Islamic University of Gaza
Faculty of Information Technology
Android Programming Lab (MOBC 2101)

Lab 2

Android XML

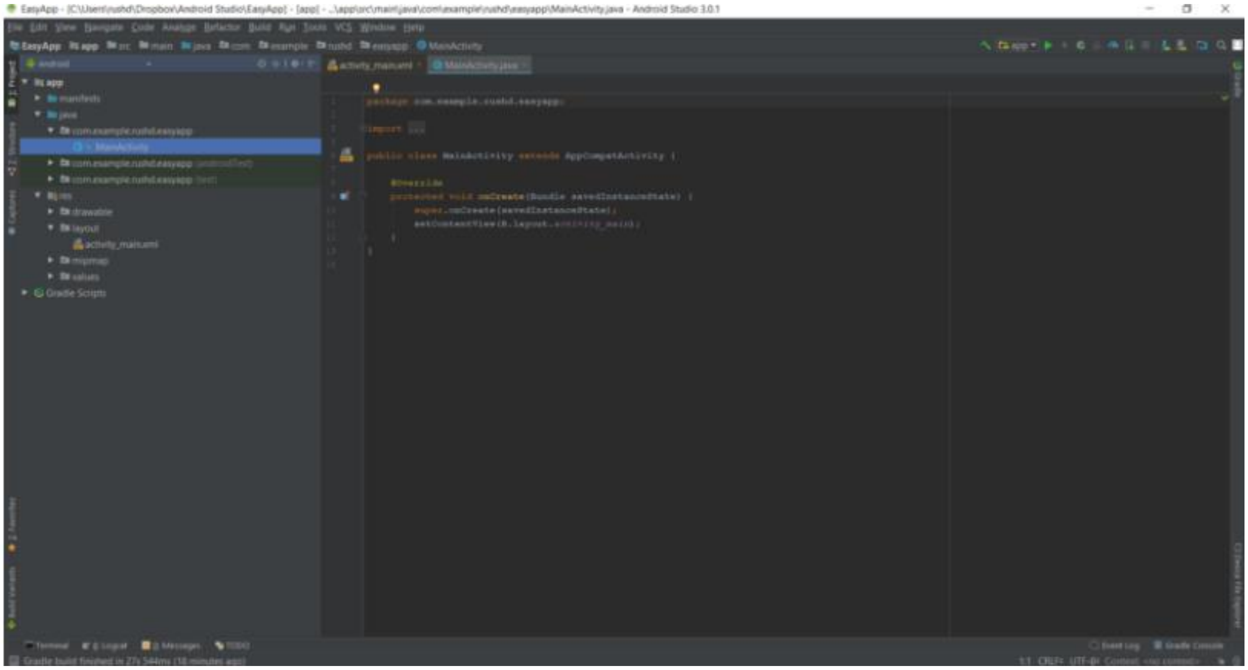
Eng. Adi.f.fayyad

XML in Android

XML stands for Extensible Markup Language. XML is a markup language much like HTML used to describe data. XML tags are not predefined in XML. We must define our own Tags. Xml as itself is well readable both by human and machine. Also, it is scalable and simple to develop. In Android we use xml for designing our layouts because xml is lightweight language so it doesn't make our layout heavy.

```
<title>Mrs.</title>
<first-name>
Mary
</first-name>
```

When you create a new project in Android Studio, you will be greeted by a hierarchy of different files and folders, which can be a little daunting for complete beginners. It's a rather jarring introduction to XML, no doubt!


EasyApp - [C:\Users\rushd\Dropbox\Android Studio\EasyApp] - [app] - ...\app\src\main\java\com\example\rushd\easyapp\MainActivity.java - Android Studio 3.0.1
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
EasyApp app src main java com example rushd easyapp MainActivity
Android
app
manifests
java
com.example.rushd.easyapp
MainActivity
com.example.rushd.easyapp (androidTest)
com.example.rushd.easyapp (test)
res
drawable
layout
activity_main.xml
mipmap
values
Gradle Scripts
activity_main.xml
MainActivity.java
Terminal
6: Logcat
0: Messages
TODO
Event Log
Gradle Console
Gradle build finished in 27s 544ms (18 minutes ago)

```
package com.example.rushd.easyapp;

import ...

public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
}
```

You just need to concentrate on two files for now: **MainActivity.java** and **activity_main.xml**.

