PIC 16, Spring 2018

MiniLecture 5M: `PyQt5` in Five Slides

Monday, April 30, 2018

Matt Haberland

Activities

- Finish 4W
- Start 5M (required for all students)

Intended Learning Outcomes

By the end of the assignment, students should be able to:

- create a simple GUI window and modify its size, location on the screen, and title;
- specify absolute positions and sizes for “widgets”, including buttons, in a GUI;
- make a program respond to a button click;
- outline and fill shapes on a widget, prescribing the color, style, and/or pattern, and
- use a QTimer to run a function at a prescribed frequency.

PyQt5

- Three important submodules:
  - `QtWidgets`: `QApplication`, `QWidget`, `QPushButton`, etc…
  - `QtGui`: `QPainter`, `QColor`
  - `QtCore`: `QTimer`
- To make a window appear:
  - Instantiate a `QApplication`
  - Instantiate a `QWidget` (or subclass)
  - `show()` the `QWidget`
  - `exec_()` the `QApplication`

To ensure that Qt behaves, should be in a function (e.g. `main`)

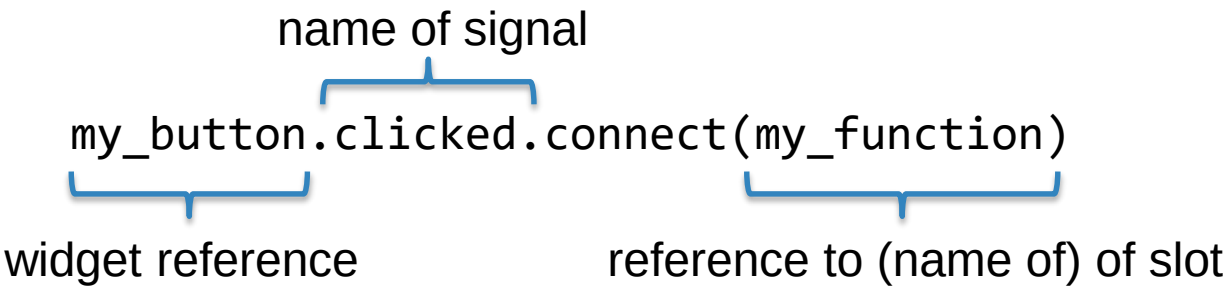
- Usually you'll subclass `QWidget`
  - Remember to initialize the `QWidget`, too!

in `__init__`: `super(MyWidget, self).__init__()`

name of subclass — reference to your subclass object

User Input

- Method 1: Signals and Slots
  - When a user interacts with a widget, a “signal” is generated
  - For your GUI to respond, you must connect this signal to a “slot”, a function or method to be called when the signal is generated



    - widget and slot references are often instance variables of a `QWidget` subclass object (so don’t forget `self.`)
    - slot reference is a *reference*; don’t *invoke* the function (no `()`)
- Method 2: Events
  - When a user interacts with a widget, an event occurs
  - For your GUI to respond, you must override the appropriate method (always has `event` in the name)

Timers

- Between instantiating and `exec_`uting a `QApplication`, create a `QTimer`
- Hold onto the `QTimer`'s reference (often as an instance variable of your custom `QWidget` subclass)
- Connect the `QTimer`'s `timeout` signal to a slot
- start the `QTimer`, passing in the period as a reference

Painting

- In the widget's `paintEvent` method:
  - Create a `QPainter`
  - `begin` the `QPainter`
  - Use the `QPainter` (often in a separate method)
    - Set the pen/brush color
    - Call appropriate `draw` method (e.g. `drawRectange`, `drawEllipse`)
  - end the `QPainter`

Tips for GUIs

- Memorize a few template code snippets
  - Showing a GUI
  - Getting a timer to run a method
  - Painting on a widget
  - Making a button do something

However, it's not all memorization. There are a few patterns we typically follow, but you need to understand how they're working to build on them.

- Look out for commonalities
  - Methods inherited from superclasses (`move`, `setGeometry`, etc…)
  - Almost all widget initializers require a parent widget
  - Coordinates are always from top-left
- Programs will crash without errors. Learn to debug.
- Search online for the widgets you need; search widget (and superclass) documentation for methods, signals, events

… and most importantly, **practice**