C346 Android Programming

LESSON 02 – SESSION 1

Learning Objectives

- List and describe UI components used to create a user interface such as
  - Text based components (e.g. TextView, EditText)
  - Button components (e.g. Button)
- Layout various UI components according to the requirement of the mobile app intended
- Create nested layouts using LinearLayout
- Configure the properties of the UI components (e.g. width, height, color)
- Understand and use string resources
- Set up Git repository using GitHub

Git

- Version control
  - Version your code!
  - As a backup and allow rollback
  - Changes to code are documented
- Portfolio of work
  - Showing technical competencies
  - Similar to Stackoverflow

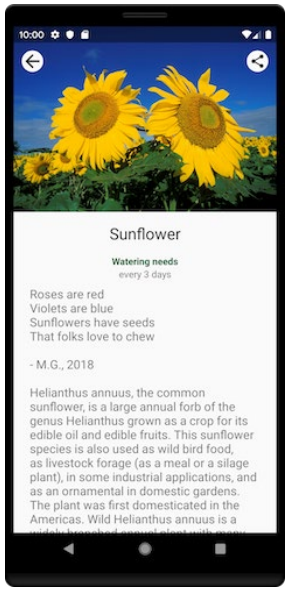
Layout

A layout defines the visual structure for a user interface in your app.

