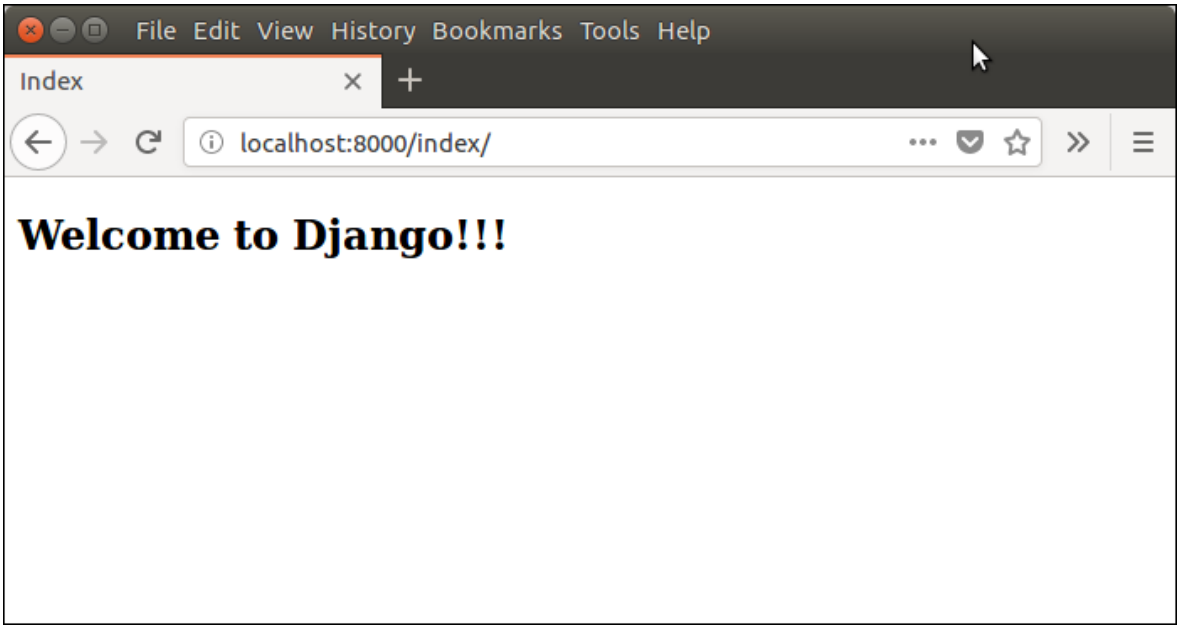
Register app inside the INSTALLED_APPS

```
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'myapp'
]
```

Run Server

Execute the following command and access the template by entering **localhost:8000/index** at the browser.

1. $ python3 manage.py runserver



ACTIVITY LOG FOR THE THRITEENTH WEEK

Day & Date	Brief description of the daily activity	Learning Outcome	Person In-Charge Signature
Day – 1 06-Apr-2023	MVT Implementation	Introduction of MVT implementation	
Day – 2 07-Apr-2023	HOLIDAY		
Day – 3 08-Apr-2023	David Warner Batting anlaysis using Pandas	Introduction of Pandas module	
Day – 4 10-Apr-2023	registration form using forms in django	Creation of registration form	
Day – 5 11-Apr-2023	Introduction of DataScience[Numpy Module]	Numpy package	
Day –6 12-Apr-2023	Programs using Numpy module	Program practice	

WEEKLY REPORT

Week -13 (From: 06/04/23 To: 12/04/23)

Objectives of the Activity Done:
Detailed Report:

This week, I focused on learning the basics of NumPy, a powerful Python library used for scientific computing and data analysis. Here are the key topics I covered:

Introduction to NumPy: I started by learning what NumPy is and how it's used to work with arrays and matrices in Python. I also learned about the key features and benefits of using NumPy for scientific computing.

Creating and Manipulating Arrays: I learned how to create NumPy arrays in Python using the np.array() function, and how to manipulate arrays using functions like np.reshape(), np.concatenate(), and np.split().

Indexing and Slicing Arrays: I learned how to access and modify individual elements in NumPy arrays using indexing and slicing techniques, as well as how to use boolean indexing to filter array elements based on specific conditions.

Basic Operations on Arrays: I learned how to perform basic arithmetic and logical operations on NumPy arrays, including addition, subtraction, multiplication, division, and comparisons.

Linear Algebra with NumPy: I learned how to use NumPy for linear algebra computations, including matrix multiplication, matrix inversion, and eigenvalue and eigenvector calculations.

Overall, this week was a great introduction to NumPy basics. I feel comfortable creating and manipulating arrays in NumPy, indexing and slicing array elements, performing basic operations on arrays, and using NumPy for linear algebra computations. I plan to continue practicing by exploring advanced NumPy features and using NumPy for data analysis and visualization tasks.

NumPy stands for numeric python which is a python package for the computation and processing of the multidimensional and single dimensional array elements.

Travis Oliphant created NumPy package in 2005 by injecting the features of the ancestor module Numeric into another module Numarray.

It is an extension module of Python which is mostly written in C. It provides various functions which are capable of performing the numeric computations with a high speed.

NumPy provides various powerful data structures, implementing multi-dimensional arrays and matrices. These data structures are used for the optimal computations regarding arrays and matrices.

In this tutorial, we will go through the numeric python library NumPy.

The need of NumPy

With the revolution of data science, data analysis libraries like NumPy, SciPy, Pandas, etc. have seen a lot of growth. With a much easier syntax than other programming languages, python is the first choice language for the data scientist.

NumPy provides a convenient and efficient way to handle the vast amount of data. NumPy is also very convenient with Matrix multiplication and data reshaping. NumPy is fast which makes it reasonable to work with a large set of data.

There are the following advantages of using NumPy for data analysis.

1. NumPy performs array-oriented computing.
2. It efficiently implements the multidimensional arrays.
3. It performs scientific computations.
4. It is capable of performing Fourier Transform and reshaping the data stored in multidimensional arrays.
5. NumPy provides the in-built functions for linear algebra and random number generation.