To make life just a little simpler, Android Studio normally opens both these files as soon as it boots up

You'll also notice that both these files have a little bit of code already in them. This is called "boilerplate code," which is code that almost every program need, and so which Android Studio will populate for you in order to save time.

One line in MainActivity.java reads:

setContentView(R.layout.acivivty_main)

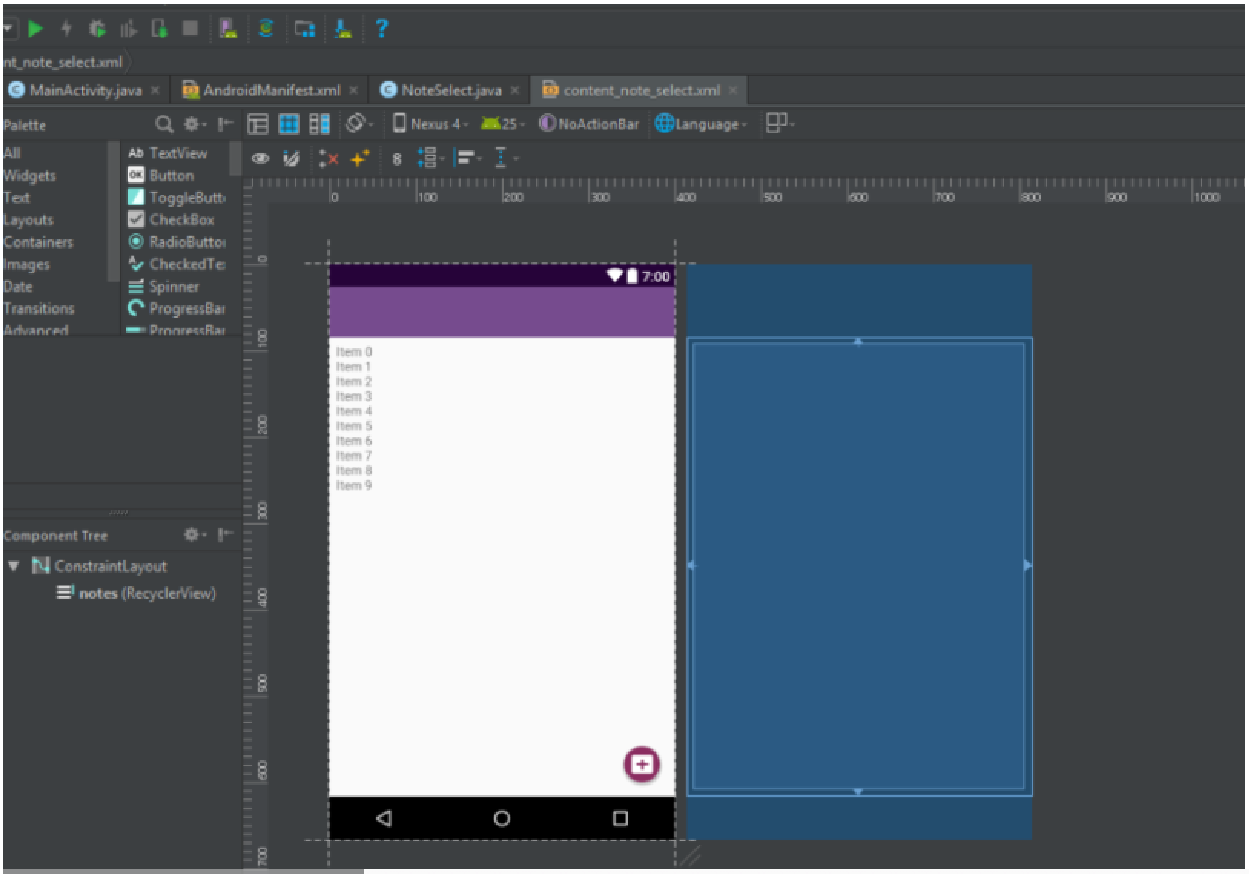
This means the activity this Java code controls will display the **activity_main.xml** file, and you may reference certain elements from that as you go.