Linear Layout

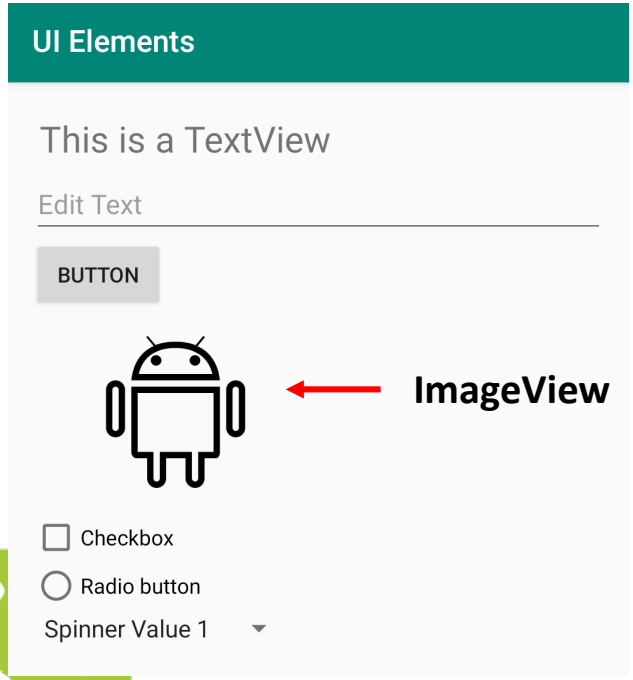
Web View

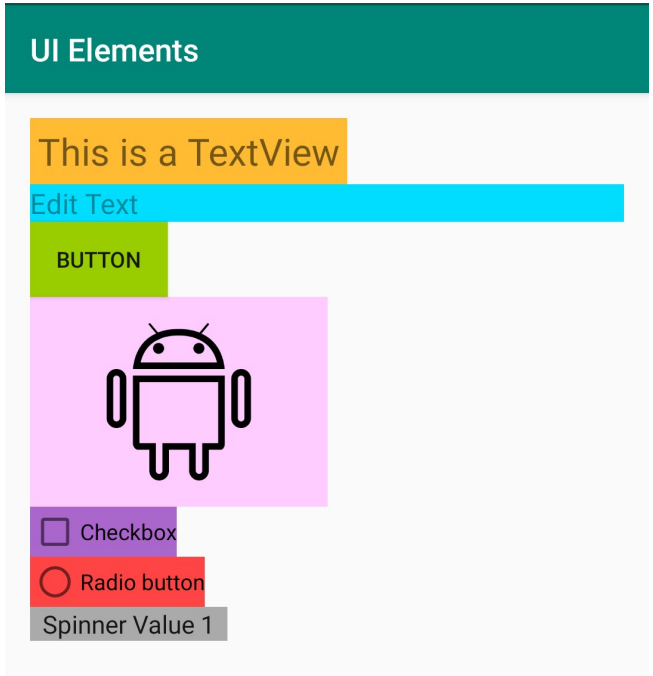
List View



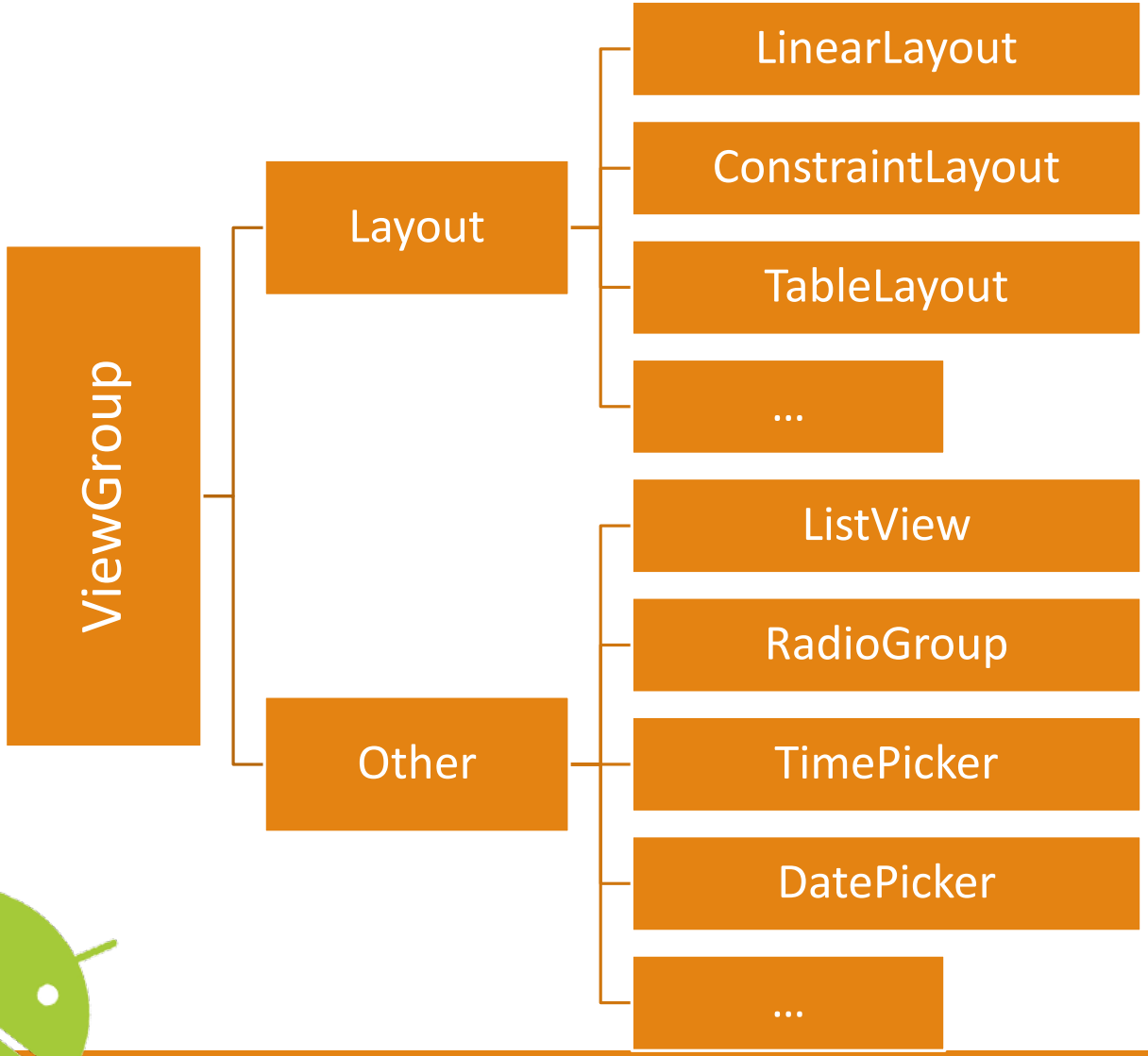
Layout

- All elements in the layout are built using a hierarchy of **View** and **ViewGroup** objects.
- A **View** usually draws something the user can see and interact with. Whereas a **ViewGroup** is an invisible container that defines the layout structure for View and other ViewGroup objects