Nowadays, NumPy in combination with SciPy and Mat-plotlib is used as the replacement to MATLAB as Python is more complete and easier programming language than MATLAB.

Python Pandas Introduction

Pandas is defined as an open-source library that provides high-performance data manipulation in Python. The name of Pandas is derived from the word **Panel Data**, which means **an Econometrics from Multidimensional data**. It is used for data analysis in Python and developed by **Wes McKinney** in **2008**.

Data analysis requires lots of processing, such as **restructuring, cleaning** or **merging**, etc. There are different tools are available for fast data processing, such as **Numpy, Scipy, Cython**, and **Panda**. But we prefer Pandas because working with Pandas is fast, simple and more expressive than other tools.

Pandas is built on top of the **Numpy** package, means **Numpy** is required for operating the Pandas.

Before Pandas, Python was capable for data preparation, but it only provided limited support for data analysis. So, Pandas came into the picture and enhanced the capabilities of data analysis. It can perform five significant steps required for processing and analysis of data irrespective of the origin of the data, i.e., **load, manipulate, prepare, model, and analyze**.

Key Features of Pandas

- It has a fast and efficient DataFrame object with the default and customized indexing.
- Used for reshaping and pivoting of the data sets.
- Group by data for aggregations and transformations.
- It is used for data alignment and integration of the missing data.
- Provide the functionality of Time Series.

- It integrates with the other libraries such as SciPy, and scikit-learn.
- Provides fast performance, and If you want to speed it, even more, you can use the **Python**.

Benefits of Pandas

The benefits of pandas over using other language are as follows:

- **Data Representation:** It represents the data in a form that is suited for data analysis through its DataFrame and Series.
- **Clear code:** The clear API of the Pandas allows you to focus on the core part of the code. So, it provides clear and concise code for the user.

Python Pandas Data Structure

The Pandas provides two data structures for processing the data, i.e., **Series** and **DataFrame**, which are discussed below:

1) Series

It is defined as a one-dimensional array that is capable of storing various data types. The row labels of series are called the **index**. We can easily convert the list, tuple, and dictionary into series using "series' method. A Series cannot contain multiple columns. It has one parameter:

Data: It can be any list, dictionary, or scalar value.

Creating Series from Array:

Before creating a Series, Firstly, we have to import the numpy module and then use array() function in the program.

```
import pandas as pd
import numpy as np
info = np.array(['P','a','n','d','a','s'])
a = pd.Series(info)
print(a)
```

Output

```
0   P
1   a
2   n
3   d
4   a
5   s
dtype: object
```

Explanation: In this code, firstly, we have imported the **pandas** and **numpy** library with the **pd** and **np** alias. Then, we have taken a variable named "info" that consist of an array of some values. We have called the **info** variable through a **Series** method and defined it in an "**a**" variable. The Series has printed by calling the **print(a)** method.

Python Pandas DataFrame

It is a widely used data structure of pandas and works with a two-dimensional array with labeled axes (rows and columns). DataFrame is defined as a standard way to store data and has two different indexes, i.e., row index and column index. It consists of the following properties:

- The columns can be heterogeneous types like int, bool, and so on.
- It can be seen as a dictionary of Series structure where both the rows and columns are indexed. It is denoted as "columns" in case of columns and "index" in case of rows.

Create a DataFrame using List:

We can easily create a DataFrame in Pandas using list.

```
import pandas as pd
# a list of strings
x = ['Python', 'Pandas']

# Calling DataFrame constructor on list
df = pd.DataFrame(x)
print(df)
```

Output

```
     0
0  Python
1  Pandas
```

Explanation: In this code, we have defined a variable named "x" that consist of string values. The DataFrame constructor is being called on a list to print the values.

ACTIVITY LOG FOR THE FOURTEENTH WEEK

Day & Date	Brief description of the daily activity	Learning Outcome	Person In-Charge Signature
Day – 1 13-Apr-2023	Matplot lib and seaborn	Introduction of matplot lib	
Day – 2 14-Apr-2023	HOLIDAY		
Day – 3 15-Apr-2023	Flask Framework	Introduction of flask framework	
Day – 4 17-Apr-2023	Regular Expression[Find all(),search(),sub() & match()]	Explanation of regular expression	
Day – 5 18-Apr-2023	Flask framework	Basics of flask framework	
Day –6 19-Apr-2023	Tkinter module	Introduction of tkinter module	

WEEKLY REPORT

Week -14 (From: 13/04/23 To: 19/04/23)

Objectives of the Activity Done:

Detailed report:

This week, I focused on learning the basics of Matplotlib, a Python library used for creating data visualizations. Here are the key topics I covered:

Introduction to Matplotlib: I started by learning what Matplotlib is and how it's used to create various types of plots, including line plots, scatter plots, bar charts, histograms, and more.

Basic Line Plots: I learned how to create simple line plots using the plt.plot() function in Matplotlib, as well as how to customize the plot by changing the colors, line styles, markers, and labels.

Basic Scatter Plots: I learned how to create scatter plots using the plt.scatter() function in Matplotlib, and how to customize the plot by changing the marker colors, sizes, and shapes.

Basic Bar Charts and Histograms: I learned how to create bar charts and histograms using the plt.bar() and plt.hist() functions in Matplotlib, and how to customize the plot by changing the bar colors, widths, and orientations.

Subplots and Axes: I learned how to create multiple subplots and customize the axes of a plot using the plt.subplots() function and the plt.axis() method in Matplotlib.

Overall, this week was a great introduction to Matplotlib basics. I feel comfortable creating various types of plots in Matplotlib, customizing plot colors, markers, and labels, and using subplots and axes to create complex visualizations. I plan to continue practicing by exploring advanced Matplotlib features and using Matplotlib for data analysis and presentation tasks.

Matplotlib

Human minds are more adaptive for the visual representation of data rather than textual data. We can easily understand things when they are visualized. It is better to represent the data through the graph where we can analyze the data more efficiently and make the specific decision according to data analysis. Before learning the matplotlib, we need to understand data visualization and why data visualization is important.