You can assign any XML file to any Java file with this, and you can create as many of both as you like. However, by default **MainActivity.java** will always be the class (java file) Android loads first when running your programs.

Using XML in your Android app

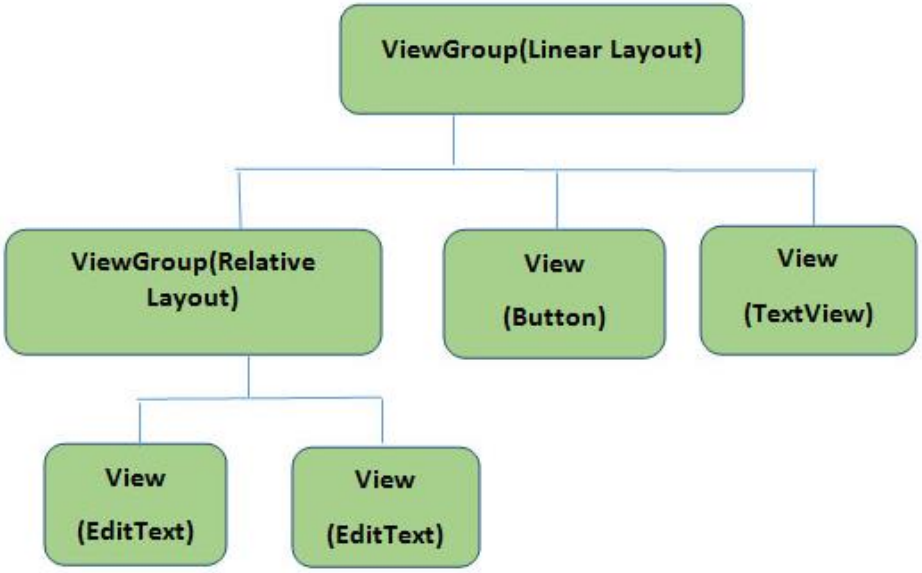
To recap, XML describes the views in your activities, and Java tells them how to behave. To make changes to the layout of your app then, you have two main options.The first is to use the Design view. Open up the **activity_main.xml** file in Android Studio and get your first introduction to XML. You'll notice there are two tabs at the bottom of that window: Design and Text. The Text view will show you the actual XML code, but the Design view will let you manually edit the layout by dragging and dropping elements into the render of your activity.