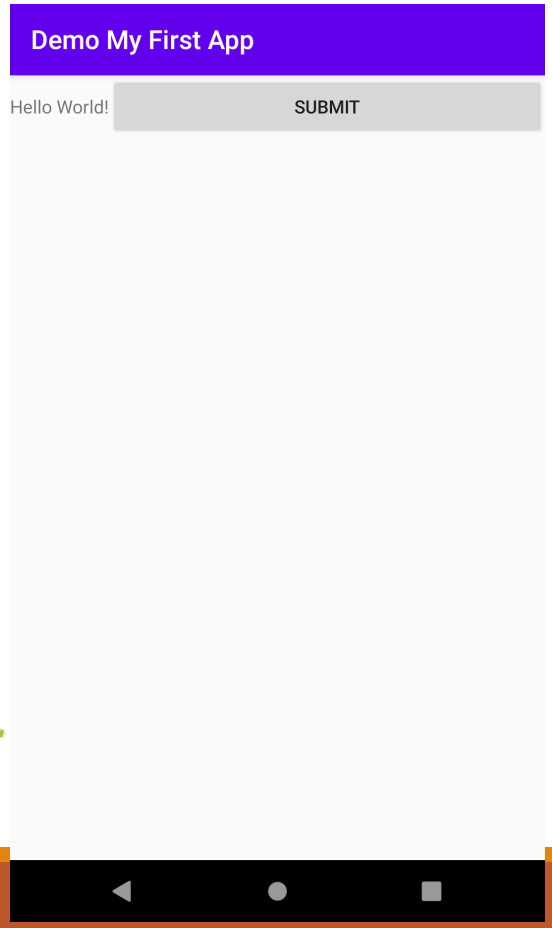


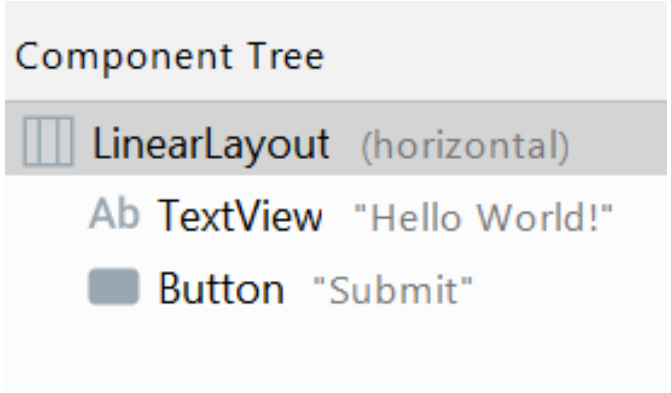
View & ViewGroup

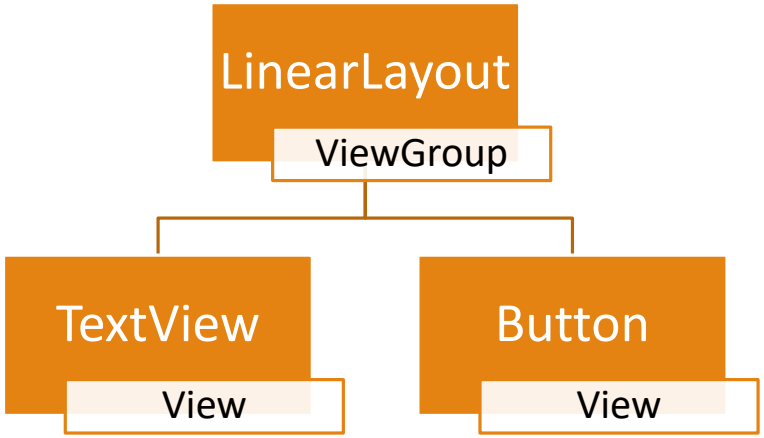


Layout

- A **ViewGroup** is an object that holds other View (and ViewGroup) objects in order to define the layout of the interface, like **LinearLayout**.







LinearLayout & UI Elements

- Note the different colors for matching.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="horizontal">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello World!" />

    <Button
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Submit" />

</LinearLayout>
```

LinearLayout

A layout that organizes its children into a single***horizontal*** or ***vertical*** row.

You can specify the layout direction with the **android:orientation** attribute.

XML namespace used to access all the Android attributes that are already defined

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="horizontal">

</LinearLayout>
```

android:orientation

Should the layout be a column or a row? Use "horizontal" for a row, "vertical" for a column.

Must be one of the following constant values.

Constant	Value	Description
horizontal	0	Defines an horizontal widget.
vertical	1	Defines a vertical widget.

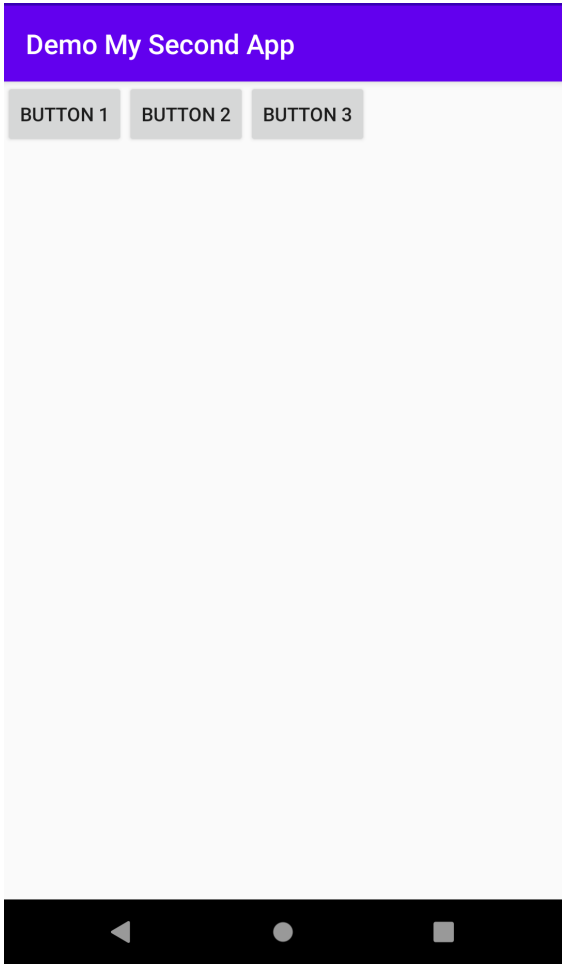
https://developer.android.com/reference/android/widget/LinearLayout.html#attr_android:orientation

Exercise 1a

The buttons are placed in a LinearLayout. What is the orientation for this LinearLayout?

Exercise 1b

The buttons are placed in a LinearLayout. What is the orientation for this LinearLayout?



UI Elements

- TextView

```
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="This is a TextView!" />
```

element

- Button

```
<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Button" />
```