Data Visualization

Graphics provides an excellent approach for exploring the data, which is essential for presenting results. Data visualization is a new term. It expresses the idea that involves more than just representing data in the graphical form (instead of using textual form).

This can be very helpful when discovering and getting to know a dataset and can help with classifying patterns, corrupt data, outliers, and much more. With a little domain knowledge, data visualizations can be used to express and demonstrate key relationships in plots and charts. The static does indeed focus on

quantitative description and estimations of data. It provides an important set of tools for gaining a qualitative understanding.

There are five key plots that are used for data visualization.

There are five phases which are essential to make the decision for the organization:

- **Visualize:** We analyze the raw data, which means it makes complex data more accessible, understandable, and more usable. Tabular data representation is used where the user will look up a specific measurement, while the chart of several types is used to show patterns or relationships in the data for one or more variables.
- **Analysis:** Data analysis is defined as cleaning, inspecting, transforming, and modeling data to derive useful information. Whenever we make a decision for the business or in daily life, is by past experience. **What will happen to choose a particular decision**, it is nothing but analyzing our past. That may be affected in the future, so the proper analysis is necessary for better decisions for any business or organization.
- **Document Insight:** Document insight is the process where the useful data or information is organized in the document in the standard format.
- **Transform Data Set:** Standard data is used to make the decision more effectively.

Why need data visualization?

Data visualization can perform below tasks:

- It identifies areas that need improvement and attention.
- It clarifies the factors.
- It helps to understand which product to place where.
- Predict sales volumes.

Python Regex

A regular expression is a set of characters with highly specialized syntax that we can use to find or match other characters or groups of characters. In short, regular expressions, or Regex, are widely used in the UNIX world.

The re-module in Python gives full support for regular expressions of Pearl style. The re module raises the re.error exception whenever an error occurs while implementing or using a regular expression.

We'll go over two crucial functions utilized to deal with regular expressions. But first, a minor point: many letters have a particular meaning when utilized in a regular expression.

re.match()

Python's re.match() function finds and delivers the very first appearance of a regular expression pattern. In Python, the RegEx Match function solely searches for a matching string at the beginning of the provided text to be searched. The matching object is produced if one match is found in the first line. If a match is found in a subsequent line, the Python RegEx Match function gives output as null.

Examine the implementation for the re.match() method in Python. The expressions ".w*" and ".w*?" will match words that have the letter "w," and anything that does not has the letter "w" will be ignored. The for loop is used in this Python re.match() illustration to inspect for matches for every element in the list of words.

Matching Characters

The majority of symbols and characters will easily match. (A case-insensitive feature can be enabled, allowing this RE to match Python or PYTHON.) The regular expression check, for instance, will match exactly the string check.

There are some exceptions to this general rule; certain symbols are special metacharacters that don't match. Rather, they indicate that they must compare something unusual, or they have an effect on other parts of the RE by recurring or modifying their meaning.

Here's the list of the metacharacters;

1. . ^ $ * + ? { } [ ] \ | ( )

Repeating Things

The ability to match different sets of symbols will be the first feature regular expressions can achieve that's not previously achievable with string techniques. On the other hand, Regexes isn't much of an improvement if that had been their only extra capacity. We can also define that some sections of the RE must be reiterated a specified number of times.

The first metacharacter we'll examine for recurring occurrences is *. Instead of matching the actual character '*,' * signals that the preceding letter can be matched 0 or even more times, rather than exactly one.

Ba*t, for example, matches 'bt' (zero 'a' characters), 'bat' (one 'a' character), 'baaat' (three 'a' characters), etc.

Greedy repetitions, such as *, cause the matching algorithm to attempt to replicate the RE as many times as feasible. If later elements of the sequence fail to match, the matching algorithm will retry with lesser repetitions.

This is the syntax of re.match() function -

```
re.match(pattern, string, flags=0)
```

Parameters

pattern:- this is the expression that is to be matched. It must be a regular expression

string:- This is the string that will be compared to the pattern at the start of the string.

flags:- Bitwise OR (|) can be used to express multiple flags. These are modifications, and the table below lists them.

Code

```
import re
line = "Learn Python through tutorials on javatpoint"
match_object = re.match( r'.w* (.w?) (.w*?)', line, re.M|re.I)

if match_object:
   print ("match object group : ", match_object.group())
   print ("match object 1 group : ", match_object.group(1))
   print ("match object 2 group : ", match_object.group(2))
else:
   print ( "There isn't any match!!" )
```

Output:

```
There isn't any match!!
```

re.search()

The re.search() function will look for the first occurrence of a regular expression sequence and deliver it. It will verify all rows of the supplied string, unlike Python's re.match(). If the pattern is matched, the re.search() function produces a match object; otherwise, it returns "null."

To execute the search() function, we must first import the Python re-module and afterward run the program. The "sequence" and "content" to check from our primary string are passed to the Python re.search() call.

This is the syntax of re.search() function -

```
re.search(pattern, string, flags=0)
```

Here is the description of the parameters -

pattern:- this is the expression that is to be matched. It must be a regular expression

string:- The string provided is the one that will be searched for the pattern wherever within it.

flags:- Bitwise OR (|) can be used to express multiple flags. These are modifications, and the table below lists them.

Code

```
import re
line = "Learn Python through tutorials on javatpoint";
search_object = re.search( r' .*t? (.*t?) (.*t?)', line)
if search_object:
   print("search object group : ", search_object.group())
   print("search object group 1 : ", search_object.group(1))
   print("search object group 2 : ", search_object.group(2))
else:
   print("Nothing found!!")
```

Output:

```
search object group :  Python through tutorials on javatpoint
search object group 1 :  on
search object group 2 :  javatpoint
```

ACTIVITY LOG FOR THE FIFTEENTH WEEK

Day & Date	Brief description of the daily activity	Learning Outcome	Person In-Charge Signature
Day – 1 20-Apr-2023	HOLIDAY		
Day – 2 21-Apr-2023	HOLIDAY		
Day – 3 22-Apr-2023	HOLIDAY		
Day – 4 24-Apr-2023	Python regular expressions	Introduction of regular expressions	
Day – 5 25-Apr-2023	Python multithreading programming	Introduction of multithreading programming	
Day –6 26-Apr-2023	Database handling with MySql	Database creation	

WEEKLY REPORT

Week -15 (From: 20/04/23 To: 26/04/23)

Objectives of the Activity Done:

Detailed report:

This week, I focused on learning the basics of Python multiprocessing, a module used for executing multiple processes in parallel on a multi-core CPU. Here are the key topics I covered:

Python Multiprocessing

In this article, we will learn how we can achieve multiprocessing using Python. We also discuss its advanced concepts.

