

Android Layouts

- Simple Layouts
 - Linear Layout
 - Frame Layout
 - Table Layout
- Complex Layouts
 - Relative Layout
 - Grid Layout
- In Complex Layouts there is deep Nesting but there will be performance problems
 - Also create Shallow hierarchy
 - Difficult to keep them stable
- Linear Layout
 - Linear Layout is used to align the UI elements either horizontally or vertically.
 - Only in one direction (horizontal or vertical).
 - **android:orientation** attribute should be used.

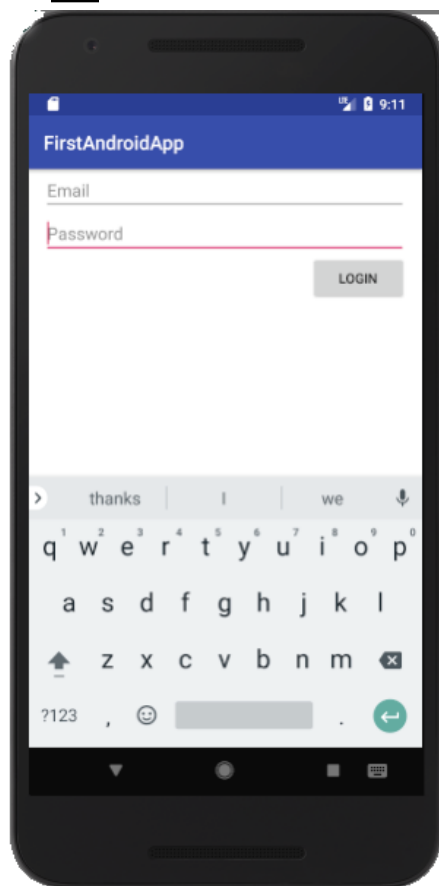
Linear layout in vertical orientation 	Linear layout in horizontal orientation 
<code>android:orientation="vertical"</code>	<code>android:orientation="horizontal"</code>

- Source:- <https://www.codzify.com/Android/linearLayouts>

o Example:-

```
<?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:paddingLeft="12dp"
    android:paddingRight="12dp"
    android:orientation="vertical" >
    <EditText
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Email" />
    <EditText
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Password" />

    <Button
        android:layout_width="100dp"
        android:layout_height="wrap_content"
        android:layout_gravity="right"
        android:text="login" />
</LinearLayout>
```



- **Frame Layout**

- Frame Layout is designed to block out an area on the screen to display a single item.
- However, add multiple children to a FrameLayout and control their position within the FrameLayout by assigning gravity to each child, using the **android:layout_gravity** attribute.
- Child views are drawn in a **stack**, with the most recently added child on top.
- The size of the FrameLayout is the **size of its largest child** (plus padding).

- **Example:-**

activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">
    <ImageView
        android:id="@+id/imgvw1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:scaleType="centerCrop"
        android:src="@drawable/flimg" />
    <TextView
        android:id="@+id/txtvw1"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="40dp"
        android:background="#4C374A"
        android:padding="10dp"
        android:text="Grand Palace, Bangkok"
        android:textColor="#FFFFFF"
        android:textSize="20sp" />
    <TextView
        android:id="@+id/txtvw2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="right|bottom"
        android:background="#AA000000"
        android:padding="10dp"
        android:text="21/Aug/2017"
        android:textColor="#FFFFFF"
```

