GDC | **March 21-25, 2022**
San Francisco, CA

Immutable data for editor for EVE Echoes

#GDC22

THE
SCOPE

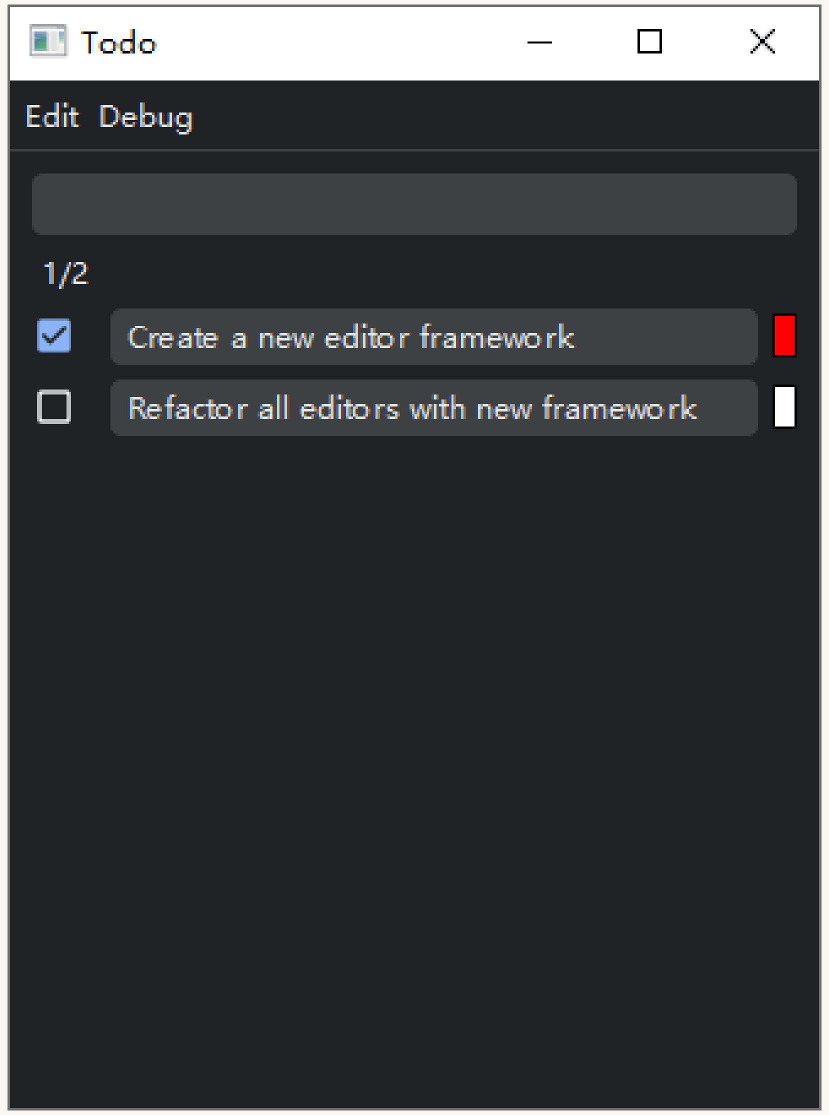
PREVIOUS WORK

Tools for 'Marvel's Spider-Man': Editing with Immutable Data
- Ron Pieket, GDC 2019

AGENDA

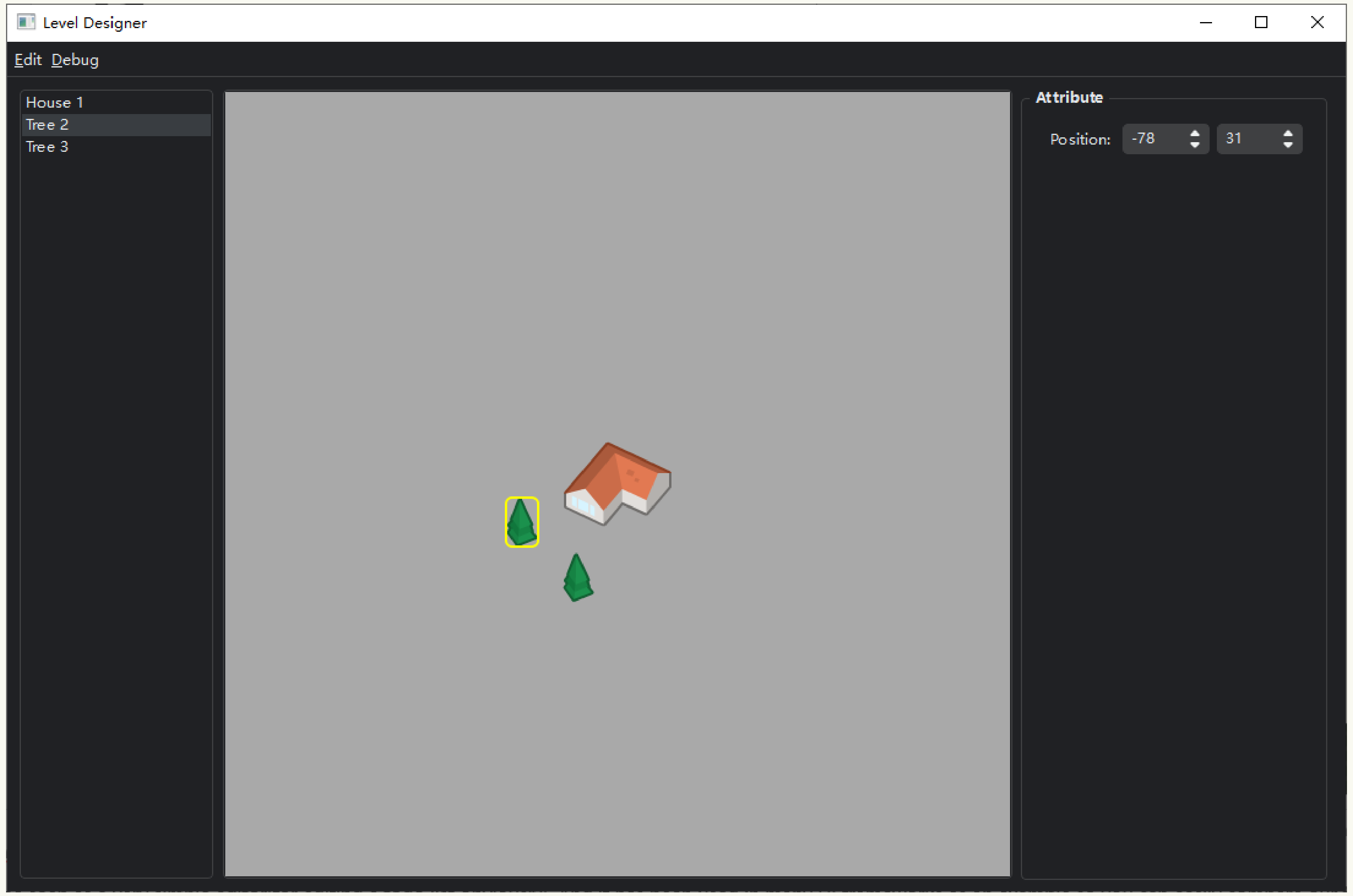
- Problems with event based editor
- How immutable data based editors work
- Automated testing
- Manage immutable data efficiently
- How is immutable data implemented

DEMO



1/2

- [x] Create a new editor framework 🟥
- [ ] Refactor all editors with new framework ⬜



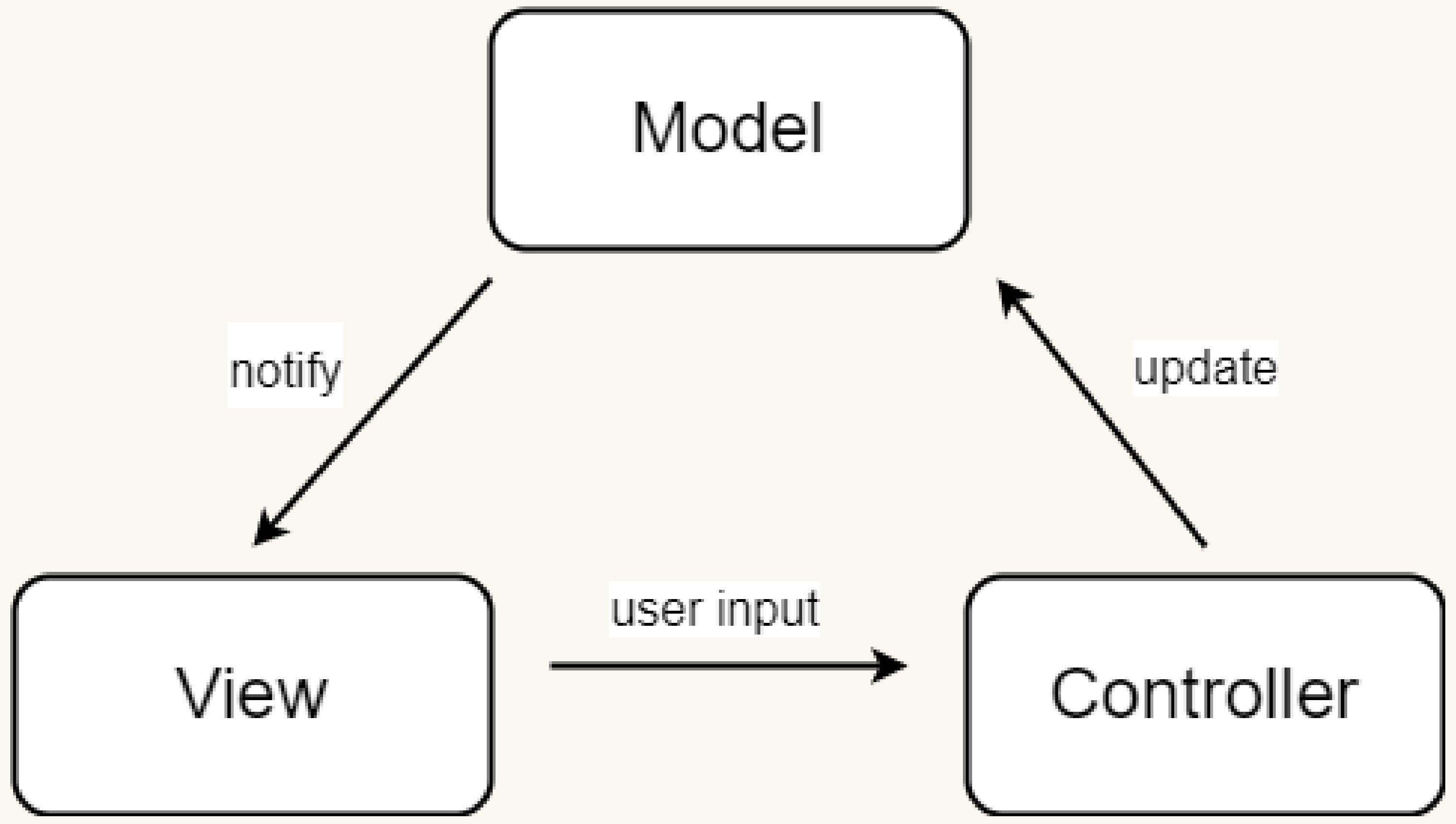
Position: -78 31

https://github.com/kkpattern/immu_editor_gdc

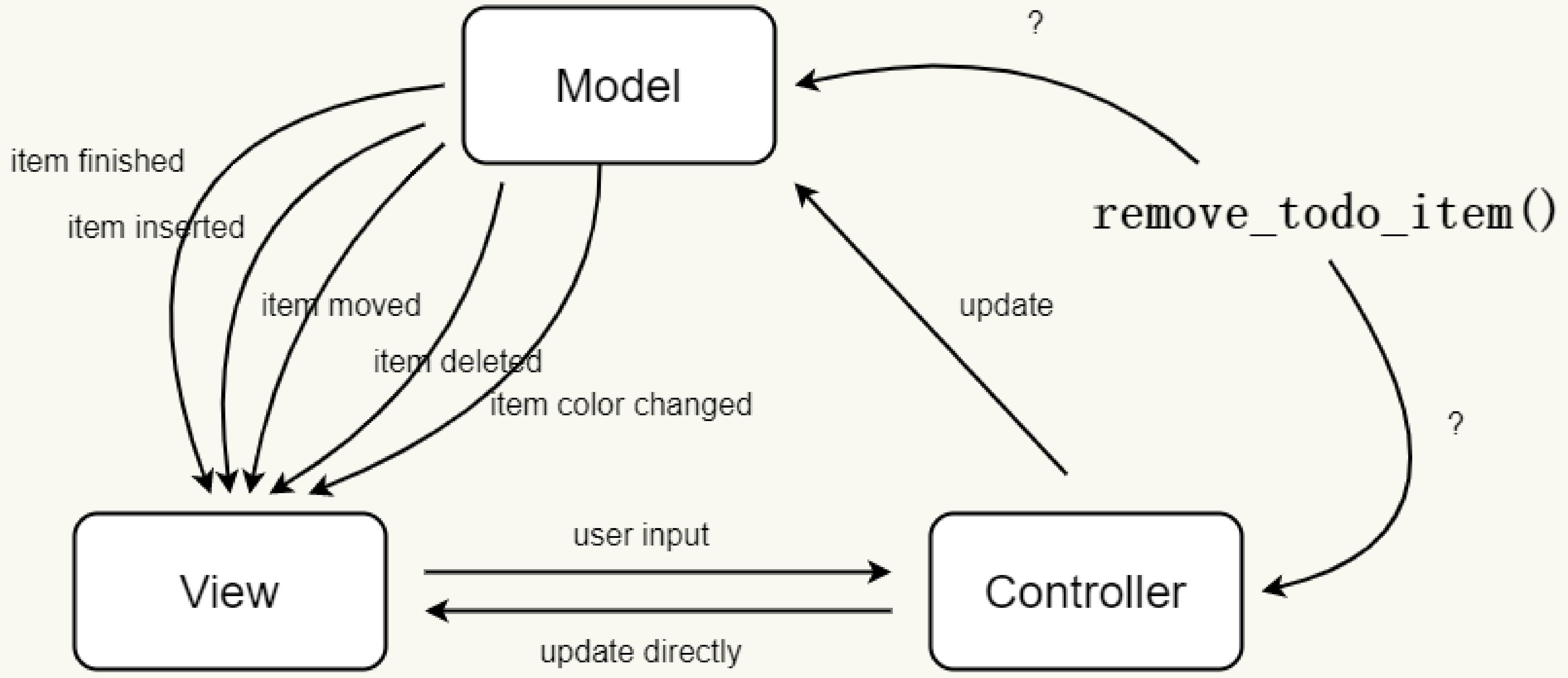
BACKGROUND

- Relatively small team
  - One dedicated tools developer
  - No dedicated tools QA
- New feature development always in a rush
- Unstable editors
- Editors no longer maintained

MVC



MVC


Model
?
item finished
item inserted
remove_todo_item()
item moved
update
item deleted
item color changed
?
user input
View
Controller
update directly

DIFF BASED UPDATE

- Hard to see the complete data
- Hard to detect and recover from out-of-sync

THE UNDO/REDO PROBLEM

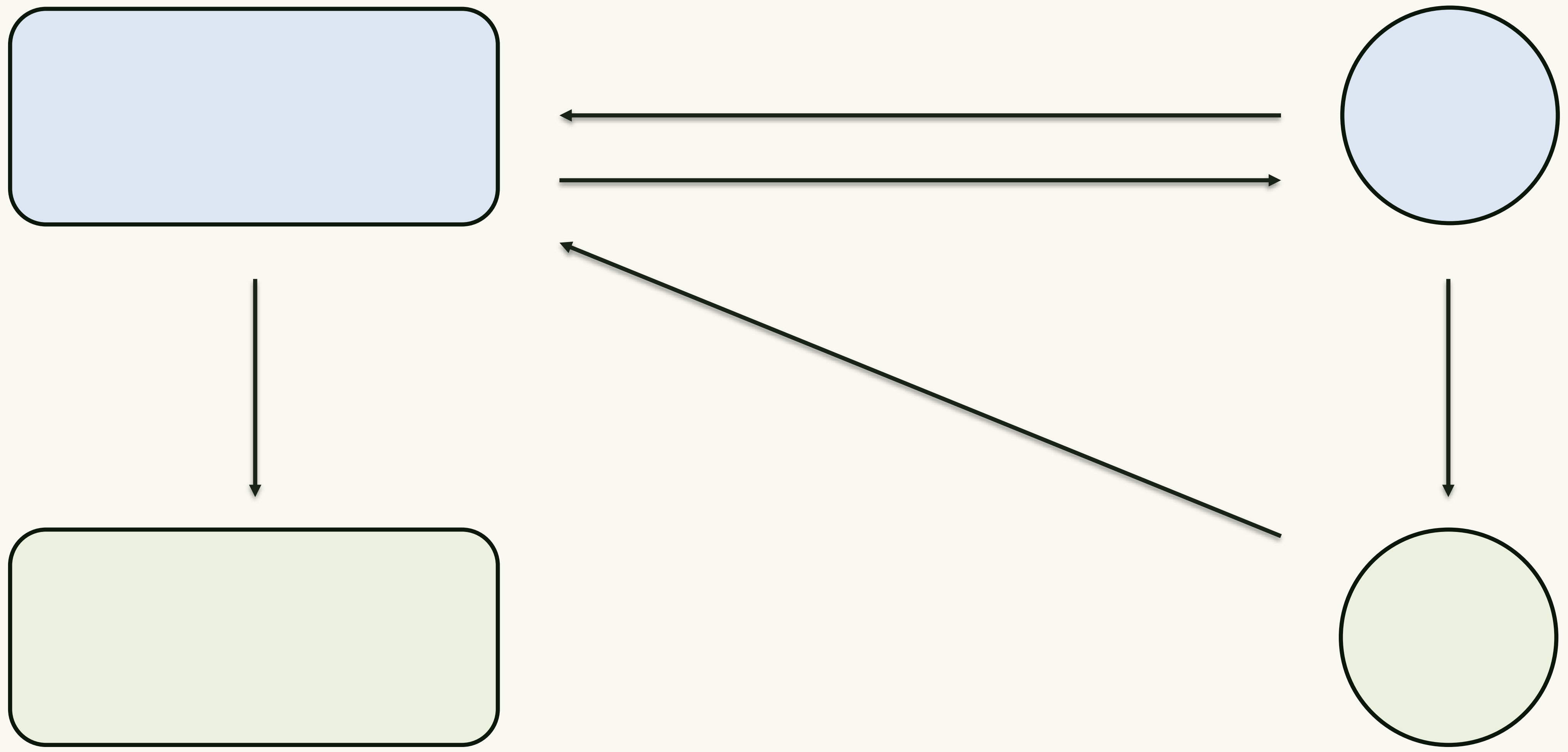
- Iteration is king
- Command pattern
- Undo/redo often non-trivial
- Developers tend not to provide new features