What is Multiprocessing?

Multiprocessing is the ability of the system to run one or more processes in parallel. In simple words, multiprocessing uses the two or more CPU within the single computer system. This method is also capable to allocate the tasks between more than one process.

Processing units share the main memory and peripherals to process programs simultaneously. Multiprocessing Application breaks into smaller parts and runs independently. Each process is allocated to the processor by the operating system.

Python provides the built-in package called multiprocessing which supports swapping processes. Before working with the multiprocessing, we must aware with the process object.

Why Multiprocessing?

Multiprocessing is essential to perform the multiple tasks within the Computer system. Suppose a computer without multiprocessing or single processor. We assign various processes to that system at the same time.

It will then have to interrupt the previous task and move to another to keep all processes going. It is as simple as a chef is working alone in the kitchen. He has to do several tasks to cook food such as cutting, cleaning, cooking, kneading dough, baking, etc.

Therefore, multiprocessing is essential to perform several task at the same time without interruption. It also makes easy to track all the tasks. That is why the concept of multiprocessing is to arise.

- Multiprocessing can be represented as a computer with more than one central processor.

- A Multi-core processor refers to single computing component with two or more independent units.

In the multiprocessing, the CPU can assign multiple tasks at one each task has its own processor.

Multiprocessing In Python

Python provides the multiprocessing module to perform multiple tasks within the single system. It offers a user-friendly and intuitive API to work with the multiprocessing.

Let's understand the simple example of multiple processing.

Example -

```
from multiprocessing import Process
  def disp():
    print ('Hello !! Welcome to Python Tutorial')
    if__name__== '_main_':
    p = Process(target=disp)
    p.start()
    p.join()
```

Output:

```
'Hello !! Welcome to Python Tutorial'
```

Creating new databases

In this section of the tutorial, we will create the new database PythonDB.

ACTIVITY LOG FOR THE SIXTEENTH WEEK

Day & Date	Brief description of the daily activity	Learning Outcome	Person In-Charge Signature
Day – 1 27-Apr-2023	Django URL Mapping	Introduction of URL mapping	
Day – 2 28-Apr-2023	Django File uploading	File uploading	
Day – 3 29-Apr-2023	Sessions, caching and comments	Creating sessions and caching	
Day – 4 01-May-2023	Handling database errors	Database errors	
Day – 5 02-May-2023	Diconnecting database	Disconnecting database	
Day –6 03-May-2023	SQL and NOSQL	Introduction of sql and nosql0020	

WEEKLY REPORT

Week -16 (From: 27/04/23 To: 03/05/23)

Objectives of the Activity Done:

Detailed report:

This week, I focused on learning the basics of URL mapping in Python Django, a web framework used for building dynamic web applications. Here are the key topics I covered:

Introduction to URL Mapping: I started by learning what URL mapping is and how it's used in Django to map URLs to views, which are Python functions that generate HTTP responses.

Basic URL Patterns: I learned how to define basic URL patterns in Django using the urls.py module, and how to map URLs to views using the path() and re_path() functions.

Capturing URL Parameters: I learned how to capture parameters from URLs using regular expressions, and how to pass these parameters to views as function arguments.

Naming URL Patterns: I learned how to give names to URL patterns in Django using the name attribute, and how to use these names to generate URLs dynamically in templates and views.

Including Other URL Patterns: I learned how to include other URL patterns in Django using the include() function, and how to organize URL patterns into multiple apps and modules for better code organization.

Overall, this week was a great introduction to Python Django URL mapping basics. I feel comfortable defining basic URL patterns, capturing URL parameters using regular expressions, giving names to URL patterns, including other URL patterns, and organizing URL patterns in multiple apps and modules. I plan to continue practicing by exploring advanced URL mapping features in Django and using URL mapping for building dynamic web applications.

Django URL Mapping

Well, till here, we have learned to create a model, view, and template. Now, we will learn about the routing of application.

Since Django is a web application framework, it gets user requests by URL locater and responds back. To handle URL, **django.urls** module is used by the framework.

Let's open the file **urls.py** of the project and see the what it looks like:

// **urls.py**

```
from django.contrib import admin
from django.urls import path

urlpatterns = [
    path('admin/', admin.site.urls),
]
```

See, Django already has mentioned a URL here for the admin. The path function takes the first argument as a route of string or regex type.

The view argument is a view function which is used to return a response (template) to the user.

The **django.urls** module contains various functions, **path(route,view,kwargs,name)** is one of those which is used to map the URL and call the specified view.

Django URL Functions

Here, we are giving some commonly used functions for URL handling and mapping.

Name	Description	Example
path(route, view, kwargs=None, name=None)	It returns an element for inclusion in urlpatterns.	path('index/', views.index, name='main-view')
re_path(route, view, kwargs=None, name=None)	It returns an element for inclusion in urlpatterns.	re_path(r'^index/$', views.index, name='index'),
include(module, namespace=None)	It is a function that takes a full Python import path to another URLconf module that should be "included" in this place.	
register_converter(converter, type_name)	It is used for registering a converter for use in path() routes.	

Django File Upload

File upload to the server using Django is a very easy task. Django provides built-in library and methods that help to upload a file to the server.

The **forms.FileField()** method is used to create a file input and submit the file to the server. While working with files, make sure the HTML form tag contains **enctype="multipart/form-data"** property.

Let's see an example of uploading a file to the server. This example contains the following files.

