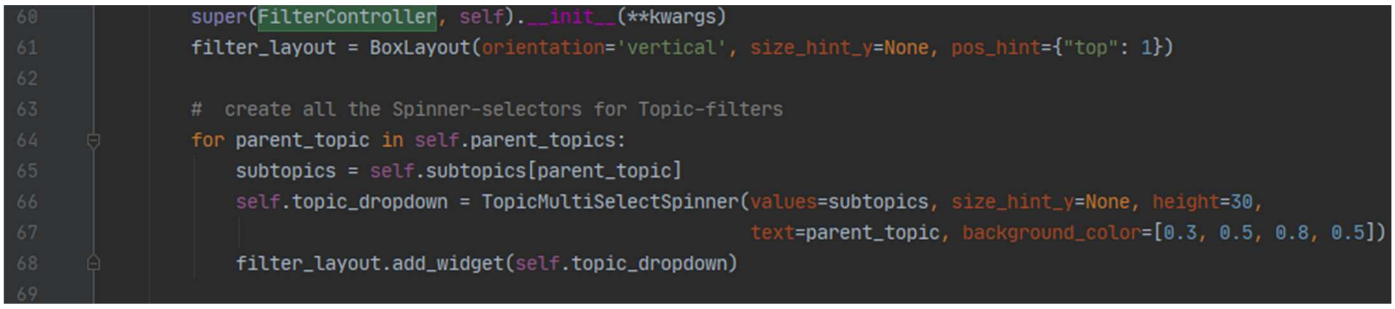
Adding Filters in FilterController/ Change filter behaviour

First you should think about if you need to create a number of widgets dynamically (e.g. a spinner that contains all topics of the database) or if you need a fixed number of buttons/spinners/… (e.g. “create pdf”-button). Generally it is easier and clearer to add widgets in the kv-file but in case of the FilterController we mostly needed widgets created dynamically so in this section I will explain that. If you only want to change the behaviour of the filtering go directly to step 3.

Step 1: First you need to add the widget that the user can click to activate the filter to the layout. In the code snipped below (fig 2) which is from ExerciseMgr.py in the class “FilterController” you can see how the spinners that contain all the subtopics are added to the layout. In line 66 a widget of the class TopicMultiSelectSpinner is created and then added to the layout in line 68 which will be done for every parent topic. Note that if you add a button to the layout you should do “self.button_count += 1”



```
60          super(FilterController, self).__init__(**kwargs)
61          filter_layout = BoxLayout(orientation='vertical', size_hint_y=None, pos_hint={"top": 1})
62
63          #  create all the Spinner-selectors for Topic-filters
64          for parent_topic in self.parent_topics:
65              subtopics = self.subtopics[parent_topic]
66              self.topic_dropdown = TopicMultiSelectSpinner(values=subtopics, size_hint_y=None, height=30,
67                                                            text=parent_topic, background_color=[0.3, 0.5, 0.8, 0.5])
68              filter_layout.add_widget(self.topic_dropdown)
69
```

Figure 1

Step 2: you will need to extend the apply_filter(self) method of the FilterController class. The method reads all the states of the different filter-widgets (mostly spinners) and bundles them in the variable “myfilter”. I found it to be easiest to read the information from the widgets the way it is done in apply_filter but note that there is a possibility to give widgets ids and find them by the id (main reason I did not use that often is that 1. the ids are not globally available and 2. kivy does not create them automatically). To work with widgets I can highly recommend using the debugger of pycharm to see which wigets are available in the current environment and to see parent-child relations.

Step 3:

After you changed myfilter you will need to change the function that processes the filter which is get_filter_matching_exercises(conn, filter) in sqlite_wrapper.py. Depending on what filter you added you will have to think of a way to filter the data from the database accordingly and return it. Sometimes it will be easier to adjust the sql-command and sometimes it will be easier to filter with python. Note that sqlite is fast but it does not support array operations.

Changing appearance of item in ExerciseLayout (middle view):

This is quite simple. Just look for “ <OneScrollItem>: “ in exercisemgr.kv. If you want to add/change/remove a button you will see a number of elements that start with “Button:” that you can remove/change/add. If you want to change the behaviour of a button you have to go to the class “OneScrollItem” in utils_gui.py and make changes to the method that is called by the button (e.g. if somewhere in the button declaration it says “on_press: root.on_previous_button()” you will have to change the method “on_previous_button(self)” in utils_gui.py). In general, all methods that are called by buttons will start with “on_”

Changes to the ExitLayout:

For the ExitLayout all buttons are defined in the kv-file so you have to go to exercisemgr.kv and search for “ <ExitLayout>:” where you will find all buttons. If you want to change the behaviour you have to edit the methods of the class “ExitLayout” in ExerciseMgr.py

Creating Custom Widgets

Since kivy does not provide many complex widgets it can be helpful to create its own custom widgets. A good example of that is the MultiSelectSpinner class in utils_gui.py. Basically, what it does is creating a button that if pressed will open a dropdown which is filled with toggle-buttons. Button, toggle-button and dropdown are all widgets provided by kivy and if you combine them you get a dropdown in which you can select multiple values at once. Normally you could only select one value in a dropdown provided by kivy. If you wanted to write a widget that is similar to the MultiSelectSpinner but needs to have a different behaviour you can write a class that inherits from MultiSelectSpinner and overwrites some methods in order to fit its purpose better like it is done with TopicMultiSelectSpinner and PopupMultiSelectSpinner.