


appears underneath the element.

- The pack button  in the toolbar provides options for packing or expanding selected UI elements. You can use it to equally arrange the `Button` elements horizontally across the layout.

Using `LinearLayout` :

- [LinearLayout](#) is a [ViewGroup](#) that arranges its collection of views in a horizontal or vertical row.
- A `LinearLayout` is required to have the `layout_width`, `layout_height`, and `orientation` attributes.
- `match_parent` for `layout_width` or `layout_height` : Expands the `View` to fill its parent by width or height. When the `LinearLayout` is the root `View`, it expands to the size of the screen (the parent `View`).
- `Wrap_content` for `layout_width` or `layout_height` : Shrinks the dimensions so the `View` is just big enough to enclose its content. If there is no content, the `View` becomes invisible.
- Fixed number of dp ([density-independent pixels](#)) for `layout_width` or `layout_height` : Specify a fixed size, adjusted for the screen density of the device. For example, `16dp` means 16 density-independent pixels.
- The `orientation` for a `LinearLayout` can be `horizontal` to arrange elements from left to right, or `vertical` to arrange elements from top to bottom.
- Specifying `gravity` and `weight` attributes gives you additional control over arranging views and content in a `LinearLayout`.
- The `android:gravity` attribute specifies the alignment of the content of a `View` within the `View` itself.
- The `android:layout_weight` attribute indicates how much of the extra space in the `LinearLayout` will be allocated to the `View`. If only one `View` has this attribute, it gets all the extra screen space. For multiple `View` elements, the space is prorated. For example, if two `Button` elements each have a weight of 1 and a `TextView` 2, totaling 4, the `Button` elements get  $\frac{1}{4}$  of the space each, and the `TextView` half.

Using `RelativeLayout` :

- A [RelativeLayout](#) is a `ViewGroup` in which each view is positioned and aligned relative to other views within the group.
- Use `android:layout_alignParentTop` to align the `View` to the top of the parent.
- Use `android:layout_alignParentLeft` to align the `View` to the left side of the parent.
- Use `android:layout_alignParentStart` to make the start edge of the `View` match the start edge of the parent. This attribute is useful if you want your app to work on devices that use different language or locale preferences. The *start* is the left edge of the screen if the preference is left-to-right, or it is the right edge of the screen if the preference is right-to-left.