Template (index.html)

It will create an HTML form which contains a file input component.

```
<body>
<form method="POST" class="post-form" enctype="multipart/form-data">
    {% csrf_token %}
    {{ form.as_p }}
    <button type="submit" class="save btn btn-default">Save</button>
</form>
</body>
```

Form (forms.py)

```
from django import forms
class StudentForm(forms.Form):
   firstname = forms.CharField(label="Enter first name",max_length=50)
   lastname = forms.CharField(label="Enter last name", max_length = 10)
   email     = forms.EmailField(label="Enter Email")
   file      = forms.FileField() # for creating file input
```

View (views.py)

Here, one extra parameter **request.FILES** is required in the constructor. This argument contains the uploaded file instance.

```
from django.shortcuts import render
from django.http import HttpResponse
from myapp.functions.functions import handle_uploaded_file
from myapp.form import StudentForm
def index(request):
   if request.method == 'POST':
      student = StudentForm(request.POST, request.FILES)
      if student.is_valid():
         handle_uploaded_file(request.FILES['file'])
         return HttpResponse("File uploaded successfuly")
   else:
      student = StudentForm()
      return render(request,"index.html",{'form':student})
```

Specify URL (urls.py)

```
from django.contrib import admin
from django.urls import path
from myapp import views
urlpatterns = [
    path('admin/', admin.site.urls),
    path('index/', views.index),
]
```

Upload Script (functions.py)

This function is used to read the uploaded file and store at provided location. Put this code into the **functions.py** file. But first create this file into the project.

```
def handle_uploaded_file(f):
    with open('myapp/static/upload/'+f.name, 'wb+') as destination:
        for chunk in f.chunks():
            destination.write(chunk)
```

Django Session

A session is a mechanism to store information on the server side during the interaction with the web application.

In Django, by default session stores in the database and also allows file-based and cache based sessions. It is implemented via a piece of middleware and can be enabled by using the following code.

Put **django.contrib.sessions.middleware.SessionMiddleware** in MIDDLEWARE and **django.contrib.sessions** in INSTALLED_APPS of settings.py file.

To set and get the session in views, we can use **request.session** and can set multiple times too.

The **class backends.base.SessionBase** is a base class of all session objects. It contains the following standard methods.

Method	Description
__getitem__(key)	It is used to get session value.
__setitem__(key, value)	It is used to set session value.
__delitem__(key)	It is used to delete session object.
__contains__(key)	It checks whether the container contains the particular session object or not.
get(key, default=None)	It is used to get session value of the specified key.

ACTIVITY LOG FOR THE SEVENTEENTH WEEK

Day & Date	Brief description of the daily activity	Learning Outcome	Person In-Charge Signature
Day – 1 04-May-2023	Document Preparation - Part1	Summary Typing	
Day – 2 05-May-2023	Document Preparation – Part2	Daily Report Typing	
Day – 3 06-May-2023	Document Preparation – Part3	Activity Report Typing	
Day – 4 08-May-2023	Document Preparation – Part4	Internship Outcomes	
Day – 5 09-May-2023	Document Preparation – Part5	Evaluation Typing	
Day –6 10-May-2023	Document Preparation – Part6	Invigilator Support	

CHAPTER 5: OUTCOMES DESCRIPTION

Describe the work environment you have experienced (in terms of people interactions, facilities available and maintenance, clarity of job roles, protocols, procedures, processes, discipline, time management, harmonious relationships, socialization, mutual support and teamwork, motivation, space and ventilation, etc.)

The work environment refers to the physical and social setting in which work is conducted. The quality of the work environment can significantly impact employee satisfaction, productivity, and well-being. In terms of people interactions, a positive work environment should promote harmonious relationships, socialization, mutual support, and teamwork among employees. Facilities available and their maintenance should be adequate and well- maintained. Clarity of job roles, protocols, procedures, and processes can help ensure employees understand their responsibilities and perform their duties efficiently. Discipline is necessary to maintain a productive work environment, and time management is essential to ensure tasks are completed on time. Ventilation and space should be considered to maintain a comfortable and safe work environment. Motivation should also be fostered to encourage employees to be engaged and productive. Overall, a positive work environment should provide employees with the necessary support and resources to excel in their roles.

Describe the real time technical skills you have acquired (in terms of the job- related skills and hands on experience)

In terms of technical skills related to Python, as a student, I have acquired a basic understanding of python language and database connection. Through my course work, I have learned python packages like numpy, pandas, matplotlib.

As a student, I have also learned about Python library used for creating interactive web applications for machine learning projects. I understand how Python allows developers to build user interfaces that enable users to interact with machine learning models and visualize data in real-time. I have developed my skills in Python programming.

Describe the managerial skills you have acquired (in terms of planning, leadership, team work, behaviour, workmanship, productive use of time, weekly improvement in competencies, goal setting, decision making, performance analysis, etc.

As a student, I have acquired several managerial skills through my academic and extracurricular activities. I have developed planning skills through setting short-term and long-term academic goals and creating study schedules. I understand the importance of effective leadership, and I have developed my leadership skills by taking on leadership roles in student organizations and group projects.

Working collaboratively with other students in group projects and extracurricular activities has helped me to develop teamwork skills. I recognize the significance of behavior in achieving academic and personal success, and I have developed good work ethics and professional behaviour.

I believe in utilizing time productively, and I have developed time management skills by prioritizing tasks and creating schedules to achieve academic and personal goals. I am committed to continuous learning and have developed weekly improvement strategies to improve my competencies in various areas.

Goal setting is an essential managerial skill, and I have learned how to set SMART (specific, measurable, attainable, relevant, and time-bound) goals for my academic and personal life. I have also learned decision-making skills by analysing information and making sound decisions based on facts and logical reasoning.

Performance analysis is crucial for self-improvement, and I have developed skills in analyzing my academic performance to identify areas that need improvement. I have also learned how to provide constructive feedback to team members and accept feedback from others to enhance team performance.

Describe how you could improve your communication skills (in terms of improvement in oral communication, written communication, conversational abilities, confidence levels while communicating, anxiety management, understanding others, getting understood by others, extempore speech, ability to articulate the key points, closing the conversation, maintaining niceties and protocols, greeting, thanking and appreciating others, etc.,)

As a student, I recognize the importance of effective communication skills, and I am committed to improving my oral and written communication abilities. To improve my oral communication skills, I plan to practice extempore speeches and presentations to enhance my ability to articulate key points clearly and confidently. I also plan to work on managing anxiety while communicating by using techniques such as deep breathing and visualization.