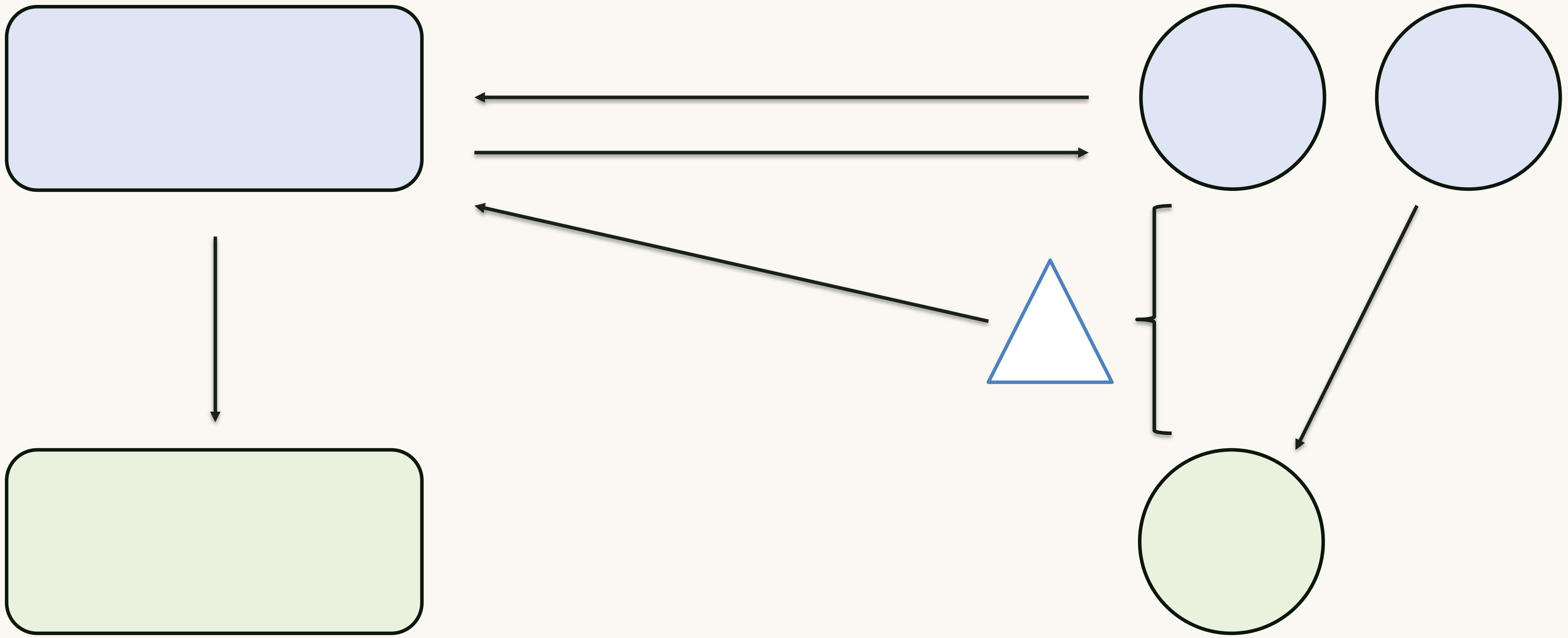
GOAL

- Increase editor stability
- Increase editor development efficiency
  - Provide a simple undo/redo solution
- Paradigm for all editors
- Fast (enough) performance

OTHER SOLUTIONS

- Declarative UI
- Immediate Mode UI





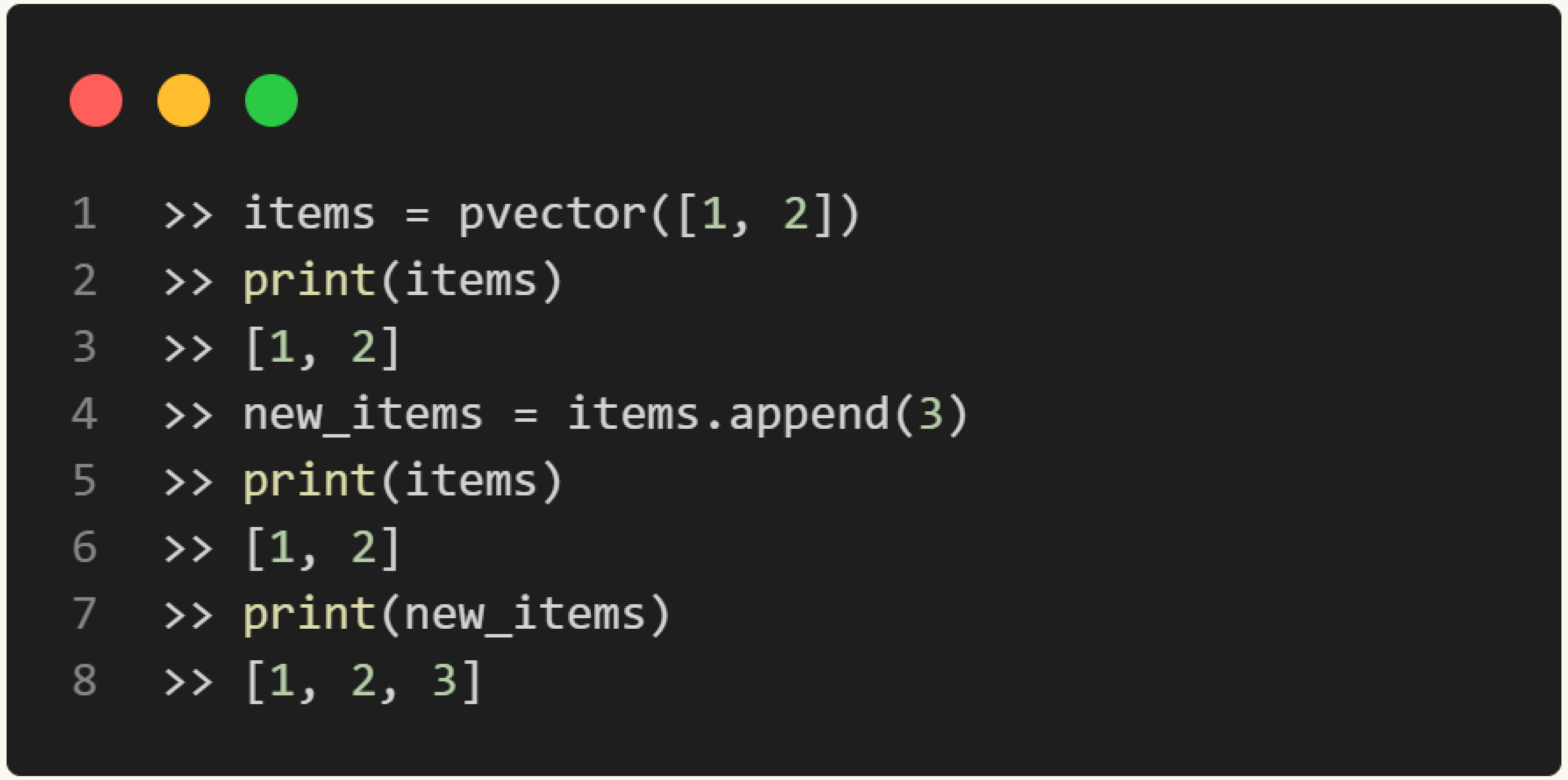
- Get the complete data
- Easier to detect and recover from out-of-sync
- Free undo/redo

- Copy can be expensive
- Copy can be hard
- Diff can be expensive
- Diff can be hard

IMMUTABLE DATA STRUCTURE

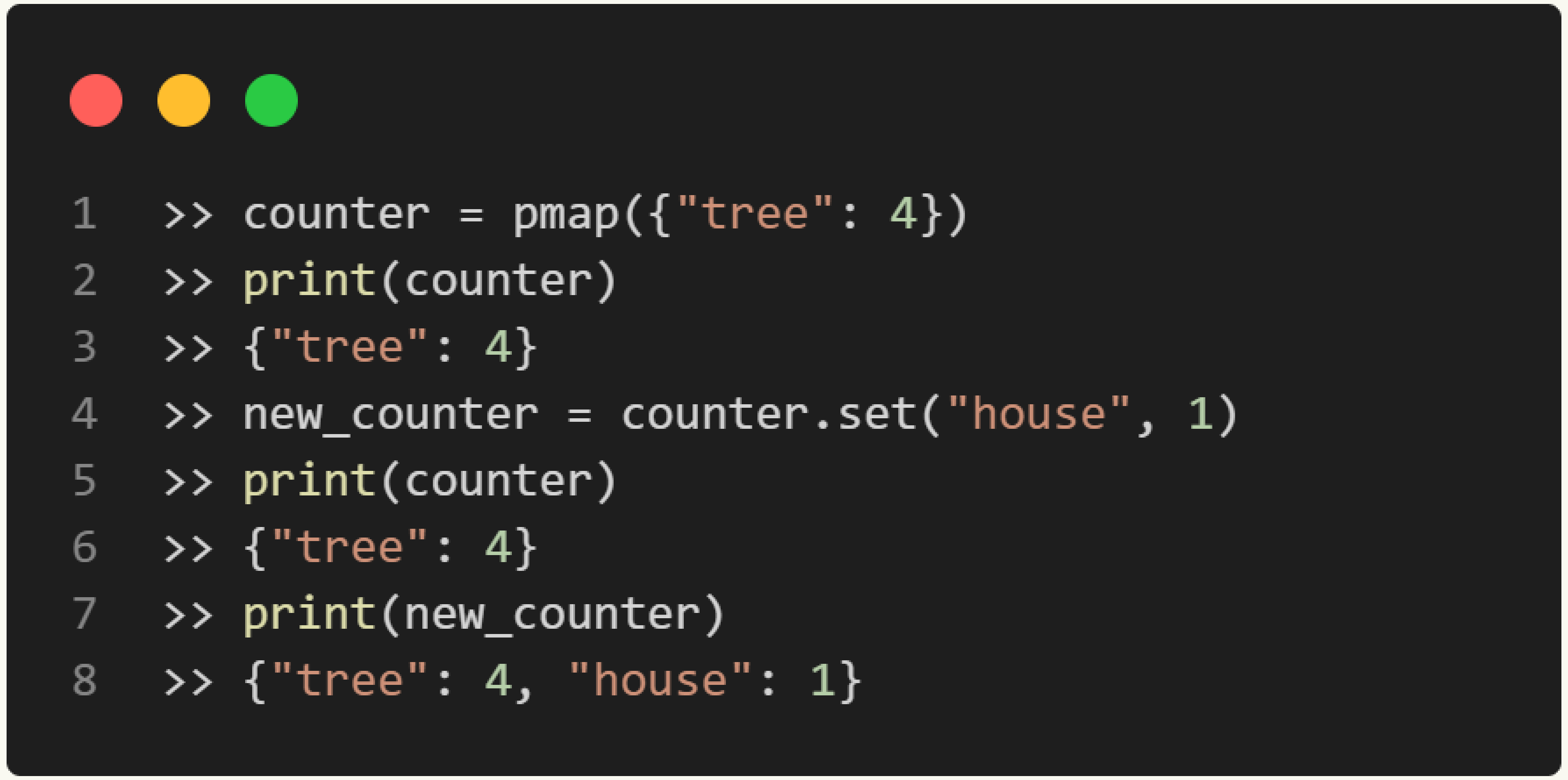
- Persistent data structure
- Cannot be changed
- Copy to modify

EXAMPLE: PYRSISTENT



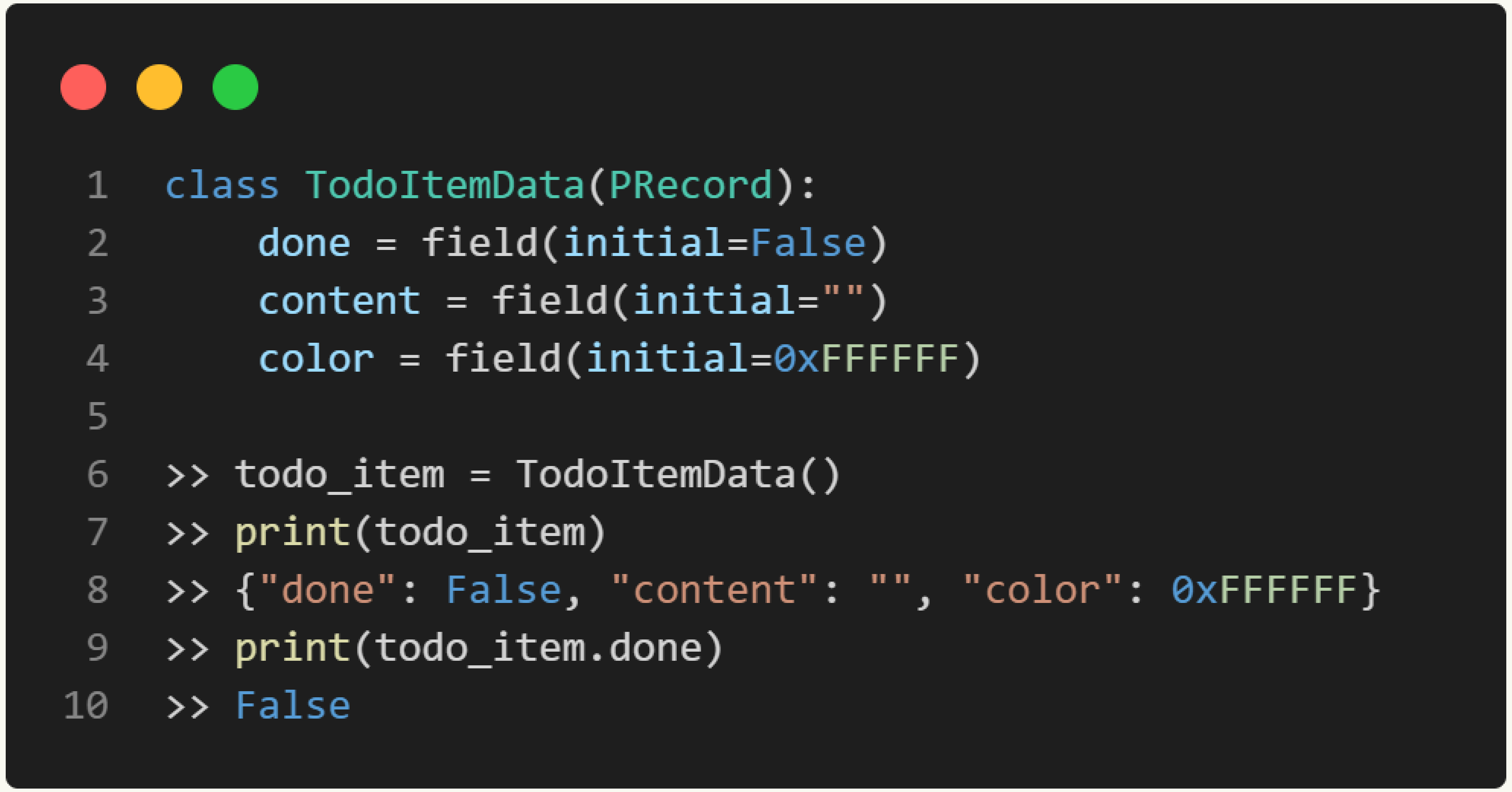
```
>> items = pvector([1, 2])
>> print(items)
>> [1, 2]
>> new_items = items.append(3)
>> print(items)
>> [1, 2]
>> print(new_items)
>> [1, 2, 3]
```

EXAMPLE: PYRSISTENT



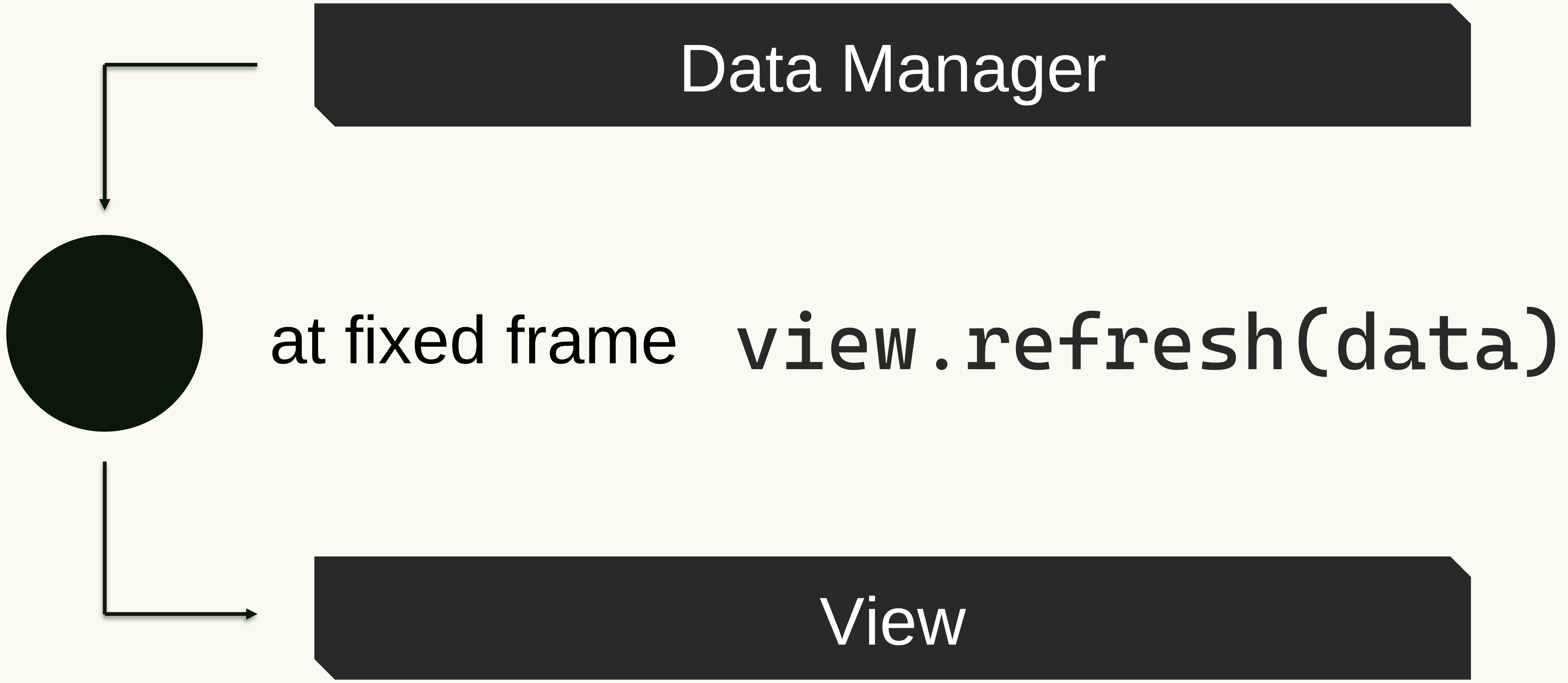
```
>> counter = pmap({"tree": 4})
>> print(counter)
>> {"tree": 4}
>> new_counter = counter.set("house", 1)
>> print(counter)
>> {"tree": 4}
>> print(new_counter)
>> {"tree": 4, "house": 1}
```