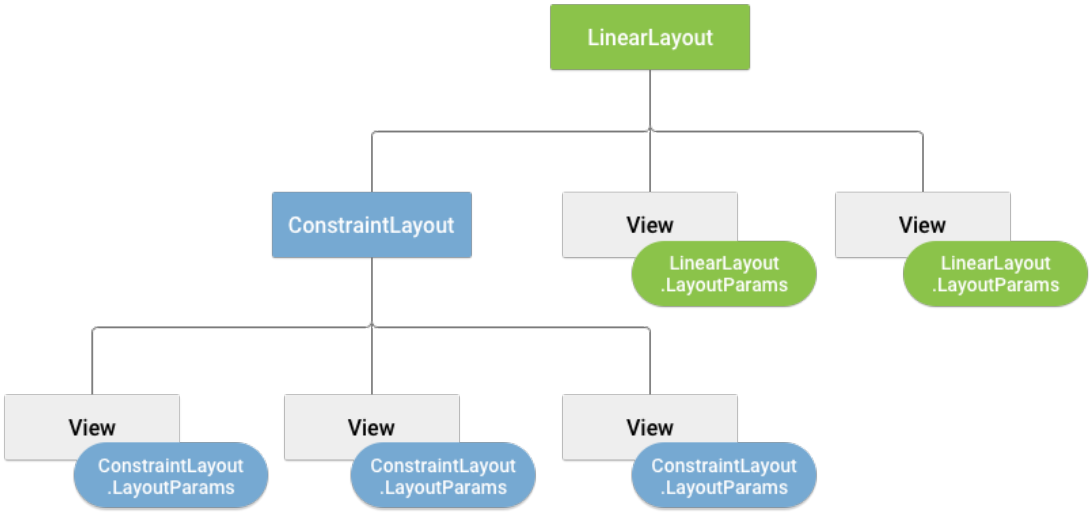
attribute

attribute name attribute value

TextView / Button

Attributes	
android:gravity	Specifies how to align the text by the view's x- and/or y-axis when the text is smaller than the view
android:height	Makes the TextView be exactly this tall
android:hint	Hint text to display when the text is empty
android:inputType	The type of data being placed in a text field, used to help an input method decide how to let the user enter text
android:numeric	If set, specifies that this TextView has a numeric input method
android:text	Text to display
android:textAppearance	Base text color, typeface, size, and style
android:textColor	Text color
android:textSize	Size of the text
android:textStyle	Style (normal, bold, italic, bold|italic) for the text
android:width	Makes the TextView be exactly this wide

Layout attributes



- All view groups include a width and height (layout_width and layout_height), and each view is required to define them.
- wrap_content - tells your view to size itself to the dimensions required by its content
- match_parent tells your view to become as big as its parent view group will allow

https://developer.android.com/guide/topics/ui/declaring-layout

XML Layout file

- activity_main.xml

activity_main.xml | MainActivity.java

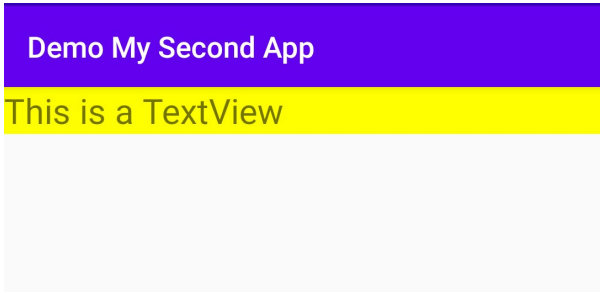
```xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="horizontal">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello World!" />

</LinearLayout>
```

Root element (parent)

Child element

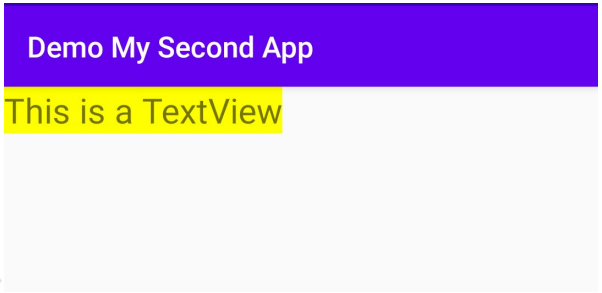
match_parent vs wrap_content

- match_parent
  - the view wants to be as big as its parent, minus the parent's padding




```
<TextView
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="This is a TextView"
    android:background="#ffff00"
    android:textSize="24sp" />
```

- wrap_content
  - the view wants to be just large enough to fit its own internal content, taking its own padding into account

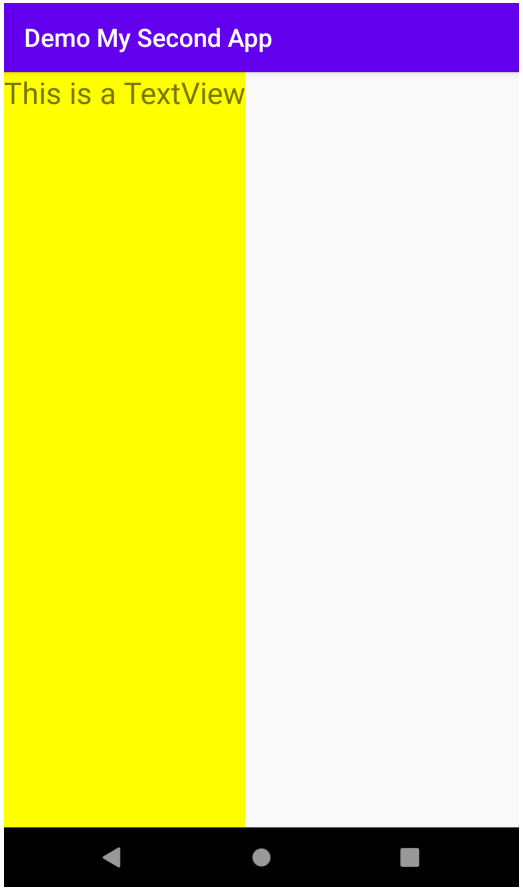



```
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="This is a TextView"
    android:background="#ffff00"
    android:textSize="24sp" />
```

Exercise 2a

Write the XML codes to display the text as shown. Font size used is 24sp.

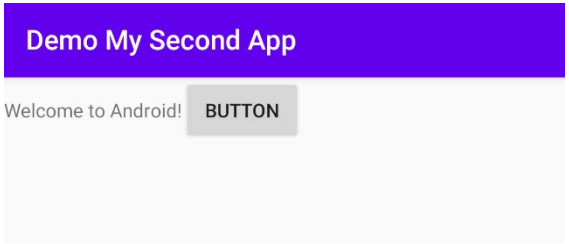
The background is #ffff00 for the text.