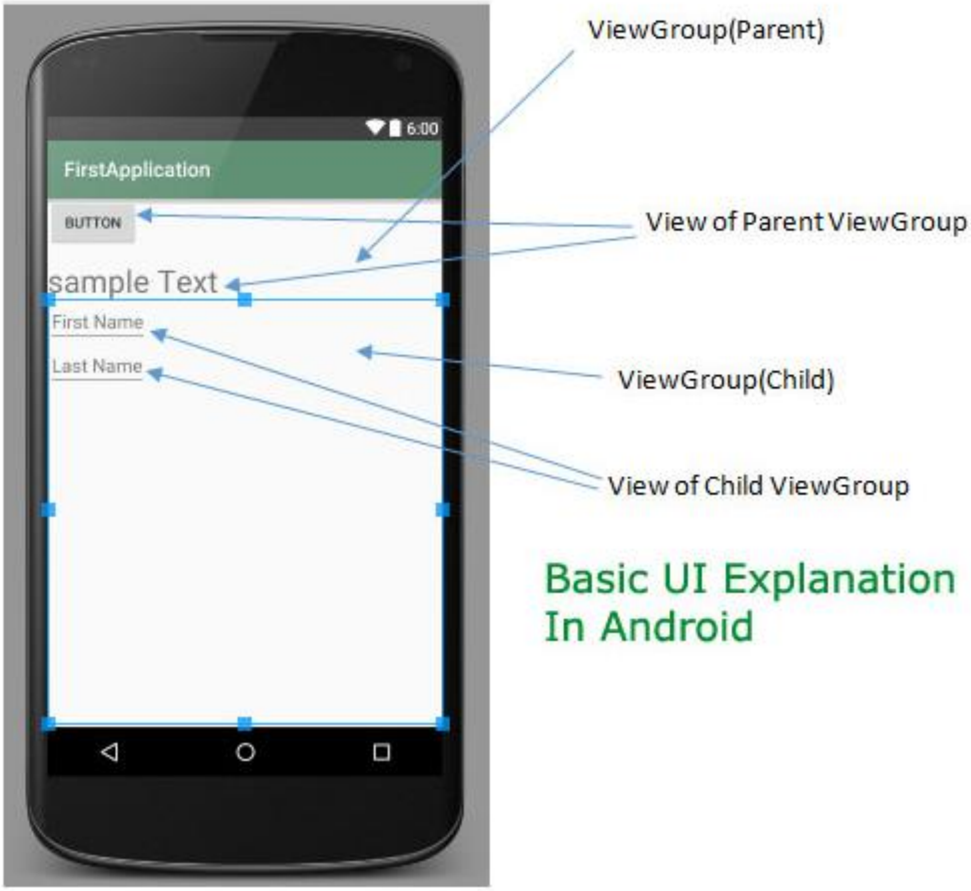
XML files can also help store strings. Using the Design view is easier for beginners, though it can lead to complications. For one, you will run into the limitations of XML early on when the designer refuses to let you drop items into specific places. Without the knowledge of why, this can make designing your app an exercise in frustration!



Managing views and viewgroups

As you may already have guessed, a ViewGroup is a group of views. These are also referred to as layouts, and serve as hierarchical arrangements of views. A linear layout for example places all its children in a linear vertical or horizontal arrangement. Meanwhile, a ConstraintLayout will allow you to define the positions of views by their relationship with other views in the layout and the boundaries of the activity. This way, the views can maintain their relative positions, even as the dimensions of the device vary.


ViewGroup(Linear Layout)
ViewGroup(Relative Layout)
View (Button)
View (TextView)
View (EditText)
View (EditText)


6:00
FirstApplication
BUTTON
sample Text
First Name
Last Name
ViewGroup(Parent)
View of Parent ViewGroup
ViewGroup(Child)
View of Child ViewGroup
Basic UI Explanation
In Android

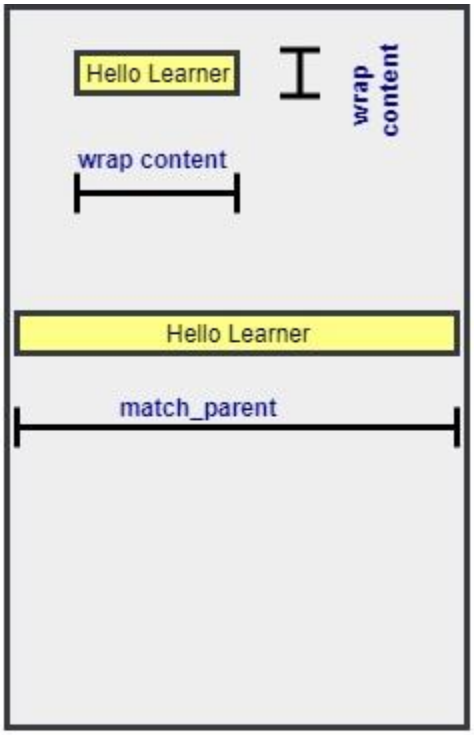
View is the basic building block of UI(User Interface) in android. View refers to the android.view.View class, which is the **super class** for all the GUI components like TextView, ImageView, Button etc.