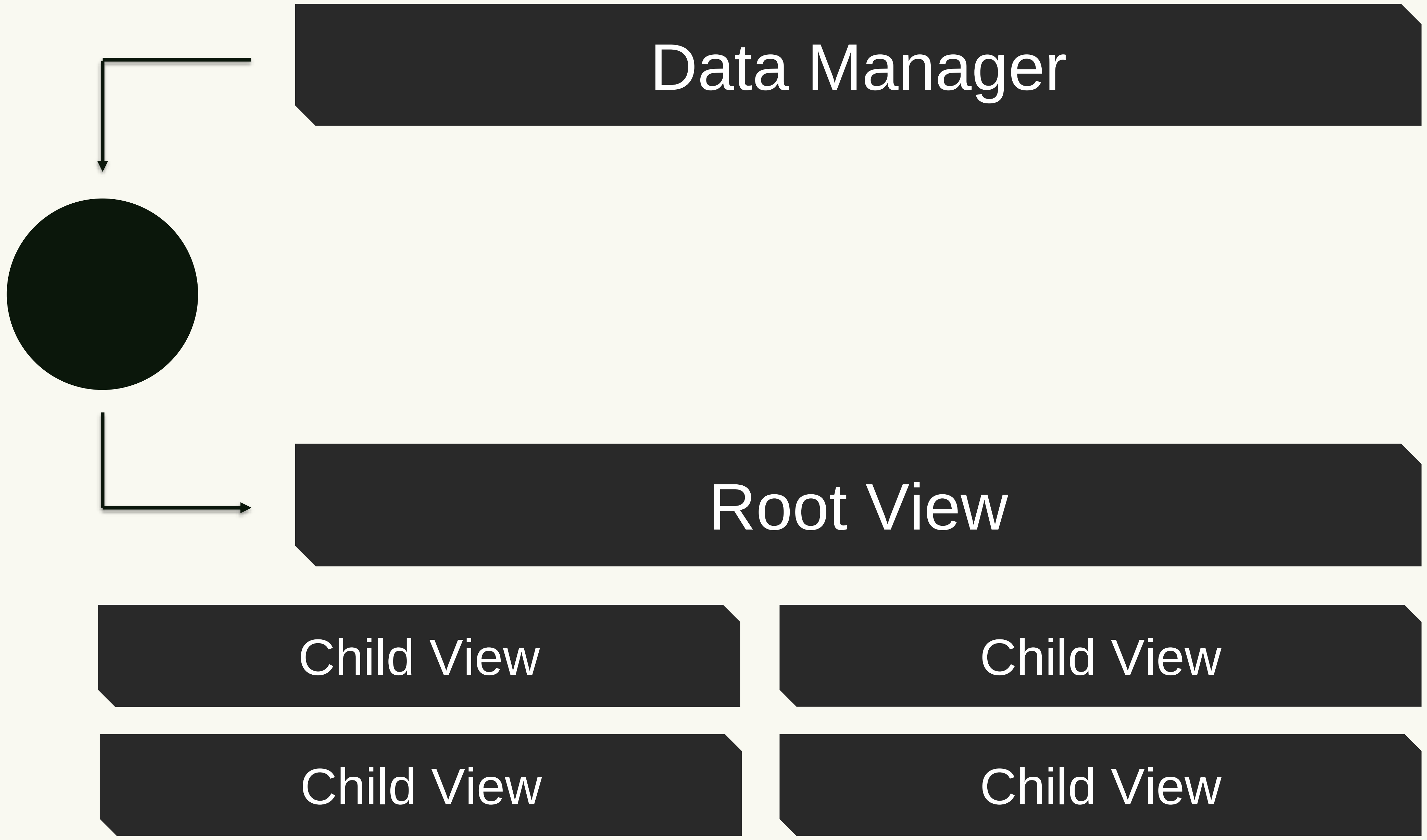
EXAMPLE: PYRSISTENT

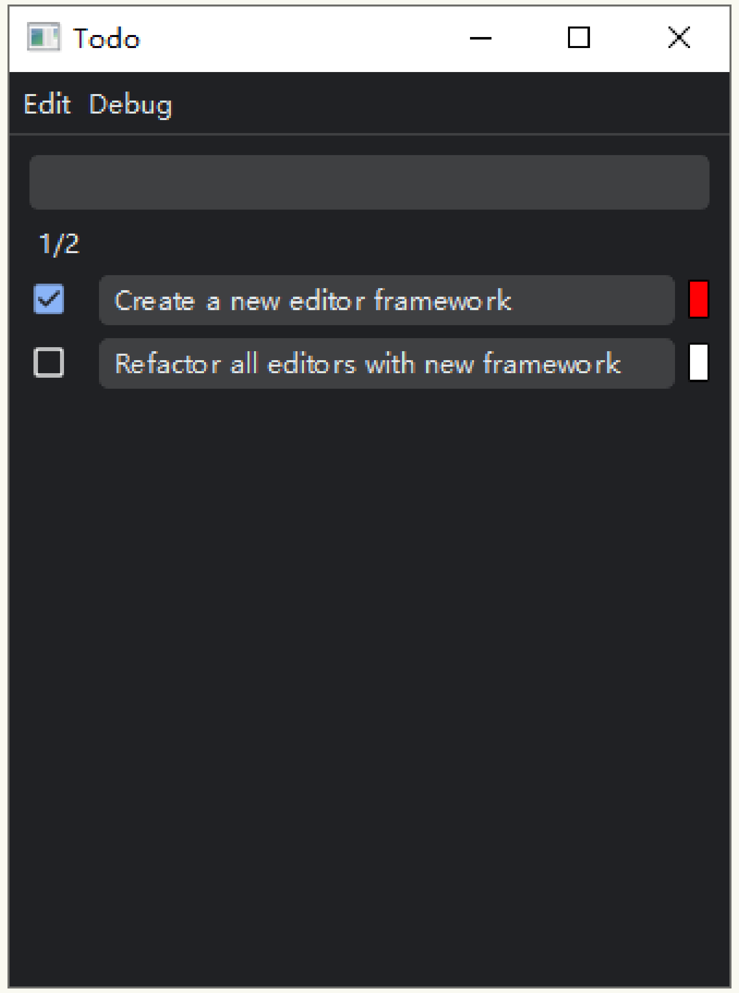


```python
class TodoItemData(PRecord):
    done = field(initial=False)
    content = field(initial="")
    color = field(initial=0xFFFFFF)

>> todo_item = TodoItemData()
>> print(todo_item)
>> {"done": False, "content": "", "color": 0xFFFFFF}
>> print(todo_item.done)
>> False
```

- Cheap copy
- Cheap diff
  - Replace comparing content with comparing memory location
  - Quickly to identify unchanged data


Data Manager
at fixed frame
view.refresh(data)
View


Data Manager
Root View
Child View
Child View
Child View
Child View

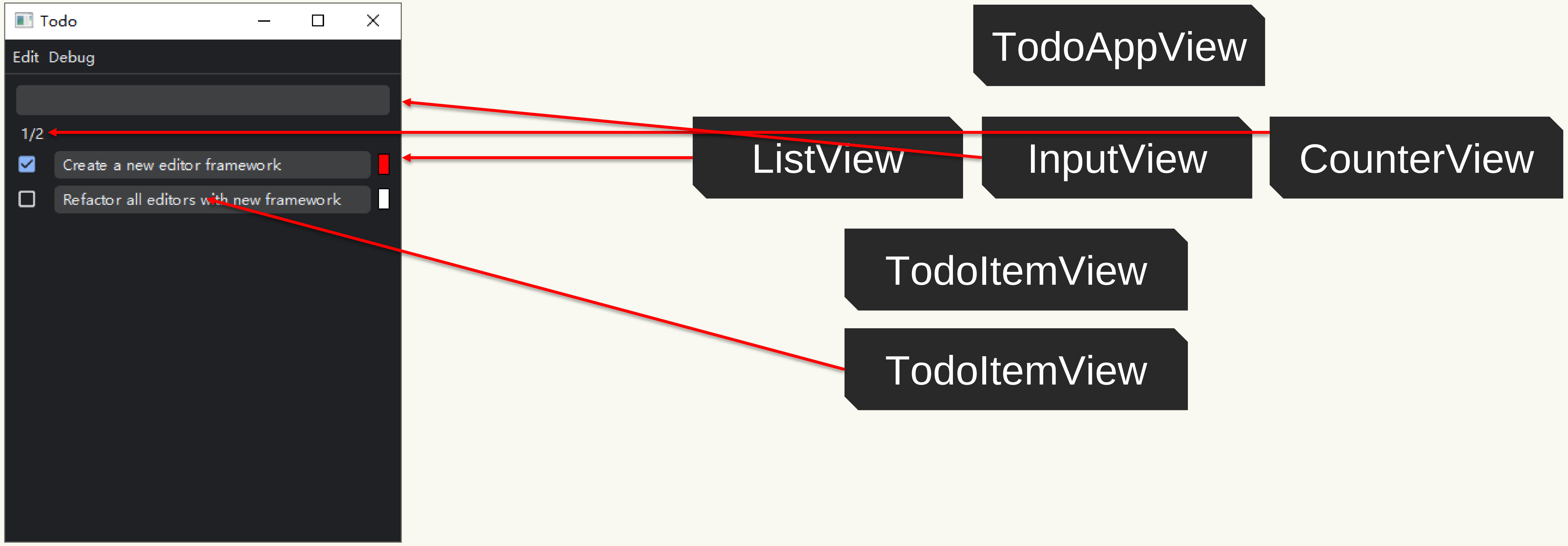


1/2

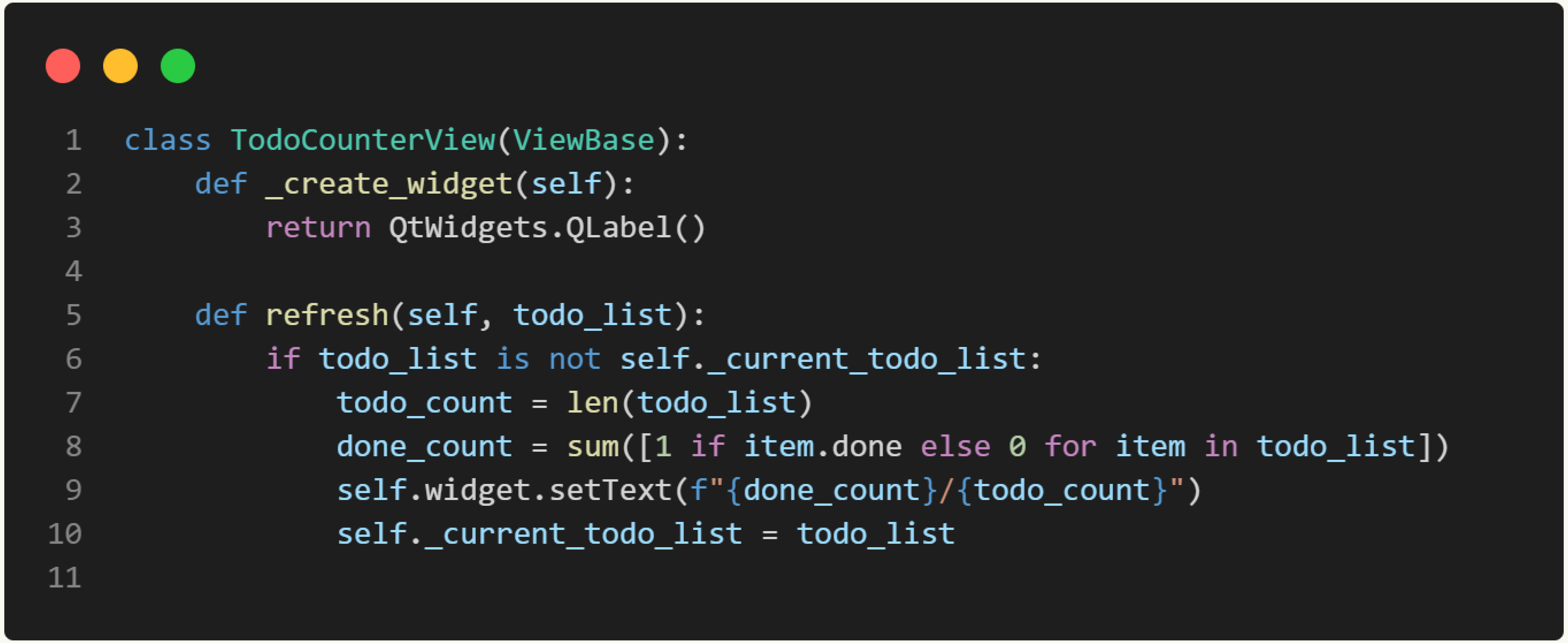
✓ Create a new editor framework

☐ Refactor all editors with new framework

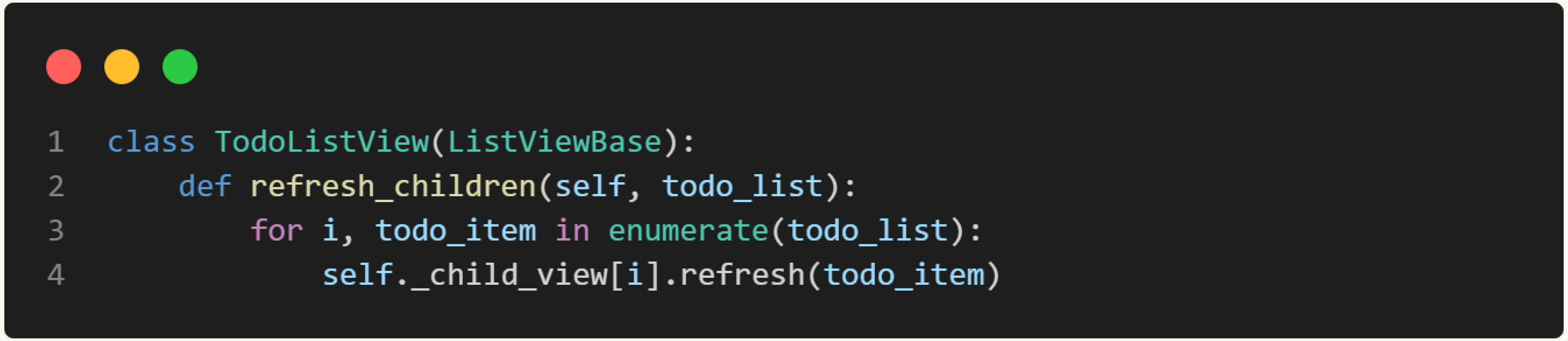
```python
class TodoItemData(RecordWithUUID):
    done = field()
    content = field()
    color = field()


class TodoAppData(PRecord):
    todo_list = field()   # a list of TodoItemData
```


Todo
Edit Debug
1/2
Create a new editor framework
Refactor all editors with new framework
TodoAppView
ListView
InputView
CounterView
TodoItemView
TodoItemView

```python
class TodoAppView(ViewBase):
    def refresh(self, todo_app_data):
        self._counter_view.refresh(todo_app_data.todo_list)
        self._list_view.refresh(todo_app_data.todo_list)
```



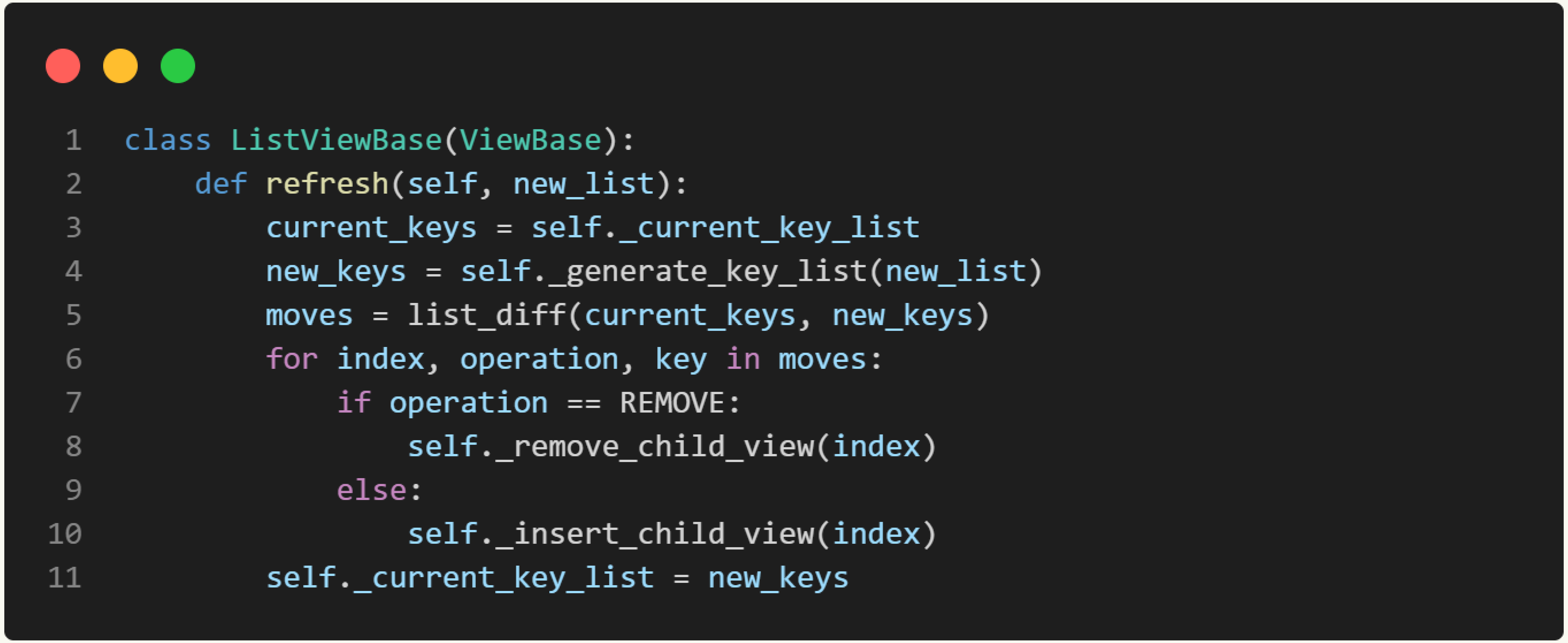
```python
class TodoCounterView(ViewBase):
    def _create_widget(self):
        return QtWidgets.QLabel()

    def refresh(self, todo_list):
        if todo_list is not self._current_todo_list:
            todo_count = len(todo_list)
            done_count = sum([1 if item.done else 0 for item in todo_list])
            self.widget.setText(f"{done_count}/{todo_count}")
            self._current_todo_list = todo_list

```



```python
class TodoListView(ListViewBase):
    def refresh_children(self, todo_list):
        for i, todo_item in enumerate(todo_list):
            self._child_view[i].refresh(todo_item)
```

```python
class TodoItemView(ViewBase):
    def refresh(self, todo_item):
        if todo_item is not self._current_todo_item:
            self._done_checkbox.setChecked(todo_item.done)
            self._push_button.setStyleSheet(
                LABEL_BUTTON.format(color=todo_item.color))
            self._content_edit.setText(todo_item.content)
            self._current_todo_item = todo_item
```

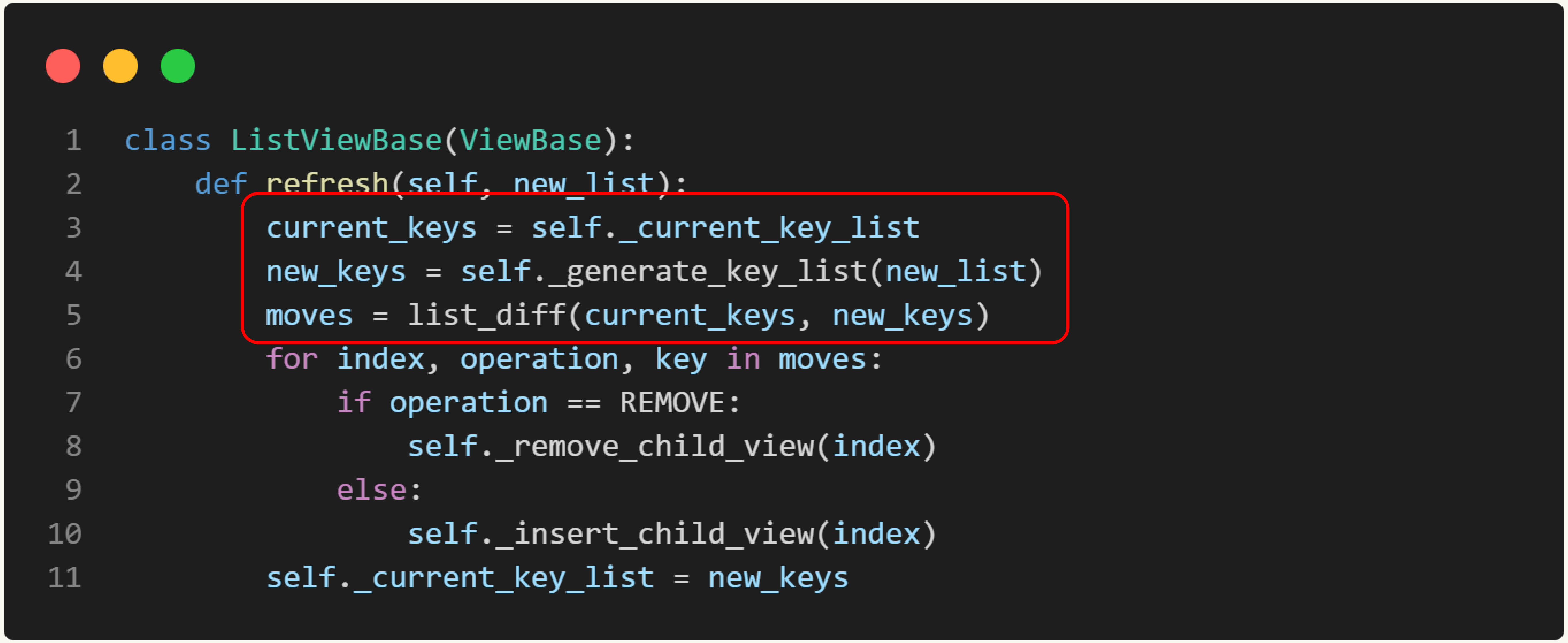
```python
class TodoItemView(ViewBase):
    def refresh(self, todo_item):
        if todo_item is not self._current_todo_item:
            self._done_checkbox.setChecked(todo_item.done)
            self._push_button.setStyleSheet(
                LABEL_BUTTON.format(color=todo_item.color))
            if todo_item.content is not self._current_todo_item.content:
                self._content_edit.setText(todo_item.content)
            self._current_todo_item = todo_item
```



```python
class ListViewBase(ViewBase):
    def refresh(self, new_list):
        current_keys = self._current_key_list
        new_keys = self._generate_key_list(new_list)
        moves = list_diff(current_keys, new_keys)
        for index, operation, key in moves:
            if operation == REMOVE:
                self._remove_child_view(index)
            else:
                self._insert_child_view(index)
        self._current_key_list = new_keys
```

