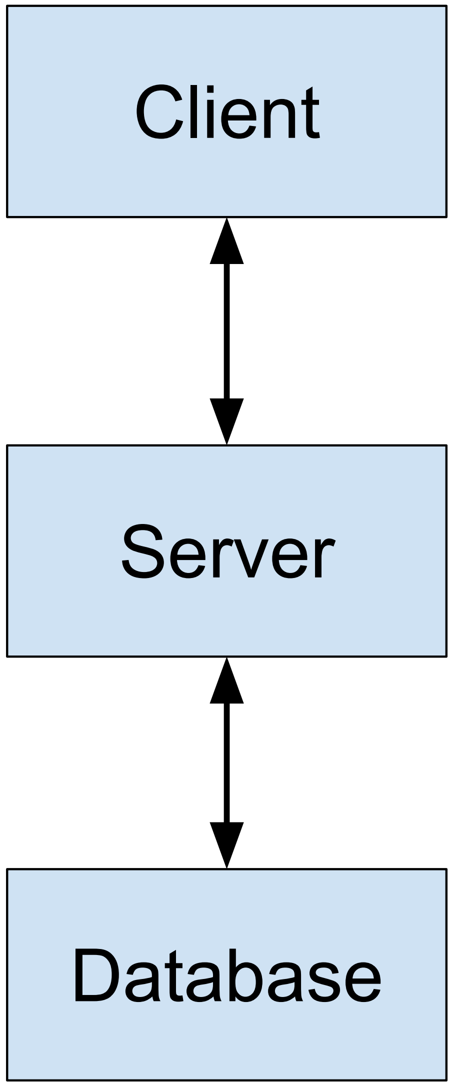
Client-Side Options (Part 1)

Objectives



Python
Browser/HTML/JavaScript (jQuery, React)
Desktop apps (PyQt5)

blue => course default
red => other option

Python
Python/Flask/Jinja2
Java/Servlets
Java/Spring/Mustache
JavaScript/Express/Mustache

SQLite
PostgreSQL

Objectives

- We will cover:
  - Desktop app programming
  - A Penny **desktop** client

Objectives

- More specifically…
- We will cover:
  - “High-level” desktop app programming
- We will not cover:
  - “Low-level” desktop app programming

Motivation

- **Question**: Why study desktop app programming?
- **Answers**:
  - Mainstream “advanced programming”
  - Illustrates event-driven programming
  - Illustrates concurrent programming
  - Important!

Agenda

- **Qt and PyQt5**
- PennyJson server
- Penny desktop client
  - Version 1: Baseline
  - Version 2: Sequential
  - Version 3: Bad
  - Version 4: Multi-threaded
  - Version 5: Stop
  - Version 6: Debouncing

Qt and PyQt5

Cross-platform desktop app libraries:

Desktop App Library	Language	Platform
GTK	C, C++	Linux, Windows, macOS
Qt	**C, C++**	**Linux, Windows, macOS**
wxWidgets	C, C++	Linux, Windows, macOS
Flutter	C, C++, Dart	Linux, WIndows, macOS, Android, iOS
Kivy	Python	Linux, WIndows, macOS, Android, iOS
Swing	Java	Linux, Windows, macOS
Tck/Tk	Tcl	Linux, Windows, macOS, Android, iOS
…	…	…
PyQt	**Python**	**Linux, Windows, macOS**

Qt and PyQt5

- ***Qt***
  - Haavard Nord and Eirik Chambe-Eng
  - Trollteck, Nokia, The Qt Company

Qt and PyQt5

- **Question**: Why study Qt?
  - Instead of some other GUI library?
- **Answers**:
  - Well designed, stable, robust
  - Excellent reputation
  - Very popular

Qt and PyQt5

- ***PyQt5***
  - Phil Thompson
  - Riverbank Computing
  - A “Python binding” of Qt

Qt and PyQt5

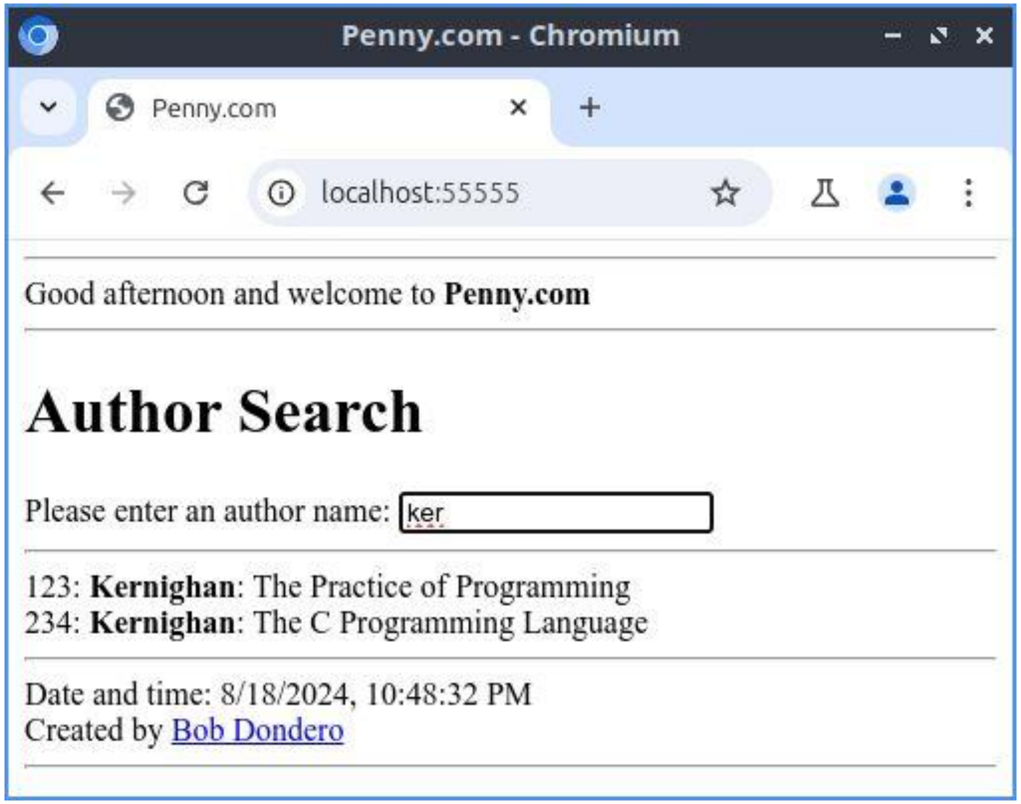
- **Question**: Why study PyQt5?
  - Instead of some other GUI library?
- **Answers**:
  - Convenient way to learn the fundamentals of Qt!