Exercise 2b

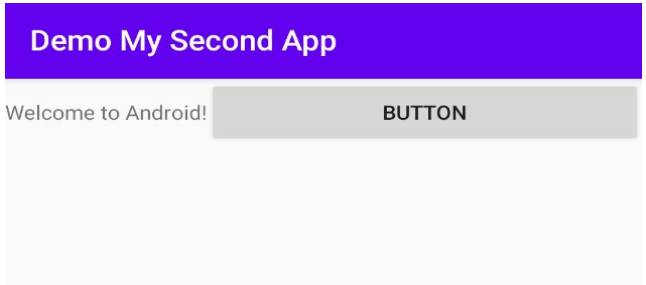
Write the XML codes to display the text and button as shown.

The default font size is used.



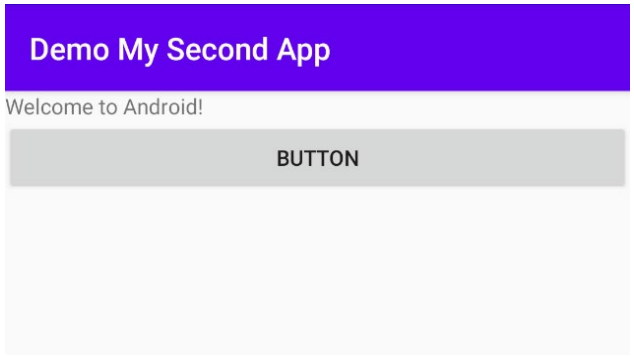
Exercise 2c

Write the XML codes to display the text and button as shown. Note that the Button takes up the remaining of the width left. The default font size is used.



Exercise 2d

Write the XML codes to display the text and button as shown. The default font size is used.



layout_weight

Layout Weight

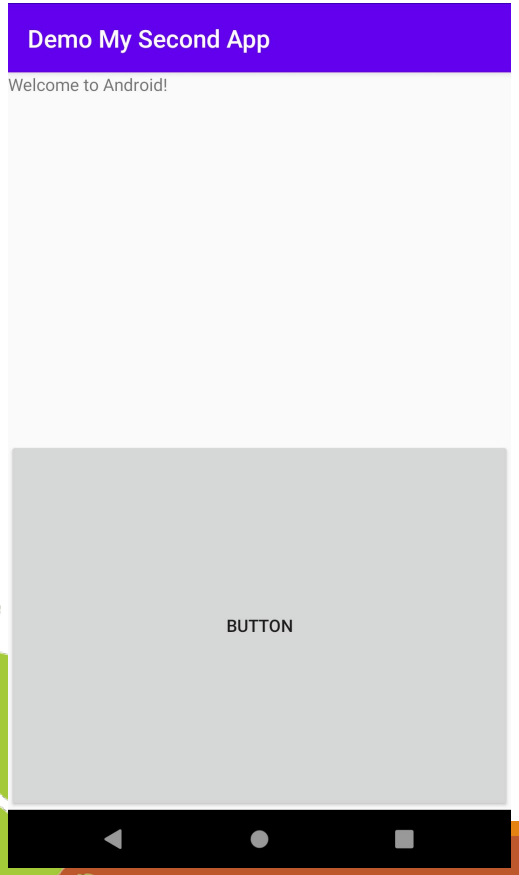
`LinearLayout` also supports assigning a *weight* to individual children with the `android:layout_weight` attribute. This attribute assigns an "importance" value to a view in terms of how much space it should occupy on the screen. A larger weight value allows it to expand to fill any remaining space in the parent view. Child views can specify a weight value, and then any remaining space in the view group is assigned to children in the proportion of their declared weight. Default weight is zero.

https://developer.android.com/guide/topics/ui/layout/linear#Weight

layout_weight

Equal distribution

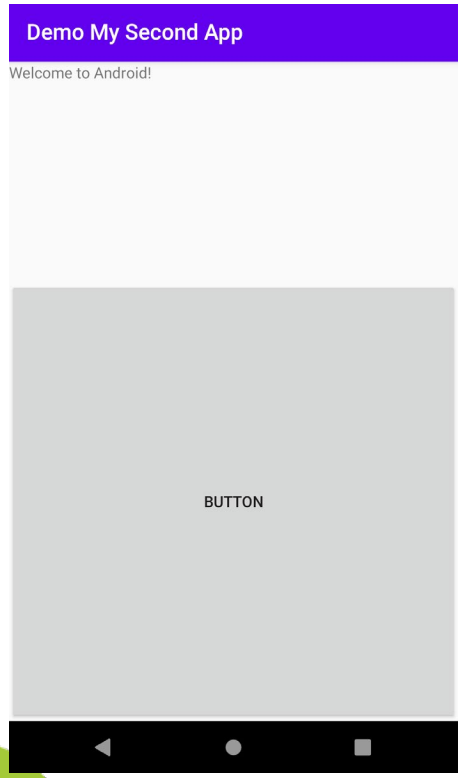
To create a linear layout in which each child uses the same amount of space on the screen, set the `android:layout_height` of each view to `"0dp"` (for a vertical layout) or the `android:layout_width` of each view to `"0dp"` (for a horizontal layout). Then set the `android:layout_weight` of each view to `"1"`.




```xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">

    <TextView
        android:id="@+id/textView"
        android:layout_width="wrap_content"
        android:layout_height="0dp"
        android:layout_weight="1"
        android:text="Welcome to Android!" />

    <Button
        android:id="@+id/button"
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="1"
        android:text="Button" />

</LinearLayout>
```

Exercise 3a

Write the XML codes to display the text and button as shown. The text takes up 1/3 of the screen and the button takes up 2/3 of the screen.

