


<b>Mobile Programming Laboratory</b> <b>Week 5: Layout and Random number</b>			
Name: Pongsaat Piniingam		ID: 6631501079	Section: 02
Date: Sep 16, 2025		Due: Midnight before the next class	

### Objective

- To experiment with layouts such as Container, Align, Row, Column etc.
- To practice UI design
- To generate random numbers

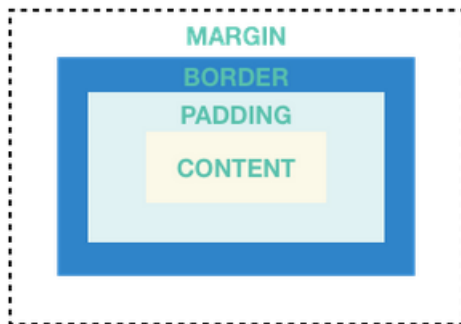
### References

- <https://flutter.dev/docs/development/ui/layout>
- <https://flutter.dev/docs/development/ui/widgets/layout>

### Layout

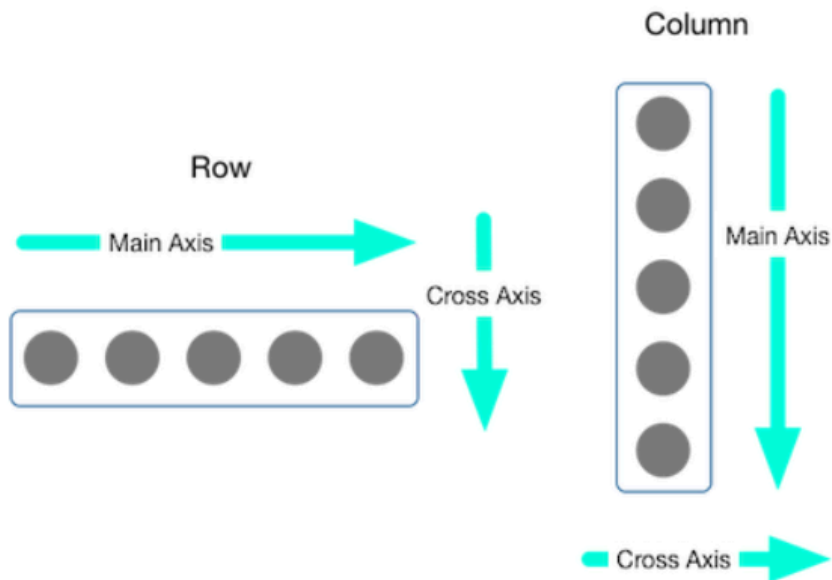
#### Single-child layout widgets

Can have only one child. The examples are **Container**, **Center**, **Padding** etc.

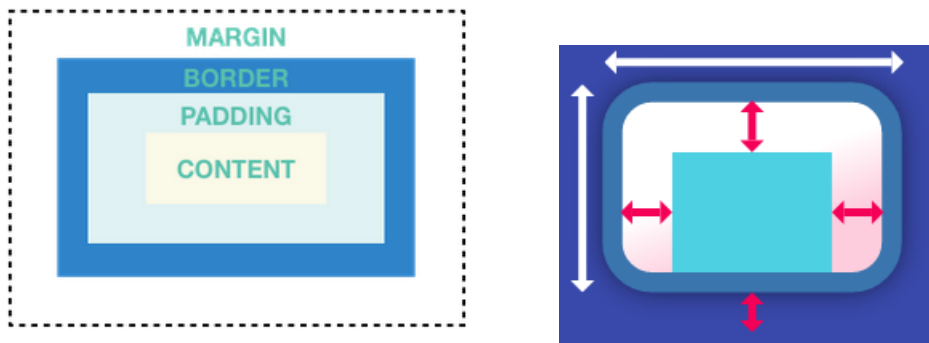


## Multi-child layout widgets

Can have many children e.g. **Column**, **Row**, **ListView** etc.

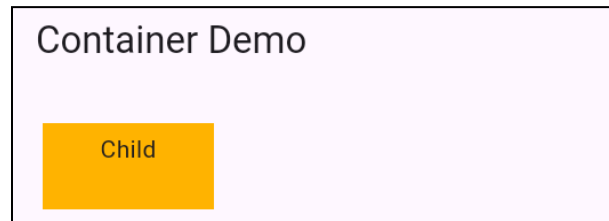


## Container



- A very general widget to contain another widget
- Good for providing padding, margins and borders
- Useful for defining background color or image
- Can only have a single child widget. However, we can use a child such as Row, Column, or others to contain multiple widgets.
- Can be sized by width and height, but when it has a child, its size fits its child and also its parent.
- Can align its child
- Can be decorated to have an image and borders using BoxDecoration

## Exercise 1 Container



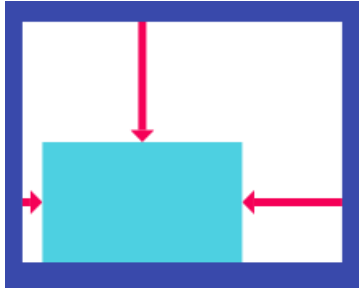
```
import 'package:flutter/material.dart';

void main() {
  runApp(
    const MaterialApp(home: ContainerDemo(), debugShowCheckedModeBanner: false),
  );
}

class ContainerDemo extends StatelessWidget {
  const ContainerDemo({super.key});

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: const Text('Container Demo')),
      body: Container(
        alignment: Alignment.topCenter,
        margin: const EdgeInsets.all(20),
        padding: const EdgeInsets.all(5),
        color: Colors.amber[600],
        width: 100.0,
        height: 50.0,
        child: const Text('Child'),
      ),
    );
  }
}
```

## Align



If you want to place a child widget at a position in a parent, you can wrap the child inside Align and put Align inside the parent.

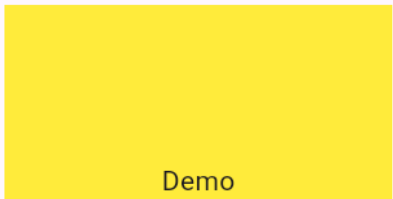
For example, if a parent is Container and a child is Text. Generally, Text is at top left of the Container. Certainly we can move Text to the center of Container using Center. But to place Text to different positions inside Container, try wrap text in Align.

### Container Demo



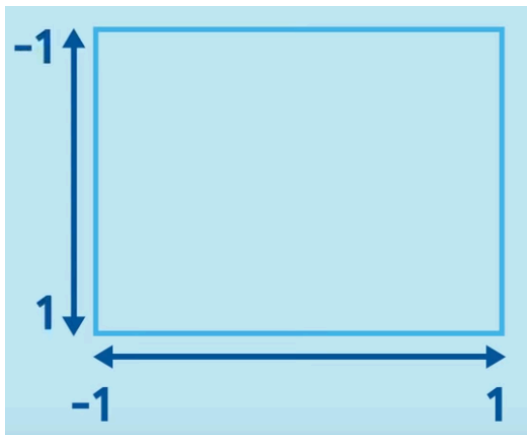
```
body: Container(  
  margin: const EdgeInsets.all(20),  
  color: Colors.yellow,  
  width: 200,  
  height: 100,  
  child: const Align(  
    alignment: Alignment.topRight,  
    child: Text('Demo'),  
  ),  
)
```

## Container Demo



```
child: const Align(  
  alignment: Alignment.bottomCenter,  
  child: Text('Demo'),  
),
```

Or we can define alignment in a format of a range from top left (-1,-1) to bottom right (1,1).