Agenda

- Qt and PyQt5
- **PennyJson server**
- Penny desktop client
  - Version 1: Baseline
  - Version 2: Sequential
  - Version 3: Bad
  - Version 4: Multi-threaded
  - Version 5: Stop
  - Version 6: Debouncing

PennyJson Server

- See **<u>PennyJson</u>** app
  - (Almost) same as **PennyAjax4** app from previous lecture



Good afternoon and welcome to **Penny.com**

Author Search

Please enter an author name: ker

123: **Kernighan**: The Practice of Programming
234: **Kernighan**: The C Programming Language

Date and time: 8/18/2024, 10:48:32 PM
Created by Bob Dondero

PennyJson Server

- See **PennyJson** app (cont.)
  - runserver.py
  - penny.sql, penny.sqlilte
  - database.py
  - index.html
    - Irrelevant for this lecture
  - **penny.py**

PennyJson/penny.py (Page 1 of 1)

```python
#!/usr/bin/env python

#-----------------------------------------------------------------------
# penny.py
# Author: Bob Dondero
#-----------------------------------------------------------------------

import os
import time
import json
import flask
import dotenv
import database

dotenv.load_dotenv()
_IO_DELAY = int(os.getenv('IO_DELAY', '0'))

#-----------------------------------------------------------------------

app = flask.Flask(__name__)

#-----------------------------------------------------------------------

@app.route('/', methods=['GET'])
@app.route('/index', methods=['GET'])
def index():

    return flask.send_file('index.html')

#-----------------------------------------------------------------------

@app.route('/searchresults', methods=['GET'])
def search_results():

    author = flask.request.args.get('author')
    if author is None:
        author = ''
    author = author.strip()

    # Simulate a slow database.
    time.sleep(_IO_DELAY)

    if author == '':
        books = []
    else:
        books = database.get_books(author) # Exception handling omitted

    json_doc = json.dumps(books)
    response = flask.make_response(json_doc)
    response.headers['Content-Type'] = 'application/json'
    return response
```

blank (Page 1 of 1)

This page is intentionally blank.

PennyJson Server

- See **PennyJson** app (cont.)
  - Relevant observations
    - Added `IO_DELAY` environment variable
      - Simulates slow DB
      - Shows how client performs with I/O bound server

Agenda

- Qt and PyQt5
- PennyJson server
- **Penny desktop client**
  - **Version 1: Baseline**
  - Version 2: Sequential
  - Version 3: Bad
  - Version 4: Multi-threaded
  - Version 5: Stop
  - Version 6: Debouncing

Penny Desktop v1

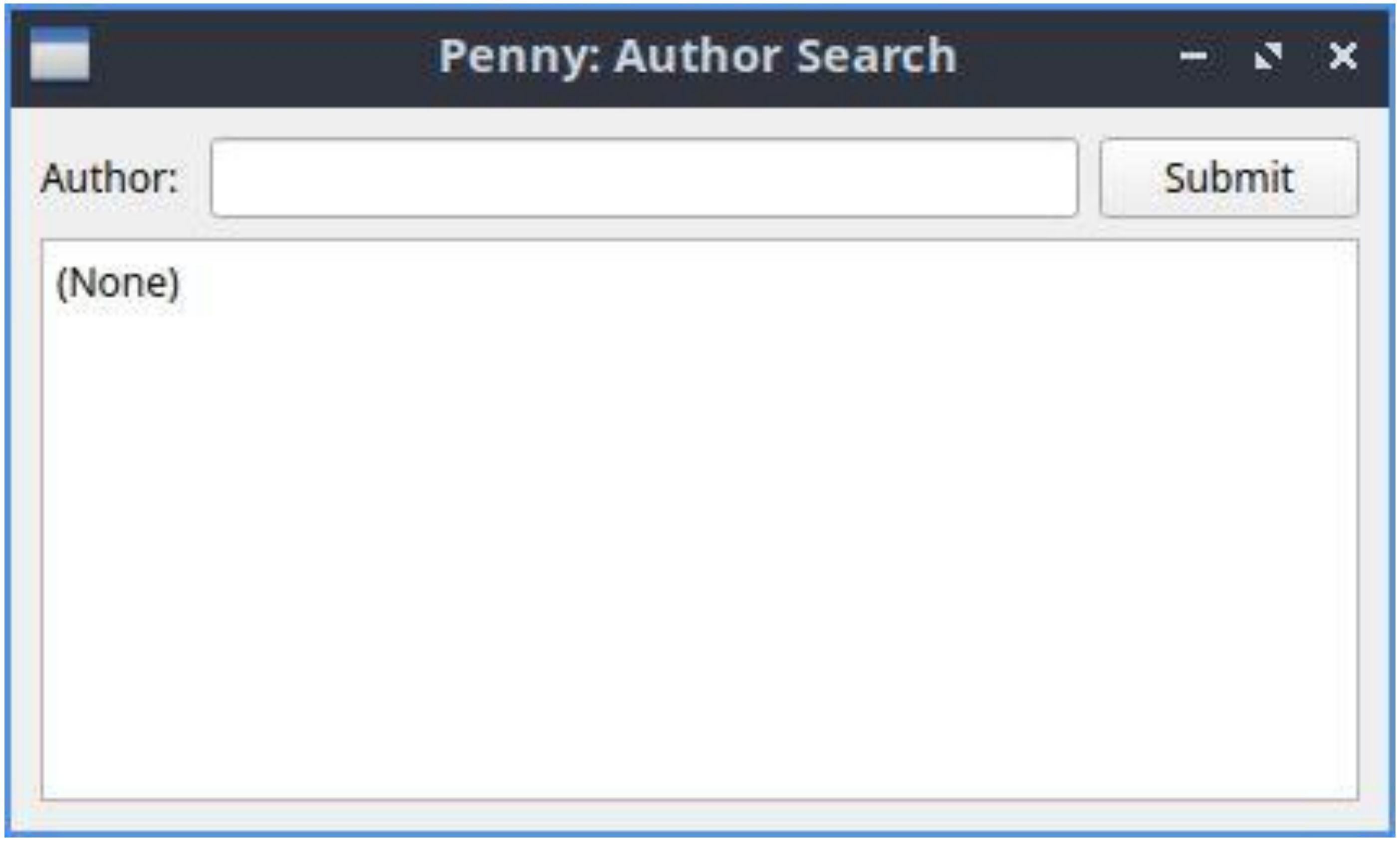
- Penny desktop v1

```
$ export IO_DELAY=1
$ python runserver.py 55555
```

```
$ python pennydesktop1base.py https://localhost:55555
```