LIST DIFF

- Inspired by list_diff in React
- Heuristic
- Remove operation before insert operation



```python
class ListViewBase(ViewBase):
    def refresh(self, new_list):
        current_keys = self._current_key_list
        new_keys = self._generate_key_list(new_list)
        moves = list_diff(current_keys, new_keys)
        for index, operation, key in moves:
            if operation == REMOVE:
                self._remove_child_view(index)
            else:
                self._insert_child_view(index)
        self._current_key_list = new_keys
```

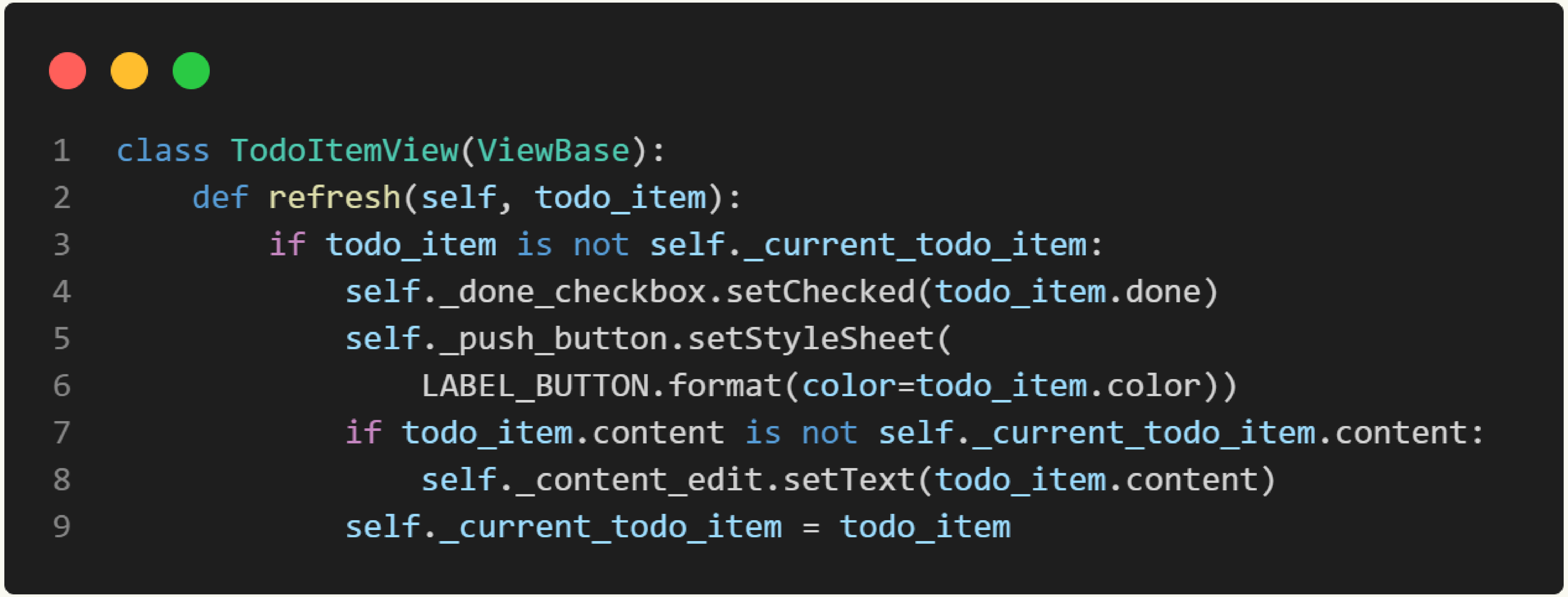
LIST DIFF

```
[TodoItemData(id_=1, done=False)]

[TodoItemData(id_=1, done=True)]

(DEL, TodoItemData(id_=1, done=False))

(ADD, TodoItemData(id_=1, done=True))
```

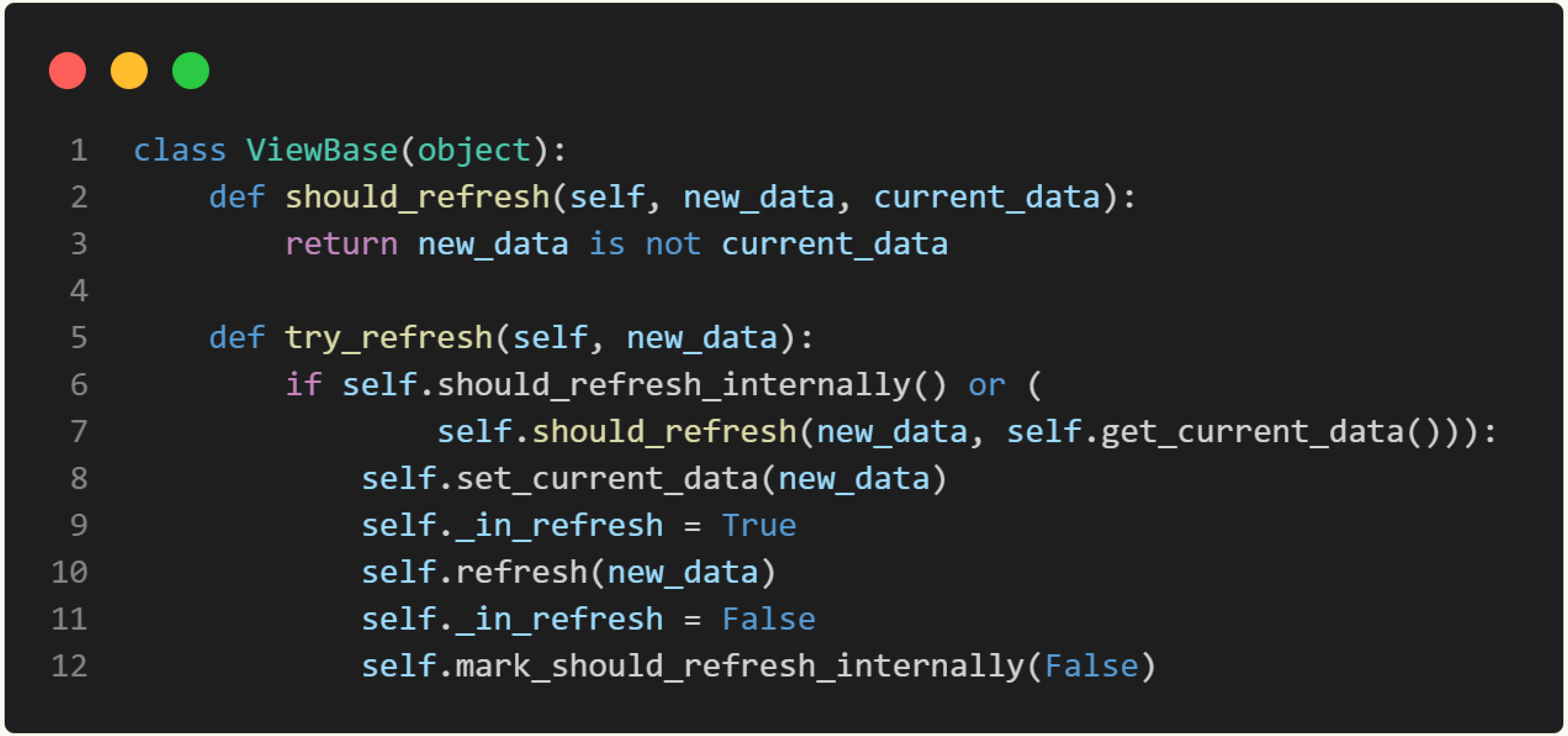
LIST DIFF

[1]

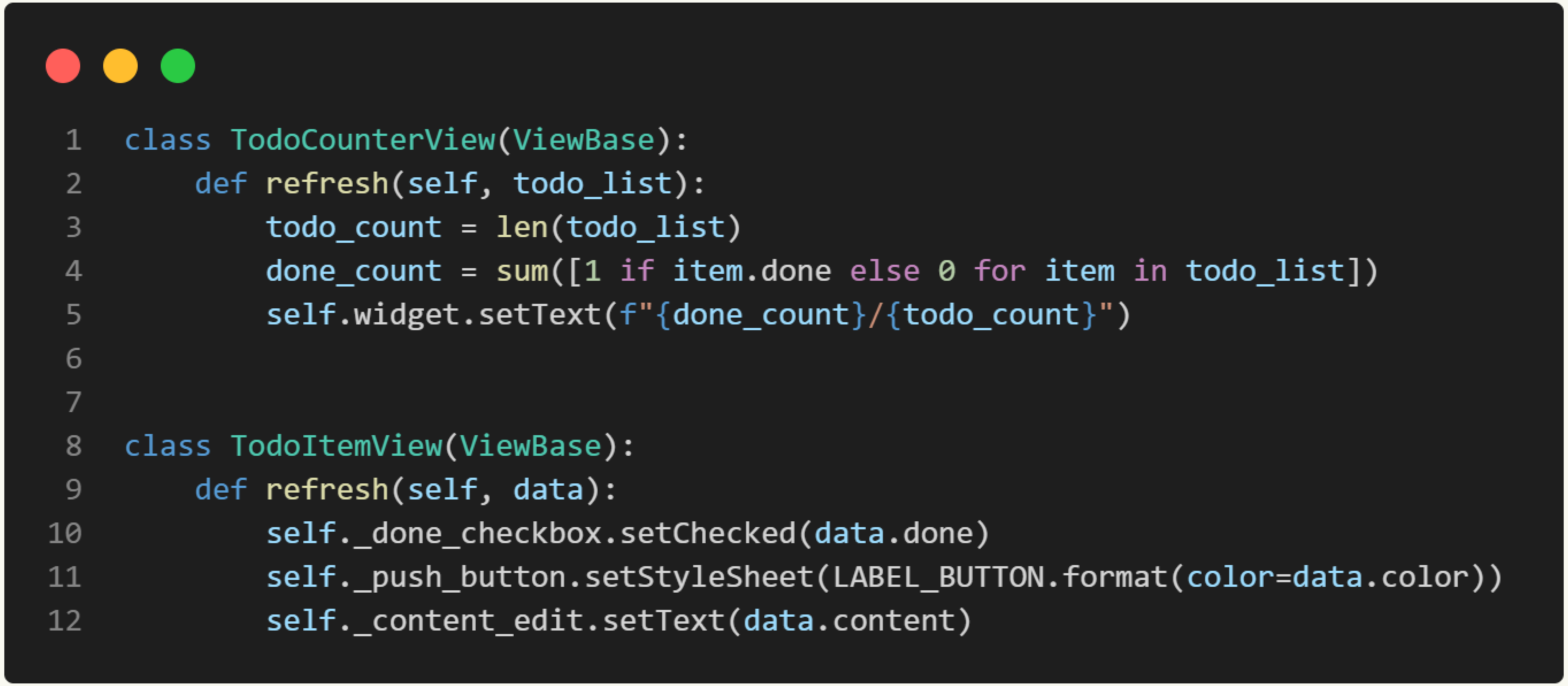
[1]



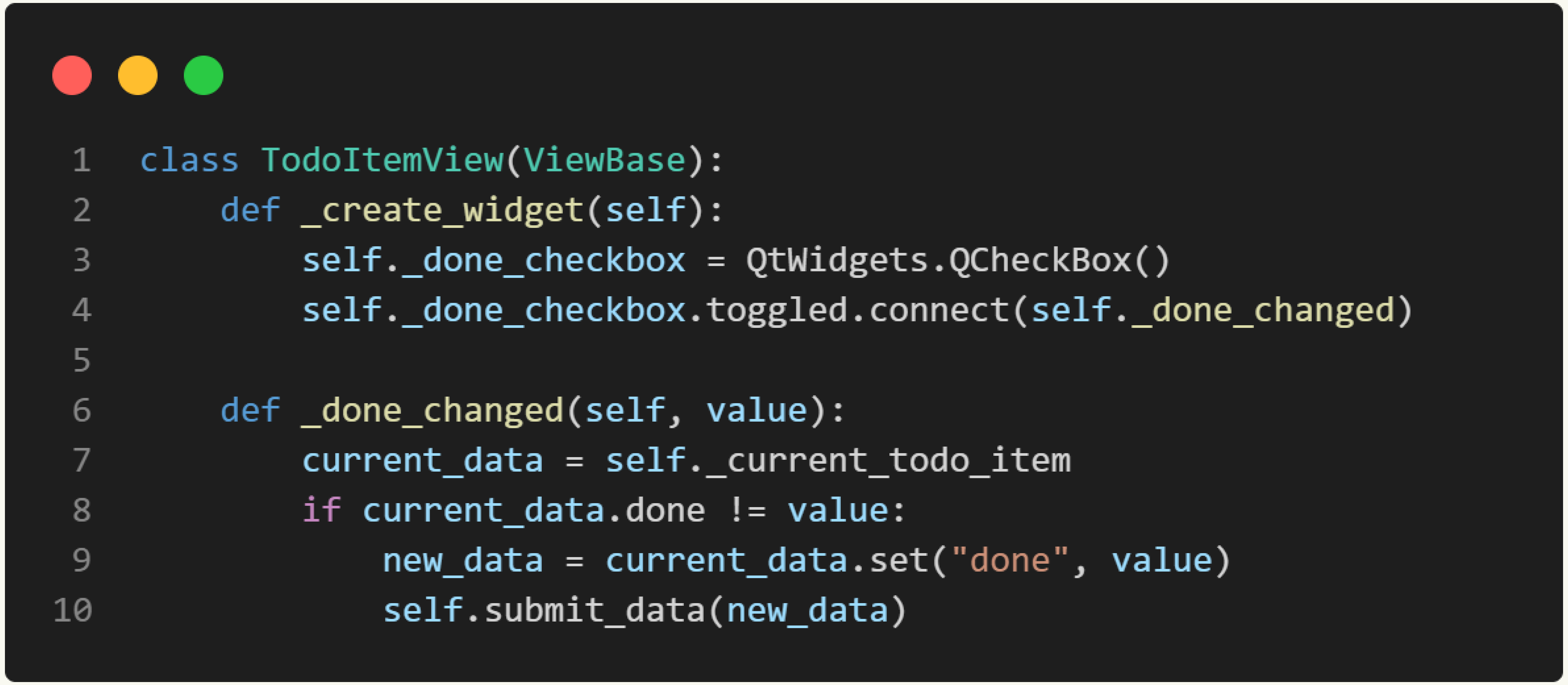
```python
class TodoItemView(ViewBase):
    def refresh(self, todo_item):
        if todo_item is not self._current_todo_item:
            self._done_checkbox.setChecked(todo_item.done)
            self._push_button.setStyleSheet(
                LABEL_BUTTON.format(color=todo_item.color))
            if todo_item.content is not self._current_todo_item.content:
                self._content_edit.setText(todo_item.content)
            self._current_todo_item = todo_item
```



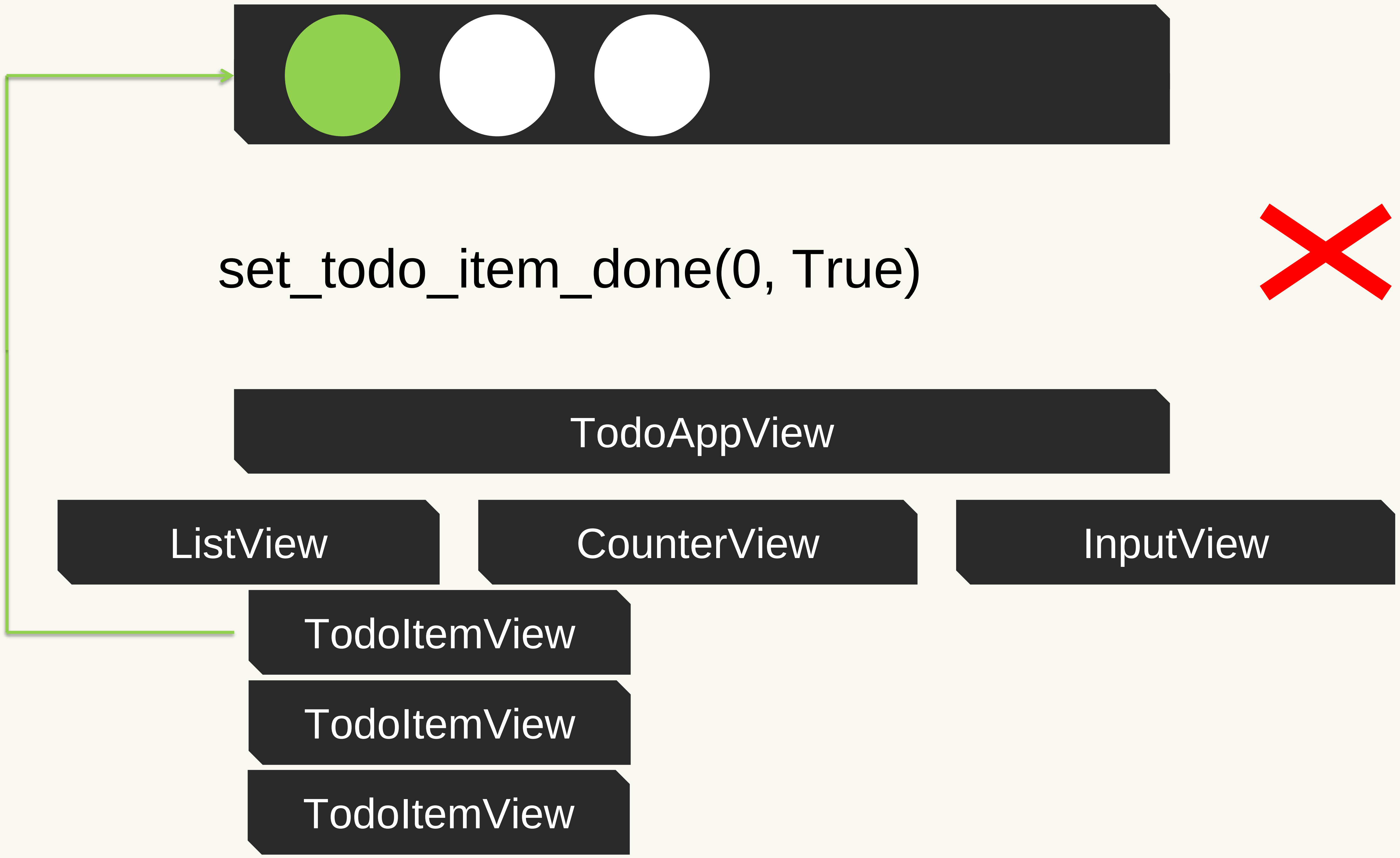
```python
class ViewBase(object):
    def should_refresh(self, new_data, current_data):
        return new_data is not current_data

    def try_refresh(self, new_data):
        if self.should_refresh_internally() or (
                self.should_refresh(new_data, self.get_current_data())):
            self.set_current_data(new_data)
            self._in_refresh = True
            self.refresh(new_data)
            self._in_refresh = False
            self.mark_should_refresh_internally(False)
```

