<https://www.geeksforgeeks.org/absolute-layout-in-android-with-example/>

<https://www.geeksforgeeks.org/relative-layout-in-android/>

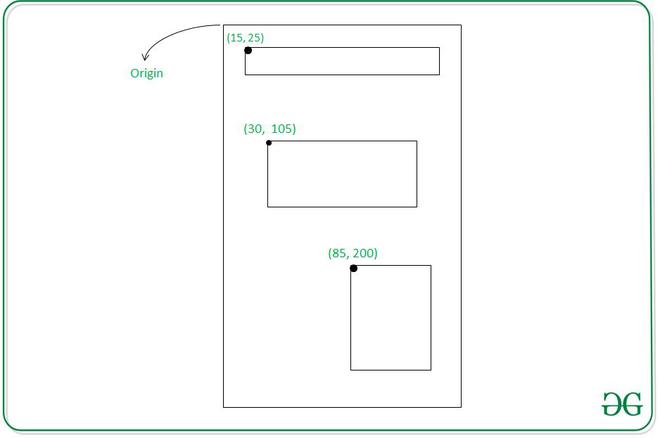
<https://www.geeksforgeeks.org/android-linearlayout-in-kotlin/>

<https://www.geeksforgeeks.org/linearlayout-and-its-important-attributes-with-examples-in-android/>

Experiment No 3.2 (No.8 in unit-3)

Implement building blocks for Android Application using different layouts (such as linear, relative and absolute).(CO5)

An **Absolute Layout** allows you to specify the exact location .i.e., X and Y coordinates of its children with respect to the origin at the top left corner of the layout. The absolute layout is less flexible and harder to maintain for varying sizes of screens that’s why it is not recommended. Although Absolute Layout is deprecated now.



Some of the important Absolute Layout attributes are the following:

1. **android:id**: It uniquely specifies the absolute layout
2. **android:layout\_x:** It specifies X-Coordinate of the Views (Possible values of this is in [density-pixel or pixel](https://www.geeksforgeeks.org/how-to-scale-different-views-to-all-screen-sizes-in-android-studio/))
3. **android:layout\_y:** It specifies Y-Coordinate of the Views (Possible values of this is in dp or px)

**The Syntax for Absolute Layout**

<AbsoluteLayout

xmlns:android="http://schemas.android.com/apk/res/android"

android:layout\_width="fill\_parent"

android:layout\_height="fill\_parent">

<!--add child views-->

</AbsoluteLayout>

**Example**

In this example, we are going to create a basic application with Absolute Layout that is having two [TextView](https://www.geeksforgeeks.org/textview-widget-in-android-using-java-with-examples/). Note that we are going to implement this project using the **Java**language.

**Step by Step Implementation**

**Step 1: Create a New Project**

o create a new project in Android Studio please refer to [How to Create/Start a New Project in Android Studio](https://www.geeksforgeeks.org/android-how-to-create-start-a-new-project-in-android-studio/). Note that select **Java**as the programming language.

**Step 2: Create the layout file**

For this go to **app > res > layout > activity\_main.xml** file and change the Constraint Layout to Absolute Layout and add [TextViews](https://www.geeksforgeeks.org/textview-widget-in-android-using-java-with-examples/). Below is the code snippet for the **activity\_mian.xml** file.

<?xml version="1.0" encoding="utf-8"?>

<AbsoluteLayout

xmlns:android="http://schemas.android.com/apk/res/android"

xmlns:app="http://schemas.android.com/apk/res-auto"

xmlns:tools="http://schemas.android.com/tools"

android:layout\_width="fill\_parent"

android:layout\_height="fill\_parent"

tools:context=".MainActivity">

<!--Setting up TextViews-->

<TextView

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_x="100px"

android:layout\_y="300px" />

<TextView

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_x="120px"

android:layout\_y="350px" />

</AbsoluteLayout>

Before moving further let’s add some color attributes in order to enhance the app bar. Go to **app > res > values > colors.xml** and add the following color attributes.