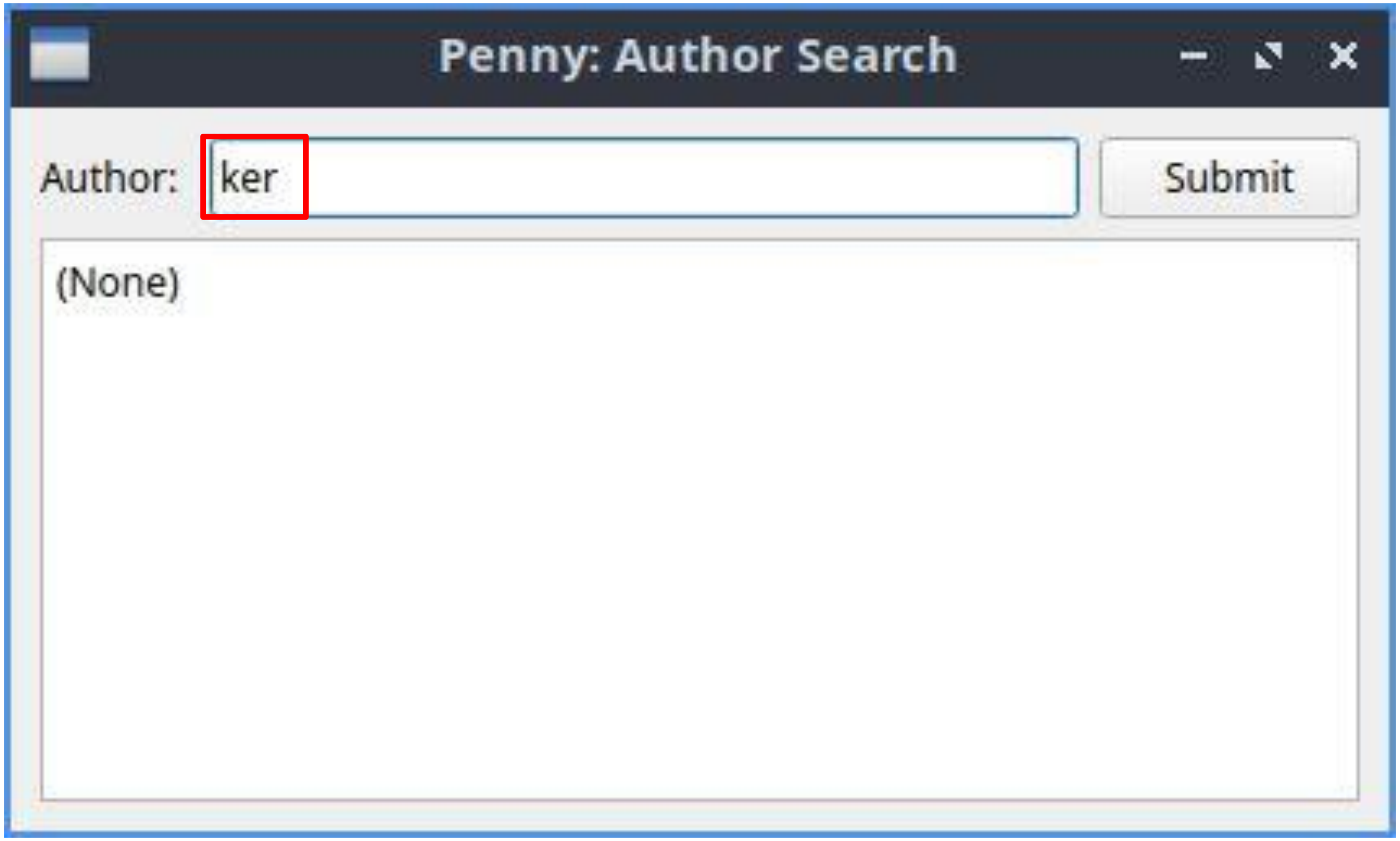
Penny Desktop v1

- Penny desktop v1



Penny Desktop v1

- Penny desktop v1

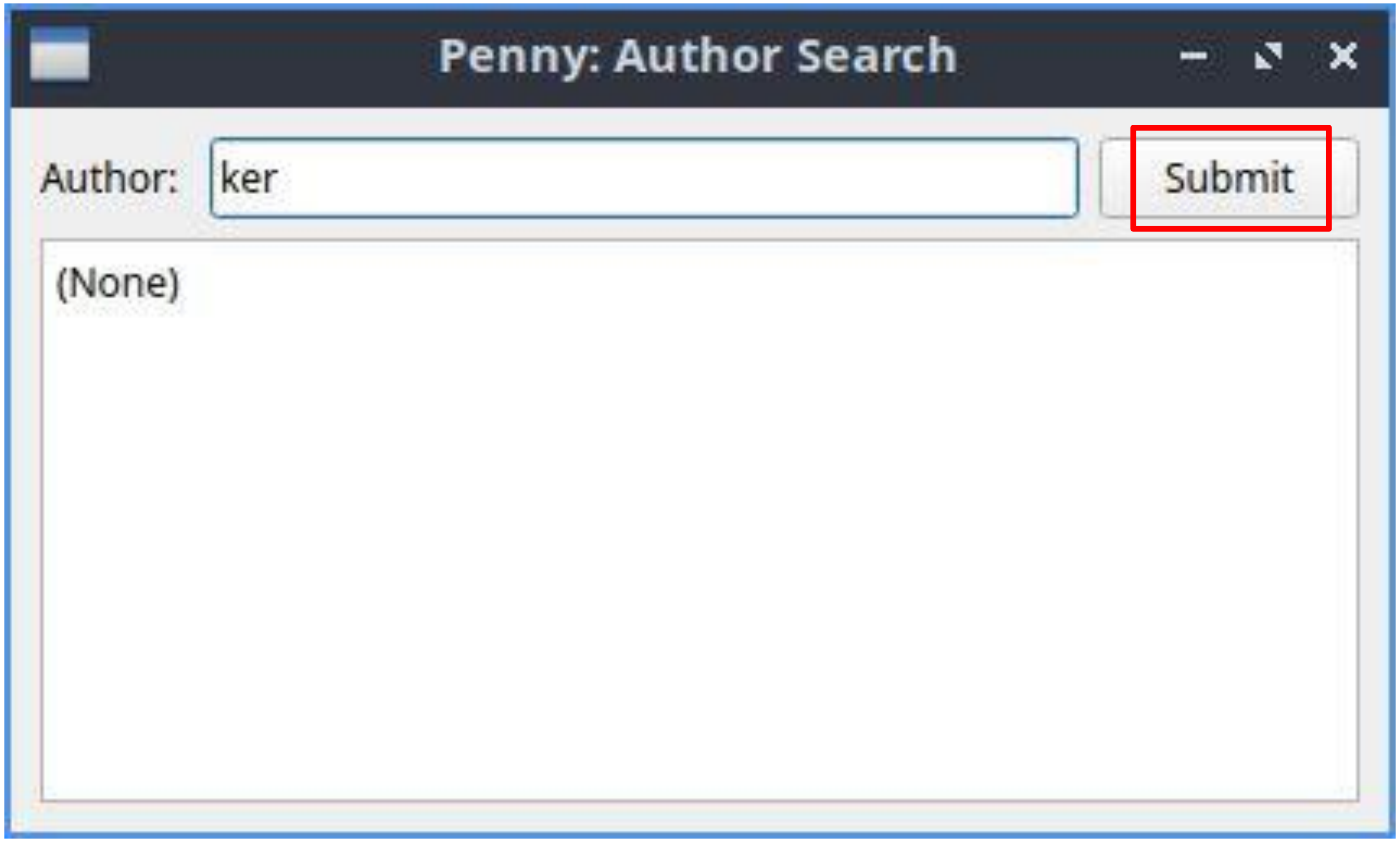


Author: ker

(None)