```
        android:textSize="18sp" />
</FrameLayout>
```

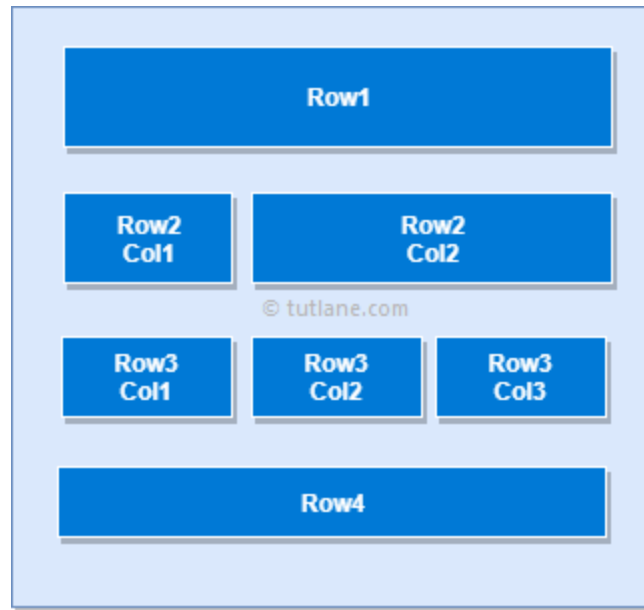


- **Source:-**

- <https://www.tutlane.com/tutorial/android/android-framelayout-with-examples>

- **Table Layout**

- A layout that arranges its children into rows and columns.
 - A TableLayout consists of a number of `TableRow` objects, each defining a row
 - Do not display border lines for their rows, columns, or cells.
 - Each row has zero or more cells;
 - Each cell can hold one `View` object.
 - Width(of a child) is always `MATCH_PARENT`.
 - We can specify certain columns as shrinkable or stretchable by calling `setColumnShrinkable()` or `setColumnStretchable()`
 - stretchable -to fit any extra space
 - shrinkable - shrunk to fit the table into its parent object



0

0

```
<?xml version="1.0" encoding="utf-8"?>
<TableLayout xmlns:android="http://schemas.android.com/apk/res
/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:layout_marginTop="100dp"
    android:paddingLeft="10dp"
    android:paddingRight="10dp" >

    <TableRow android:background="#0079D6" android:padding="5dp">
        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:text="UserId" />
        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:text="User Name" />
        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:text="Location" />
    </TableRow>

    <TableRow android:background="#DAE8FC" android:padding="5dp">
```

```

        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:text="1" />
        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:text="Suresh Dasari" />
        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:text="Hyderabad" />
    </TableRow>

    <TableRow android:background="#DAE8FC" android:padding="5dp">
        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:text="2" />
        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:text="Rohini Alavala" />
        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:text="Guntur" />
    </TableRow>

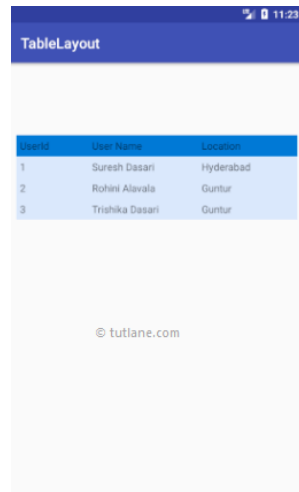
    <TableRow android:background="#DAE8FC" android:padding="5dp">
        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:text="3" />
        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"

```

```

        android:layout_weight="1"
        android:text="Trishika Dasari" />
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:text="Guntur" />
    </TableRow>
</TableLayout>

```



o

o **Source:-**

<https://www.tutlane.com/tutorial/android/android-tablelayout-with-examples>

RelativeLayout

- A Layout where the positions of the children can be described in relation to each other or to the parent.

Attribute	Description
layout_alignParentTop	If it specified "true", the top edge of view will match the top edge of parent.
layout_alignParentBottom	If it specified "true", the bottom edge of view will match the bottom edge of parent.
layout_alignParentLeft	If it specified "true", the left edge of view will match the left edge of parent.
layout_alignParentRight	If it specified "true", the right edge of view will match the right edge of parent.

layout_centerInParent	If it specified "true", the view will be aligned to centre of parent.
layout_centerHorizontal	If it specified "true", the view will be horizontally centre aligned within its parent.
layout_centerVertical	If it specified "true", the view will be vertically centre aligned within its parent.
layout_above	It accepts another sibling view id and places the view above the specified view id.
layout_below	It accepts another sibling view id and places the view below the specified view id.
layout_toLeftOf	It accepts another sibling view id and places the view left of the specified view id.
layout_toRightOf	It accepts another sibling view id and places the view right of the specified view id.
layout_toStartOf	It accepts another sibling view id and places the view to start of the specified view id.
layout_toEndOf	It accepts another sibling view id and places the view to end of the specified view id.

● **Example:-**