View can be considered as a rectangle on the screen that shows some type of content. It can be an image, a piece of text, a button or anything that an android application can display. The rectangle here is actually invisible, but every view occupies a rectangle shape.

The question that might be bothering you would be , what can be the size of this rectangle?

The answer is either we can set it manually, by specifying the exact size(with proper units) or by using some predefined values. These predefined values are **match_parent** and **wrap_content.**

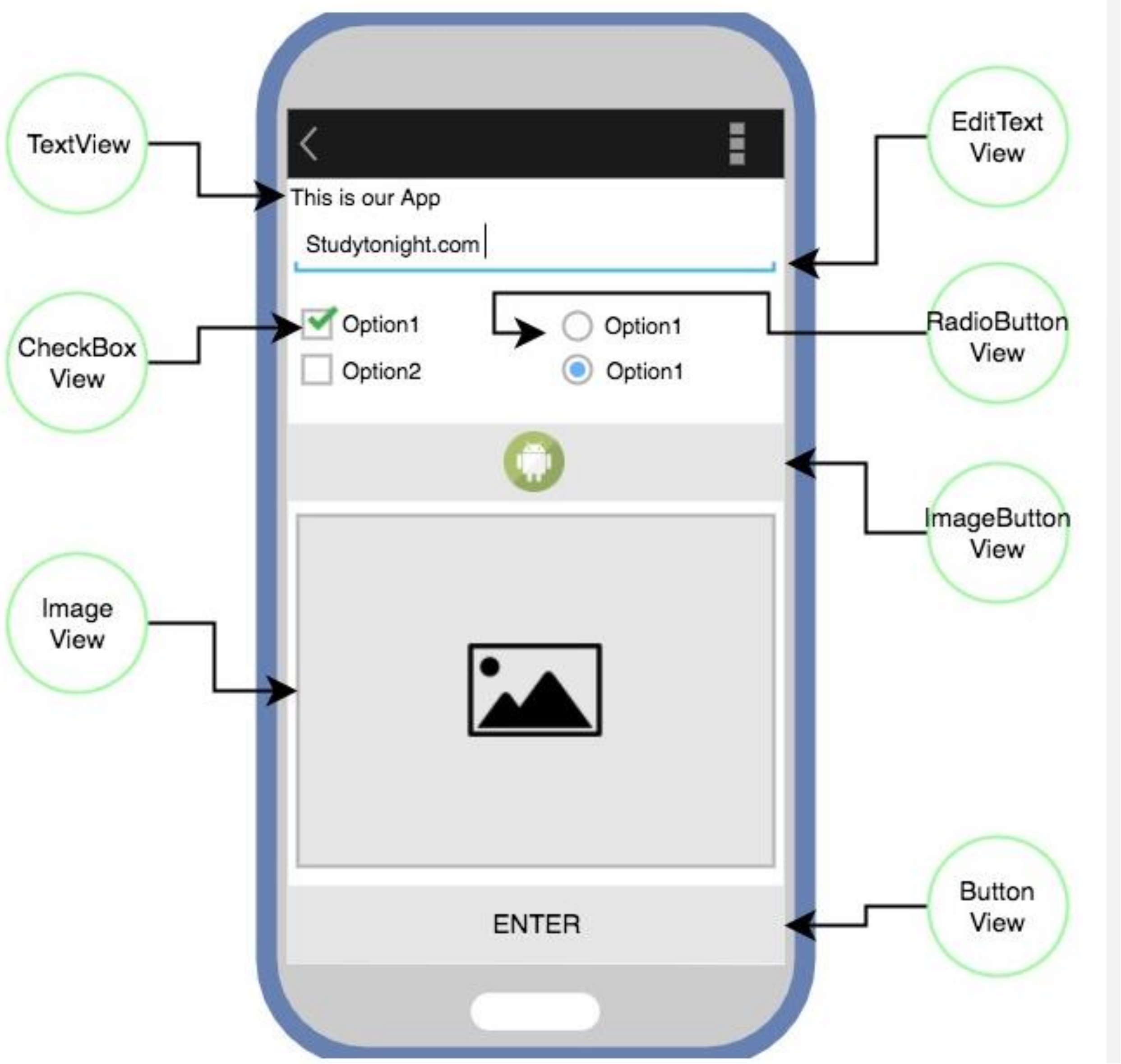
match_parent means it will occupy the complete space available on the display of the device. Whereas, **wrap_content** means it will occupy only that much space as required for its content to display.