Penny Desktop v1

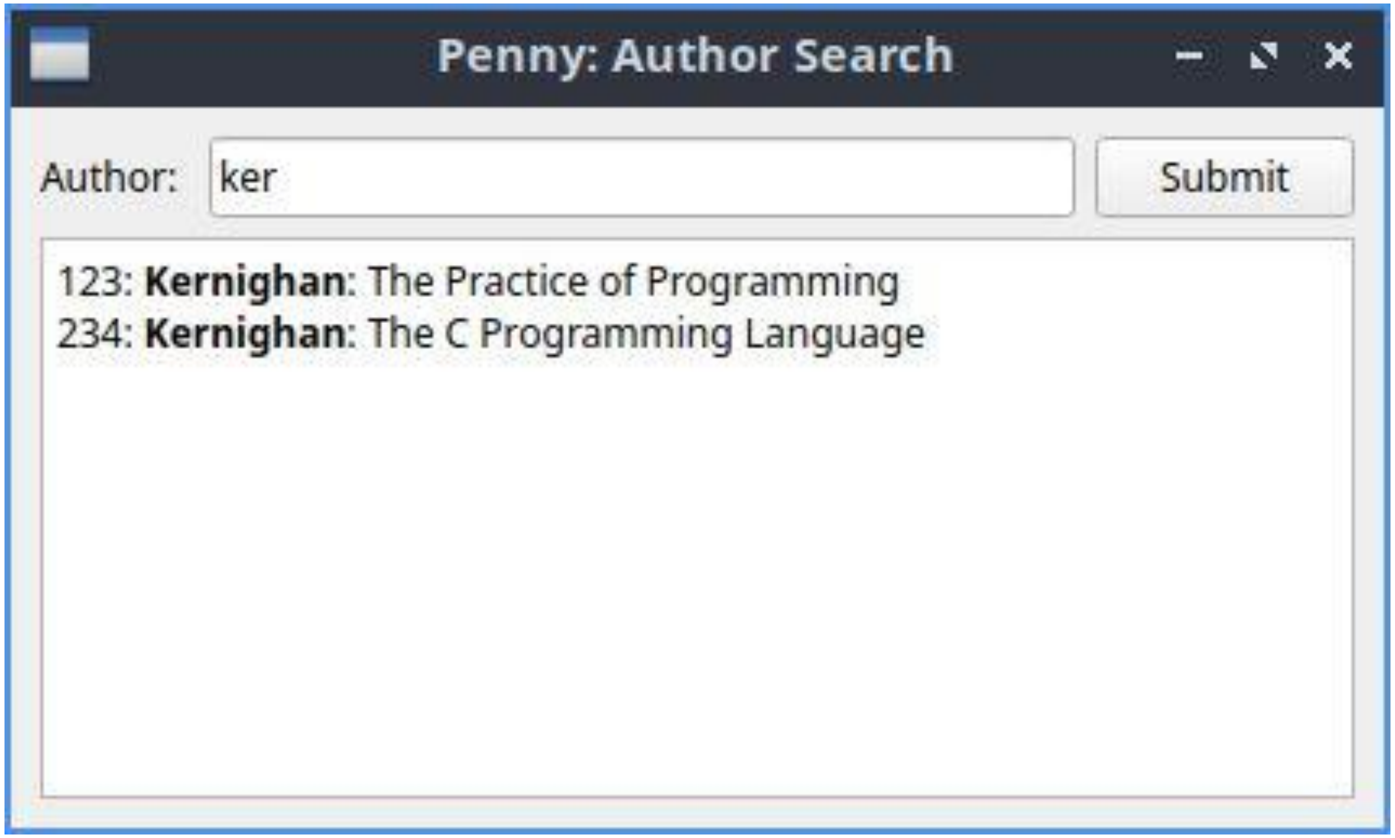
- Penny desktop v1



(None)

Penny Desktop v1

- Penny desktop v1



ker

```
123: Kernighan: The Practice of Programming
234: Kernighan: The C Programming Language
```

After a 1 second delay

Penny Desktop v1

- Penny desktop v1 (cont.)
  - See **pennydesktop1base.py**

PennyPyqt/pennydesktop1base.py (Page 1 of 2)

```python
#!/usr/bin/env python

#-----------------------------------------------------------------------
# pennydesktop1base.py
# Author: Bob Dondero
#-----------------------------------------------------------------------

import sys
import urllib.parse
import urllib.request
import json
import PyQt5.QtWidgets

#-----------------------------------------------------------------------

def create_widgets():

    author_label = PyQt5.QtWidgets.QLabel('Author: ')
    author_lineedit = PyQt5.QtWidgets.QLineEdit()
    submit_button = PyQt5.QtWidgets.QPushButton('Submit')
    books_textedit = PyQt5.QtWidgets.QTextEdit()
    books_textedit.setReadOnly(True)

    layout = PyQt5.QtWidgets.QGridLayout()
    layout.addWidget(author_label, 0, 0)
    layout.addWidget(author_lineedit, 0, 1)
    layout.addWidget(submit_button, 0, 2)
    layout.addWidget(books_textedit, 1, 0, 1, 3)
    layout.setRowStretch(0, 0)
    layout.setRowStretch(1, 1)
    layout.setColumnStretch(0, 0)
    layout.setColumnStretch(1, 1)
    layout.setColumnStretch(2, 0)

    frame = PyQt5.QtWidgets.QFrame()
    frame.setLayout(layout)

    window = PyQt5.QtWidgets.QMainWindow()
    window.setWindowTitle('Penny: Author Search')
    window.setCentralWidget(frame)
    screen_size = PyQt5.QtWidgets.QDesktopWidget().screenGeometry()
    window.resize(screen_size.width()//2, screen_size.height()//2)

    return (window, author_lineedit, submit_button, books_textedit)

#-----------------------------------------------------------------------

def author_slot_helper(server_url, author_lineedit, books_textedit):

    author = author_lineedit.text()
    encoded_author = urllib.parse.quote_plus(author)
    url = server_url + '/searchresults?author=' + encoded_author

    books_textedit.clear()

    try:
        with urllib.request.urlopen(url) as in_flo:
            response = in_flo.read()
            json_doc = response.decode('utf-8')
            books = json.loads(json_doc)

            if len(books) == 0:
                books_textedit.insertPlainText('(None)')
            else:
                pattern = '%s: <strong>%s</strong>: %s<br>'
```

