**ASSIGNMENT**

1. On LinearLayout and RelativeLayout

**LinearLayout**

* A layout that organizes its children into a single horizontal or vertical row. It creates a scrollbar if the length of the window exceeds the length of the screen.
* It can align views one by one (vertically/ horizontally).
* To create a linear layout in which each child uses the same amount of space on the screen, set the [android:layout\_height](https://developer.android.com/reference/android/view/ViewGroup.LayoutParams.html#attr_android:layout_height) of each view to "0dp" (for a vertical layout) or the [android:layout\_width](https://developer.android.com/reference/android/view/ViewGroup.LayoutParams.html#attr_android:layout_width) of each view to "0dp" (for a horizontal layout). Then set the [android: layout\_weight](https://developer.android.com/reference/android/widget/LinearLayout.LayoutParams.html#attr_android:layout_weight) of each view to "1".

**RelativeLayout**

* [RelativeLayout](https://developer.android.com/reference/android/widget/RelativeLayout.html) is a view group that displays child views in relative positions. The position of each view can be specified as relative to sibling elements (such as to the left-of or below another view) or in positions relative to the parent [RelativeLayout](https://developer.android.com/reference/android/widget/RelativeLayout.html) area (such as aligned to the bottom, left or center)..
* Enables you to specify the location of child objects relative to each other (child A to the left of child B) or to the parent (aligned to the top of the parent).

Some of the many layout properties available to views in a [RelativeLayout](https://developer.android.com/reference/android/widget/RelativeLayout.html) include:

[android:layout\_alignParentTop](https://developer.android.com/reference/android/widget/RelativeLayout.LayoutParams.html#attr_android:layout_alignParentTop)

If "true", makes the top edge of this view match the top edge of the parent.

[android:layout\_centerVertical](https://developer.android.com/reference/android/widget/RelativeLayout.LayoutParams.html#attr_android:layout_centerVertical)

If "true", centers this child vertically within its parent.

[android:layout\_below](https://developer.android.com/reference/android/widget/RelativeLayout.LayoutParams.html#attr_android:layout_below)

Positions the top edge of this view below the view specified with a resource ID.

[android:layout\_toRightOf](https://developer.android.com/reference/android/widget/RelativeLayout.LayoutParams.html#attr_android:layout_toRightOf)

Positions the left edge of this view to the right of the view specified with a resource ID.

1. Why we use DP or DIP?

Ans:

* The first pitfall you must avoid is using pixels to define distances or sizes. Defining dimensions with pixels is a problem because different screens have different pixel densities, so the same number of pixels may correspond to different physical sizes on different devices.
* To preserve the visible size of your UI on screens with different densities, you must design your UI using density-independent pixels (dp) as your unit of measurement.
* One dp is a virtual pixel unit that's roughly equal to one pixel on a medium-density screen (160dpi; the "baseline" density). Android translates this value to the appropriate number of real pixels for each other density.
* If you were to define a view to be "100px" wide, it will appear much larger on the device on the left. So you must instead use "100dp" to ensure it appears the same size on both screens.
* We will need to express dimensions in dp and then convert them to pixels. The conversion of dp units to screen pixels is simple:

px = dp \* (dpi / 160)