To improve my written communication skills, I plan to read more extensively to improve my vocabulary and grammar skills. I also plan to practice writing and seek feedback from others to improve my writing style and clarity.

I am aware that conversational abilities play a vital role in effective communication, and I plan to work on my active listening skills to understand others better. I also plan to practice summarizing what others have said to ensure that I have understood them correctly.

Confidence is an essential aspect of effective communication, and I plan to work on building my confidence levels by practicing communication in various contexts. I will also focus on maintaining niceties and protocols, such as greeting, thanking, and appreciating others, to build positive relationships.

I understand the importance of being understood by others, and I plan to work on my clarity of expression by paying attention to my tone, pace, and pronunciation. I will also practice using simple and straightforward language to convey my thoughts and ideas effectively.

Closing a conversation is also a crucial aspect of effective communication, and I plan to work on wrapping up discussions concisely and positively. I will practice ending conversations on a positive note to build a good rapport with others.

Describe how could you could enhance your abilities in group discussions, participation in teams, contribution as a team member, leading a team/activity.

As a student, I recognize the importance of participating effectively in group discussions, contributing to teams, and leading team activities. To enhance my abilities in group discussions, I plan to improve my active listening skills and learn how to ask effective questions that encourage discussion. I also plan to research and prepare before the discussion to contribute to the conversation effectively.

To improve my participation as a team member, I plan to develop my teamwork skills, including effective communication, collaboration, and problem-solving. I will also work on being reliable and accountable by meeting deadlines and fulfilling commitments. I believe that constructive feedback is essential to improving as a team member, and I plan to seek feedback from others to identify areas for improvement.

As I develop my skills as a team member, I plan to work towards taking on leadership roles in team activities. To develop my leadership skills, I will observe and learn from effective leaders and seek opportunities to lead team activities. I will also work on my ability to motivate and inspire team members to achieve common goals.

I understand that effective team leadership requires a clear understanding of team dynamics, and I plan to develop my skills in this area. I will learn how to delegate tasks, provide constructive feedback, and resolve conflicts within the team. I also plan to work on my decision-making skills to make informed and timely decisions for the benefit of the team.

Overall, I recognize that contributing effectively as a team member and leading a team requires a combination of communication, collaboration, and leadership skills. Through practice and seeking feedback, I plan to develop and refine these skills to achieve success as a team member and leader.

Student Self Evaluation of the Short-Term Internship

Student Name: Reddy Cherala Gurucharan varma **Registration No:** K2001543

Term of Internship: **From:** 09/01/2023 **To:** 10/05/2023

Date of Evaluation:

Organization Name & Address: ANVHAYA Technical Solutions Private Limited, Plot No.8.9.10. 4th Floor, Puppalaguda, Rajendranager, Manikonda, ,Hyderabad -500 089.

Please rate your performance in the following areas:

Rating Scale: **Letter grade of CGPA calculation to be provided**

1	**Oral Communication**	**1**	**2**	**3**	**4**	**5**
2	Written Communication	1	2	3	4	5
3	Proactiveness	1	2	3	4	5
4	Interaction ability with community	1	2	3	4	5
5	Positive Attitude	1	2	3	4	5
6	Self-confidence	1	2	3	4	5
7	Ability to learn	1	2	3	4	5
8	Work Plan and organization	1	2	3	4	5
9	Professionalism	1	2	3	4	5
10	Creativity	1	2	3	4	5
11	Quality of work done	1	2	3	4	5
12	Time Management	1	2	3	4	5
13	Understanding the Community	1	2	3	4	5
14	Achievements of Desired Outcome	1	2	3	4	5
15	**OVERALL PERFORMANCE**	**1**	**2**	**3**	**4**	**5**

Date: **Signature of the Student**

Evaluation of the Supervisor of the Intern Organization

Student Name: Reddy Cherala Gurucharan varma **Registration No:** K2001543

Term of Internship: **From:** 09/01/2023 **To:** 10/05/2023

Date of Evaluation:

Organization Name & Address: ANVHAYA Technical Solutions Private Limited, Plot No.8.9.10. 4th Floor, Puppalaguda, Rajendranager, Manikonda ,Hyderabad -500 089.

Please rate your performance in the following areas:

Rating Scale: **Letter grade of CGPA calculation to be provided**

1	**Oral Communication**	**1**	**2**	**3**	**4**	**5**
2	Written Communication	1	2	3	4	5
3	Proactiveness	1	2	3	4	5
4	Interaction ability with community	1	2	3	4	5
5	Positive Attitude	1	2	3	4	5
6	Self-confidence	1	2	3	4	5
7	Ability to learn	1	2	3	4	5
8	Work Plan and organization	1	2	3	4	5
9	Professionalism	1	2	3	4	5
10	Creativity	1	2	3	4	5
11	Quality of work done	1	2	3	4	5
12	Time Management	1	2	3	4	5
13	Understanding the Community	1	2	3	4	5
14	Achievements of Desired Outcome	1	2	3	4	5
15	**OVERALL PERFORMANCE**	**1**	**2**	**3**	**4**	**5**