PennyPyqt/pennydesktop1base.py (Page 2 of 2)

```python
                for book in books:
                    books_textedit.insertHtml(pattern %
                        (book['isbn'], book['author'], book['title']))

    except Exception as ex:
        books_textedit.insertPlainText(str(ex))

    books_textedit.repaint()  # Required on Mac.

#-----------------------------------------------------------------------

def main():

    if len(sys.argv) != 2:
        print('Usage: penny serverURL', file=sys.stderr)
        sys.exit(1)

    server_url = sys.argv[1]

    # Create and lay out the widgets.

    app = PyQt5.QtWidgets.QApplication(sys.argv)
    window, author_lineedit, submit_button, books_textedit = (
        create_widgets())

    # Handle signals.

    def author_slot():
        author_slot_helper(server_url, author_lineedit, books_textedit)
    submit_button.clicked.connect(author_slot)

    # Start up.

    window.show()
    author_slot()  # Populate books_textedit initially.
    sys.exit(app.exec_())

if __name__ == '__main__':
    main()
```

Penny Desktop v1

- Penny desktop v1
  - **(maybe) Problem**:
    - Inconsistent window state after typing author and before clicking Submit button
  - **Solution: redesign**…
    - Eliminate Submit button
    - GUI refreshes with each keystroke

Agenda

- Qt and PyQt5
- PennyJSON server
- **Penny desktop client**
  - Version 1: Baseline
  - **Version 2: Sequential**
  - Version 3: Bad
  - Version 4: Multi-threaded
  - Version 5: Stop
  - Version 6: Debouncing

Penny Desktop v2

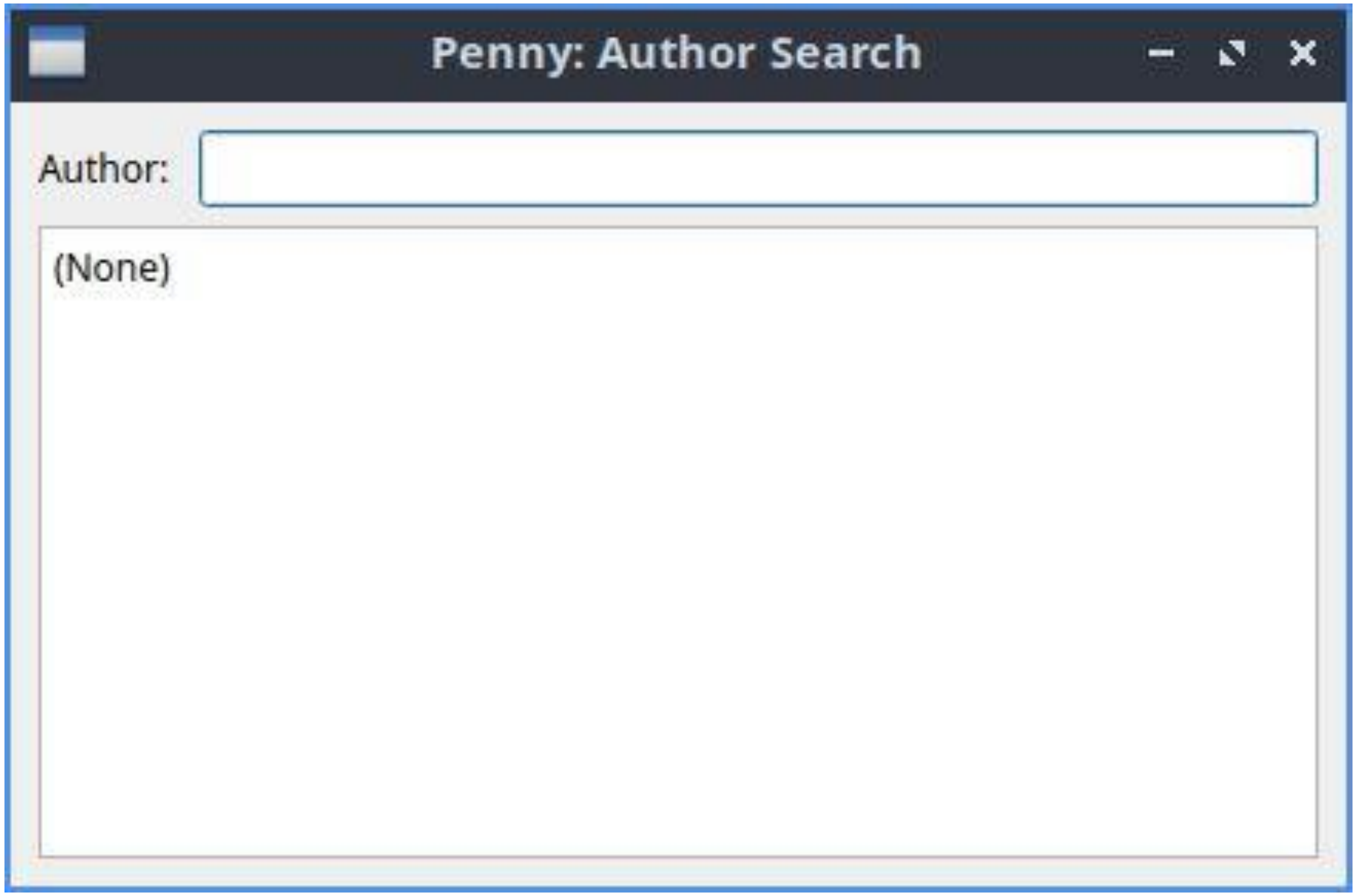
- Penny desktop v2

```
$ export IO_DELAY=1
$ python runserver.py 55555
```

```
$ python pennydesktop2seq.py https://localhost:55555
```

