Cheatography

Kivy Interactive Applications and Games in Python Cheat Sheet

by Ibrahim Omar via cheatography.com/58047/cs/15365/

Layouts

proportional coordinates	("percentages" of the total size of the window) rather than fixed coordinates (exact pixels).[10]
size_hint	size_hint: .4, .3
pos_hint: {'x': 0, 'top': 1}	pos_hint: {'right': 1, 'y': 0}
pos_hint: {'center_x':.5, 'center_y':.5 }	would align a widget in the middle no matter the size of the window.
pos: root.x, root.top - self.height	we should have used: x: 0 top: root.height

The properties x, center_x, right, y, center_y, and top always specify fixed coordinates (pixels), and not proportional ones. If we want to use proportional coordinates, we have to be inside a Layout (or an App) and use the pos_hint property.

Drawing basic shapes 2 Chapter 2

Ellipse	works very similar to Rectangle , but it has three new properties: angle_start, angle_end, and segments.

http://kivy.org/docs/api-kivy.graphics.vertex_instructions.html

layouts 2

Property	Value	For layouts	For widgets
size_hint	A pair w, h: w, and h express a proportion (from 0 to 1 or None).	Yes	No

layouts 2 (cont)

size_ hint_x size_ hint_y	A proportion from 0 to 1 or None, indicating width (size_ hint_x) or height (size_ hint_y).	Yes	No
pos_hint	Dictionary with one x-axis key (x, center_x, or right) and one y-axis key (y, center_y, or top). The values are proportions from 0 to 1.	Yes	No
size	A pair w, h: w and h indicating fixed width and height in pixels.	Yes, but set size_ hint: (None, None)	Yes
width	Fixed number of pixels.	Yes, but set size_ hint_x: None	Yes
height	Fixed number of pixels.	Yes, but set size_ hint_y: None	Yes
pos	A pair x, y indicating a fixed coordinate (x, y) in pixels.	Yes, but don't use pos_hint	Yes
x, right or center_x	Fixed number of pixels.	Yes, but don't use x, right or center_x in pos_hint	Yes

layouts 2 (cont)

y, top or center_y	Fixed number of pixels.	Yes, but don't use y, top or center_y in pos_hint	Yes

Graphics – the Canvas Chapter 2

Kivy Canvas	is a set of drawing instructions that define the graphical representation of Widget .
coordinate space	the place in which we draw.All the Kivy widgets share the same coordinate space, and a Canvas instance, the instructions to draw on it. A coordinate space is not restricted to the size of the window or the application screen, which means that we can draw outside of the visible area.

Understanding the canvas Chapter 2

• The coordinate space refers to the place in which we draw, which is not restricted to the windows size
• A Canvas object is a set of instructions to draw in the coordinate space, not the place we draw in
• All Widget objects contain their own Canvas (canvases, which we will see later) but all of them share the same coordinate space, the one in the App object.
All the graphics instructions added to different Canvas objects, which at the same time belong to different Widget objects, affect the same coordinate space. It is our task to

By **Ibrahim Omar**
cheatography.com/ibrahim-omar/

Not published yet.
Last updated 3rd April, 2018.

Understanding the canvas Chapter 2 (cont)

make sure that the coordinate space is in its original state
after modifying it with the graphics instructions. A Widget is also a place marker (with its position and size),
but not necessarily a placeholder. The instructions of the
canvas of a widget are not restricted to the specific area of
the widget but to the whole coordinate space.

Drawing basic shapes Chapter 2

The vertex instructions inherit from the VertexInstruction base
class, and allow us to draw vector shapes in the coordinate space.
The context instructions (Color, Rotate, Translate, and Scale)
inherit from the ContextInstruction base class, and let us apply
transformations to the coordinate space context. By coordinate space
context, we mean the conditions in which the shapes (specified in the
vertex instructions) are drawn in the coordinate space.
Basically, vertex instructions are what we draw and context instructions affect
where and how we draw.
This means that we cannot use the size_hint or pos_hint properties as we did
with the widgets in Chapter 1, GUI Basics – Building an Interface. However, we can
use the properties of self to achieve similar results (Line 18 and 19).
Rectangle is a good starting point because it resembles the way we set properties
in widgets. We just have to set the pos and size properties.
The pos and size properties of the vertex instructions are different

Drawing basic shapes Chapter 2 (cont)

from the pos and size properties of Widget, since they belong to
the VertexInstruction base class. All the values to specify the
properties of the vertex instructions are given in fixed values.

float layout

We can also force a Layout to use fixed values, but there can be conflicts if we are
not careful with the properties. If we use any Layout ; pos_hint and size_hint take
priority. If we want to use fixed positioning properties (pos , x , center_x , right ,
y , center_y , top), we have to ensure that we are not using the pos_hint property.
Second, if we want to use the size , height , or width properties, then we need to
set a None value to the size_hint axis we want to use with absolute values.[13]

LAYOUT 3

Layout	Details
FloatLayout	Organizes the widgets with proportional coordinates by the size_hint and pos_hint properties. The values are numbers between 0 and 1, indicating a proportion to the window size.
RelativeLayout	Operates in the same way that FloatLayout does, but the positioning properties (pos, x, center_x, right, y, center_y, top) are relative to the Layout size and not the window size.

LAYOUT 3 (cont)

GridLayout	Organizes widgets in a grid. You have to specify at least one of two properties – cols (for columns) or rows (for rows).
BoxLayout	Organizes widgets in one row or one column depending on whether the value of the orientation property is horizontal or vertical.
StackLayout	Similar to BoxLayout, but it goes to the next row or column when it runs out of space. There is more flexibility to set the orientation. For example, rl-bt organizes the widgets in right-to-left, bottom-to-top order. Any combination of lr (left to right), rl (right to left), tb (top to bottom), and bt (bottom to top) is allowed.
ScatterLayout	Works in a similar manner to RelativeLayout but allows multitouch gesturing for rotating, scaling, and translating. It is slightly different in its implementation, so we will review it later on.
PageLayout	Stacks widgets on top of each other, creating a multipage effect that allows flipping of pages using side borders. Very often, we will use another layout to organize elements inside each of the pages, which are simply widgets.
Anchor layout	The AnchorLayout aligns its children to a border (top, bottom, left, right) or center.

Embedding layouts

Note that pos_hint always uses relative coordinates, no matter the
layout we are using. In other words, the previous example wouldn't
have worked if we were using pos_hint instead of pos.[17]

PageLayout – swiping pages

The PageLayout works in a different manner from other layouts. It is a dynamic
layout, in the sense that it allows flipping through pages using its borders. The idea
is that its components are stacked in front of each other, and we can just see the one
that is on top.

PageLayout – swiping pages 2

If we want to apply changes to all the child widgets that have
a common base class (such as Layout), we can introduce those
changes in the base class. Kivy will apply the changes to all the
classes that derive from it.[21]

Our project – Comic Creator Chapter 1

Kivy id	Kivy id which allows us to refer to other components inside the Kivy language.
ToggleButton	The difference with the normal Button is that it stays pressed until we click on it again.
group property	A ToggleButton instance can be associated with other ToggleButton instances, so just one of them is clicked on at a time. We can achieve this by assigning the same group property (line 250) to the ToggleButton instances that we want to react together.

Our project – Comic Creator Chapter 1 (cont)

markup	a nice feature for styling the text of the Label class. It works in a similar manner to XML-based languages.documentation for Label (http://kivy.org/docs/api-kivy.uix.label.html).