**Python**

**List**

If we want to represent a group of values as a single entity where insertion order is preserved and duplicates are also allowed then we go for list.

* We create list as [ ].
* Heterogeneous objects are allowed and list are mutable. Eg: a=[10,20,True, “hi”].
* Indexing and slicing concept is applicable in list.
* We can add outer value such as a.append(30).
* We can remove the values as a.remove(30), if there are duplicates then the first occurrence value are removed.
* If we want single value we can simply iterate it. For x in a: print(x)

**Tuples**

It is exactly same as list but immutable i.e cannot be changed. Tuples are also called immutable list.

* We can create tuples as t=(10,20,30).
* For single value of tuple we do t=(10,).
* We cannot add or remove values from tuples.

**Set**

If we want to represent values without duplicate where insertion order is not preserved then we go for set.

* Heterogeneous object are allowed.
* It is mutable.
* Indexing and slicing concept is not applicable as there is not ordering concept.
* We can add or remove items as a.add(10) and a.remove(10).
* We create set as ‘{}’ : a={10,20,’hi’}.
* We create single set as a=set()

**Frozen Set**

Similar to set it is immutable i.e set cannot be changed. To make a set frozen set we do as: s={10,20}

fs=frozenset(s).

**Dict**

If we want to represent the value with its key then we prefer dict. We create it as d={200:”hari”, 300:”Bhaskar”}.

* If we want to add new value then we do as : d[400]=”Shyam”.
* d[200] = “sita” will replace hari by sita. So duplication in key is not possible as it will replace its value.
* Duplication in value is possible as d[500] =”bhaskar” then it will be added in the dict.
* It is mutable but not ordered.
* Indexing and slicing is not possible.

**Range**

Range is a certain interval that represent a sequence of numbers. It only accept the numeric values.

* Form 1: range(10) This represent the numbers from 0 to 9.
* Form 2: range(10,20) This represents the numbers from 10 to 19.
* Form 3: range(10,20,2) This represents the numbers from 10 to 19 in 2 gap interval as 10,12,14,16,18.
* For values we iterate the range as for i in range(10): print(i)
* It can also go in reverse order i.e range(20,10,-1). Here interval is compulsory.
* In range we cannot modify values.

Bytes

It is the collection of bytes.

* The values of bytes range from 0 to 255 only. The values out of the range is unacceptable.
* It is represented as b=[10,20,30] by= bytes(b)
* Indexing and slicing is not possible.
* Immutable so we cannot add or remove the values
* Bytes don’t accept heterogeneous values.

**Bytes array**

Exactly same as bytes but it is mutable i.e we can add or remove the values.

* We represent it as by= bytearray(b).
* We can do the operation as b[0] =200.

**None**

None means nothing. There is no value.

* It makes eligible for garbage collection.
* When there is not a return value mentioned then None is returned.

**HTML**

**Module 4: Images, Multimedia, and Semantic HTML**

**Working with Images**

**1. The <img> Tag**

The <img> tag is used to embed images in a webpage. It is an empty tag, meaning it does not have a closing tag. Eg: <img src="image.jpg" alt="A beautiful landscape" width="400" height="300">

src: The source URL of the image.

* Example: src="image.jpg"

alt: Alternative text for the image, displayed if the image fails to load or for screen readers.

* Example: alt="A description of the image"

width: Defines the width of the image (can be in pixels or percentages).

* Example: width="300" or width="50%"

height: Defines the height of the image (can also be in pixels or percentages).

* Example: height="200"

**2. Responsive Images**

To make images responsive, ensuring they adapt to different screen sizes, we can do it by using srcset attribute directly in the <img> tag to serve different images based on screen size or resolution.

Eg: <img src="default.jpg"

srcset="small.jpg 300w, medium.jpg 768w, large.jpg 1200w"

sizes="(max-width: 600px) 300px,

(max-width: 768px) 100vw,

1200px"

alt="A responsive image">

* srcset: A set of image source, each followed by its width descriptor(eg 30w).
* sizes: Defines how much space image should take up on different screen sizes(100vw refers to 100% of viewport width.

Multimedia in HTML

**1. Embedding Audio (<audio>)**

The <audio> tag is used to embed sound content, such as music or voice recordings, on a webpage. You can provide multiple audio sources to support different formats (like MP3).