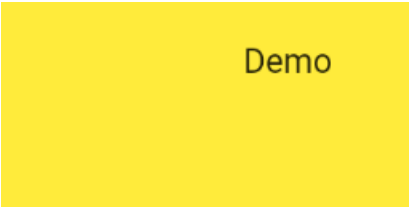


```
child: Align(  
  alignment: Alignment(0, 0),  
  child: Text('Demo'),  
),
```

Demo

```
child: Align(  
  alignment: Alignment(-1, -1),  
  child: Text('Demo'),  
),
```

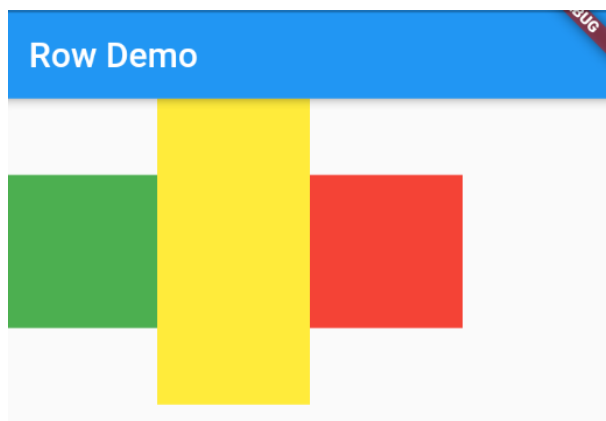
Demo

<pre>child: Align(   alignment: Alignment(1, 1),   child: Text('Demo'), )</pre>	
<pre>child: Align(   alignment: Alignment(0.5, -0.5),   child: Text('Demo'), )</pre>	

## Row



- A container arranges children horizontally
- By default, the row's width is the screen width and the row's height is the max height of the children
- To fit the row's width to its children, set `mainAxisSize` property to `MainAxisSize.min`
- Cannot scroll (use `ListView` instead)
- We can use 'Expanded' widget to let a row's child to expand to cover the row's width
- `MainAxisAlignment`: center, start, end, spaceAround, spaceBetween, spaceEvenly
- `CrossAxisAlignment`: center, start, end, stretch



### Exercise 2 Row

```
import 'package:flutter/material.dart';

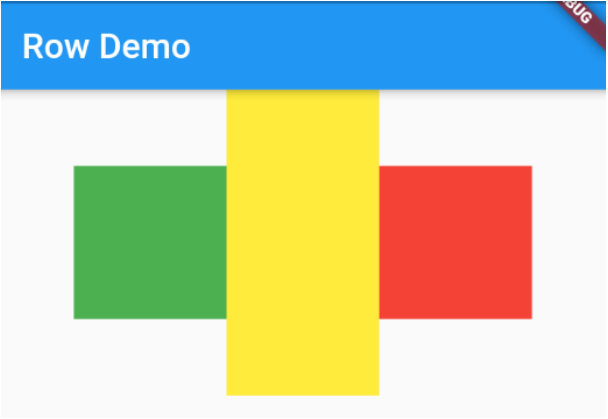
class RowDemo extends StatelessWidget {
  const RowDemo({super.key});

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: Scaffold(
        appBar: AppBar(
```


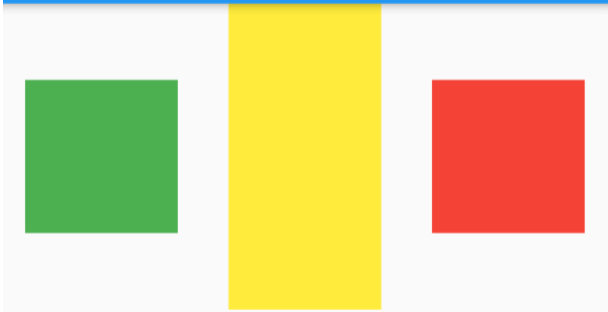


```

    title: const Text('Row Demo'),
  ),
  body: Row(
    children: <Widget>[
      Container(
        color: Colors.green,
        width: 100,
        height: 100,
      ),
      Container(
        color: Colors.yellow,
        width: 100,
        height: 200,
      ),
      Container(
        color: Colors.red,
        width: 100,
        height: 100,
      ),
    ],
  ),
);
}

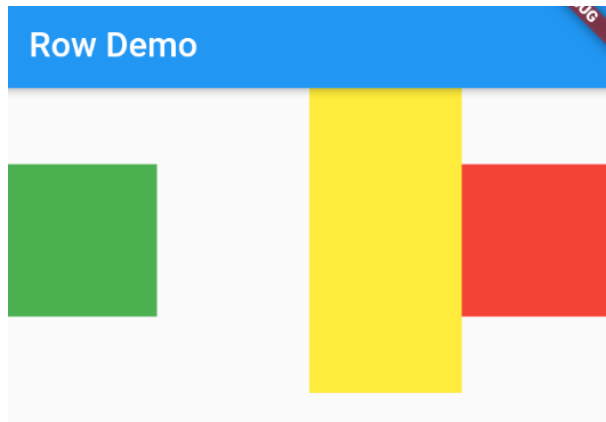
```

	<pre> body: Row(   mainAxisAlignment: MainAxisAlignment.center, </pre>
---	--



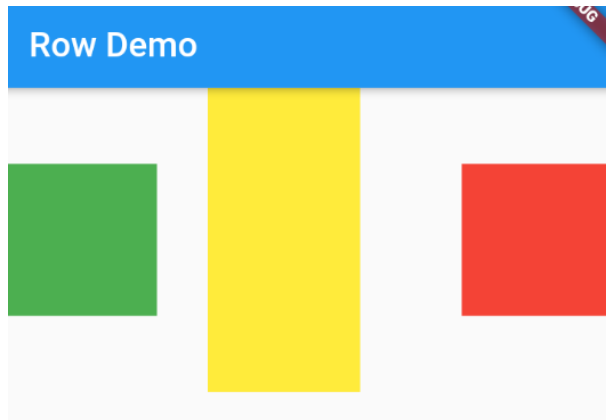
<p>Row Demo</p>  A visual demonstration of MainAxisAlignment.end. It shows a horizontal row of three colored squares: green, yellow, and red. The yellow square is positioned at the far right end of the row, while the green and red squares are to its left.	<pre>mainAxisAlignment: MainAxisAlignment.end,</pre>
<p>Row Demo</p>  A visual demonstration of MainAxisAlignment.spaceAround. It shows a horizontal row of three colored squares: green, yellow, and red. The yellow square is centered, and the green and red squares are positioned to its left and right, respectively, with equal space between them.	<pre>mainAxisAlignment: MainAxisAlignment.spaceAround</pre>
<p>Row Demo</p>  A visual demonstration of MainAxisAlignment.spaceBetween. It shows a horizontal row of three colored squares: green, yellow, and red. The yellow square is centered, and the green and red squares are positioned to its left and right, respectively, with equal space between them.	<pre>mainAxisAlignment: MainAxisAlignment.spaceBetween</pre>
<p>Row Demo</p>  A visual demonstration of MainAxisAlignment.spaceEvenly. It shows a horizontal row of three colored squares: green, yellow, and red. The yellow square is centered, and the green and red squares are positioned to its left and right, respectively, with equal space between them.	<pre>mainAxisAlignment: MainAxisAlignment.spaceEvenly</pre>

If you want to organize spaces between widgets inside Row more precisely, you can use 'Spacer'.



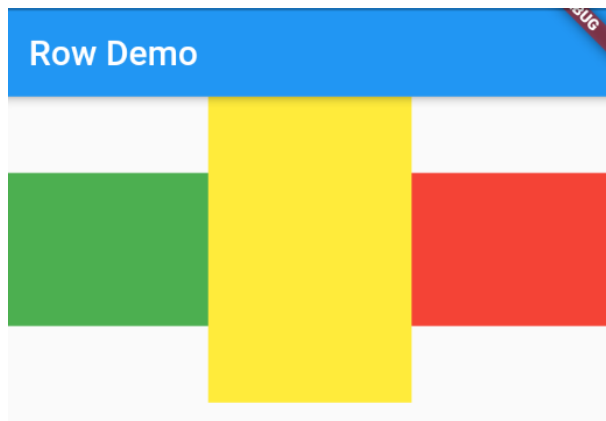
```
body: Row(  
  children: <Widget>[  
    Container(  
      color: Colors.green,  
      width: 100,  
      height: 100,  
    ),  
    Spacer(),  
    Container(  
      color: Colors.yellow,  
      width: 100,  
      height: 200,  
    ),  
    Container(  
      color: Colors.red,  
      width: 100,  
      height: 100,  
    ),  
  ],  
)
```

And we can set flex of Spacer to create a proportion of spaces.



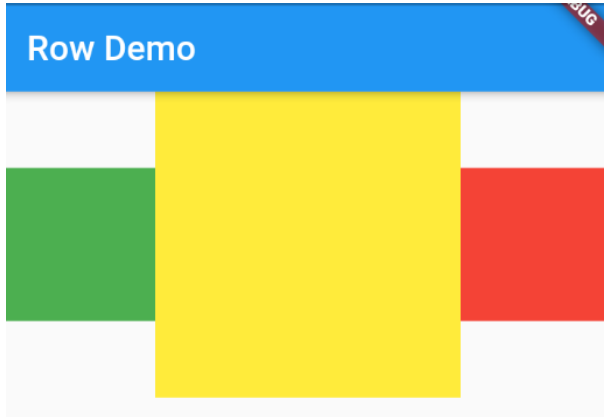
```
body: Row(  
  children: <Widget>[  
    Container(  
      color: Colors.green,  
      width: 100,  
      height: 100,  
    ),  
    Spacer(), //same as Spacer(flex: 1)  
    Container(  
      color: Colors.yellow,  
      width: 100,  
      height: 200,  
    ),  
    Spacer(flex: 2),  
    Container(  
      color: Colors.red,  
      width: 100,  
      height: 100,  
    ),  
  ],  
)
```

To expand children to the entire row's width, use 'Expanded' widget. It also helps to fit large children to the screen's width or height.