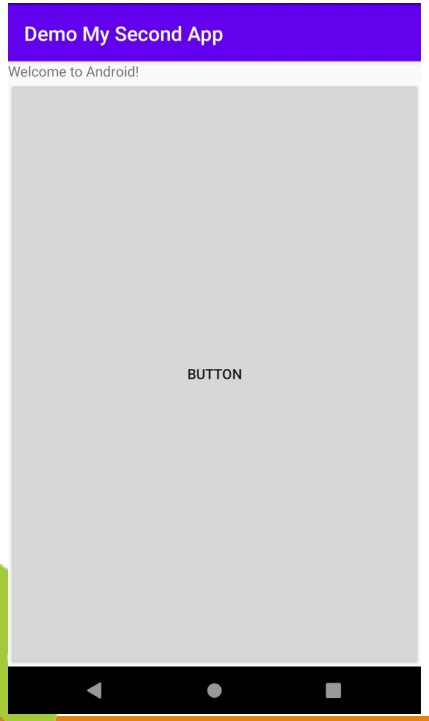


layout_weight

Unequal distribution

You can also create linear layouts where the child elements use different amounts of space on the screen:

- If there are three text fields and two of them declare a weight of 1, while the other is given no weight, the third text field without weight doesn't grow. Instead, this third text field occupies only the area required by its content. The other two text fields, on the other hand, expand equally to fill the space remaining after all three fields are measured.

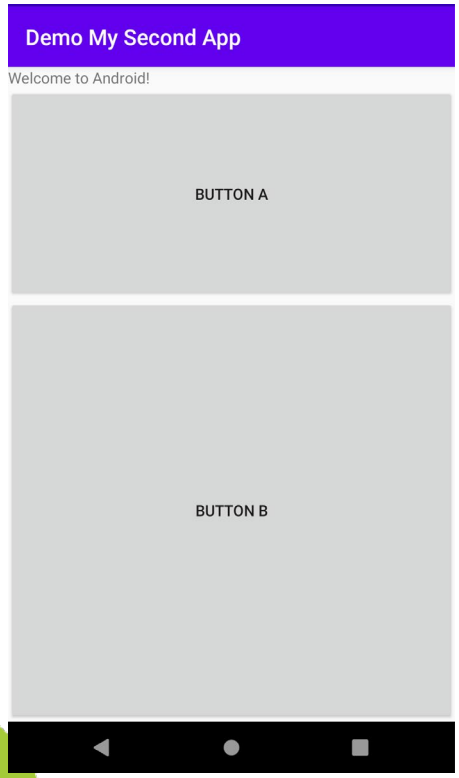



```xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">

    <TextView
        android:id="@+id/textView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Welcome to Android!" />

    <Button
        android:id="@+id/button"
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="1"
        android:text="Button" />

</LinearLayout>
```

layout_weight

Unequal distribution

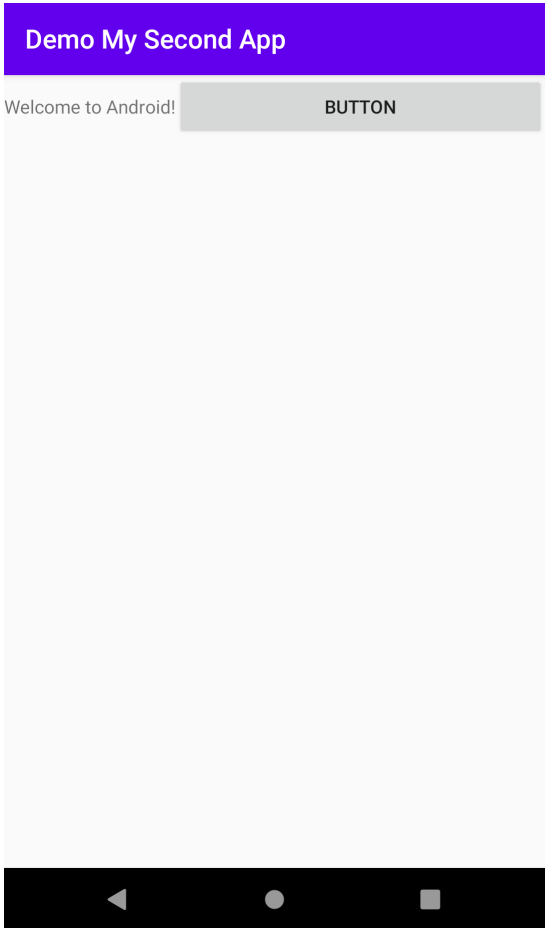
- If there are three text fields and two of them declare a weight of 1, while the third field is then given a weight of 2 (instead of 0), then it's now declared more important than both the others, so it gets half the total remaining space, while the first two share the rest equally.




```xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">

    <TextView
        android:id="@+id/textView"
        android:layout_width="wrap_content"
        android:layout_height="0dp"
        android:layout_weight="1"
        android:text="Welcome to Android!" />

    <Button
        android:id="@+id/button1"
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="1"
        android:text="Button A" />

    <Button
        android:id="@+id/button2"
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="2"
        android:text="Button B" />

</LinearLayout>
```