Date: **Signature of the Supervisor**

PHOTOS & VIDEO LINKS

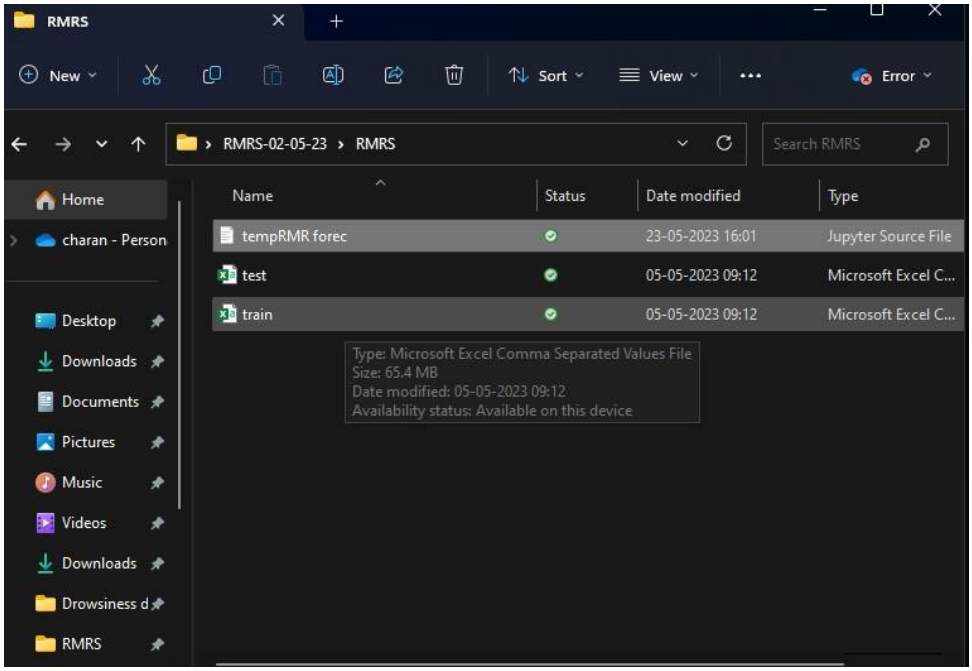
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import GridSearchCV
import time
```

```
train=pd.read_csv('train.csv')
train
```

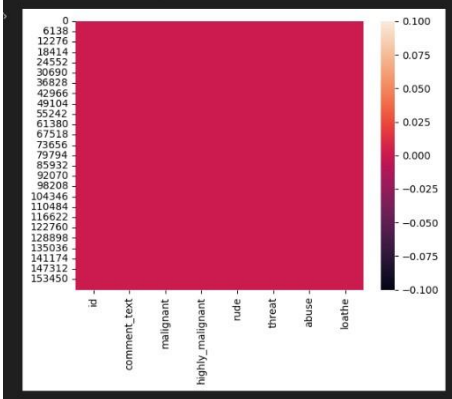
	id	comment_text	malignant	highly_malignant	rude	threat	abuse	loathe
0	0000997932d777bf	Explanation\nWhy the edits made under my usern...	0	0	0	0	0	0
1	000103f0d9cfb60f	D'aww! He matches this background colour I'm s...	0	0	0	0	0	0
2	000113f07ec002fd	Hey man, I'm really not trying to edit war. It...	0	0	0	0	0	0
3	0001b41b1c6bb37e	"\nMore\nI can't make any real suggestions on ...	0	0	0	0	0	0
4	0001d958c54c6e35	You, sir, are my hero. Any chance you remember...	0	0	0	0	0	0
...	...	...	...	...	...	...	...	...
159566	ffe987279560d7ff	":::::And for the second time of asking, when ...	0	0	0	0	0	0
159567	ffea4adeee384e90	You should be ashamed of yourself \n\nThat is ...	0	0	0	0	0	0
159568	ffee36eab5c267c9	Spitzer \n\nUmm, theres no actual article for ...	0	0	0	0	0	0
159569	fff125370e4aaaf3	And it looks like it was actually you who put ...	0	0	0	0	0	0
159570	fff46fc426af1f9a	"\nAnd ... I really don't think you understand...	0	0	0	0	0	0

159571 rows × 8 columns



```
# checking null values
print(train.isnull().sum())
print(sns.heatmap(train.isnull()))
```

```
id                0
comment_text      0
malignant         0
highly_malignant  0
rude              0
threat            0
abuse             0
loathe            0
dtype: int64
Axes(0.125,0.11;0.62x0.77)
```



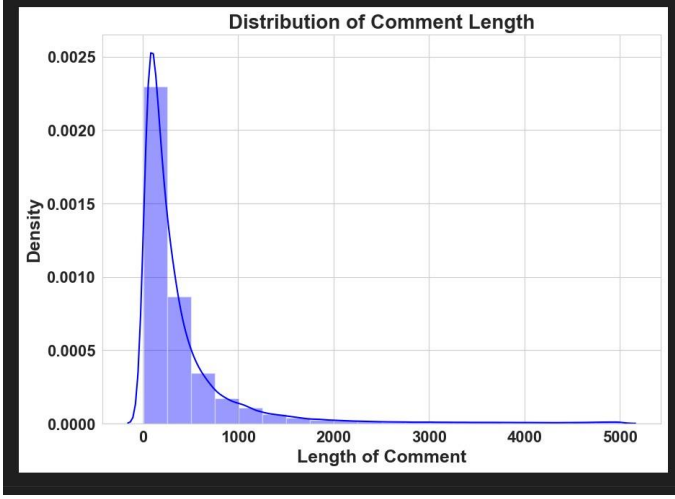

```
C:\Users\charan varma\AppData\Local\Temp\ipykernel_22956\2080975542.py:5: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(comment_len, bins=20, color = 'blue')
```

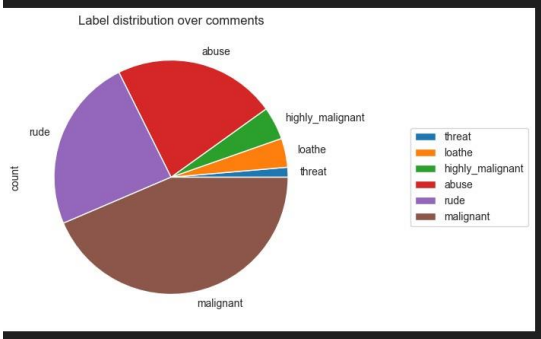