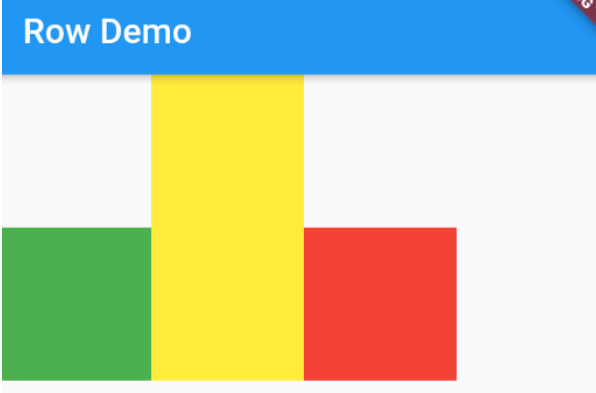
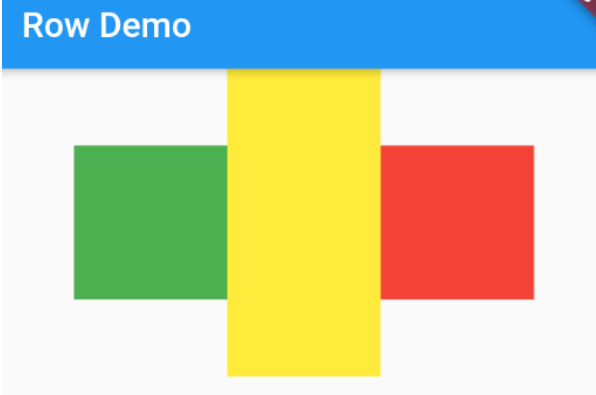

```
body: Row(  
  children: <Widget>[  
    Expanded(  
      child: Container(  
        color: Colors.green,  
        width: 100,  
        height: 100,  
      ),  
    ),  
    Expanded(  
      child: Container(  
        color: Colors.yellow,  
        width: 100,  
        height: 200,  
      ),  
    ),  
    Expanded(  
      child: Container(  
        color: Colors.red,  
        width: 100,  
        height: 100,  
      ),  
    ),  
  ],  
)
```

## Expansion with proportion



```
body: Row(  
  children: <Widget>[  
    Expanded(  
      child: Container(  
        color: Colors.green,  
        width: 100,  
        height: 100,  
      ),  
    ),  
    Expanded(  
      flex: 2,  
      child: Container(  
        color: Colors.yellow,  
        width: 100,  
        height: 200,  
      ),  
    ),  
    Expanded(  
      child: Container(  
        color: Colors.red,  
        width: 100,  
        height: 100,  
      ),  
    ),  
  ],  
)
```

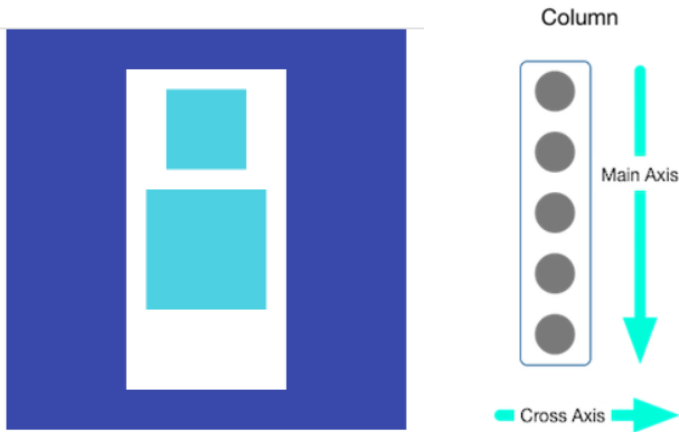
## Cross axis alignment

	<pre>body: Row(   crossAxisAlignment: CrossAxisAlignment.end,   children: &lt;Widget&gt;[</pre>
	<pre>    mainAxisAlignment: MainAxisAlignment.center,     crossAxisAlignment: CrossAxisAlignment.center,     children: &lt;Widget&gt;[</pre>
	<pre>    crossAxisAlignment: CrossAxisAlignment.stretch,     children: &lt;Widget&gt;[</pre>

Another way to simply get the above result is to neglect the Container's height so that the height will be as large as possible or the screen's height.

```
body: Row(  
  children: <Widget>[  
    Container(  
      color: Colors.green,  
      width: 100,  
    ),  
    Container(  
      color: Colors.yellow,  
      width: 100,  
    ),  
    Container(  
      color: Colors.red,  
      width: 100,  
    ),  
  ],  
)
```

## Column



- A container arranges children vertically
- By default, the column's height is the screen height and the column's width is the max width of its children
- To fit the column's height to its children, set `mainAxisSize` property to `MainAxisSize.min`
- Cannot scroll (use `ListView` instead)
- We can use 'Expanded' widget to let a column's child to expand to cover the row's height
- `MainAxisAlignment`: center, start, end, spaceAround, spaceBetween, spaceEvenly
- `CrossAxisAlignment`: center, start, end, stretch

### Column Demo





### Exercise 3 Column

```
import 'package:flutter/material.dart';

class ColumnDemo extends StatelessWidget {
  const ColumnDemo({super.key});

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text('Column Demo'),
      ),
      body: Column(
        children: <Widget>[
          Container(
            color: Colors.green,
            width: 100,
            height: 100,
          ),
          Container(
            color: Colors.yellow,
            width: 200,
            height: 100,
          ),
          Container(
            color: Colors.red,
            width: 100,
            height: 100,
          ),
        ],
      ),
    );
  }
}
```

Column Demo



```
body: Column(  
  mainAxisAlignment: MainAxisAlignment.center,
```

Column Demo



```
  mainAxisAlignment: MainAxisAlignment.end,
```

Column Demo

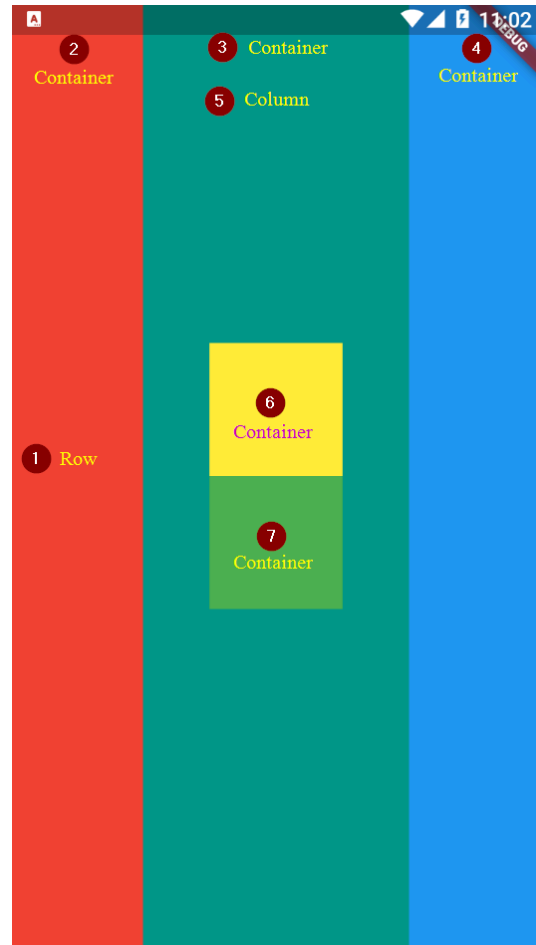
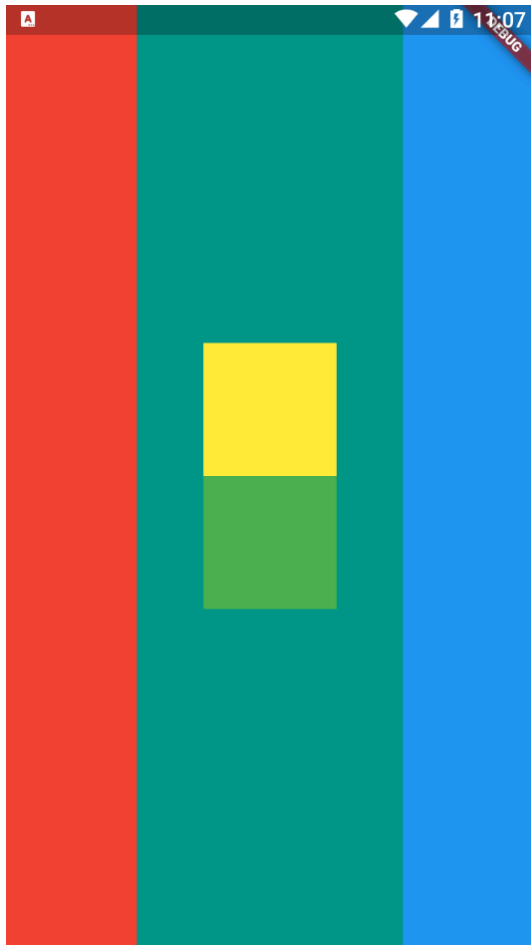


```
  mainAxisAlignment: MainAxisAlignment.spaceAround
```

