CSS FlexBox :-

Question 1: What is CSS Flexbox, and how is it useful for layout design? Explain the terms flex-container and flex-item.

Ans:- CSS Flexbox (Flexible Box Layout) is a CSS layout module designed to arrange elements efficiently and responsively within a container. It simplifies alignment, spacing, and ordering of elements, especially when compared to traditional layout methods like floats and positioning.

Key Features of Flexbox

- Provides a **flexible and responsive layout** for elements.
- Supports both **horizontal** and **vertical alignment** of items.
- Automatically adjusts the size of items to best fit the available space.
- Makes it easy to **center elements**, distribute space evenly, and reorder items.

1. **Flex-Container**

The parent element that defines the flex layout. Any element can be made a flex-container by applying the CSS property display: flex or display: inline-flex.

```
Eg:- .flex-container

{

   display: flex;

}
```

Characteristics:

- Child elements of the flex-container become **flex-items**.
- Controls the layout and alignment of its child elements.

2. Flex-Item

The direct child elements of a flex-container. These items are arranged and controlled by the container's flex properties.

```
<div class="flex-container">

  <div class="flex-item">Item 1</div>

  <div class="flex-item">Item 2</div>

  <div class="flex-item">Item 3</div>

</div>
```

CSS Flexbox (Flexible Box Layout) is a CSS layout module designed to arrange elements efficiently and responsively within a container. It simplifies alignment, spacing, and ordering of elements, especially when compared to traditional layout methods like floats and positioning.

Key Features of Flexbox

- Provides a **flexible and responsive layout** for elements.
- Supports both **horizontal** and **vertical alignment** of items.
- Automatically adjusts the size of items to best fit the available space.
- Makes it easy to **center elements**, distribute space evenly, and reorder items.

Question 2: Describe the properties justify-content, align-items, and flex-direction used in Flexbox.

Ans:-

1. justify-content

- **Purpose**: Aligns flex items along the **main axis**.
- **Values**:
    - flex-start (default): Items align at the start of the main axis.
    - flex-end: Items align at the end of the main axis.
    - center: Items are centered along the main axis.
    - space-between: Items have equal space between them.
    - space-around: Items have equal space around them.
    - space-evenly: Items have equal space between and around them.

Eg:-
```
<div class="container">

  <div class="item">1</div>

  <div class="item">2</div>

  <div class="item">3</div>

</div>
```

Style.css

```
.container {

  display: flex;

  justify-content: space-around; /* Items are evenly spaced */

  background-color: lightgray;

  height: 100px;

}
```

```
.item {

   background-color: steelblue;

   color: white;

   padding: 20px;

}
```

2. align-items

- **Purpose**: Aligns flex items along the **cross axis**.
- **Values**:
    - stretch (default): Items stretch to fill the container along the cross axis.
    - flex-start: Items align at the start of the cross axis.
    - flex-end: Items align at the end of the cross axis.
    - center: Items are centered along the cross axis.
    - baseline: Items align along their text baselines.

Eg:-

```
<div class="container">

   <div class="item">A</div>

   <div class="item">B</div>

   <div class="item">C</div>

</div>
```

Style.css :-

```
.container {
  display: flex;

  align-items: center; /* Items are vertically centered */

  background-color: lightgray;

  height: 150px;

}



.item {

  background-color: coral;

  color: white;

  padding: 20px;

  height: 50px;

}
```

3. flex-direction

- **Purpose**: Defines the direction of the **main axis** and the order of flex items.
- **Values**:
    - row (default): Items are placed in a row (left to right).
    - row-reverse: Items are placed in a row (right to left).
    - column: Items are placed in a column (top to bottom).
    - column-reverse: Items are placed in a column (bottom to top).

Eg:-

```
<div class="container">
```

```
    <div class="item">X</div>

    <div class="item">Y</div>

    <div class="item">Z</div>

</div>
```

Style.css

```
.container {

    display: flex;

    flex-direction: column; /* Items are stacked vertically */

    background-color: lightgray;

    height: 200px;

}


.item {

    background-color: darkorange;

    color: white;

    padding: 20px;

}
```