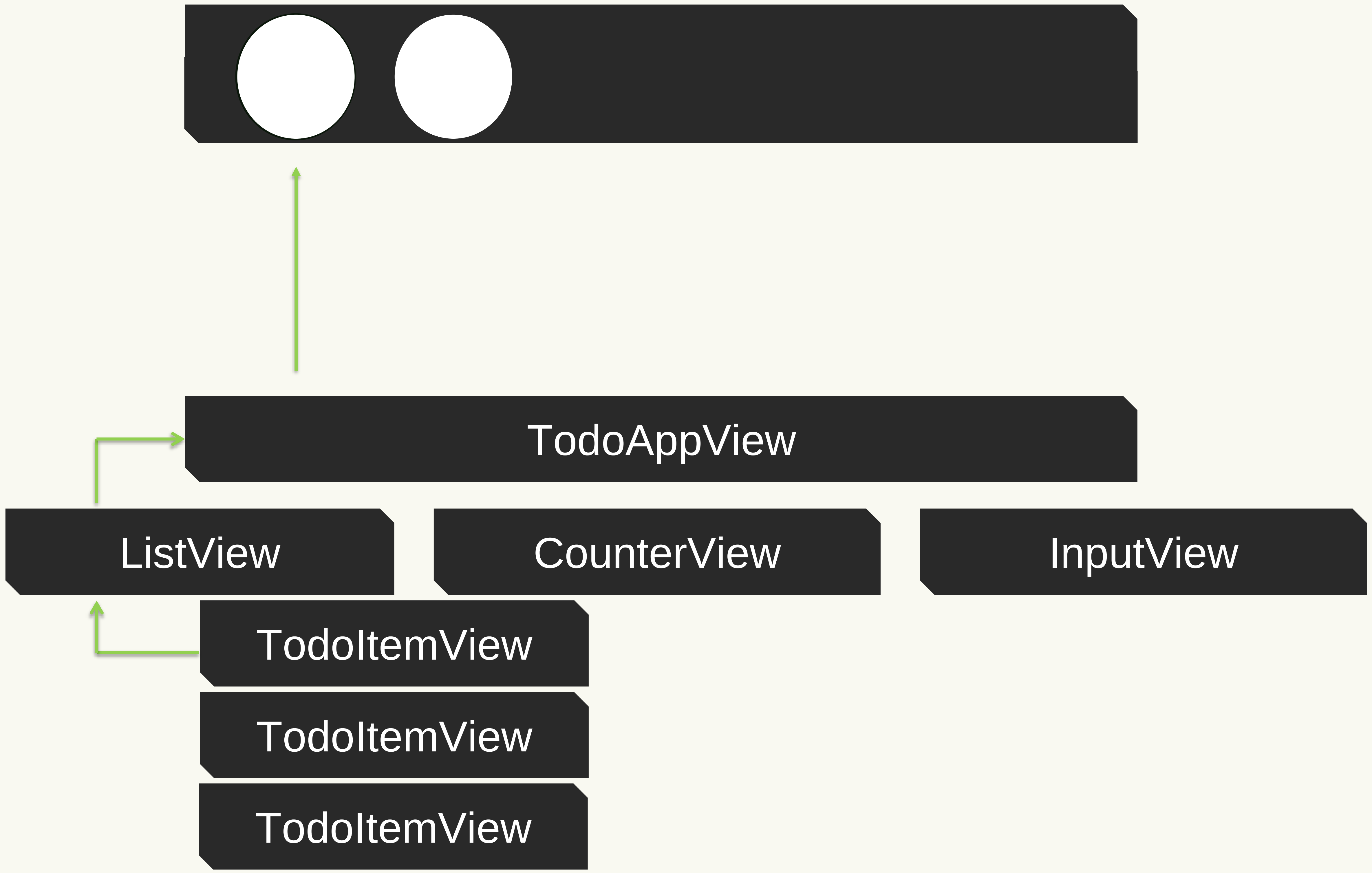


```python
class TodoCounterView(ViewBase):
    def refresh(self, todo_list):
        todo_count = len(todo_list)
        done_count = sum([1 if item.done else 0 for item in todo_list])
        self.widget.setText(f"{done_count}/{todo_count}")


class TodoItemView(ViewBase):
    def refresh(self, data):
        self._done_checkbox.setChecked(data.done)
        self._push_button.setStyleSheet(LABEL_BUTTON.format(color=data.color))
        self._content_edit.setText(data.content)
```



```python
class TodoItemView(ViewBase):
    def _create_widget(self):
        self._done_checkbox = QtWidgets.QCheckBox()
        self._done_checkbox.toggled.connect(self._done_changed)

    def _done_changed(self, value):
        current_data = self._current_todo_item
        if current_data.done != value:
            new_data = current_data.set("done", value)
            self.submit_data(new_data)
```

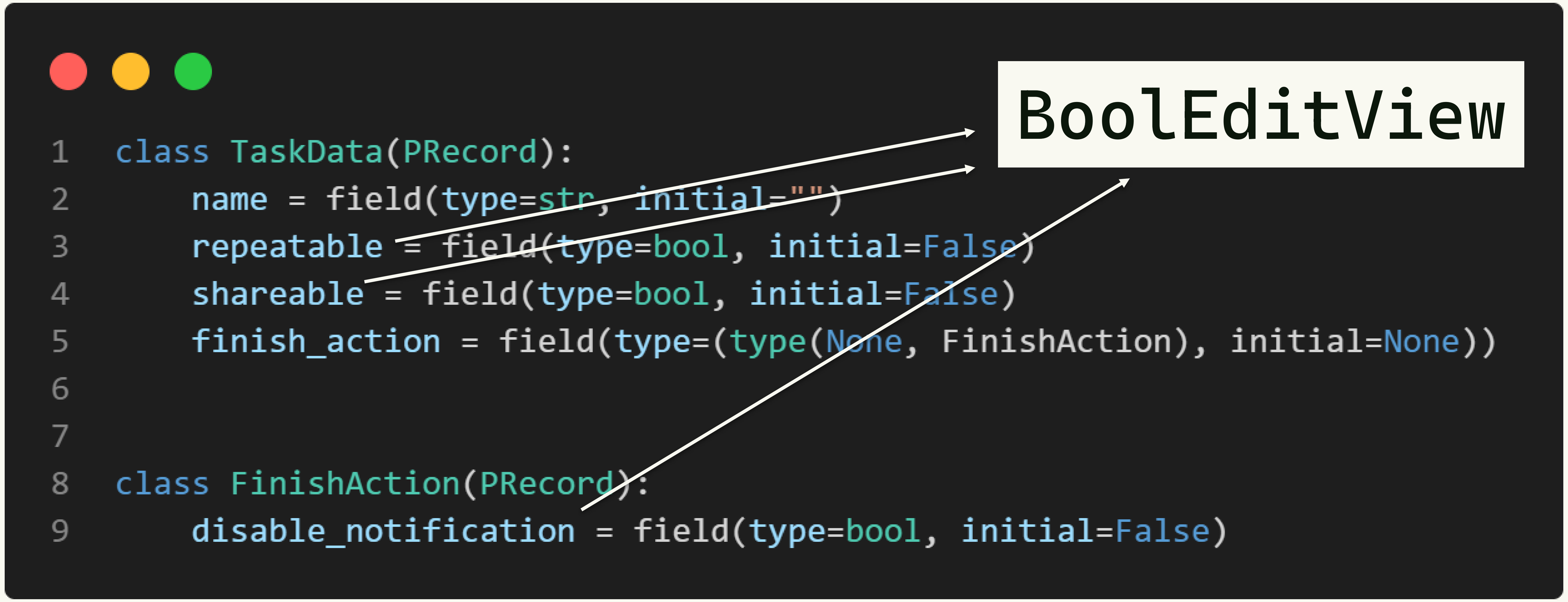

set_todo_item_done(0, True)
TodoAppView
ListView
CounterView
InputView
TodoItemView
TodoItemView
TodoItemView


TodoAppView
ListView
CounterView
InputView
TodoItemView
TodoItemView
TodoItemView

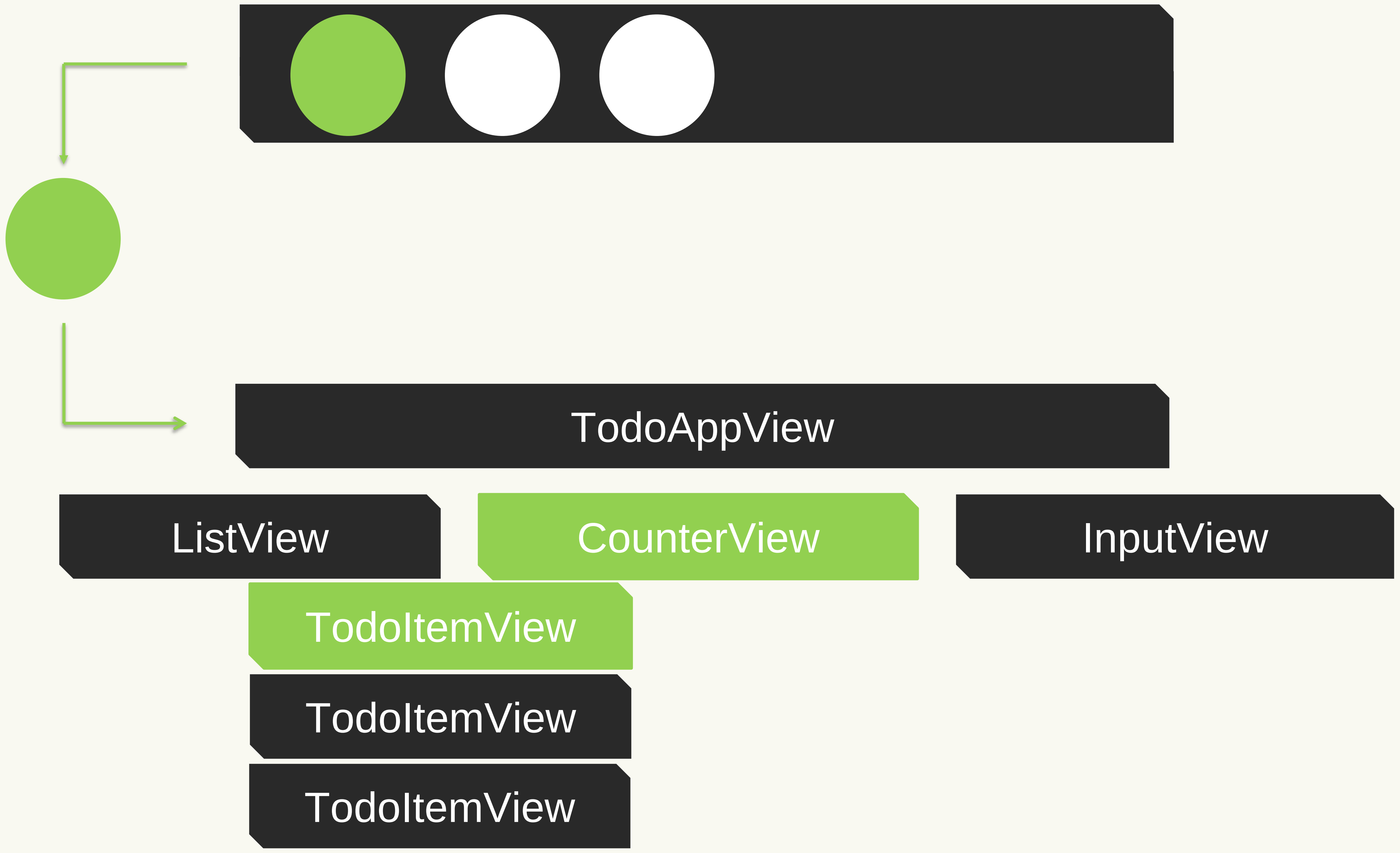
SUBMIT DATA

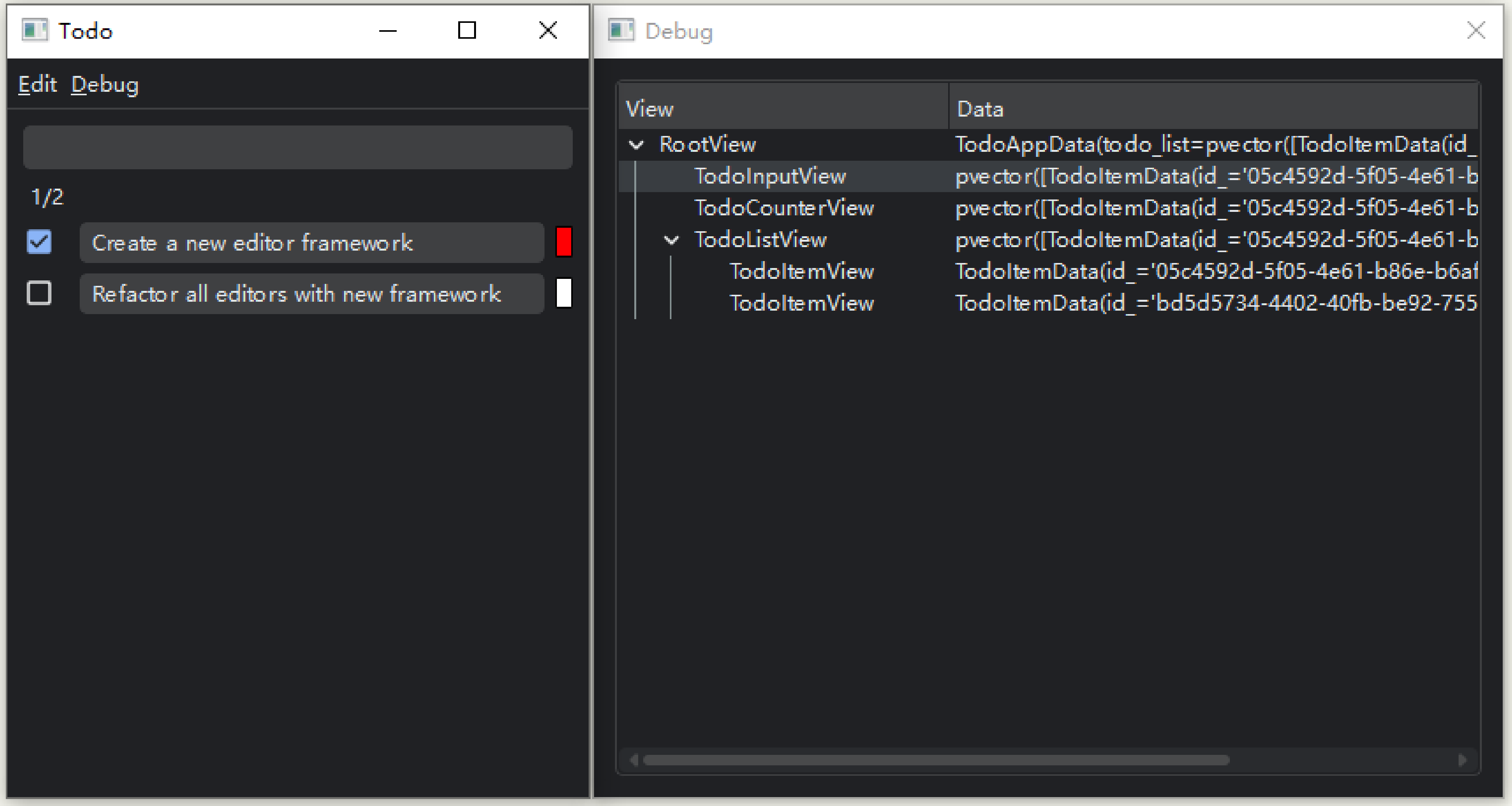
- Always submit data to parent
- Zero dependency to other classes
- Maximize reusability

SUBMIT DATA



```
class TaskData(PRecord):
    name = field(type=str, initial="")
    repeatable = field(type=bool, initial=False)
    shareable = field(type=bool, initial=False)
    finish_action = field(type=(type(None), FinishAction), initial=None))


class FinishAction(PRecord):
    disable_notification = field(type=bool, initial=False)
```


TodoAppView
ListView
CounterView
InputView
TodoItemView
TodoItemView
TodoItemView

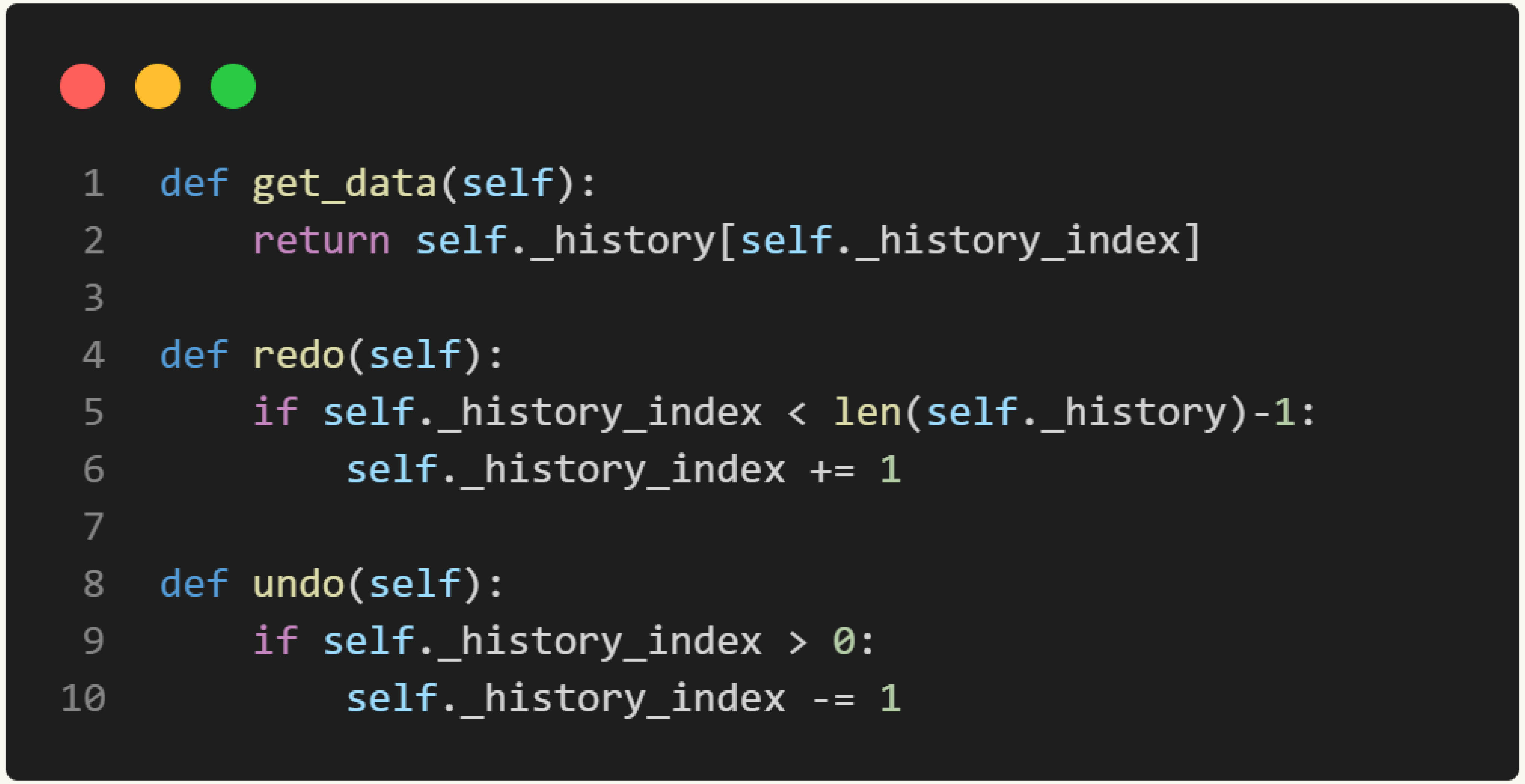


1/2

- [x] Create a new editor framework
- [ ] Refactor all editors with new framework

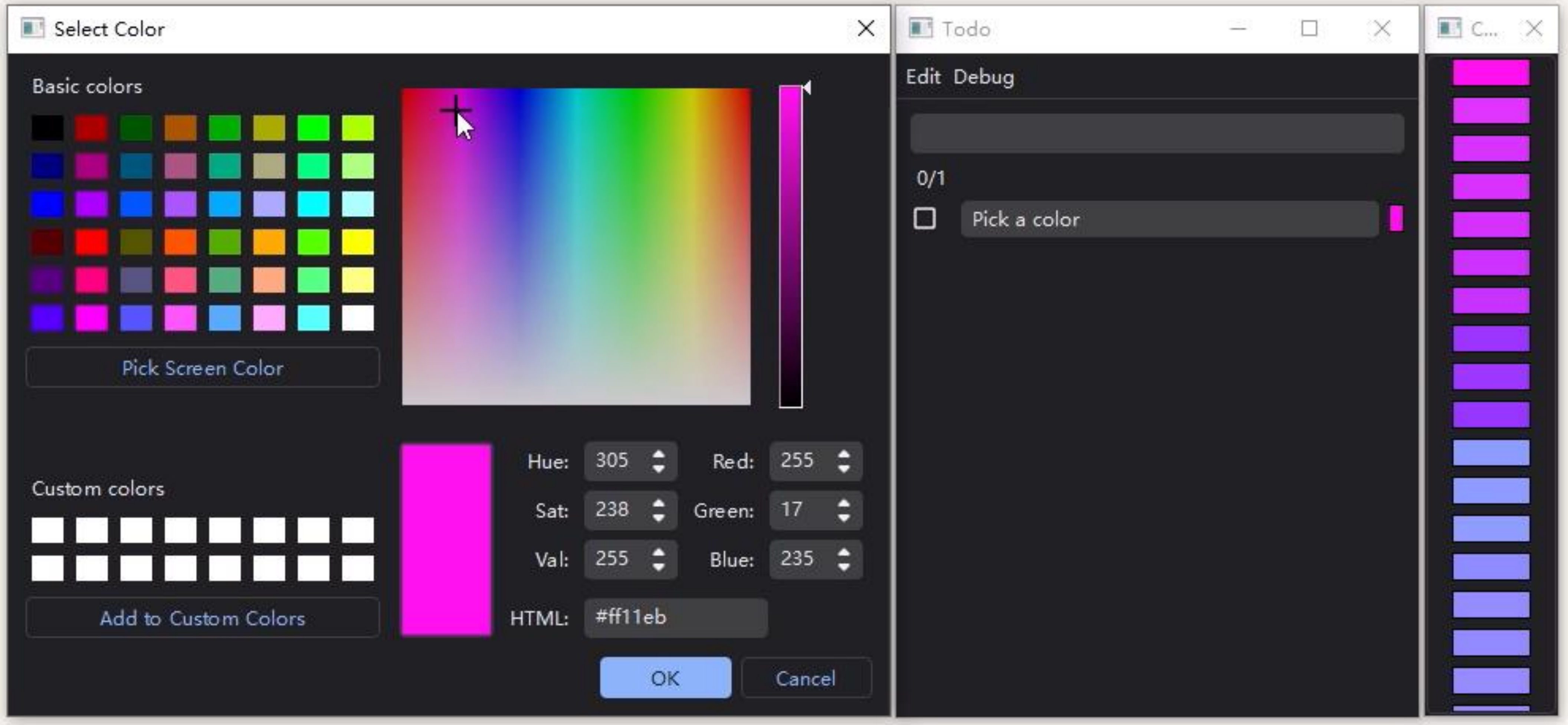
View	Data
RootView	TodoAppData(todo_list=pvector([TodoItemData(id_
TodoInputView	pvector([TodoItemData(id_='05c4592d-5f05-4e61-b
TodoCounterView	pvector([TodoItemData(id_='05c4592d-5f05-4e61-b
TodoListView	pvector([TodoItemData(id_='05c4592d-5f05-4e61-b
TodoItemView	TodoItemData(id_='05c4592d-5f05-4e61-b86e-b6af
TodoItemView	TodoItemData(id_='bd5d5734-4402-40fb-be92-755