<div data-bbox="134 199 435 239">Column Demo</div> 	<pre>mainAxisAlignment: MainAxisAlignment.spaceBetween</pre>
<div data-bbox="134 739 435 779">Column Demo</div> 	<pre>mainAxisAlignment: MainAxisAlignment.spaceEvenly</pre>
<div data-bbox="134 1278 435 1318">Column Demo</div> 	<pre>mainAxisAlignment: MainAxisAlignment.start, crossAxisAlignment: CrossAxisAlignment.end,</pre>

<div data-bbox="134 199 435 237">Column Demo</div> 	<pre>mainAxisAlignment: MainAxisAlignment.center, crossAxisAlignment: CrossAxisAlignment.center,</pre>
<div data-bbox="134 737 435 774">Column Demo</div> 	<pre>mainAxisAlignment: MainAxisAlignment.center, crossAxisAlignment: CrossAxisAlignment.stretch,</pre>

#### Exercise 4 Design the following UI.

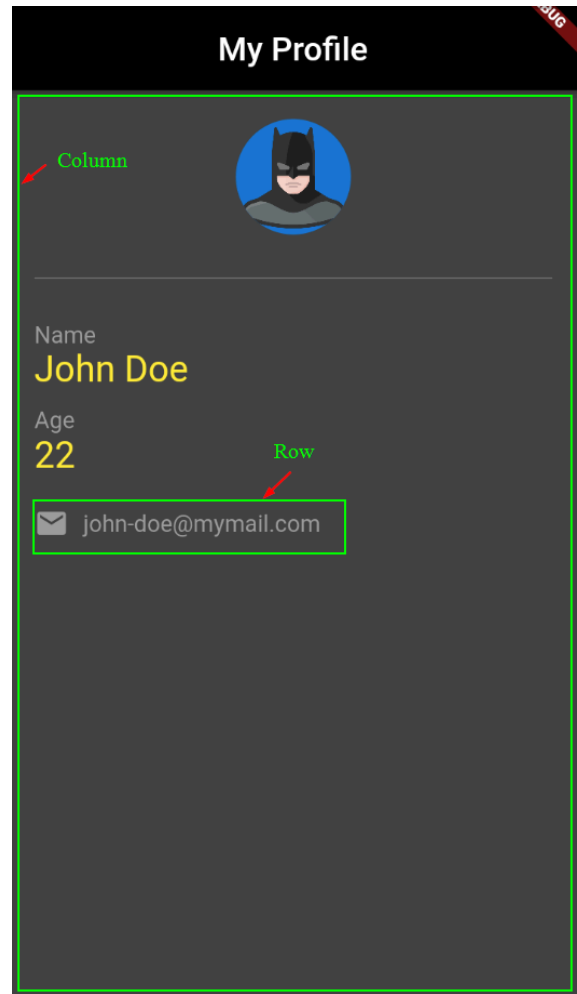
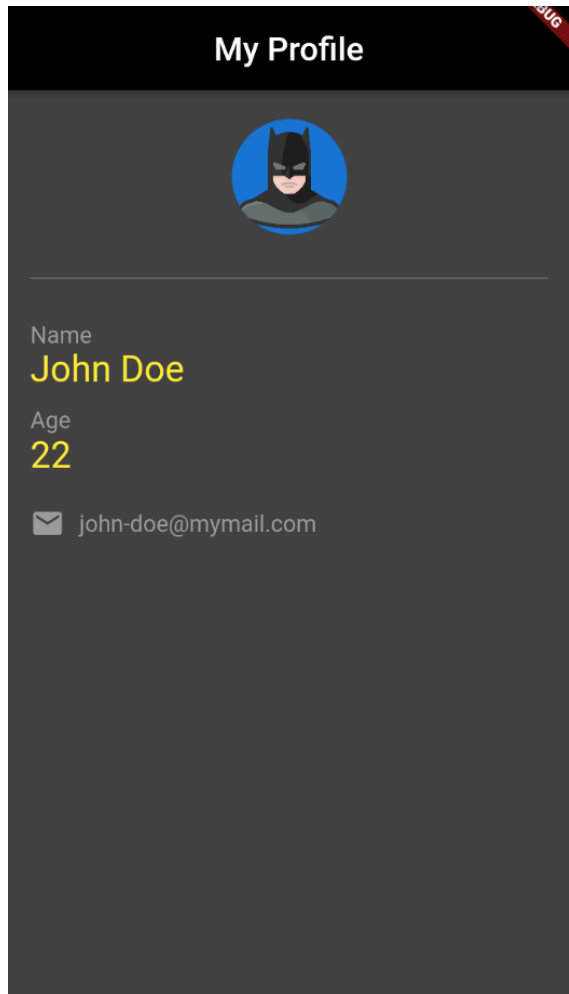


```
class ContainerDemo extends StatelessWidget {  
  const ContainerDemo({super.key});  
  
  @override  
  Widget build(BuildContext context) {  
    return Scaffold(  
      body: Row(  
        children: [  
          Container(color: Colors.red, width: 100),  
          Expanded(  
            child: Container(  
              color: Colors.teal,  
              width: 100,  
              child: Column(  
                mainAxisAlignment: MainAxisAlignment.center,  

```

```
        children: [
            Container(width: 100, height: 100, color: Colors.yellow),
            Container(width: 100, height: 100, color: Colors.green),
        ],
    ),
),
Container(color: Colors.blue, width: 100),
],
),
);
}
```