* controls: Displays the browser’s default audio controls (play, pause, volume, etc.).
* autoplay: Automatically starts playing the audio when the page loads.
* loop: Makes the audio loop continuously after it ends.
* muted: Starts the audio muted by default.

**2. Embedding Video (<video>)**

The <video> tag is used to embed video files. Like audio, multiple sources can be provided for different video formats.

* controls: Adds video controls such as play, pause, volume, etc.
* autoplay: Starts playing the video automatically when the page loads.
* loop: Replays the video after it finishes.
* muted: Mutes the video by default.
* width/height: Specifies the dimensions of the video play.

### 3. Using the <source> Element for Multiple Formats

Both the <audio> and <video> tags support the <source> element to specify different media formats. This ensures compatibility across different browsers that might support only certain formats.

* **For Audio**: Use different formats like MP3 and OGG.
* **For Video**: Use formats like MP4, WebM, and OGG.

<embed>: The <embed> tag is used to embed external content such as PDFs, media players, or other types of content that are not natively supported by the browser.

Eg: <embed src="document.pdf" width="600" height="500" type="application/pdf">

<iframe>: The <iframe> tag is used to embed another HTML page or resource into the current page.

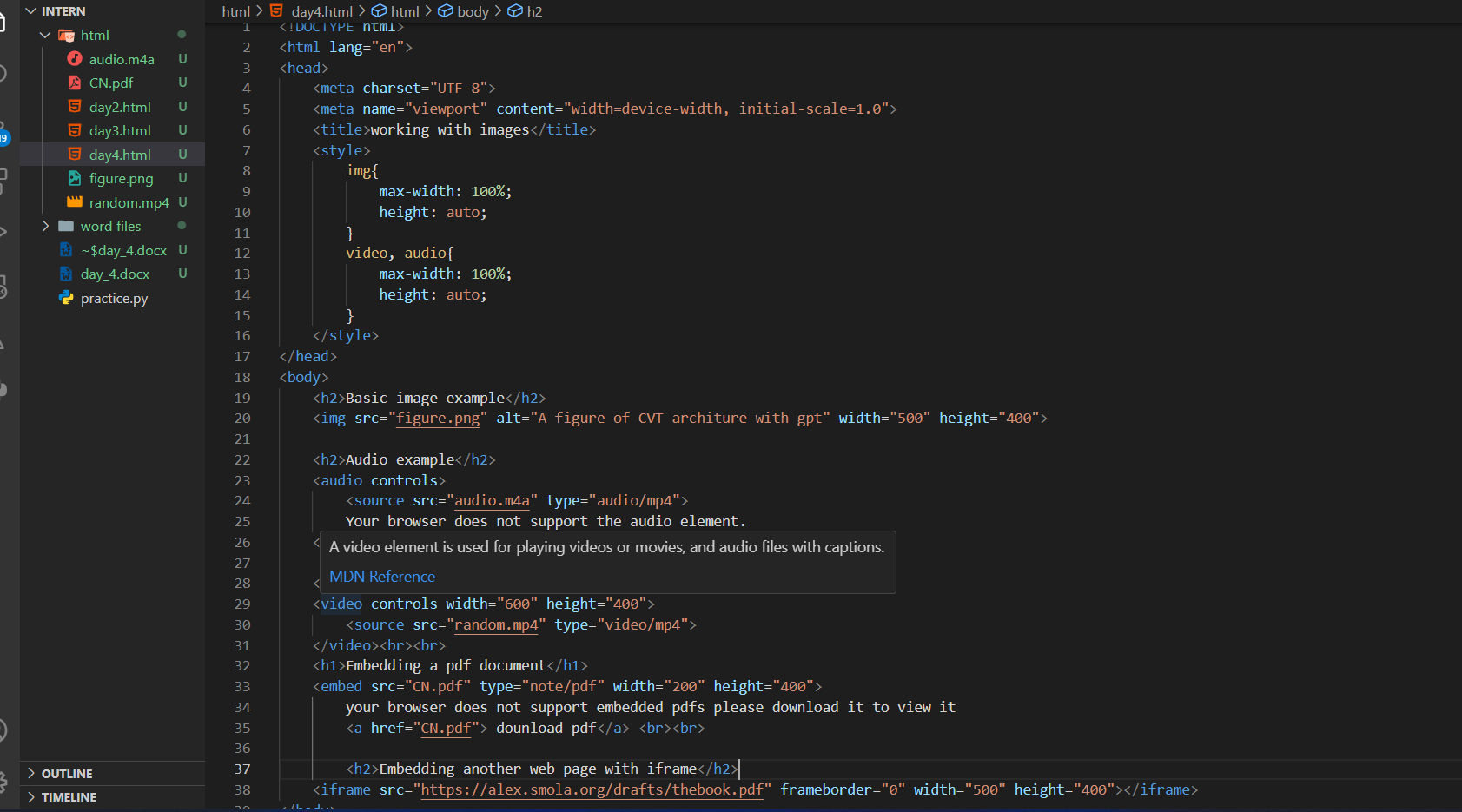
Eg: <iframe src="https://www.example.com" width="600" height="400"></iframe>