<resources>

<color name="colorPrimary">#0F9D58</color>

<color name="colorPrimaryDark">#16E37F</color>

<color name="colorAccent">#03DAC5</color>

</resources>

**Step 3: Working with the MainActivity.java file**

In this step, we will initialize the TextViews in our **MainActivity.java** file.

import androidx.appcompat.app.AppCompatActivity;

import android.os.Bundle;

import android.widget.TextView;

public class MainActivity extends AppCompatActivity {

TextView heading, subHeading;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

// Referencing the TextViews

heading = (TextView) findViewById(R.id.heading);

subHeading = (TextView) findViewById(R.id.subHeading);

// Setting text dynamically

heading.setText("Computer Science Portal");

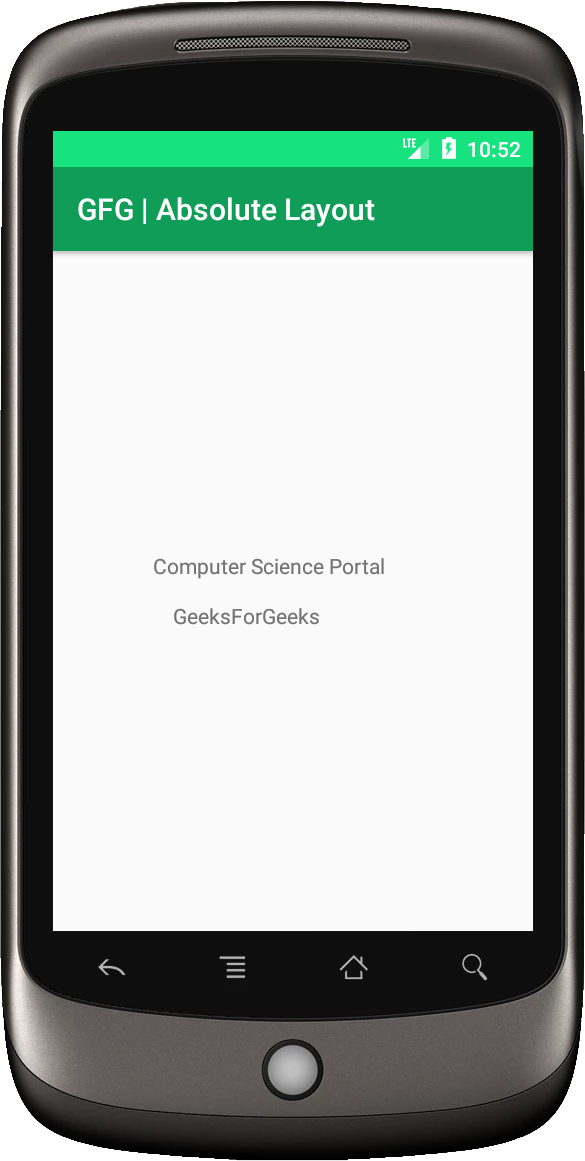
subHeading.setText("GeeksForGeeks");

}

}

**Output: Run On Emulator**

You will see that TextViews are having fixed X and Y Coordinates.



# Relative Layout in Android

The relative layout is used to arrange the child views in a proper order which means arranging the child objects relative to each other. Generally, if we create an application using a linear layout that consists of 5 buttons. Even if we specify weight and gravity properties they will not be relatively arranged. To arrange them in a proper order we need to use the relative layout. To arrange them we need some advanced properties. Basically, we use layout\_width, layout\_height, layout\_text properties while creating an application using the linear layout. But we need some more advanced properties which are supported by relative layout. There are so many properties that are supported by relative layout. some of the most used properties are listed below

* layout\_alignParentTop
* layout\_alignParentBottom
* layout\_alignParentRight
* layout\_alignParentLeft
* layout\_centerHorozontal
* layout\_centerVertical
* layout\_above
* layout\_below

To execute the below code we need to install Android studio. After installing follow the below steps.

1. Create a new project
2. Select the language as java
3. select the API level of 26(android OREO 8.0)
4. Click on the ok button. Wait until UI displays.
5. Then clean the project. Because gradle may not build.
6. After that, we can see two code files. One is activity\_main.xml and MainActivity.java.
7. Go to the activity\_main.xml file.
8. In that, we can see two modes. One is design mode and another one is code mode. Then go to code mode.
9. And Use the below code.
10. We do not need to change the default java code. Because we are not performing any onClick activities.

Let us see an XML code that shows the usage of the above-mentioned properties.

### ****Step by Step Implementation****

**Step 1: Create a New Project**

To create a new project in Android Studio please refer to [How to Create/Start a New Project in Android Studio](https://www.geeksforgeeks.org/android-how-to-create-start-a-new-project-in-android-studio/). Note that select **Java** as the programming language.