```

0 <?xml version="1.0" encoding="utf-8"?>
  <RelativeLayout xmlns:android="http://schemas.android.com/apk/
res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingLeft="10dp"
    android:paddingRight="10dp">
    <Button
        android:id="@+id/btn1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:text="Button1" />
    <Button
        android:id="@+id/btn2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentRight="true"
        android:layout_centerVertical="true"

```

```

        android:text="Button2" />
    <Button
        android:id="@+id/btn3"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_centerVertical="true"
        android:text="Button3" />

    <Button
        android:id="@+id/btn4"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_alignParentBottom="true"
        android:text="Button4" />
    <Button
        android:id="@+id/btn5"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignBottom="@+id/btn2"
        android:layout_centerHorizontal="true"
        android:text="Button5" />
    <Button
        android:id="@+id/btn6"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_above="@+id/btn4"
        android:layout_centerHorizontal="true"
        android:text="Button6" />
    <Button
        android:id="@+id/btn7"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_toEndOf="@+id/btn1"
        android:layout_toRightOf="@+id/btn1"
        android:layout_alignParentRight="true"
        android:text="Button7" />
</RelativeLayout>

```



0

0 **Source:-**

<https://www.tutlane.com/tutorial/android/android-relativelayout-with-examples>

• **GridLayout**

0 The grid is composed of a set of infinitely thin lines that separate the viewing area into *cells*.

0 A grid with N columns has $N + 1$ grid indices that run from 0 through N inclusive.

0

```
<GridLayout
xmlns:android="http://schemas.android.com/apk/res/android"
xmlns:tools="http://schemas.android.com/tools"
android:layout_width="match_parent"
android:layout_height="match_parent"
android:columnCount="4"
android:rowCount="3"
tools:context="com.android_examples.com.gridlayout.MainActivity"
>
<TextView
android:id="@+id/textView1"
android:layout_row="0"
android:layout_column="0"
android:text="A"
android:textAppearance="?android:attr/textAppearanceLarge"
android:padding="30dp"/>
<TextView
android:id="@+id/textView2"
android:layout_row="0"
android:layout_column="1"
android:text="B"
android:padding="30dp"
android:textAppearance="?android:attr/textAppearanceLarge" />
```

```
<TextView
android:id="@+id/textView3"
android:layout_row="0"
android:layout_column="2"
android:text="C"
android:padding="30dp"
android:textAppearance="?android:attr/textAppearanceLarge" />
```

```
<TextView
android:id="@+id/textView4"
android:layout_row="0"
android:layout_column="3"
android:text="D"
android:padding="30dp"
android:textAppearance="?android:attr/textAppearanceLarge" />
```

//2nd row starts from here.

```
<TextView
android:id="@+id/textView5"
android:layout_row="1"
android:layout_column="0"
android:text="E"
android:textAppearance="?android:attr/textAppearanceLarge"
android:padding="30dp"/>
```

```
<TextView
android:id="@+id/textView6"
android:layout_row="1"
android:layout_column="1"
android:text="F"
android:padding="30dp"
android:textAppearance="?android:attr/textAppearanceLarge" />
```

```
<TextView
android:id="@+id/textView7"
android:layout_row="1"
android:layout_column="2"
android:text="G"
android:padding="30dp"
android:textAppearance="?android:attr/textAppearanceLarge" />
```

```
<TextView
android:id="@+id/textView8"
android:layout_row="1"
android:layout_column="3"
android:text="H"
android:padding="30dp"
android:textAppearance="?android:attr/textAppearanceLarge" />
```

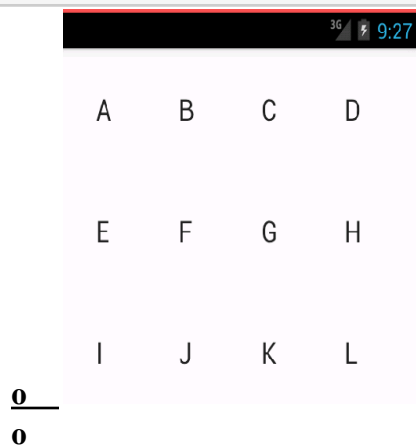
//3rd row starts from here.