Most commonly used Android View classes

Here we have some of the most commonly used android View classes:

- TextView
- EditText
- Button
- ImageView
- ImageButton
- CheckBox
- RadioButton
- ListView
- GridView
- DatePicker
- Spinner, etc.

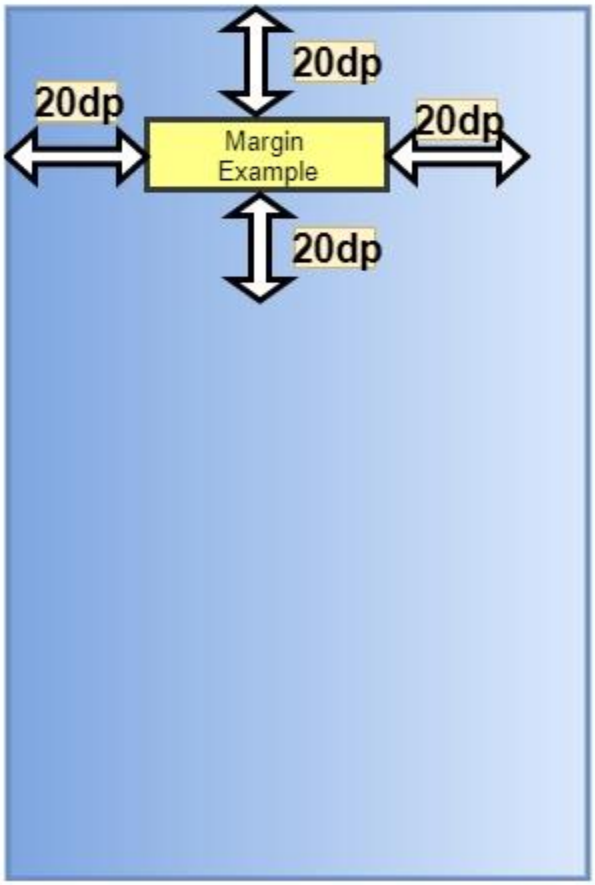


Margin, padding, gravity, layout-gravity, background

Margin

Margin is a way for a view to enforce some distance from others views. By specifying margin for a view, we say that keep this much distance from this view. Android has 5 kinds of margins.

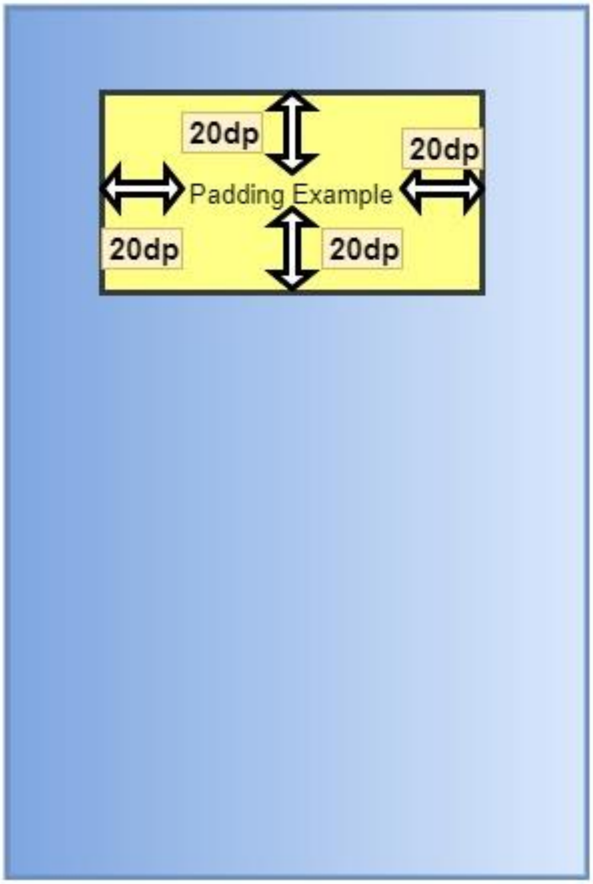
1. **margin** – keep distance on all the four sides
2. **marginLeft** – keep distance from left side of the view
3. **marginRight** – keep distance from right side of the view
4. **marginTop** – keep distance from top of the view
5. **marginBottom** – keep distance from bottom of the view