**Step 2: Working with the activity\_main.xml file**

Navigate to the **app > res > layout > activity\_main.xml**and add the below code to that file. Below is the code for the **activity\_main.xml** file.

<?xml version="1.0" encoding="utf-8"?>

<RelativeLayout

android:layout\_width="fill\_parent"

android:layout\_height="fill\_parent"

xmlns:android="http://schemas.android.com/apk/res/android">

<Button

android:id="@+id/button1"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Top Left Button"

android:layout\_alignParentLeft="true"

android:layout\_alignParentTop="true"/>

<Button

android:id="@+id/button2"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Top Right Button"

android:layout\_alignParentTop="true"

android:layout\_alignParentRight="true"/>

<Button

android:id="@+id/button3"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Bottom Left Button"

android:layout\_alignParentLeft="true"

android:layout\_alignParentBottom="true"/>

<Button

android:id="@+id/button4"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Bottom Right Button"

android:layout\_alignParentRight="true"

android:layout\_alignParentBottom="true"/>

<Button

android:id="@+id/button5"

android:layout\_width="fill\_parent"

android:layout\_height="wrap\_content"

android:text="Middle Button"

android:layout\_centerVertical="true"

android:layout\_centerHorizontal="true"/>

</RelativeLayout>

**Step 4: Working with the** **MainActivity.java file**

Go to the **MainActivity.java** file and refer to the following code. Below is the code for the **MainActivity.java** file. There is nothing to write inside the MainActivity.java file.

package com.example.example1;

import androidx.appcompat.app.AppCompatActivity;

import android.os.Bundle;

public class MainActivity extends AppCompatActivity {

@Override

protected void onCreate(Bundle savedInstanceState) {

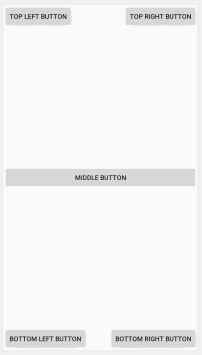
super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

}

}

**Output:**



# Android LinearLayout in Kotlin

Android **LinearLayout** is a ViewGroup subclass, used to provide child View elements one by one either in a particular direction either horizontally or vertically based on the orientation property. We can specify the linear layout orientation using **android:orientation** attribute.

All the child elements arranged one by one in multiple rows and multiple columns.

1. **Horizontal list:** One row, multiple columns.
2. **Vertical list:** One column, multiple rows.

In this article, we are going to discuss declaration and implementation of LinearLayout.

**How to declare LinearLayout in XML file?**

First of all, we should declare the LinearLayout in layout file using the below code.

* 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" >

// Add another child elements here like

// EditText, button etc

</LinearLayout>

## LinearLayout in activity\_main.xml file

Following is the code for LinearLayout in xml file.

<?xml version="1.0" encoding="utf-8"?>

<LinearLayout

xmlns:android="http:// schemas.android.com/apk/res/android"

xmlns:tools="http:// schemas.android.com/tools"

android:orientation="vertical"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

tools:context=".MainActivity">

<TextView

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:layout\_margin="16dp"

android:text="Enter your name here:"

android:textSize="24dp"

android:id="@+id/txtVw"/>

<EditText

android:id="@+id/editText"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:layout\_margin="16dp"

android:hint="Name"

android:inputType="text"/>

<Button

android:id="@+id/showInput"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_gravity="center\_horizontal"

android:text="show"

android:backgroundTint="@color/colorPrimary"

android:textColor="@android:color/white"/>

</LinearLayout>

## MainActivity.kt file

When we have created layout, we need to load the XML layout resource from our activity **onCreate()** callback method and access the UI element form the XML using **findViewById**.

package com.geeksforgeeks.myfirstkotlinapp

import androidx.appcompat.app.AppCompatActivity import android.os.Bundle import android.widget.Button import android.widget.EditText import android.widget.TextView

class MainActivity : AppCompatActivity() {

override fun onCreate(savedInstanceState: Bundle?)

{

super.onCreate(savedInstanceState)

setContentView(R.layout.activity\_main)

// finding the UI elements

val showButton

= findViewById<Button>(R.id.showInput)

val editText

= findViewById<EditText>(R.id.editText)

val textView

= findViewById<TextView>(R.id.txtVw)

}

}

## LinearLayout Output:

We can run the application using the Android Virtual Device(AVD) to get the output of the above code.