```
<TextView
android:id="@+id/textView9"
android:layout_row="2"
android:layout_column="0"
android:text="I"
android:textAppearance="?android:attr/textAppearanceLarge"
android:padding="30dp"/>

<TextView
android:id="@+id/textView10"
android:layout_row="2"
android:layout_column="1"
android:text="J"
android:padding="30dp"
android:textAppearance="?android:attr/textAppearanceLarge" />

<TextView
android:id="@+id/textView11"
android:layout_row="2"
android:layout_column="2"
android:text="K"
android:padding="30dp"
android:textAppearance="?android:attr/textAppearanceLarge" />

<TextView
android:id="@+id/textView12"
android:layout_row="2"
android:layout_column="3"
android:text="L"
android:padding="30dp"
android:textAppearance="?android:attr/textAppearanceLarge" />
</GridLayout>
```



Layout Width and Height Attributes

- **wrap_content**: widget as wide/high as its content (e.g. text)
- **match_parent**: widget as wide/high as its parent layout box
- **fill_parent**: older form of **match_parent**

Text widget width
should be as wide as
its parent (the layout)

Text widget height
should be as wide as
the content (text)

Screen (Hardware)
↑
Linear Layout
↑
TextView

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

    <TextView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="@string/hello"
    />

</LinearLayout>
```

The View inside the
layout is a TextView, a
View specifically made
to display text.

XML
main.xml

The ViewGroup, in this
case a LinearLayout
fills the screen.



Adding Padding

- Paddings sets space between layout sides and its parent

```
<RelativeLayout ...
```

```
    android:paddingBottom="16dp"
```

```
    android:paddingLeft="16dp"
```

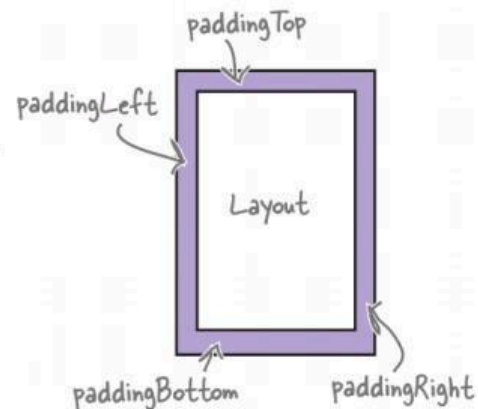
```
    android:paddingRight="16dp"
```

```
    android:paddingTop="16dp">
```

```
    ...
```

```
</RelativeLayout>
```

Add padding of 16dp.



0

Linear Layout Weight Attribute

- Specifies "importance", larger weights takes up more space
- Can set width, height = 0 then
 - weight = percent of height/width you want element to cover

```
<?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">

    <Button
        android:layout_width="match_parent"
        android:layout_height="0dip"
        android:layout_weight="50"
        android:text="@string/fifty_percent"/>

    <Button
        android:layout_width="match_parent"
        android:layout_height="0dip"
        android:layout_weight="30"
        android:text="@string/thirty_percent"/>

    <Button
        android:layout_width="match_parent"
        android:layout_height="0dip"
        android:layout_weight="20"
        android:text="@string/twenty_percent"/>

</LinearLayout>
```



0

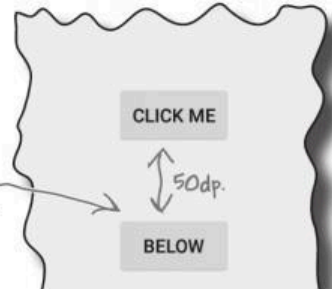
Setting Margins



- Can increase gap (margin) between adjacent widgets
- E.g. To add margin between two buttons, in declaration of bottom button

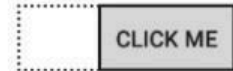
```
<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignLeft="@+id/button_click_me"
    android:layout_below="@+id/button_click_me"
    android:layout_marginTop="50dp"
    android:text="@string/button_below" />
</RelativeLayout>
```

Adding a margin to the top of the bottom button adds extra space between the two views.



- Other options

android:layout_marginLeft



android:layout_marginRight



0