## Exercise 5 Simple profile app design



```
class Profile extends StatelessWidget {  
  const Profile({super.key});  
  
  @override  
  Widget build(BuildContext context) {  
    return Scaffold(  
      backgroundColor: Colors.grey[800],  
      appBar: AppBar(  
        title: const Text('My Profile', style: TextStyle(color: Colors.white)),  
        centerTitle: true,  
        backgroundColor: Colors.black,  
      ),  
      body: const Padding(  
        padding: EdgeInsets.all(20.0),  
      ),  
    );  
  }  
}
```

```

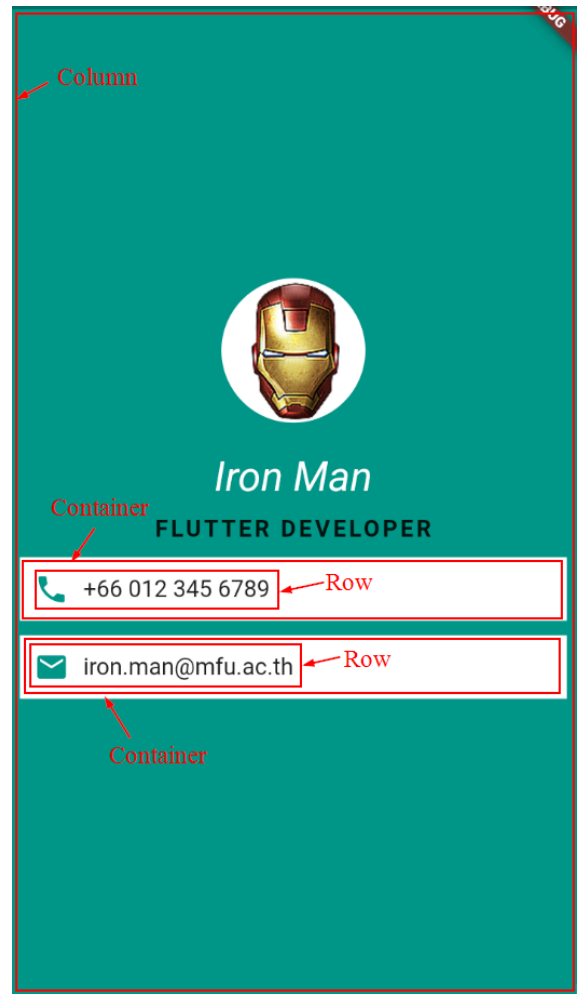
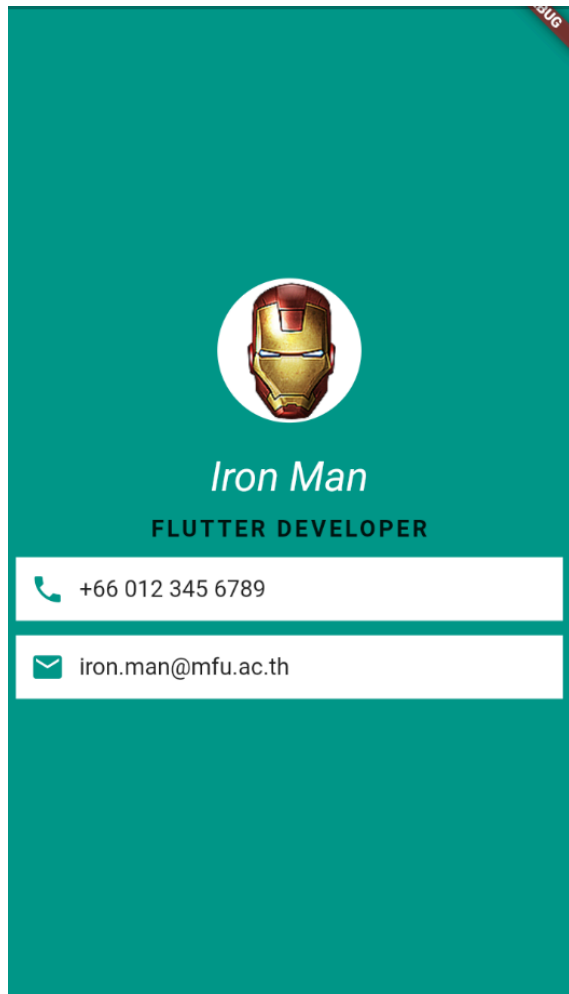
child: Column(
  crossAxisAlignment: CrossAxisAlignment.start,
  children: [
    Center(
      child: CircleAvatar(
        backgroundColor: Colors.blue,
        backgroundImage: NetworkImage(
'https://icons.iconarchive.com/icons/diversity-avatars/avatars/512/batman-icon.png',
      ),
      radius: 40,
    ),
    Divider(color: Colors.grey, height: 60),
    Text('Name', style: TextStyle(color: Colors.grey)),
    Text(
      'John Doe',
      style: TextStyle(color: Colors.yellow, fontSize: 22),
    ),
    SizedBox(height: 10),
    Text('Age', style: TextStyle(color: Colors.grey)),
    Text('22', style: TextStyle(color: Colors.yellow, fontSize: 22)),
    SizedBox(height: 20),
    Row(
      children: [
        Icon(Icons.mail, color: Colors.grey),
        SizedBox(width: 10),
        Text(
          'john-doe@mymail.com',
          style: TextStyle(color: Colors.grey),
        ),
      ],
    ),
  ],
),
);

```



```
}  
}
```

## Exercise 6 Another profile app design



```
class Profile2 extends StatelessWidget {  
  const Profile2({super.key});  
  
  @override  
  Widget build(BuildContext context) {  
    return Scaffold(  
      backgroundColor: Colors.teal,  
      body: Column(  
        mainAxisAlignment: MainAxisAlignment.center,  
        children: [  
          const CircleAvatar(  
            backgroundImage: NetworkImage(  

```

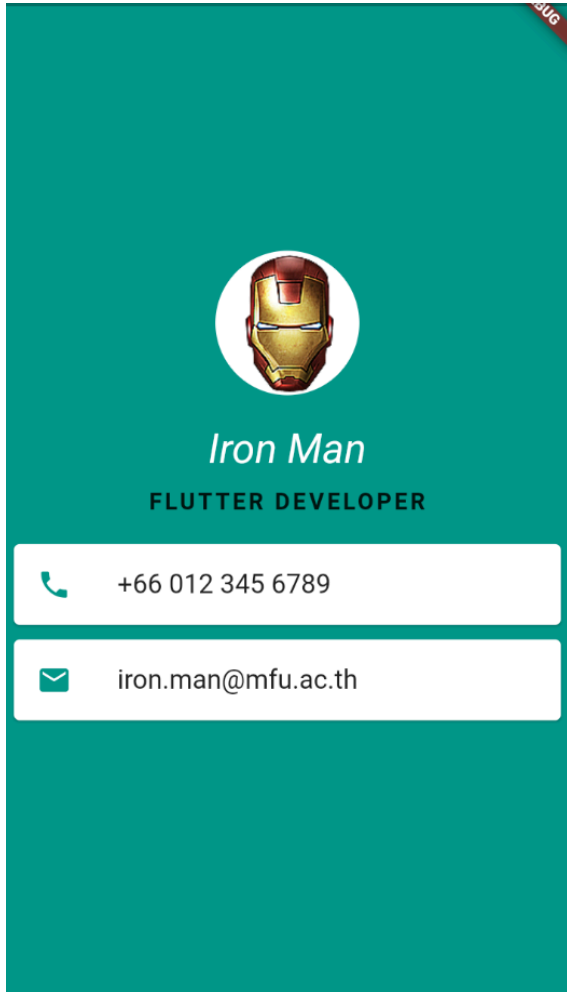
```

'https://icons.iconarchive.com/icons/kidaubis-design/cool-heroes/128/Ironman-icon.png',
    ),
    radius: 50,
    backgroundColor: Colors.white,
  ),
  const SizedBox(height: 20),
  const Text(
    'Iron Man',
    style: TextStyle(
      color: Colors.white,
      fontSize: 25,
      fontStyle: FontStyle.italic,
    ),
  ),
),
const SizedBox(height: 10),
const Text(
  'FLUTTER DEVELOPER',
  style: TextStyle(fontWeight: FontWeight.bold, letterSpacing: 2),
),
const SizedBox(height: 10),
Container(
  color: Colors.white,
  padding: const EdgeInsets.all(8),
  margin: const EdgeInsets.symmetric(horizontal: 8),
  child: const Row(
    children: [
      Icon(Icons.phone, color: Colors.teal),
      SizedBox(width: 8),
      Text('+66 012 345 6789'),
    ],
  ),
),
const SizedBox(height: 8),
Container(
  color: Colors.white,
  padding: const EdgeInsets.all(8),

```

```
margin: const EdgeInsets.symmetric(horizontal: 8),
child: const Row(
  children: [
    Icon(Icons.mail, color: Colors.teal),
    SizedBox(width: 8),
    Text('iron.man@mfu.ac.th'),
  ],
),
),
],
),
);
}
}
```

Alternatively, we can use **Card** and **ListTile** instead of the container.



```
import 'package:flutter/material.dart';

class Profile2 extends StatelessWidget {
  const Profile2({super.key});

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      backgroundColor: Colors.teal,
      body: Column(
        mainAxisAlignment: MainAxisAlignment.center,
        children: [
          const CircleAvatar(
            backgroundImage: NetworkImage(
```

```

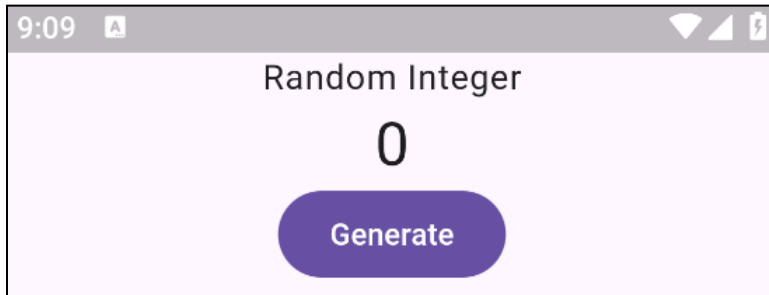
'https://icons.iconarchive.com/icons/kidaubis-design/cool-heroes/128/Ironman-icon.png',
    ),
    radius: 50,
    backgroundColor: Colors.white,
  ),
  const SizedBox(height: 20),
  const Text(
    'Iron Man',
    style: TextStyle(
      color: Colors.white,
      fontSize: 25,
      fontStyle: FontStyle.italic,
    ),
  ),
),
const SizedBox(height: 10),
const Text(
  'FLUTTER DEVELOPER',
  style: TextStyle(fontWeight: FontWeight.bold, letterSpacing: 2),
),
const SizedBox(height: 10),
Container(
  color: Colors.white,
  padding: const EdgeInsets.all(8),
  margin: const EdgeInsets.symmetric(horizontal: 8),
  child: const Row(
    children: [
      Icon(Icons.phone, color: Colors.teal),
      SizedBox(width: 8),
      Text('+66 012 345 6789'),
    ],
  ),
),
const SizedBox(height: 8),
Container(
  color: Colors.white,
  padding: const EdgeInsets.all(8),