UNDO/REDO



```python
def get_data(self):
    return self._history[self._history_index]

def redo(self):
    if self._history_index < len(self._history)-1:
        self._history_index += 1

def undo(self):
    if self._history_index > 0:
        self._history_index -= 1
```

PREVIEW CHANGES

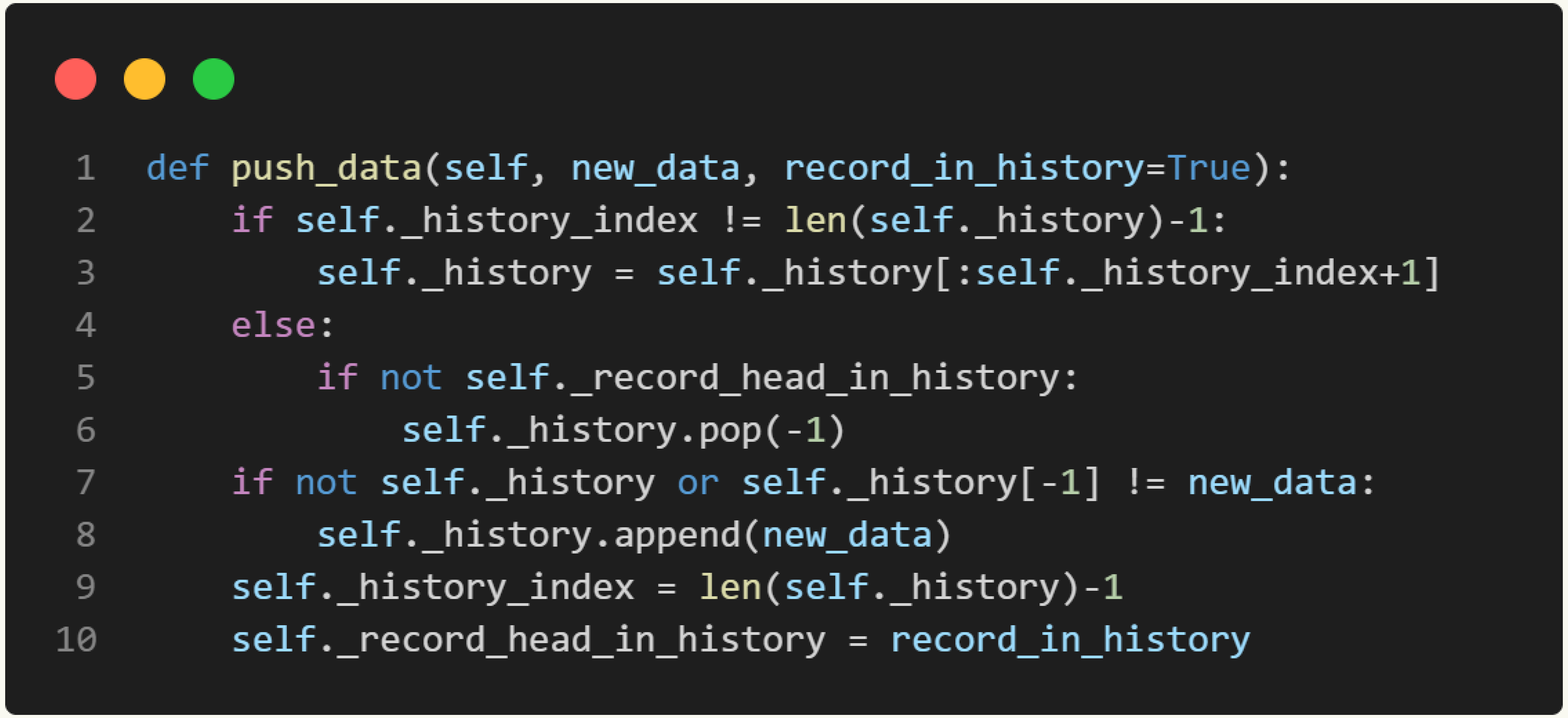


Hue: 305
Sat: 238
Val: 255
Red: 255
Green: 17
Blue: 235
HTML: #ff11eb

0/1

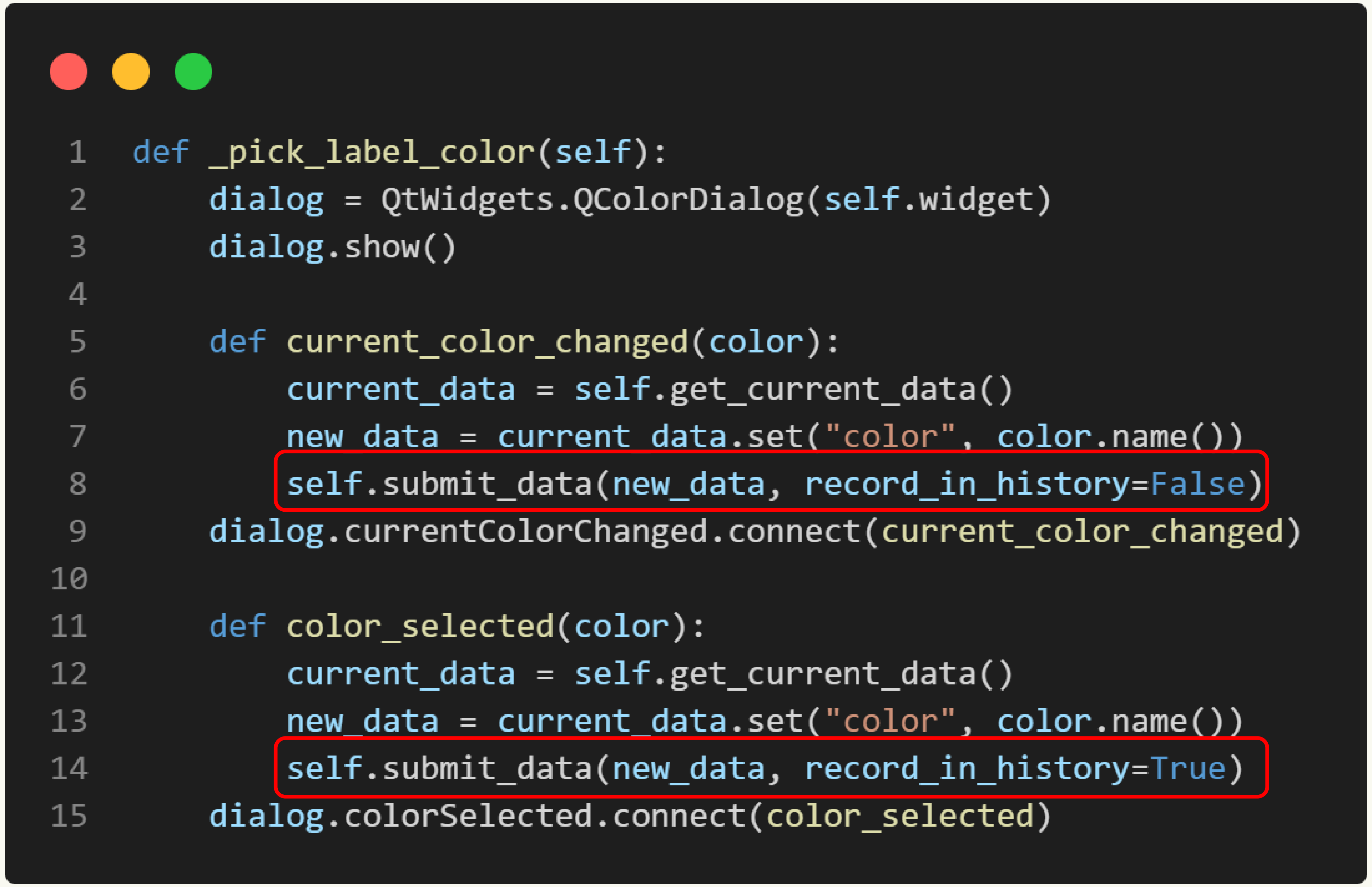
Pick a color

UNDO/REDO

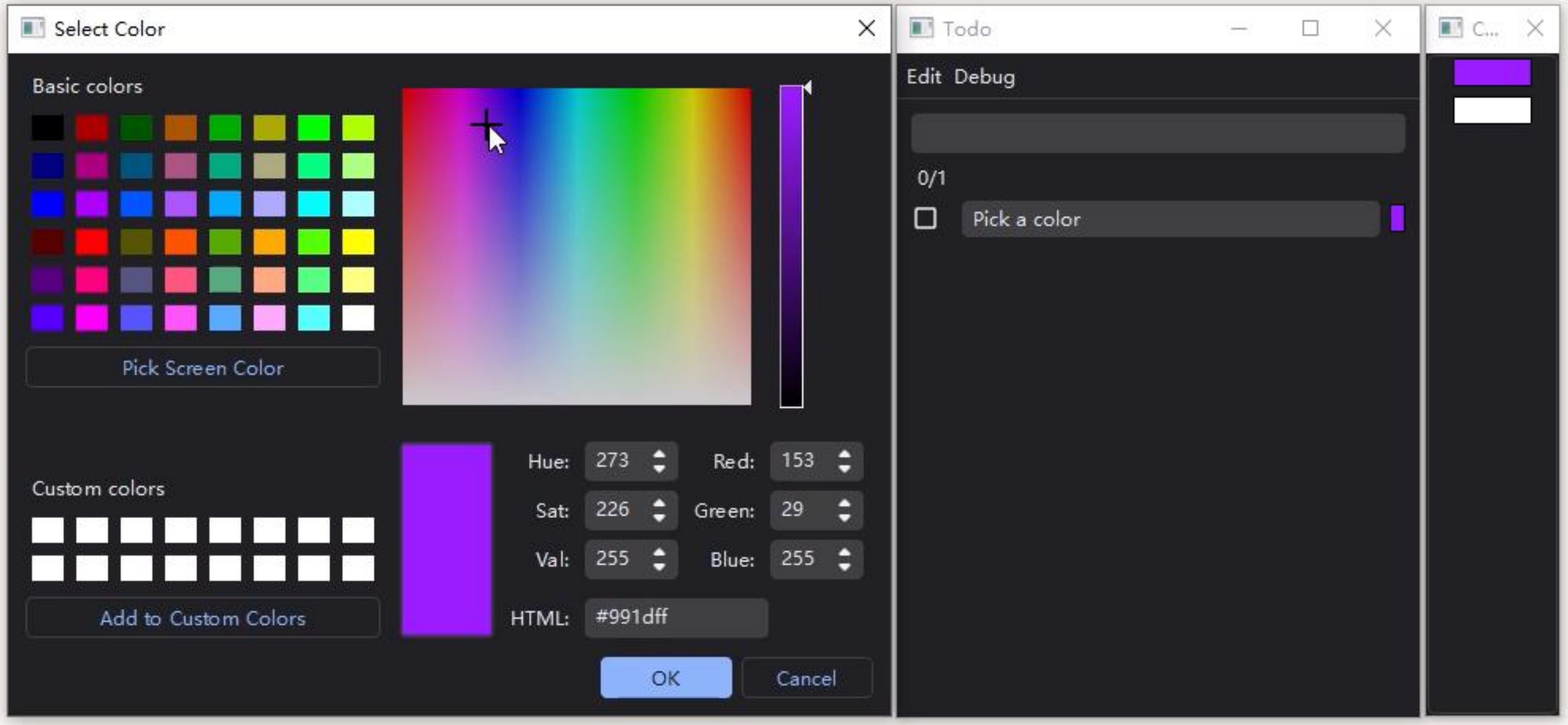


```python
def push_data(self, new_data, record_in_history=True):
    if self._history_index != len(self._history)-1:
        self._history = self._history[:self._history_index+1]
    else:
        if not self._record_head_in_history:
            self._history.pop(-1)
    if not self._history or self._history[-1] != new_data:
        self._history.append(new_data)
    self._history_index = len(self._history)-1
    self._record_head_in_history = record_in_history
```

UNDO/REDO



```python
def _pick_label_color(self):
    dialog = QtWidgets.QColorDialog(self.widget)
    dialog.show()

    def current_color_changed(color):
        current_data = self.get_current_data()
        new_data = current_data.set("color", color.name())
        self.submit_data(new_data, record_in_history=False)
    dialog.currentColorChanged.connect(current_color_changed)

    def color_selected(color):
        current_data = self.get_current_data()
        new_data = current_data.set("color", color.name())
        self.submit_data(new_data, record_in_history=True)
    dialog.colorSelected.connect(color_selected)
```

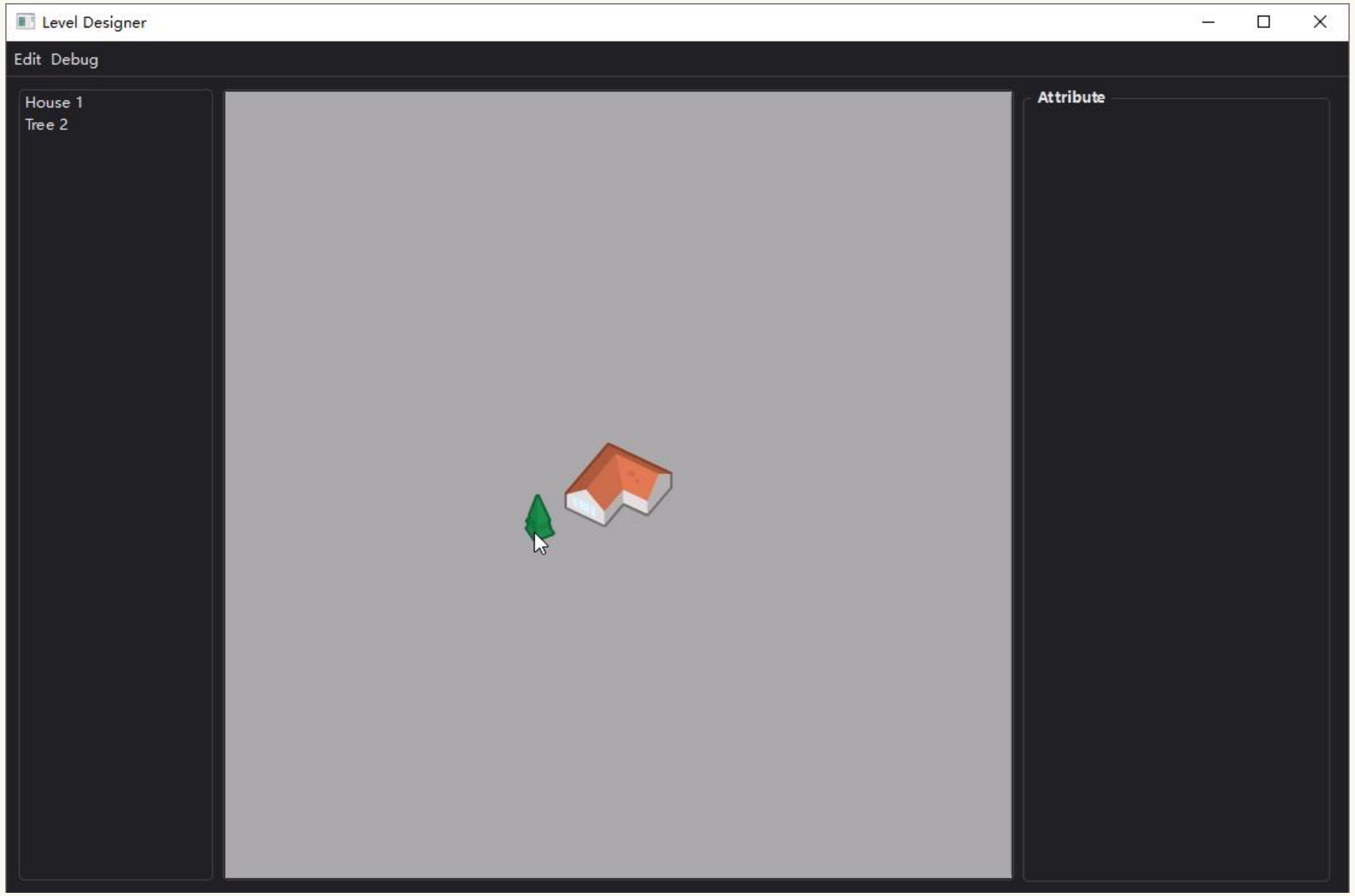


Hue: 273
Sat: 226
Val: 255
Red: 153
Green: 29
Blue: 255
HTML: #991dff

0/1

Pick a color

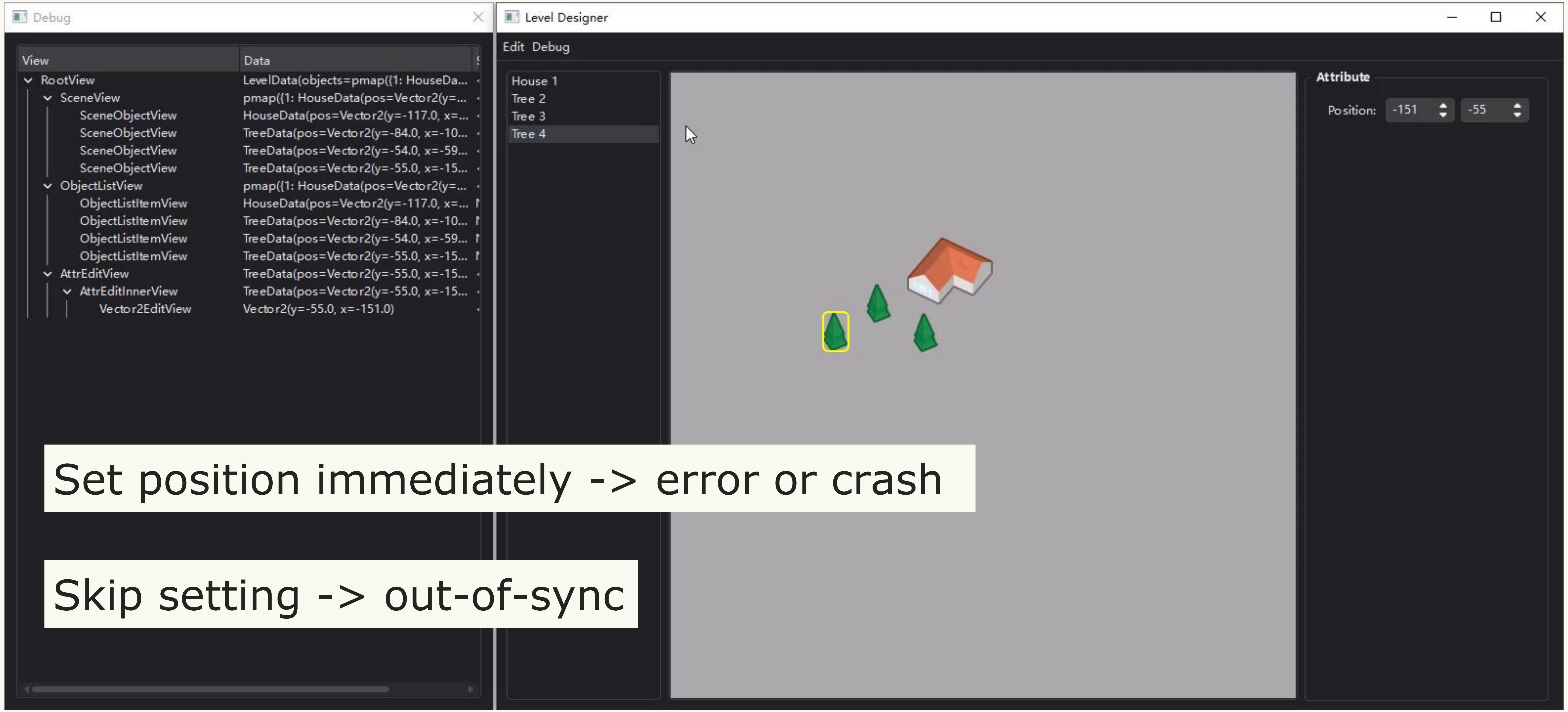
CONTINUOUS CHANGE



What about 3D models?