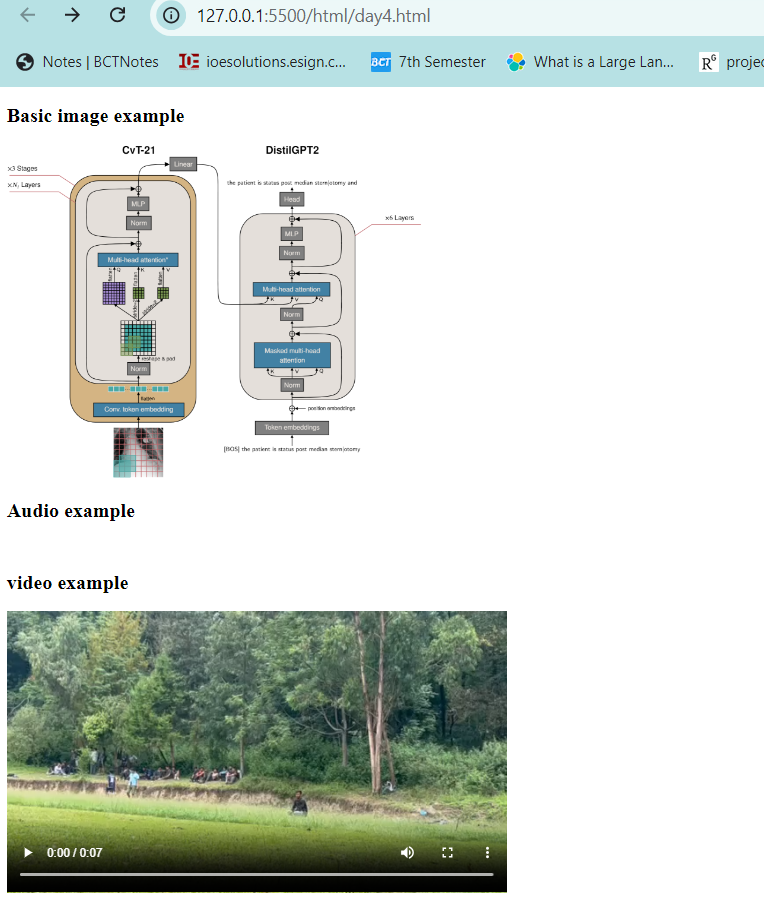
<object>: The <object> tag is more versatile than <embed>, as it can embed different types of content, such as multimedia, webpages, or interactive objects.