```

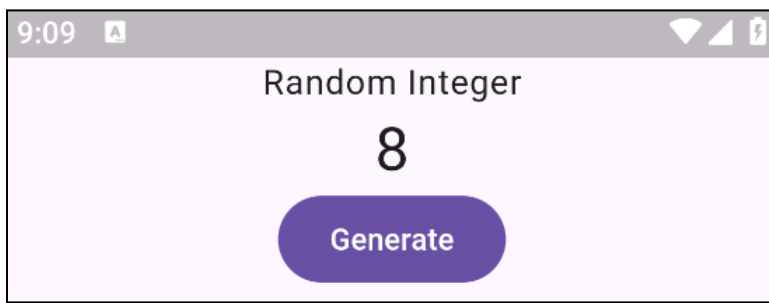
```
margin: const EdgeInsets.symmetric(horizontal: 8),
child: const Row(
  children: [
    Icon(Icons.mail, color: Colors.teal),
    SizedBox(width: 8),
    Text('iron.man@mfu.ac.th'),
  ],
),
),
],
),
);
}
}
```

### Exercise 7 Generate a random integer number

Use “Random.nextInt(max)” to generate an integer number in [0, max).



Clicking a button will generate a random number in a range 0-9.



```
import 'dart:math';
import 'package:flutter/material.dart';

class RandomDemo extends StatefulWidget {
  const RandomDemo({super.key});

  @override
  State<RandomDemo> createState() => _RandomDemoState();
}

class _RandomDemoState extends State<RandomDemo> {
  int num = 0;

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      body: SafeArea(
```

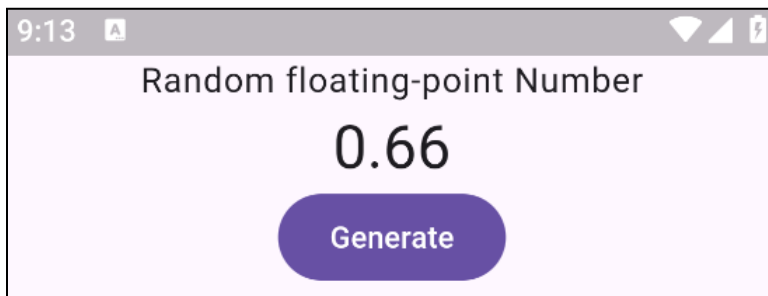
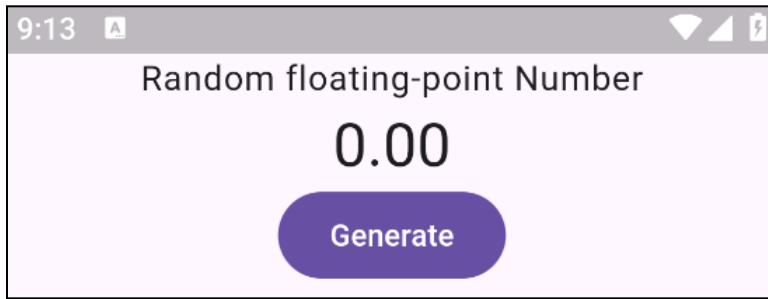


```

        child: Center(
child: Column(
  children: [
    Text(
      'Random Integer',
      style: Theme.of(context).textTheme.bodyLarge,
    ),
    Text(
      '$num',
      style: Theme.of(context).textTheme.headlineMedium,
    ),
    FilledButton(
      onPressed: () {
        setState(() {
          // generate a random number 0-9
          num = Random().nextInt(10);
        });
      },
      child: const Text('Generate'),
    ),
  ],
),
)),
);
}
}

```

**Exercise 8** Generate a random floating-point number



```
import 'dart:math';
import 'package:flutter/material.dart';

class RandomDemo extends StatefulWidget {
  const RandomDemo({super.key});

  @override
  State<RandomDemo> createState() => _RandomDemoState();
}

class _RandomDemoState extends State<RandomDemo> {
  double num = 0;

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      body: SafeArea(
        child: Center(
          child: Column(
            children: [
              Text(
```

```

        'Random floating-point Number',
        style: Theme.of(context).textTheme.bodyLarge,
    ),
    Text(
        num.toStringAsFixed(2),
        style: Theme.of(context).textTheme.headlineMedium,
    ),
    FilledButton(
        onPressed: () {
            setState(() {
                // generate a random number 0.00-0.99
                num = Random().nextDouble();
            });
        },
        child: const Text('Generate'),
    ),
],
),
)),
);
}
}

```

If we want to get the random number immediately after the app launches, we can use the “initState()” function.

Random floating-point Number

0.20

```

@override
void initState() {
    super.initState();
    num = Random().nextDouble();
}

```

**Exercise 9** Generate a random number from min to max

Random Integer Number

min

---

max

---

Generate

Random Integer Number

3

---

9

---

7

Generate

```
import 'dart:math';
import 'package:flutter/material.dart';

class RandomDemo extends StatefulWidget {
  const RandomDemo({super.key});

  @override
  State<RandomDemo> createState() => _RandomDemoState();
}

class _RandomDemoState extends State<RandomDemo> {
```

```

String result = '';
TextEditingController tcMin = TextEditingController();
TextEditingController tcMax = TextEditingController();

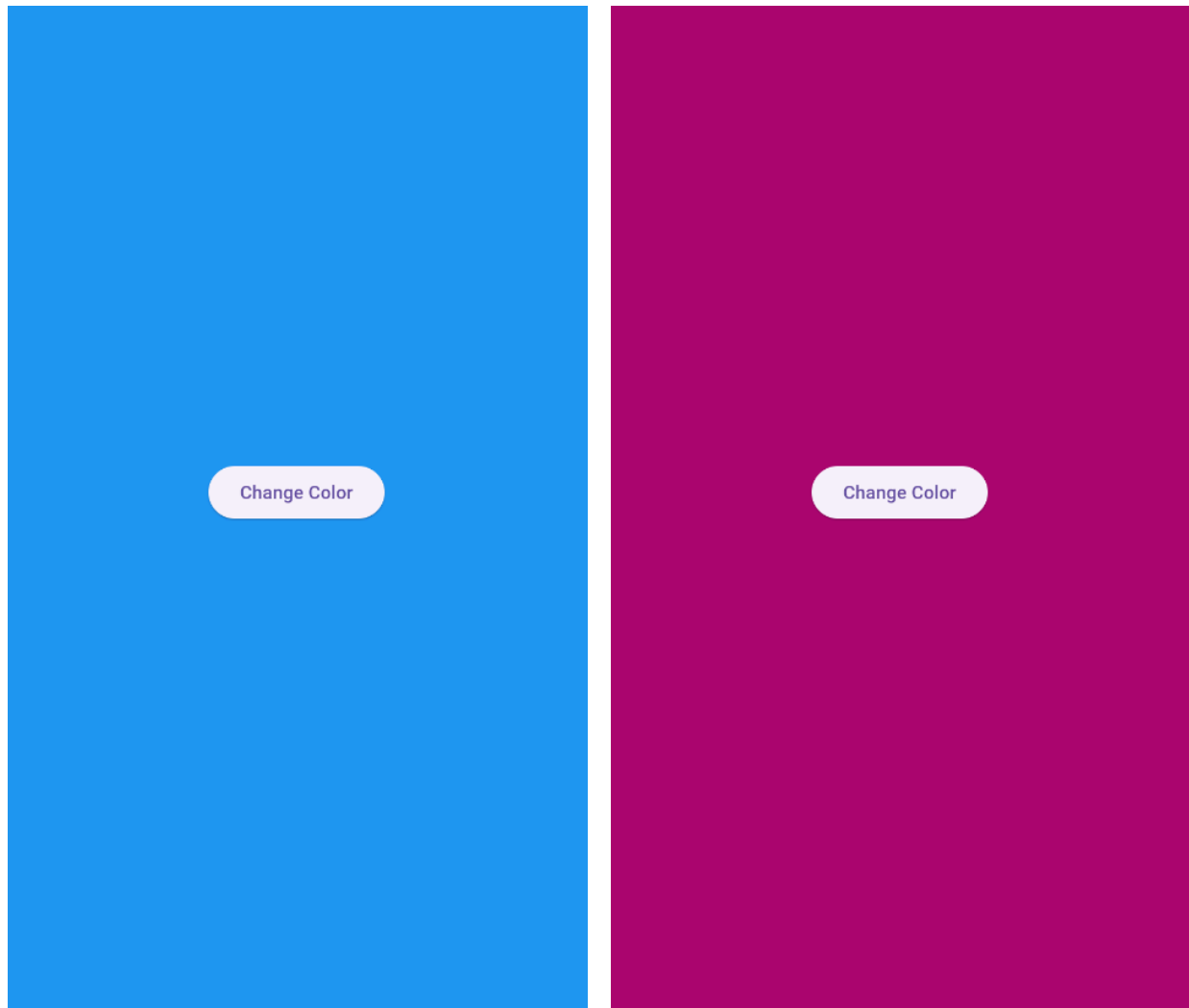
@override
Widget build(BuildContext context) {
  return Scaffold(
    body: SafeArea(
      child: Padding(
        padding: const EdgeInsets.all(16.0),
        child: Column(
          children: [
            Text(
              'Random Integer Number',
              style: Theme.of(context).textTheme.bodyLarge,
            ),
            SizedBox(height: 8),
            TextField(
              controller: tcMin,
              keyboardType: TextInputType.number,
              decoration: InputDecoration(hintText: 'min'),
            ),
            SizedBox(height: 8),
            TextField(
              controller: tcMax,
              keyboardType: TextInputType.number,
              decoration: InputDecoration(hintText: 'max'),
            ),
            SizedBox(height: 16),
            Text(result, style: Theme.of(context).textTheme.headlineMedium),
            SizedBox(height: 8),
            FilledButton(
              onPressed: () {
                int? min = int.tryParse(tcMin.text);
                int? max = int.tryParse(tcMax.text);
                if (min == null || max == null || min > max) {
                  setState(() {
                    result = 'Please check your inputs';
                  });
                }
              },
            ),
          ],
        ),
      ),
    ),
  );
}