3D models are actually UIs

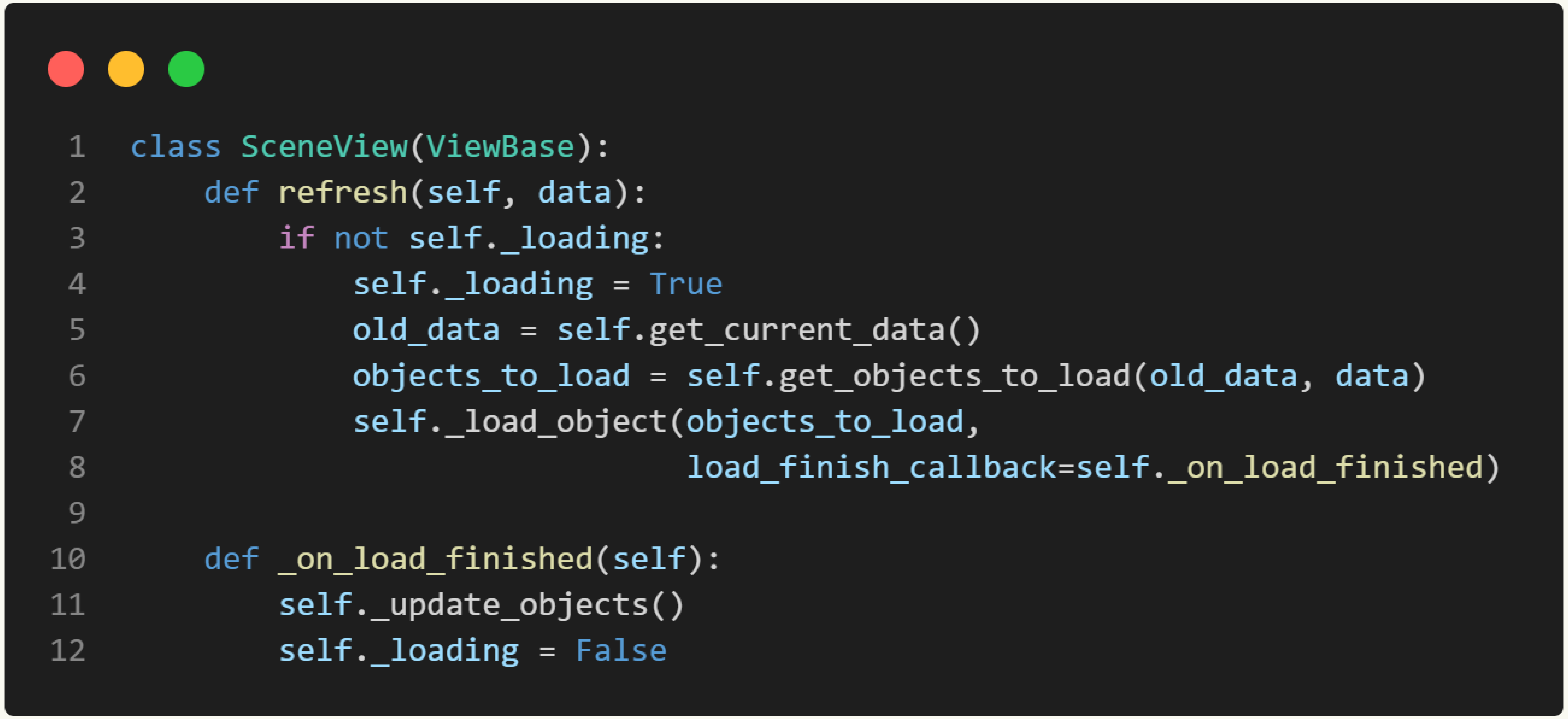
Decouple UI update with data update



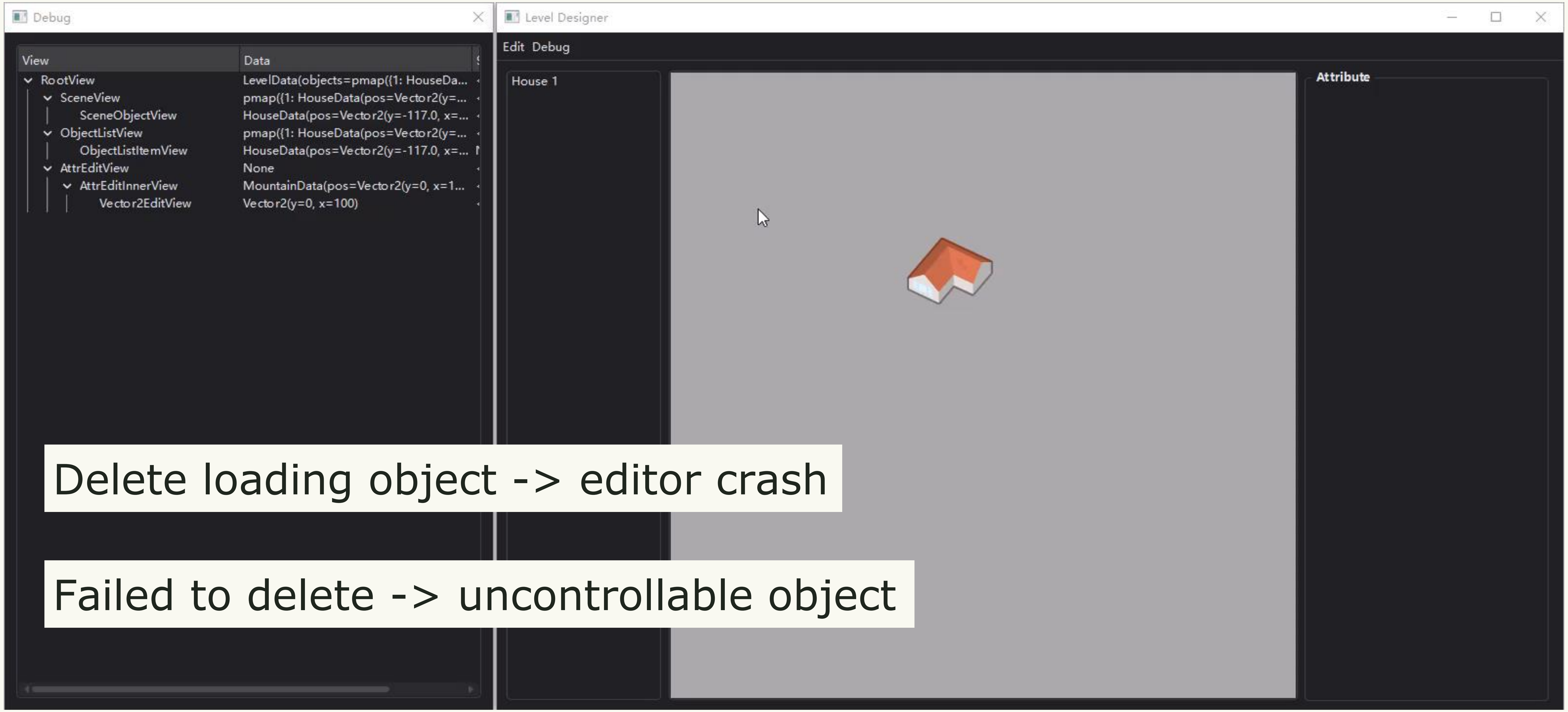
```
View                          Data
RootView                      LevelData(objects=pmap({1: HouseDa...
  SceneView                   pmap({1: HouseData(pos=Vector2(y=...
    SceneObjectView           HouseData(pos=Vector2(y=-117.0, x=...
    SceneObjectView           TreeData(pos=Vector2(y=-84.0, x=-10...
    SceneObjectView           TreeData(pos=Vector2(y=-54.0, x=-59...
    SceneObjectView           TreeData(pos=Vector2(y=-55.0, x=-15...
  ObjectListView              pmap({1: HouseData(pos=Vector2(y=...
    ObjectListItemView        HouseData(pos=Vector2(y=-117.0, x=...
    ObjectListItemView        TreeData(pos=Vector2(y=-84.0, x=-10...
    ObjectListItemView        TreeData(pos=Vector2(y=-54.0, x=-59...
    ObjectListItemView        TreeData(pos=Vector2(y=-55.0, x=-15...
  AttrEditView                TreeData(pos=Vector2(y=-55.0, x=-15...
    AttrEditInnerView         TreeData(pos=Vector2(y=-55.0, x=-15...
      Vector2EditView         Vector2(y=-55.0, x=-151.0)
```

Set position immediately -> error or crash

Skip setting -> out-of-sync



```python
class SceneView(ViewBase):
    def refresh(self, data):
        if not self._loading:
            self._loading = True
            old_data = self.get_current_data()
            objects_to_load = self.get_objects_to_load(old_data, data)
            self._load_object(objects_to_load,
                              load_finish_callback=self._on_load_finished)

    def _on_load_finished(self):
        self._update_objects()
        self._loading = False
```



View	Data
RootView	LevelData(objects=pmap({1: HouseDa...
SceneView	pmap({1: HouseData(pos=Vector2(y=...
SceneObjectView	HouseData(pos=Vector2(y=-117.0, x=...
ObjectListView	pmap({1: HouseData(pos=Vector2(y=...
ObjectListItemView	HouseData(pos=Vector2(y=-117.0, x=...
AttrEditView	None
AttrEditInnerView	MountainData(pos=Vector2(y=0, x=1...
Vector2EditView	Vector2(y=0, x=100)

Delete loading object -> editor crash

Failed to delete -> uncontrollable object

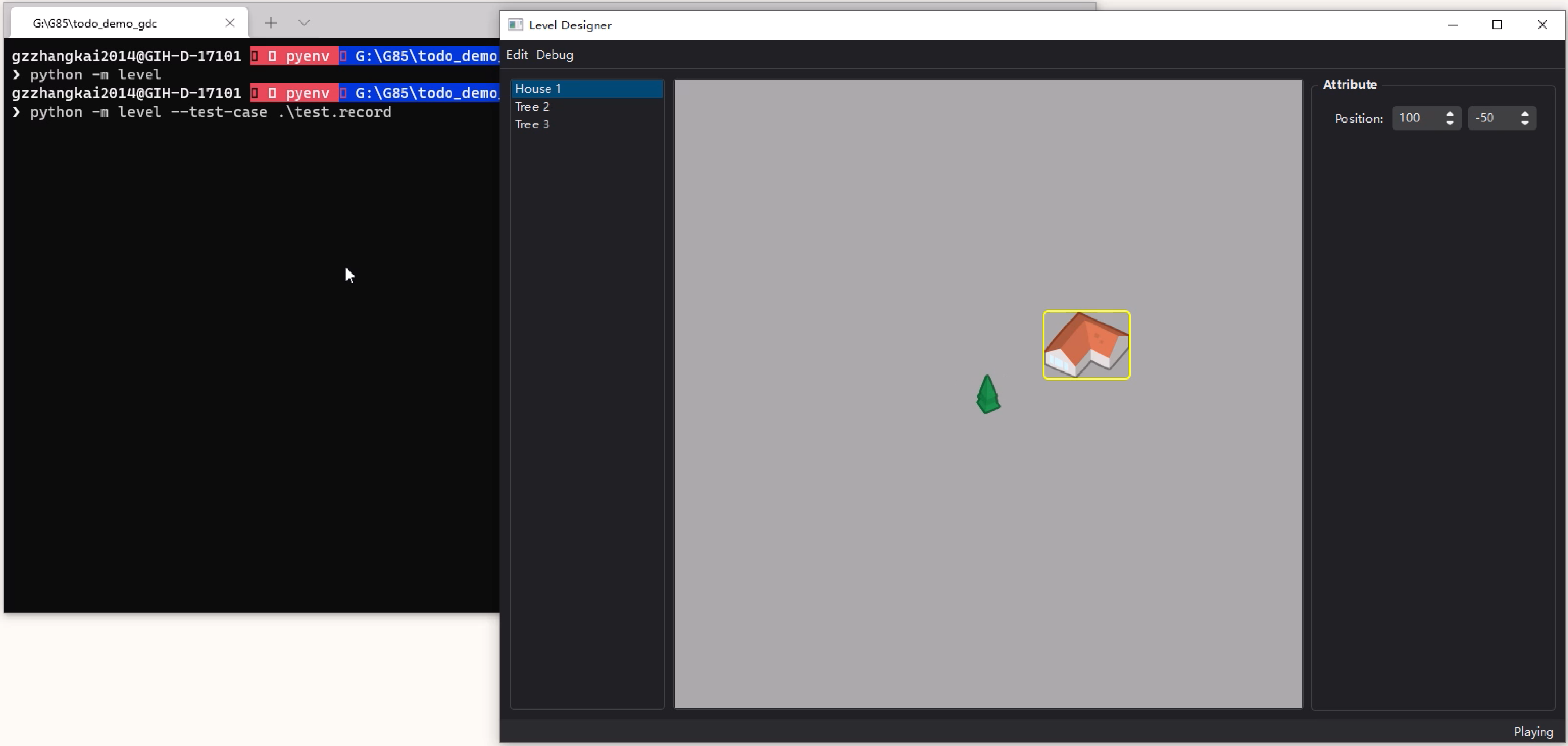
Decouple UI update with data update

- Game object loading
- UI animation
- Expensive data operation on another thread
- Auto save on another thread

AUTO TEST

Data operations → unittest

View logics → ?



```
G:\G85\todo_demo_gdc
gzzhangkai2014@GIH-D-17101   pyenv  G:\G85\todo_demo_
❯ python -m level
gzzhangkai2014@GIH-D-17101   pyenv  G:\G85\todo_demo_
❯ python -m level --test-case .\test.record
```

GOAL

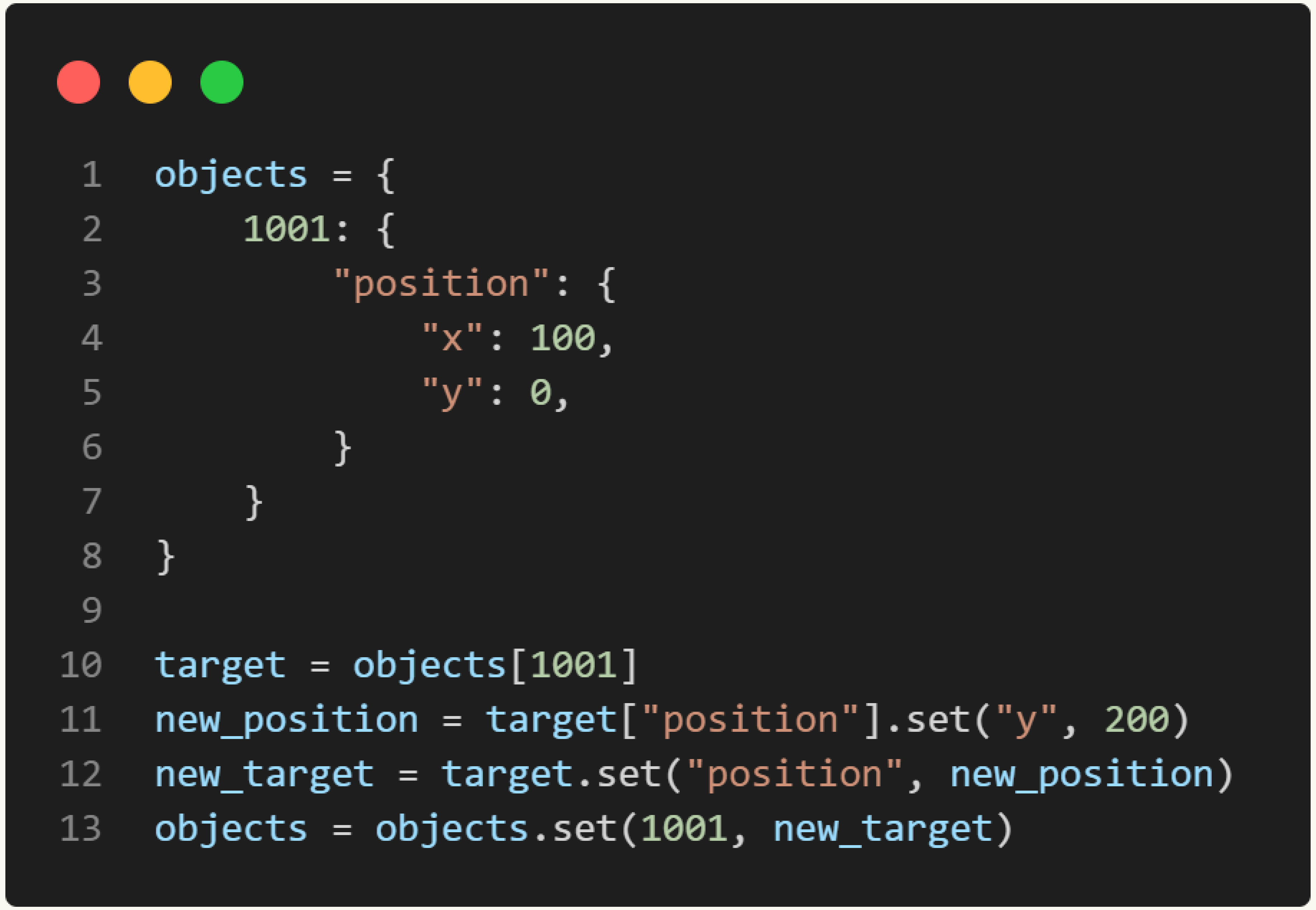
- Increase editor stability
- Increase editor development efficiency
  - Provide a simple undo/redo solution
- Paradigm for all editors
- Fast (enough) performance