Penny Desktop v2

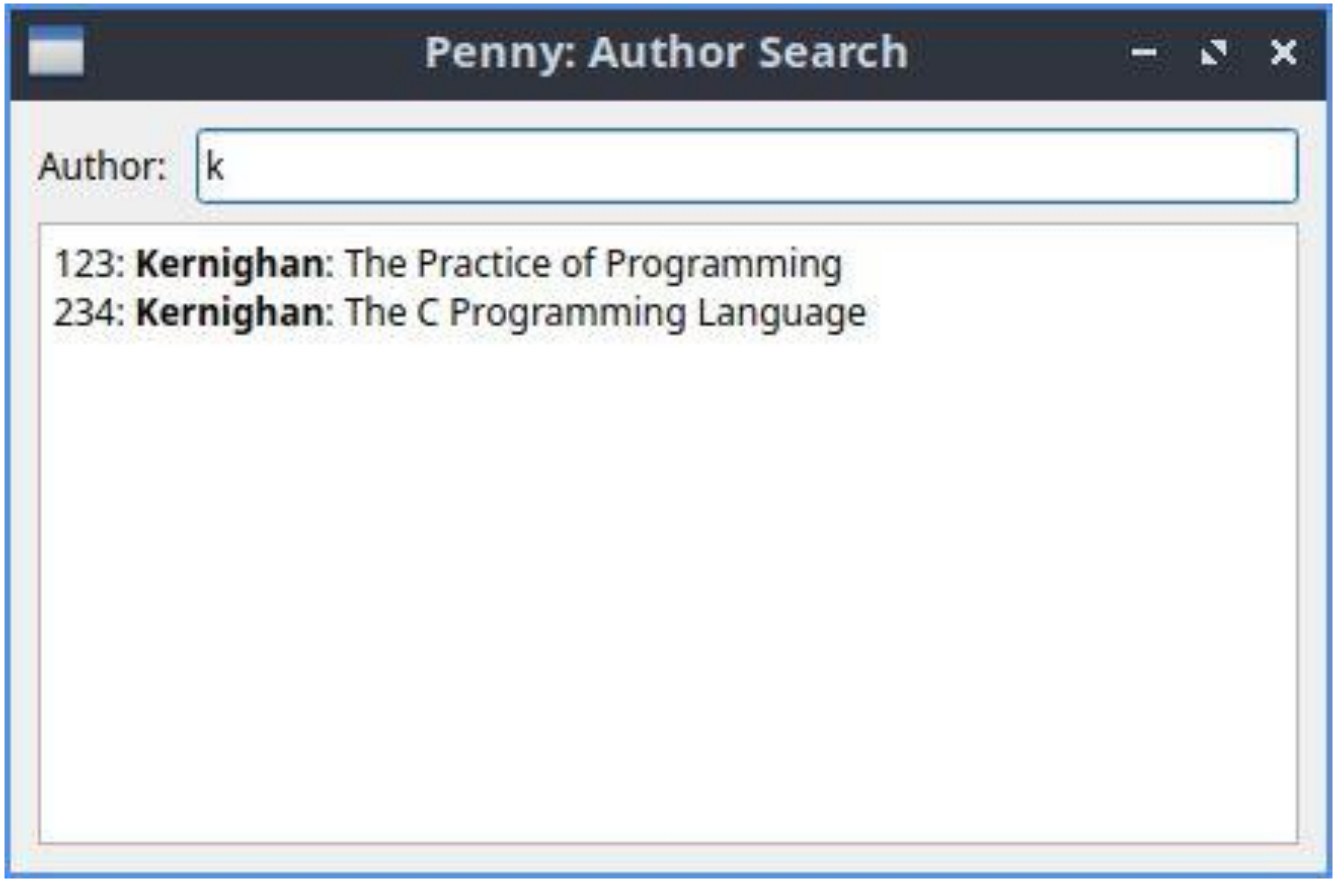
- Penny desktop v2 (cont.)



(None)

Penny Desktop v2

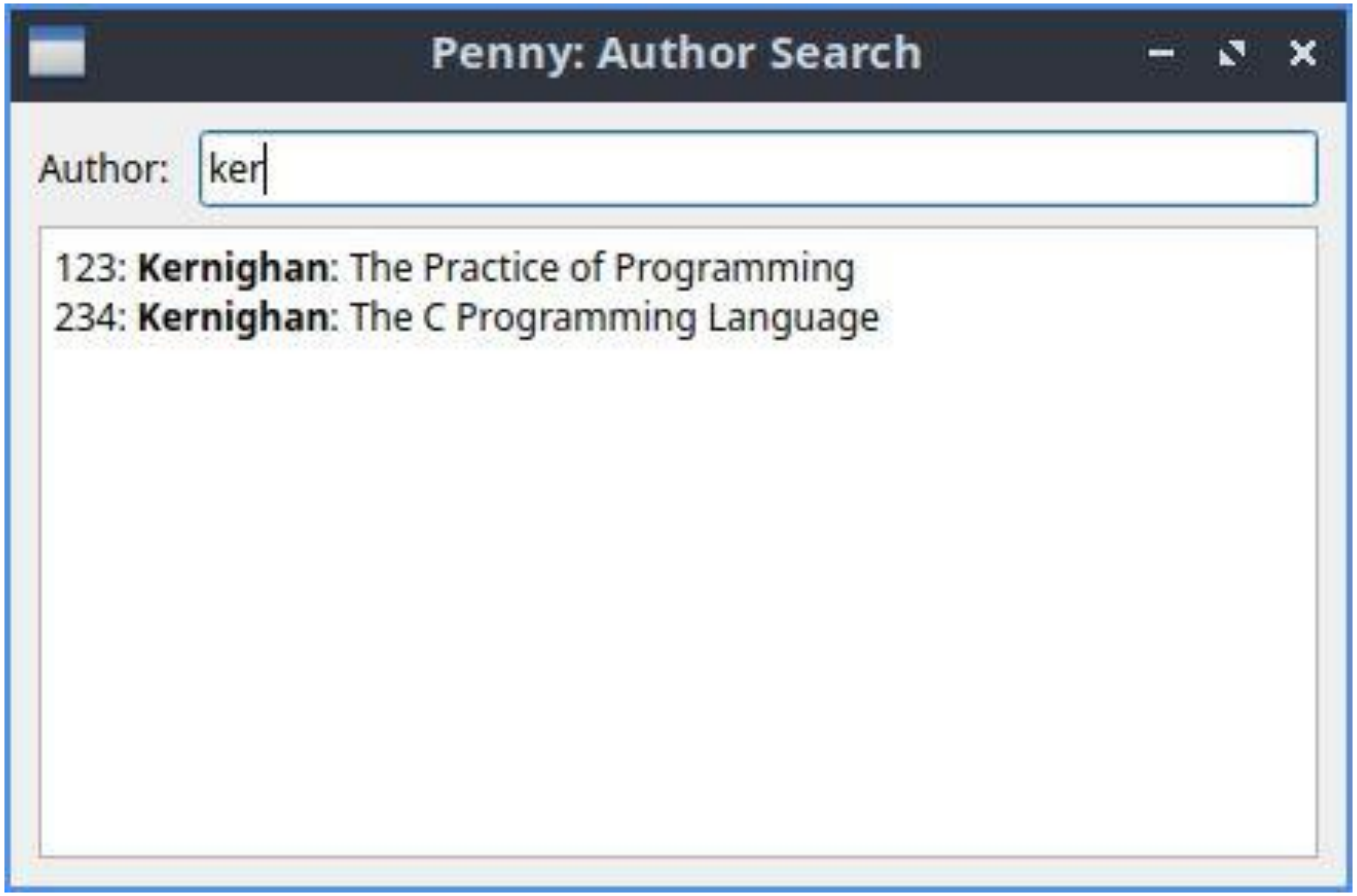
- Penny desktop v2 (cont.)