```

```
        });  
        return;  
    }  
    // generate a random number min to max  
    int num = min + Random().nextInt(max - min + 1);  
    setState(() {  
        result = num.toString();  
    });  
    },  
    child: const Text('Generate'),  
  ),  
],  
),  
),  
),  
);  
}  
}
```

## Exercise 10 Random background color

Clicking a button will change the background color randomly.



```
import 'dart:math';
import 'package:flutter/material.dart';

class RandomColorButton extends StatefulWidget {
  const RandomColorButton({super.key});

  @override
  State<RandomColorButton> createState() => _RandomColorButtonState();
}
```

```
class _RandomColorButtonState extends State<RandomColorButton> {
  Color _backgroundColor = Colors.blue;
  final Random _random = Random();

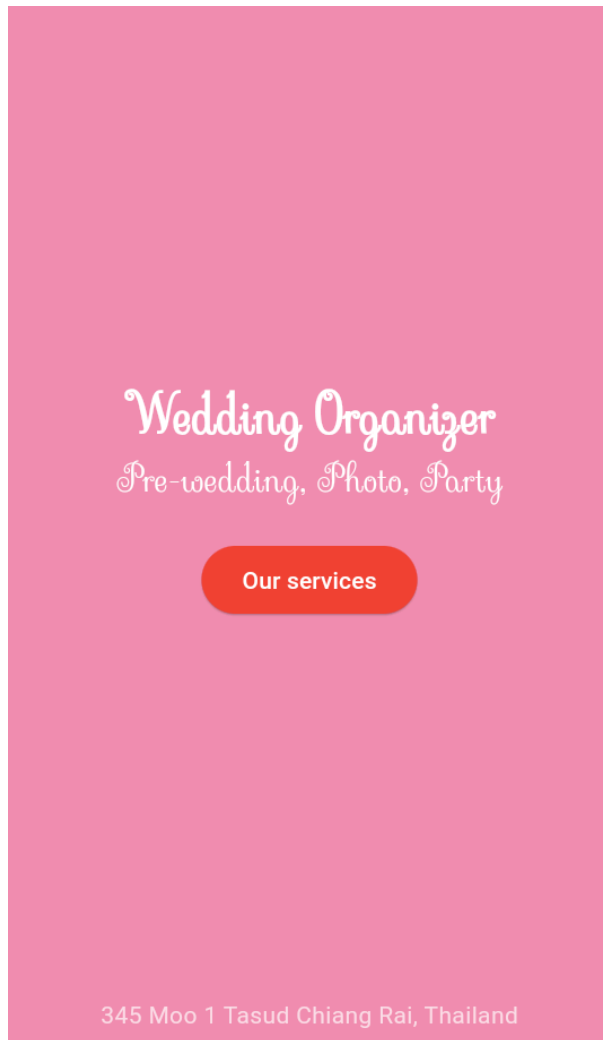
  void _changeColor() {
    setState(() {
      _backgroundColor = Color.fromRGBO(
        _random.nextInt(256), // Red
        _random.nextInt(256), // Green
        _random.nextInt(256), // Blue
        1.0, // Opacity
      );
    });
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      backgroundColor: _backgroundColor,
      body: Center(
        child: ElevatedButton(
          onPressed: _changeColor,
          child: Text('Change Color'),
        ),
      ),
    );
  }
}
```



**Assignment 1** Design an app interface below

Note that the font we use is <https://fonts.google.com/specimen/Sevillana>



Paste your code below.

```
import 'package:flutter/material.dart';

class Asm1 extends StatelessWidget {
  const Asm1({super.key});

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      backgroundColor: const Color.fromARGB(255, 255, 157, 189),
```

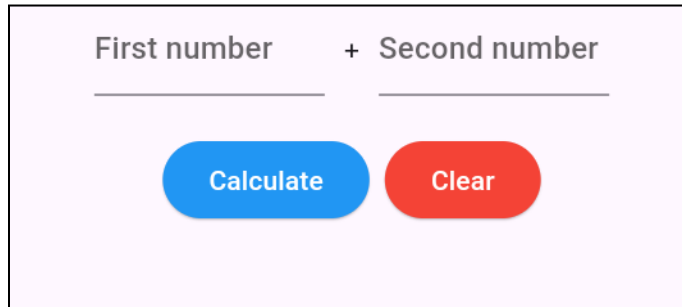
```
body: Column(
  mainAxisAlignment: MainAxisAlignment.spaceBetween,
  children: [
    Row(),
    Row(
      mainAxisAlignment: MainAxisAlignment.center,
      children: [
        Column(
          children: [
            const Text(
              "Wedding Organizer",
              style: TextStyle(
                fontFamily: "Sevillana",
                fontSize: 40,
                color: Colors.white,
              ),
            ),
            const Text(
              "Pre-wedding, Photo, Party",
              style: TextStyle(
                fontFamily: "Sevillana",
                fontSize: 25,
                color: Colors.white,
              ),
            ),
            ElevatedButton(
              onPressed: () {},
              style: ElevatedButton.styleFrom(
                backgroundColor: Colors.red,
                foregroundColor: Colors.white,
              ),
              child: const Text("Our Service"),
            ),
          ],
        ),
      ],
    ),
  ],
),
padding(
```

```
padding: const EdgeInsets.all(8.0),
child: Row(
  mainAxisAlignment: MainAxisAlignment.center,
  children: [Text("345 Moo 1 Tasud Chiang Rai, Thailand")],
),
),
],
),
);
}
```

## **Assignment 2** Create a simple summation app

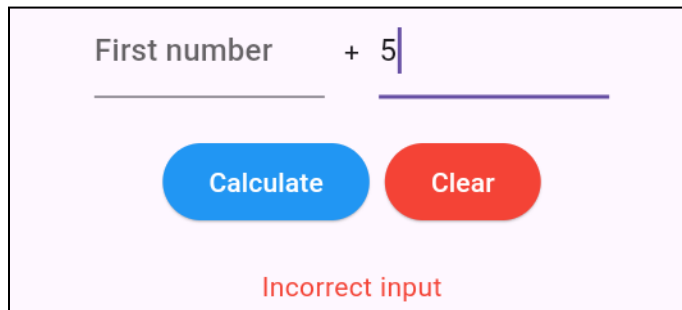
Hint: if you find an error due to TextField's size, try

<https://stackoverflow.com/questions/45986093/textfield-inside-of-row-causes-layout-exception-unable-to-calculate-size>



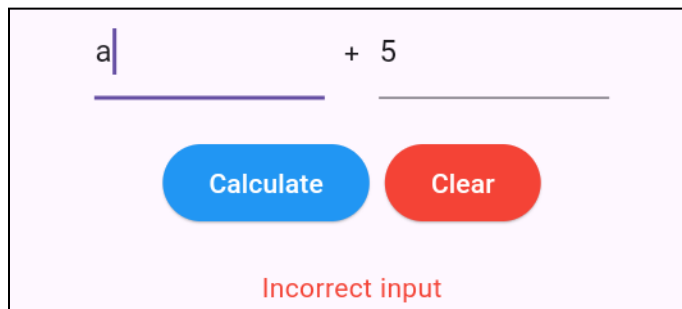
The app interface shows two empty text input fields labeled "First number" and "Second number" separated by a "+" sign. Below the inputs are two buttons: a blue "Calculate" button and a red "Clear" button.