```
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
```

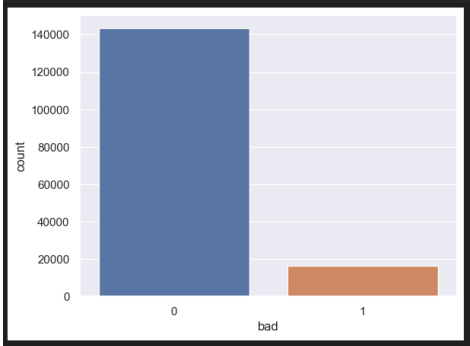
```
cols_target = ['malignant','highly_malignant','rude','threat','abuse','loathe']
df_distribution = train[cols_target].sum()\
                  .to_frame()\
                  .rename(columns={0: 'count'})\
                  .sort_values('count')

df_distribution.plot.pie(y='count',
                         title='Label distribution over comments',
                         figsize=(5, 5))\
                  .legend(loc='center left', bbox_to_anchor=(1.3, 0.5))
```

```
<matplotlib.legend.Legend at 0x290e2dacdc0>
```



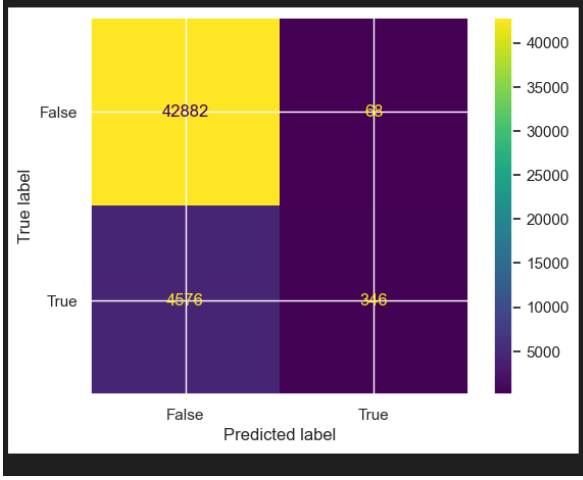

```
sns.set()
sns.countplot(x="bad" , data = train)#bad and good comments
plt.show()
```



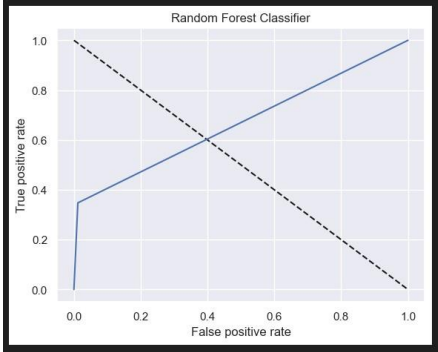
```python
from sklearn import metrics
confusion_matrix = metrics.confusion_matrix(y_test_normal,y_pred_test1)

cm_display = metrics.ConfusionMatrixDisplay(confusion_matrix = confusion_matrix, display_labels = [False, True])

cm_display.plot()
plt.show()
```




```python
fpr,tpr,thresholds=roc_curve(y_test_normal,y_pred_test3)
roc_auc=auc(fpr,tpr)
plt.plot([0,1],[1,0],'k--')
plt.plot(fpr,tpr,label = 'Random Forest Classifier')
plt.xlabel('False positive rate')
plt.ylabel('True positive rate')
plt.title('Random Forest Classifier')
plt.show()
```



EVALUATION

Internal & External Evaluation for Semester Internship

Objectives:

- Explore career alternatives prior to graduation.
- To assess interests and abilities in the field of study.
- To develop communication, interpersonal and other critical skills in the future job.
- To acquire additional skills required for the world of work.
- To acquire employment contacts leading directly to a full-time job following graduation from college.

Assessment Model:

- There shall be both internal evaluation and external evaluation
- The Faculty Guide assigned is in-charge of the learning activities of the students and for the comprehensive and continuous assessment of the students.
- The assessment is to be conducted for 200 marks. Internal Evaluation for 50 marks and External Evaluation for 150 marks
- The number of credits assigned is 12. Later the marks shall be converted into grades and grade points to include finally in the SGPA and CGPA.
- The weightings for Internal Evaluation shall be:
    - Activity Log 10marks
    - Internship Evaluation 30marks
    - Oral Presentation 10marks
- The weightings for External Evaluation shall be:
    - Activity Log 10marks
    - Internship Evaluation 30marks
    - Oral Presentation 10marks
- The External Evaluation shall be conducted by an Evaluation Committee comprising of the Principal, Faculty Guide, Internal Expert and External Expert nominated by the affiliating University. The Evaluation Committee shall also consider the grading given by the Supervisor of the Intern Organization.
- Activity Log is the record of the day-to-day activities. The Activity Log is assessed on an individual basis, thus allowing for individual members within groups to be assessed this way. The assessment will take into consideration the individual student's involvement in the assigned work.

- While evaluating the student's Activity Log, the following shall be considered –
    - a. The individual student's effort and commitment.
    - b. The originality and quality of the work produced by the individual student.
    - c. The student's integration and co-operation with the work assigned.
    - d. The completeness of the
- The Internship Evaluation shall include the following components and based on Weekly Reports and Outcomes Description
    - Description of the Work Environment.
    - Real Time Technical Skills acquired.
    - Managerial Skills acquired.
    - Improvement of Communication Skills.
    - Team Dynamics.
    - Technological Developments recorded.

MARKS STATEMENT

(To be used by the Examiners)

INTERNAL ASSESSMENT STATEMENT

Name of the Student :

Programme of Study :

Year of Study :

Group :

Register No/H.T. No :

Name of the College :

University :

Sl. No.	Evaluation Criterion	Maximum Marks	Marks Awarded
1.	Activity Log	10	
2.	Internship Evaluation	30	
3.	Oral Presentation	10	
	GRAND TOTAL	50	

Date: **Signature of Faculty Guide**

EXTERNAL ASSESSMENT STATEMENT

Name of the Student :

Programme of Study :

Year of Study :

Group :

Register No/H.T. No :

Name of the College :

University :

Sl. No.	Evaluation Criterion	Maximum Marks	Marks Awarded
1.	Internship Evaluation	80	
2.	For the grading giving by the Supervisor of Intern Organization	20	
3.	Viva Voice	50	
	TOTAL	150	
GRAND TOTAL (EXT 150M + INT. 50M)		200	

Signature of Faculty Guide

Signature of Internal Expert

Signature of External Expert

Signature of Principal with Seal