MODIFY IMMUTABLE DATA

Transformation

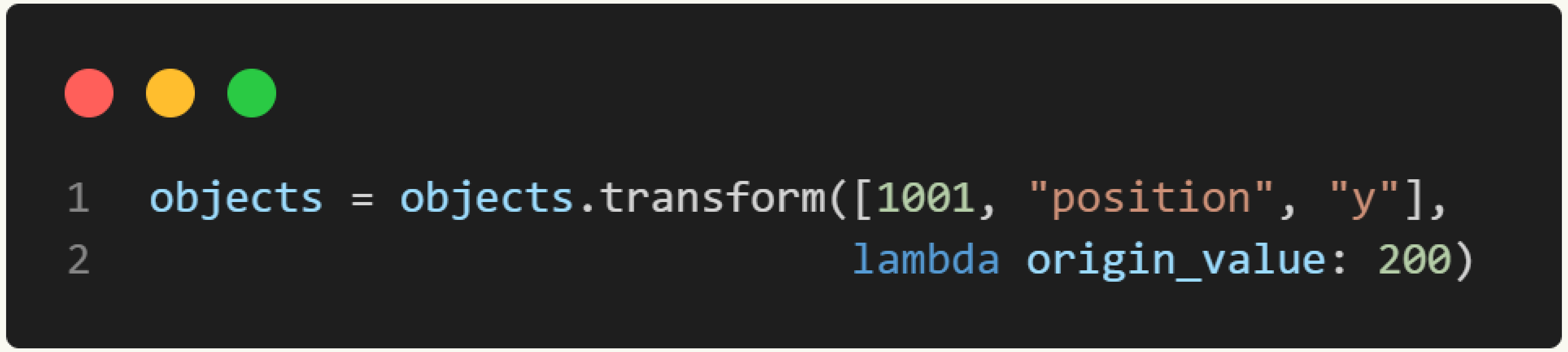
Evolver

TRANSFORMATION



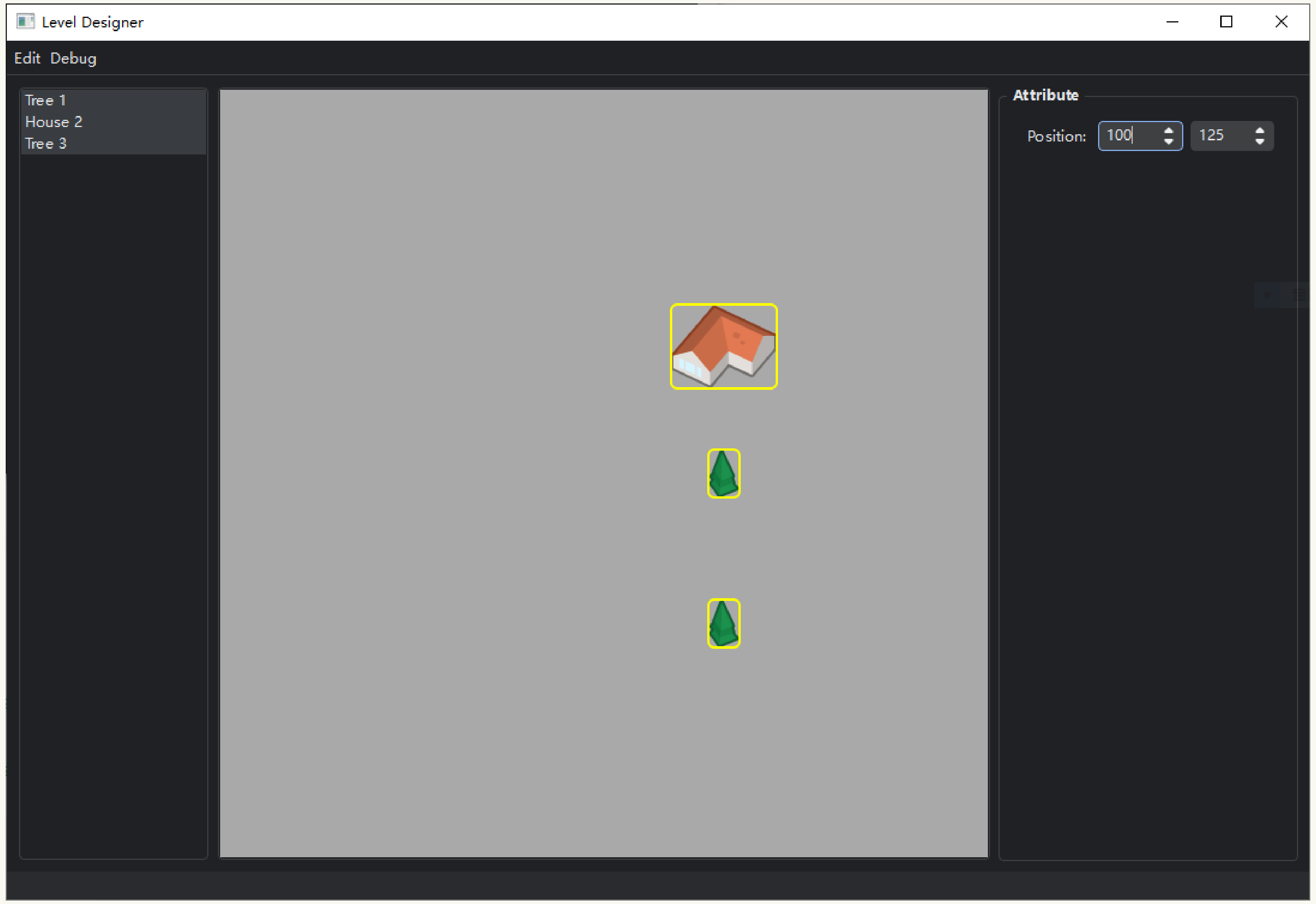
```
objects = {
    1001: {
        "position": {
            "x": 100,
            "y": 0,
        }
    }
}

target = objects[1001]
new_position = target["position"].set("y", 200)
new_target = target.set("position", new_position)
objects = objects.set(1001, new_target)
```

TRANSFORMATION

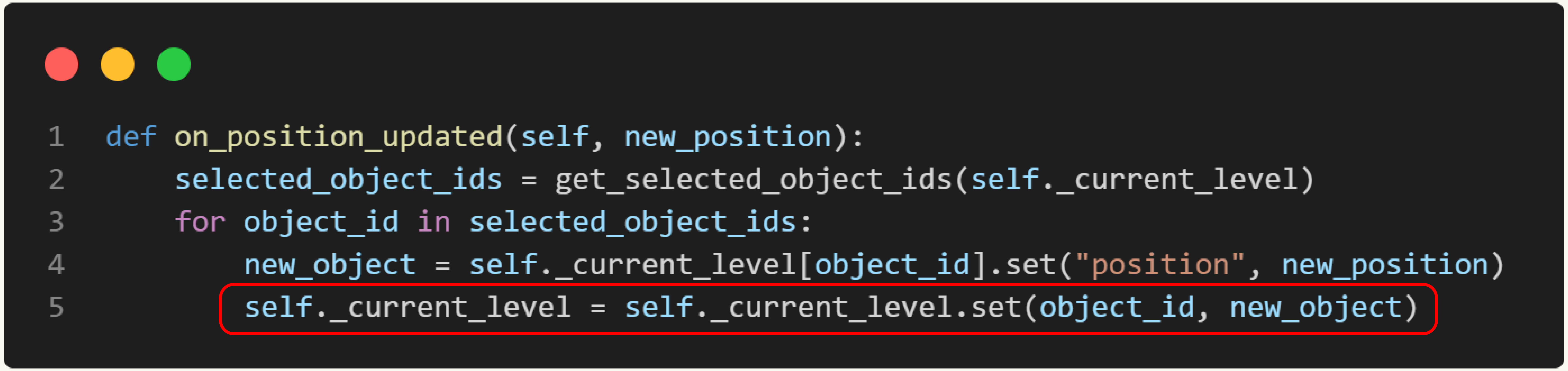


```python
objects = objects.transform([1001, "position", "y"],
                            lambda origin_value: 200)
```

EVOLVER

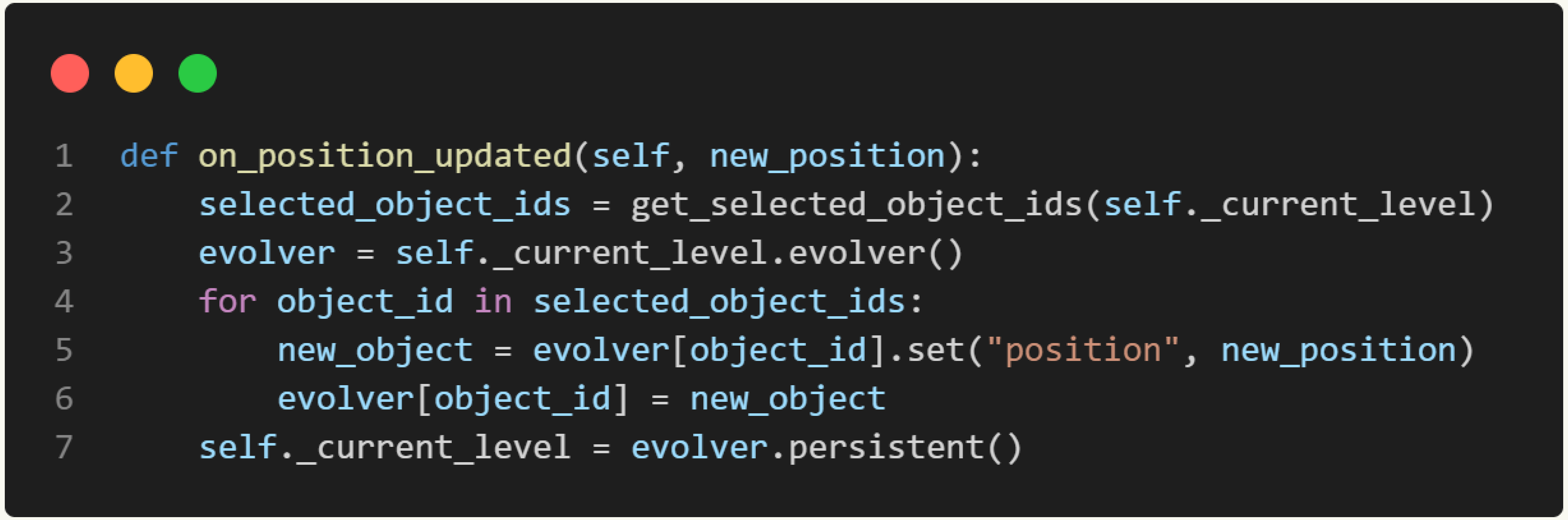

Level Designer
Edit Debug
Tree 1
House 2
Tree 3
Attribute
Position: 100 125

EVOLVER



```python
def on_position_updated(self, new_position):
    selected_object_ids = get_selected_object_ids(self._current_level)
    for object_id in selected_object_ids:
        new_object = self._current_level[object_id].set("position", new_position)
        self._current_level = self._current_level.set(object_id, new_object)
```

EVOLVER

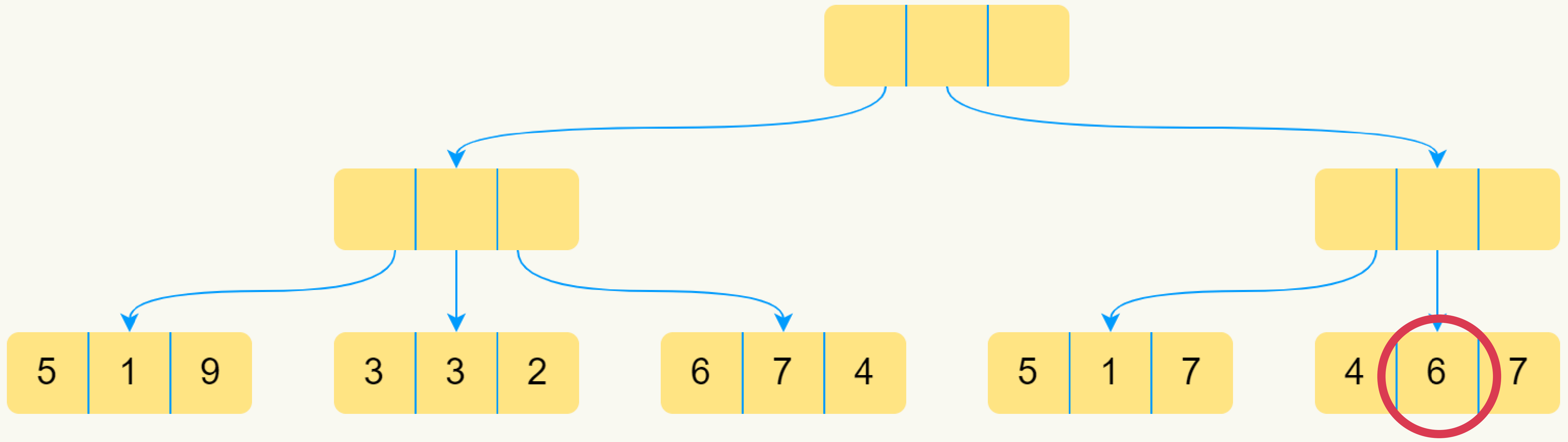


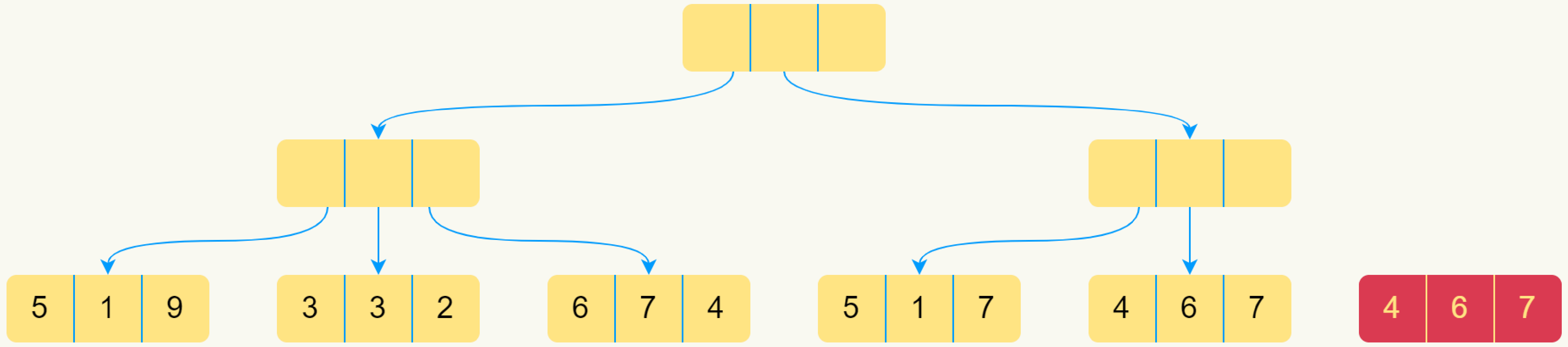
```python
def on_position_updated(self, new_position):
    selected_object_ids = get_selected_object_ids(self._current_level)
    evolver = self._current_level.evolver()
    for object_id in selected_object_ids:
        new_object = evolver[object_id].set("position", new_position)
        evolver[object_id] = new_object
    self._current_level = evolver.persistent()
```

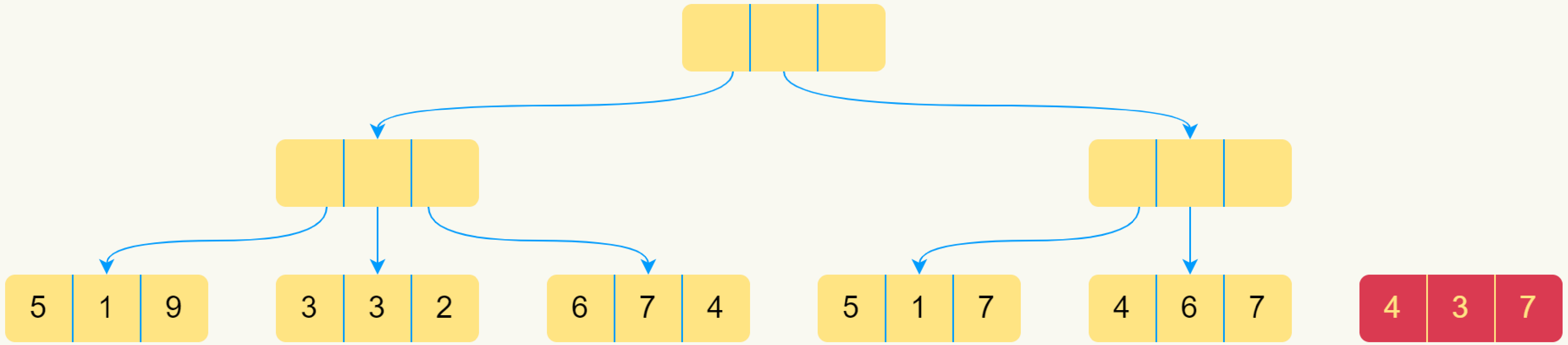
IMMUTABLE DATA

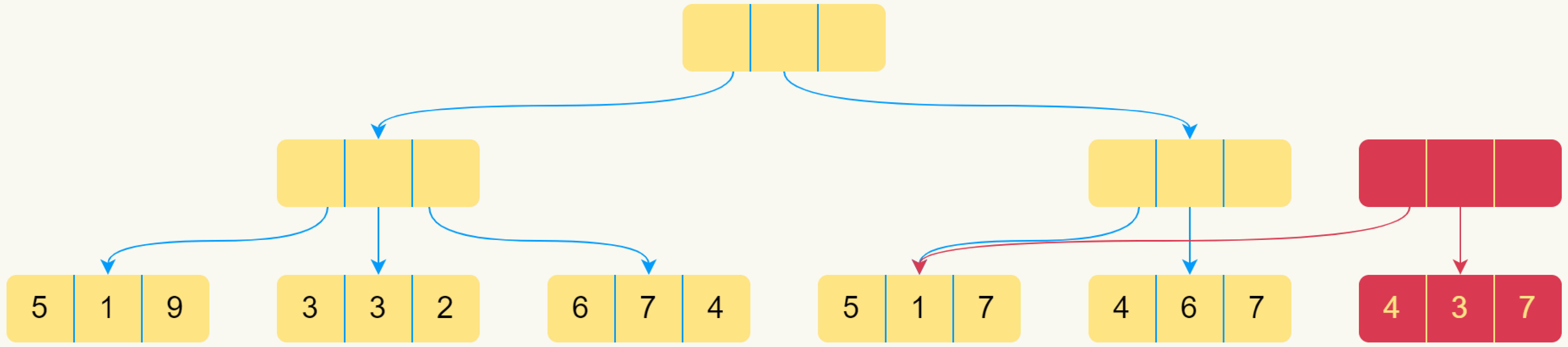
By Rich Hichkey in Clojure

Hash Array Mapped Trie by Phil Begwell

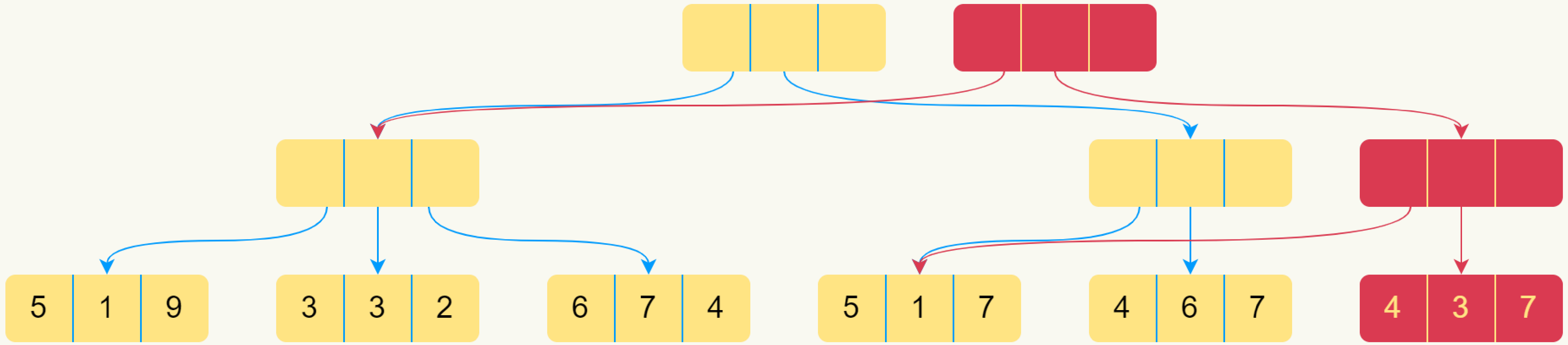

5 1 9
3 3 2
6 7 4
5 1 7
4 6 7


5 1 9
3 3 2
6 7 4
5 1 7
4 6 7
4 6 7


5 1 9
3 3 2
6 7 4
5 1 7
4 6 7
4 3 7


5 1 9
3 3 2
6 7 4
5 1 7
4 6 7
4 3 7

structural sharing



LIBRARY

C++

- Immer

  https://github.com/arximboldi/immer

  CppCon'17 Talk: Postmodern Immutable Data Structures

Python

- Pyrsistent

  https://github.com/tobgu/pyrsistent

- Immutables

  https://github.com/MagicStack/immutables

FUTURE WORK

Real-time collaborative editor

Innovate beyond set boundaries. Relentless pursuit of quality products.
NetEase Games is hiring in multiple locations around the world,
including the US, UK, Canada, Singapore and Japan.

Join us!

China: https://hr.game.163.com/recruit.html

Overseas: https://www.neteasegames.com/careers

Twitter Facebook Youtube

#GDC22
#NetEase Games

THANKS

@kkpattern

https://github.com/kkpattern/immu_editor_gdc