Exercise 3b

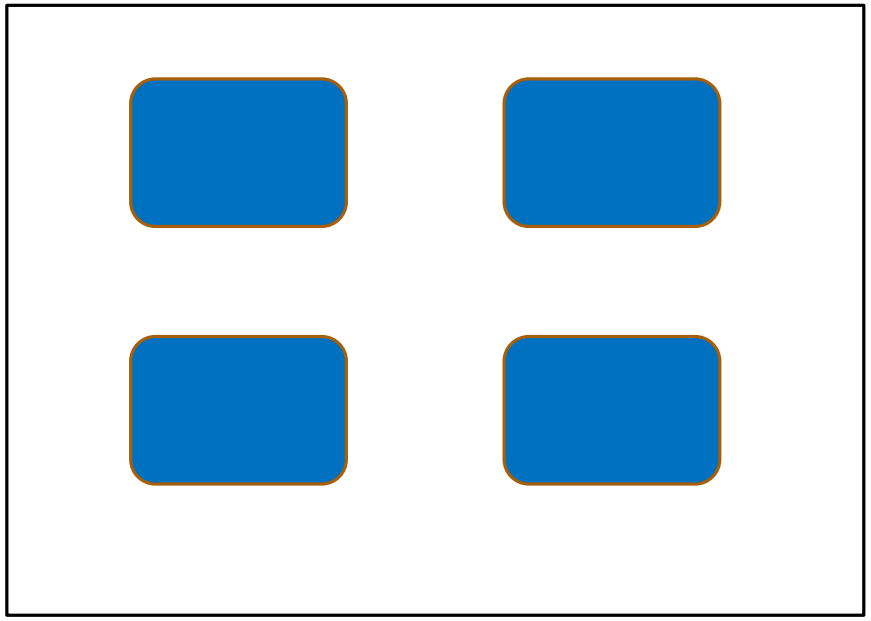
Write the XML codes to display the text and button as shown. The text takes up only the width that it needs. The button fills the remaining space.



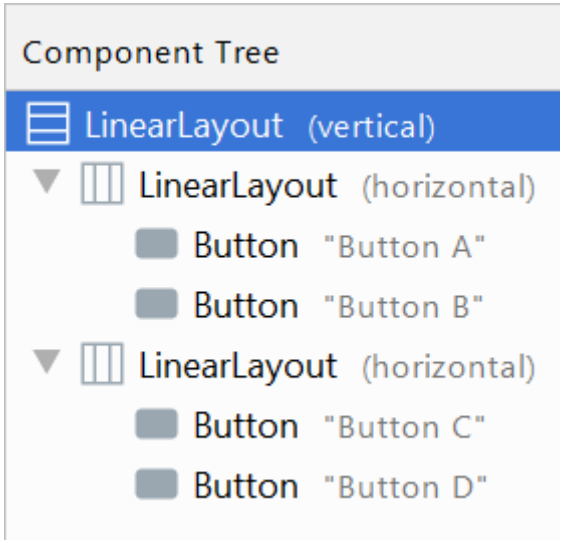
Nested Layouts

What we have done so far is implementing only one LinearLayout for the App. To solve some of the real life problems, we may need to implement more than 1 layout type or even nested layouts into the App. To support the layout design, Android OS does allow more than one LinearLayout in a canvas.

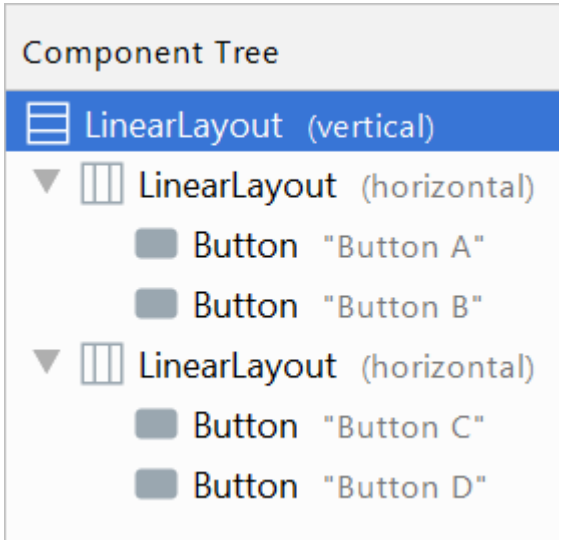
Given this scenario, how many layout(s) is/are required?



Nested Layout



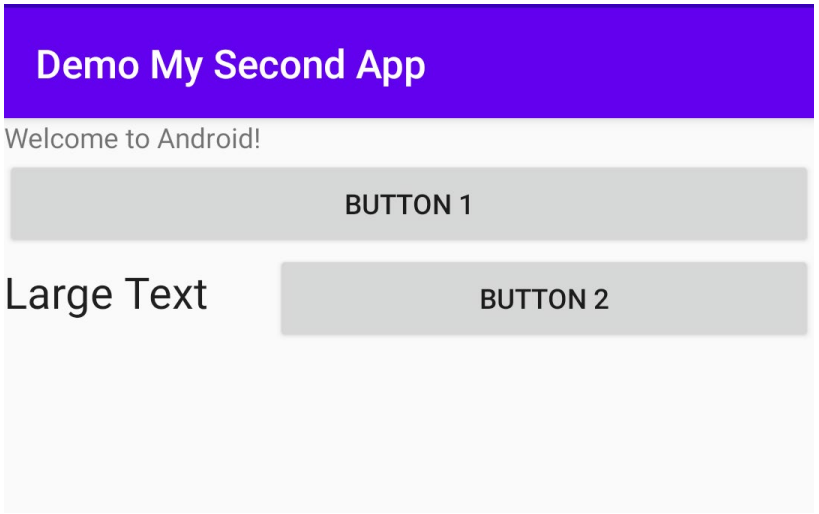
Nested Layout




```xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">

    <LinearLayout
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:orientation="horizontal">
        <Button
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Button A"/>

        <Button
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Button B"/>
    </LinearLayout>

    <LinearLayout
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:orientation="horizontal">

        <Button
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Button C"/>

        <Button
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Button D"/>
    </LinearLayout>

</LinearLayout>
```