When any input is missing and click the 'Calculate' button,



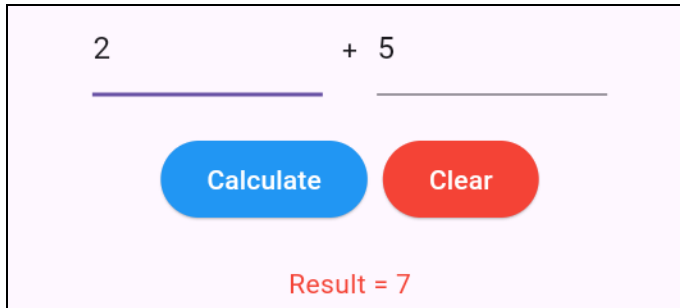
The "Second number" field now contains the value "5". The "Calculate" button is highlighted in blue. Below the buttons, the text "Incorrect input" is displayed in red.

When any input is not a number and click the 'Calculate' button,

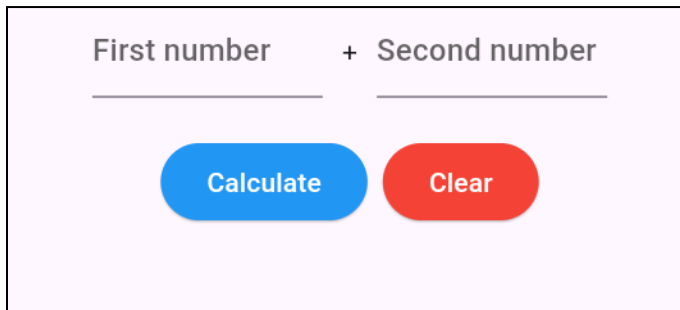


The "First number" field now contains the value "a". The "Calculate" button is highlighted in blue. Below the buttons, the text "Incorrect input" is displayed in red.

When all inputs are correct and click the 'Calculate' button,



Clicking the 'Clear' button clear all inputs and result.



Paste your code below.

```
import 'package:flutter/material.dart';

class Asm2 extends StatefulWidget {
  const Asm2({super.key});

  @override
  State<Asm2> createState() => _Asm2State();
}

class _Asm2State extends State<Asm2> {
  TextEditingController tcNum1 = TextEditingController();
  TextEditingController tcNum2 = TextEditingController();

  String ans = "";

  void Sum(String n1, String n2) {
    int? num1 = int.tryParse(n1);
    int? num2 = int.tryParse(n2);
```

```

    if (num1 != null && num2 != null) {
        setState(() {
            String value = (num1 + num2).toString();
            ans = "Result = $value";
        });
    } else {
        setState(() {
            ans = "Incorrect input";
        });
    }
}

void Clear() {
    setState(() {
        ans = "";
        tcNum1.clear();
        tcNum2.clear();
    });
}

@override
Widget build(BuildContext context) {
    return Scaffold(
        backgroundColor: const Color.fromARGB(255, 255, 229, 254),
        body: Padding(
            padding: const EdgeInsets.all(8.0),
            child: Column(
                mainAxisAlignment: MainAxisAlignment.center,
                children: [
                    Padding(
                        padding: const EdgeInsets.all(8.0),
                        child: Row(
                            mainAxisAlignment: MainAxisAlignment.center,
                            children: [
                                Expanded(
                                    child: TextField(
                                        controller: tcNum1,
                                        decoration: InputDecoration(labelText: "First Number"),

```

```

    ),
  ),
  Padding(
    padding: const EdgeInsets.all(8.0),
    child: Text(" + "),
  ),
  Expanded(
    child: TextField(
      controller: tcNum2,
      decoration: InputDecoration(labelText: "Second Number"),
    ),
  ),
],
),
),
Row(
  mainAxisAlignment: MainAxisAlignment.center,
  children: [
    Padding(
      padding: const EdgeInsets.all(8.0),
      child: ElevatedButton(
        onPressed: () {
          Sum(tcNum1.text, tcNum2.text);
        },
        style: ElevatedButton.styleFrom(
          backgroundColor: Colors.blue,
          foregroundColor: Colors.white,
        ),
        child: const Text("Calculate"),
      ),
    ),
    Padding(
      padding: const EdgeInsets.all(8.0),
      child: ElevatedButton(
        onPressed: () {
          Clear();
        },
        style: ElevatedButton.styleFrom(

```

```
        backgroundColor: Colors.red,  
        foregroundColor: Colors.white,  
      ),  
      child: const Text("Clear"),  
    ),  
  ),  
],  
,  
,  
Row(  
  mainAxisAlignment: MainAxisAlignment.center,  
  children: [Text(ans, style: TextStyle(color: Colors.red))],  
,  
],  
,  
,  
,  
);  
}  
}
```



**Assignment 3** Create a guess game. The program will randomize a number from 0-9 and ask a player to guess that number. There are three chances for guessing.

### Guess a number game

Guess a number 0-9

---

Guess

If you enter a number and press Guess, the app will show whether the guessing number is too large or too small compared to the answer.

### Guess a number game

5

---

5 is too small, 2 chance(s) left!

Guess

### Guess a number game

8

---

8 is too large, 1 chance(s) left!

Guess

# Guess a number game

7

---

Sorry, you lose. The answer is 6

Replay

Clicking “Replay” will reset the chance and generate a new random answer.

# Guess a number game

Guess a number 0-9

---

Guess

If your guess is right within 3 times,

# Guess a number game

3

---

Correct, you win!

Replay

Paste your code here.

```
import 'package:flutter/material.dart';
import 'dart:math';

class Asm3 extends StatefulWidget {
  const Asm3({super.key});

  @override
  State<Asm3> createState() => _Asm3State();
}

class _Asm3State extends State<Asm3> {
  TextEditingController guess = TextEditingController();

  String result = "";
  String buttonText = "Guess";
  bool gameOver = false;

  int ans = Random().nextInt(10);

  int chance = 3;

  void checkGuess(String num) {
    int? n = int.tryParse(num);

    if (gameOver) {
      setState(() {
        result = "";
        buttonText = "Guess";
        gameOver = false;
        chance = 3;
        ans = Random().nextInt(10);
        guess.clear();
      });
      return;
    } else {
      if (n != null && n >= 0 && n <= 9) {
```

```

    if (n == ans) {
      setState(() {
        result = "Correct, You win!";
        buttonText = "Replay";
        gameOver = true;
      });
    } else {
      chance--;
      if (chance > 0) {
        if (n < ans) {
          setState(() {
            result = "$n is too small, $chance chance(s) left.";
          });
        } else {
          setState(() {
            result = "$n is too large, $chance chance(s) left.";
          });
        }
      } else {
        setState(() {
          result = "Sorry, you lose. The answer is $ans";
          buttonText = "Replay";
          gameOver = true;
        });
      }
    }
  } else {
    setState(() {
      result = "Incorrect input";
    });
  }
}
}

@override
Widget build(BuildContext context) {
  return Scaffold(
    body: Center(

```

```

    child: Column(
      mainAxisAlignment: MainAxisAlignment.center,
      children: [
        Text("Guess a number game", style: TextStyle(fontSize: 25)),
        SizedBox(
          width: 300,
          child: TextField(
            controller: guess,
            decoration: InputDecoration(labelText: "Guess a number 0-9"),
          ),
        ),
        Padding(
          padding: const EdgeInsets.all(8.0),
          child: Text(
            result,
            style: TextStyle(color: Colors.red, fontSize: 15),
          ),
        ),
        OutlinedButton(
          onPressed: () {
            checkGuess(guess.text);
          },
          child: Text(buttonText),
        ),
      ],
    ),
  ),
);
}
}

```