123: **Kernighan**: The Practice of Programming
234: **Kernighan**: The C Programming Language

Penny Desktop v2

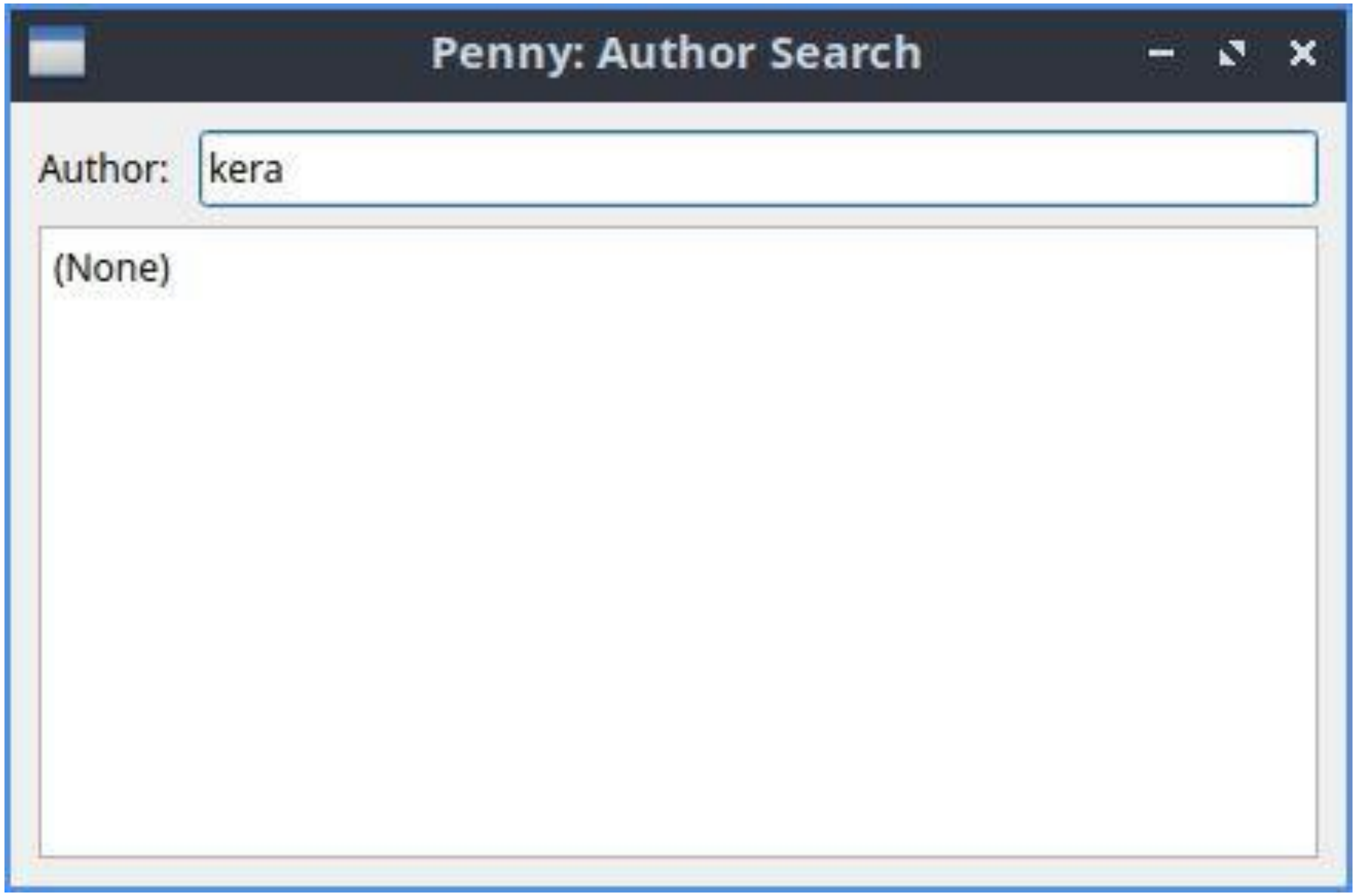
- Penny desktop v2 (cont.)



123: **Kernighan**: The Practice of Programming
234: **Kernighan**: The C Programming Language

Penny Desktop v2

- Penny desktop v2 (cont.)



Author: kera

(None)

Penny Desktop v2

- Penny desktop v2 (cont.)
  - See **<u>pennyclient2seq.py</u>**

Penny Desktop v2

- Penny desktop v2 (cont.)
  - See **pennyclient2seq.py**

PennyPyqt/pennydesktop2seq.py (Page 1 of 2)

```
#!/usr/bin/env python

#-----------------------------------------------------------------------
# pennydesktop1seq.py
# Author: Bob Dondero
#-----------------------------------------------------------------------

import sys
import urllib.parse
import urllib.request
import json
import PyQt5.QtWidgets

#-----------------------------------------------------------------------

def create_widgets():

    author_label = PyQt5.QtWidgets.QLabel('Author: ')
    author_lineedit = PyQt5.QtWidgets.QLineEdit()
    books_textedit = PyQt5.QtWidgets.QTextEdit()
    books_textedit.setReadOnly(True)

    layout = PyQt5.QtWidgets.QGridLayout()
    layout.addWidget(author_label, 0, 0)
    layout.addWidget(author_lineedit, 0, 1)
    layout.addWidget(books_textedit, 1, 0, 1, 2)
    layout.setRowStretch(0, 0)
    layout.setRowStretch(1, 1)
    layout.setColumnStretch(0, 0)
    layout.setColumnStretch(1, 1)
    layout.setColumnStretch(2, 0)

    frame = PyQt5.QtWidgets.QFrame()
    frame.setLayout(layout)

    window = PyQt5.QtWidgets.QMainWindow()
    window.setWindowTitle('Penny: Author Search')
    window.setCentralWidget(frame)
    screen_size = PyQt5.QtWidgets.QDesktopWidget().screenGeometry()
    window.resize(screen_size.width()//2, screen_size.height()//2)

    return (window, author_lineedit, books_textedit)

#-----------------------------------------------------------------------

def author_slot_helper(server_url, author_lineedit, books_textedit):

    author = author_lineedit.text()
    encoded_author = urllib.parse.quote_plus(author)
    url = server_url + '/searchresults?author=' + encoded_author

    books_textedit.clear()

    try:
        with urllib.request.urlopen(url) as in_flo:
            response = in_flo.read()
            json_doc = response.decode('utf-8')
            books = json.loads(json_doc)

            if len(books) == 0:
                books_textedit.insertPlainText('(None)')
            else:
                pattern = '%s: <strong>%s</strong>: %s<br>'
                for book in books:
                    books_textedit.insertHtml(pattern %
```