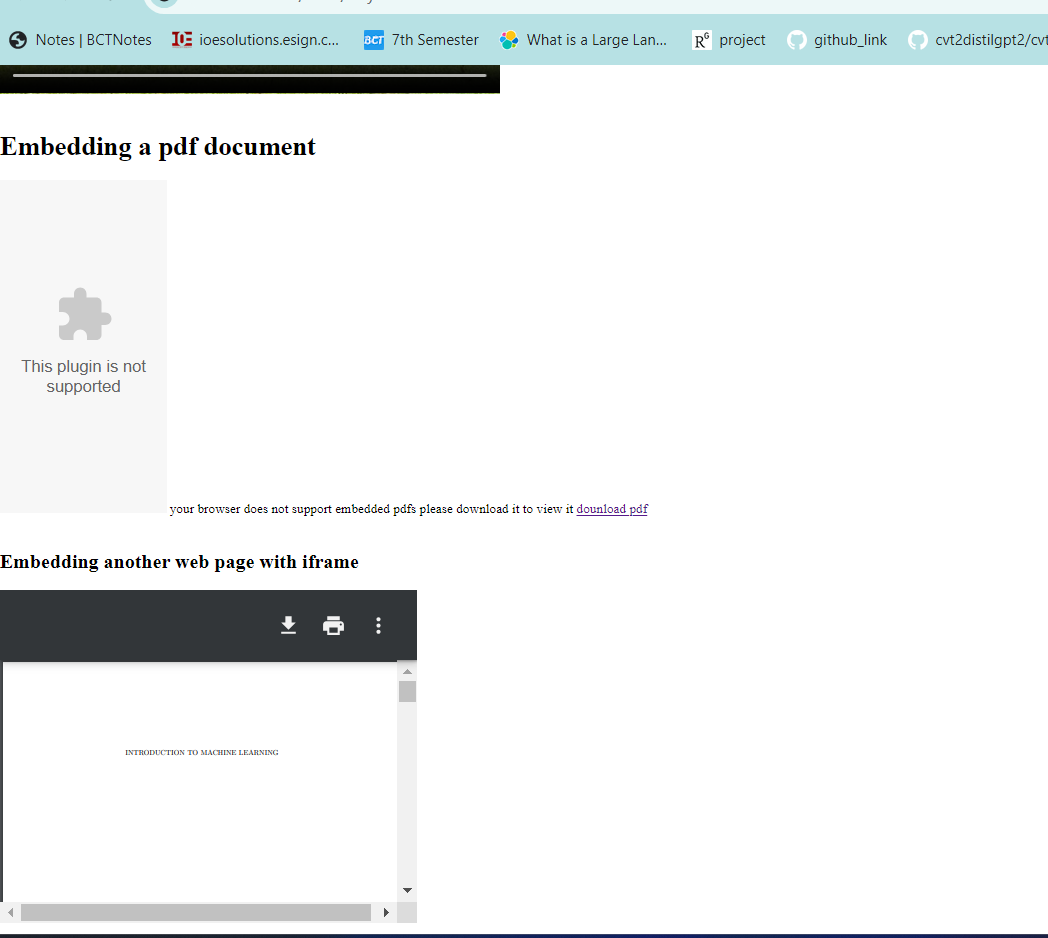
Eg: <object data="media.mp4" width="600" height="400" type="video/mp4">

Your browser does not support this media.

</object>







**Introduction to Semantics**

Semantic HTML refers to the use of HTML tags that convey meaning about the content they contain. Unlike non-semantic tags like <div> and <span>, which tell nothing about their content, semantic tags like <article>, <nav>, and <header> describe the purpose of the content, making it easier for both developers and machines (e.g., search engines, screen readers) to understand the structure of a webpage

**<header>**: Represents the header of a section or page, often containing introductory content or navigation links.

**<footer>**: Defines the footer of a section or page, commonly containing metadata, links, or copyright information.

**<section>**: Represents a standalone section of content, typically with a heading. It is used to group related content in a document.

**<article>**: Represents a self-contained piece of content that can be independently distributed or reused. It's typically used for blog posts, news articles, or forum posts.

**<nav>**: Used to define a section of navigation links. This helps both users and search engines locate the main navigation on a page.

**<aside>**: Represents content that is tangentially related to the content around it, like a sidebar or a pull quote. It is used for side content or supplementary information.

**Importance of Using Semantic Tags for SEO and Accessibility**

1. **SEO (Search Engine Optimization)**
   * **Improved Understanding**: Search engines can better understand the structure and relevance of content when semantic tags are used. For example, <article> helps identify content that is a distinct piece of information, potentially improving its ranking.
   * **Better Indexing**: Semantic tags help search engines index content more effectively, leading to better visibility in search results.
2. **Accessibility**
   * **Enhanced Navigation**: For users relying on screen readers or other assistive technologies, semantic tags provide context and structure. Tags like <nav>, <header>, and <footer> allow users to navigate pages more efficiently.
   * **Meaningful Content**: Semantic tags give meaning to the content, helping users understand the purpose and context of different sections, which is crucial for those with visual impairments or other disabilities.