Exercise 4a

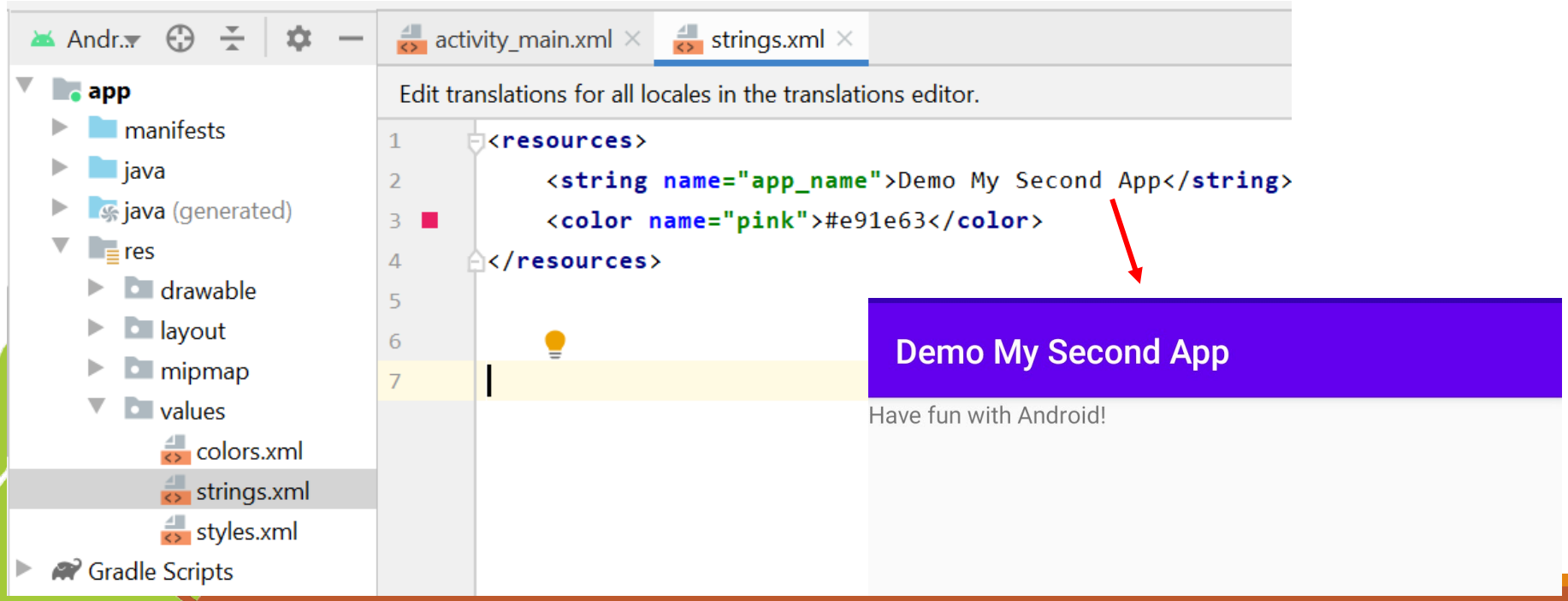
Write the XML codes to display the screen as shown.



Using String resources

- A string resource provides text strings for your application with optional text styling and formatting.

String	XML resource that provides a single string
String Array	XML resource that provides an array of strings
Color	XML resource that provides hexadecimal color codes



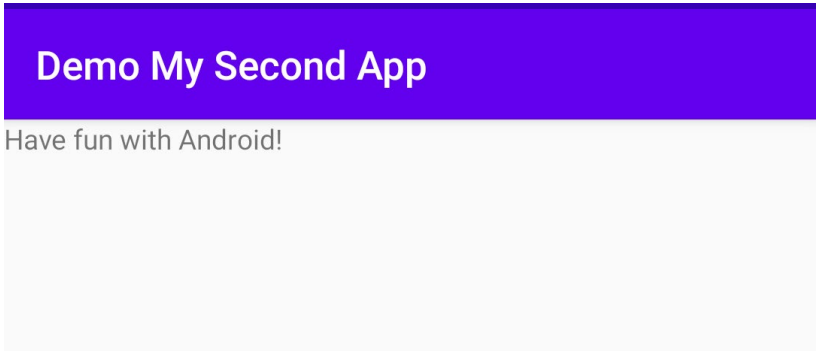
Using String resources

- A string resource provides text strings for your application with optional text styling and formatting.

strings.xml

```
1  <resources>
2      <string name="app_name">My Android App</string>
3      <string name="message">Have fun with Android!</string>
4  </resources>
```

```
<TextView
    android:text="@string/message"
    android:layout_width="match_parent"
    android:layout_height="wrap_content" />
```



- @string refers to the data type, not the file name

Exercise 4b

Remove the hardcoded text messages and replace them using string resources.