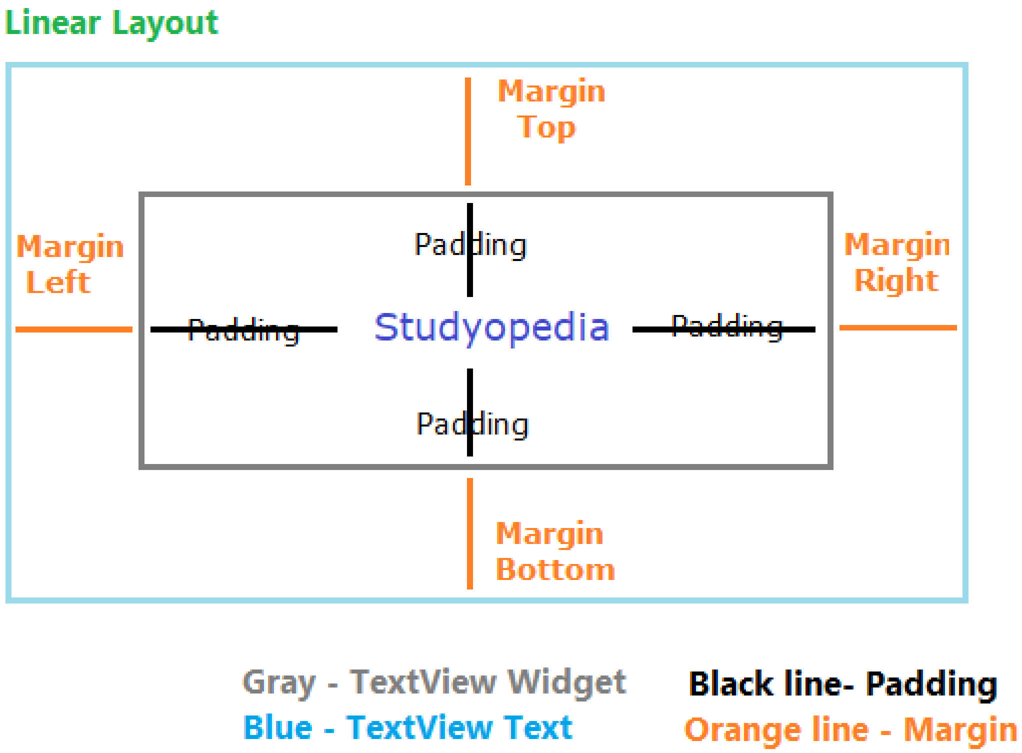


Padding

Padding is a way to push the contents away from view's inner boundary. When we specify the padding of a view, we say the content to keep this much distance from your inner boundary(left, right, top or bottom). Like margin, padding is also of 5 types.

1. **padding** – keep distance from all the inner boundaries
2. **paddingLeft** – keep distance from the left inner boundary
3. **paddingRight** – keep distance from the right inner boundary
4. **paddingTop** – keep distance from the top inner boundary
5. **paddingBottom** – keep distance from the bottom inner boundary




Linear Layout
Margin Top
Margin Left
Padding
Margin Right
Padding
Studyopedia
Padding
Padding
Margin Bottom
Gray - TextView Widget
Blue - TextView Text
Black line- Padding
Orange line - Margin

Thanks, 😊