PennyPyqt/pennydesktop2seq.py (Page 2 of 2)

```
                        (book['isbn'], book['author'], book['title']))

    except Exception as ex:
        books_textedit.insertPlainText(str(ex))

    books_textedit.repaint()  # Required on Mac.

#-----------------------------------------------------------------------

def main():

    if len(sys.argv) != 2:
        print('Usage: penny serverURL', file=sys.stderr)
        sys.exit(1)

    server_url = sys.argv[1]

    # Create and lay out the widgets.

    app = PyQt5.QtWidgets.QApplication(sys.argv)
    window, author_lineedit, books_textedit = create_widgets()

    # Handle signals.

    def author_slot():
        author_slot_helper(server_url, author_lineedit, books_textedit)
    author_lineedit.textChanged.connect(author_slot)

    # Start up.

    window.show()
    author_slot()  # Populate books_textedit initially.
    sys.exit(app.exec_())

if __name__ == '__main__':
    main()
```

Penny Desktop v2

- Penny desktop v2 (cont.)
  - **Problem:**
    - Serious GUI lag
  - **Solution:** redesign…
    - Use threads!!!

Agenda

- Qt and PyQt5
- PennyJson server
- **Penny desktop client**
  - Version 1: Baseline
  - Version 2: Sequential
  - **Version 3: Bad**
  - Version 4: Multi-threaded
  - Version 5: Stop
  - Version 6: Debouncing

Penny Desktop v3

- Penny desktop v3
  - See **<u>pennyclient3bad.py</u>**

PennyPyqt/pennydesktop3bad.py (Page 1 of 2)

```python
#!/usr/bin/env python

#-----------------------------------------------------------------------
# pennydesktop3bad.py
# Author: Bob Dondero
#-----------------------------------------------------------------------

import sys
import threading
import urllib.parse
import urllib.request
import json
import PyQt5.QtWidgets

#-----------------------------------------------------------------------

def create_widgets():

    author_label = PyQt5.QtWidgets.QLabel('Author: ')
    author_lineedit = PyQt5.QtWidgets.QLineEdit()
    books_textedit = PyQt5.QtWidgets.QTextEdit()
    books_textedit.setReadOnly(True)

    layout = PyQt5.QtWidgets.QGridLayout()
    layout.addWidget(author_label, 0, 0)
    layout.addWidget(author_lineedit, 0, 1)
    layout.addWidget(books_textedit, 1, 0, 1, 2)
    layout.setRowStretch(0, 0)
    layout.setRowStretch(1, 1)
    layout.setColumnStretch(0, 0)
    layout.setColumnStretch(1, 1)
    layout.setColumnStretch(2, 0)

    frame = PyQt5.QtWidgets.QFrame()
    frame.setLayout(layout)

    window = PyQt5.QtWidgets.QMainWindow()
    window.setWindowTitle('Penny: Author Search')
    window.setCentralWidget(frame)
    screen_size = PyQt5.QtWidgets.QDesktopWidget().screenGeometry()
    window.resize(screen_size.width()//2, screen_size.height()//2)

    return (window, author_lineedit, books_textedit)

#-----------------------------------------------------------------------

class WorkerThread (threading.Thread):

    def __init__(self, server_url, author, books_textedit):
        threading.Thread.__init__(self)
        self._server_url = server_url
        self._author = author
        self._books_textedit = books_textedit

    def run(self):
        encoded_author = urllib.parse.quote_plus(self._author)
        url = self._server_url
        url += '/searchresults?author=' + encoded_author

        self._books_textedit.clear()

        try:
            with urllib.request.urlopen(url) as in_flo:
                response = in_flo.read()
                json_doc = response.decode('utf-8')
```

PennyPyqt/pennydesktop3bad.py (Page 2 of 2)

```python
                books = json.loads(json_doc)

                if len(books) == 0:
                    self._books_textedit.insertPlainText('(None)')
                else:
                    pattern = '%s: <strong>%s</strong>: %s<br>'
                    for book in books:
                        self._books_textedit.insertHtml(pattern %
                            (book['isbn'], book['author'],
                             book['title']))

        except Exception as ex:
            self._books_textedit.insertPlainText(str(ex))

        self._books_textedit.repaint()  # Required on Mac.

#-----------------------------------------------------------------------

def main():

    if len(sys.argv) != 2:
        print('Usage: penny serverURL', file=sys.stderr)
        sys.exit(1)

    server_url = sys.argv[1]

    # Create and lay out the widgets.

    app = PyQt5.QtWidgets.QApplication(sys.argv)
    window, author_lineedit, books_textedit = create_widgets()

    # Handle signals.

    def author_slot():
        author = author_lineedit.text()
        worker_thread = WorkerThread(server_url, author, books_textedit)
        worker_thread.start()
    author_lineedit.textChanged.connect(author_slot)

    # Start up.

    window.show()
    author_slot()  # Populate books_textedit initially.
    sys.exit(app.exec_())

if __name__ == '__main__':
    main()
```

Penny Desktop v3

- Penny desktop v3 (cont.)

```
$ export IO_DELAY=1
$ python runserver.py 55555
```

```
$ python pennydesktop3bad.py http://localhost:55555
QObject: Cannot create children for a parent that is
in a different thread.
(Parent is QTextDocument(0x25db0c0), parent's thread
is QThread(0x2213e40), current thread is
QThread(0x79c7cc000cd0)
Segmentation fault (core dumped)
$
```

Penny Desktop v3

- **Problem**:
  - PyQt5 widgets are not thread safe
  - So PyQt5 prohibits worker thread from updating widgets
- **Solution**: redesign...
  - Worker thread communicates book list to main thread
  - Main thread updates GUI

Agenda

- Qt and PyQt5
- PennyJson server
- **Penny desktop client**
  - Version 1: Baseline
  - Version 2: Sequential
  - Version 3: Bad
  - **Version 4: Multi-threaded**
  - Version 5: Stop
  - Version 6: Debouncing

Penny Desktop v4

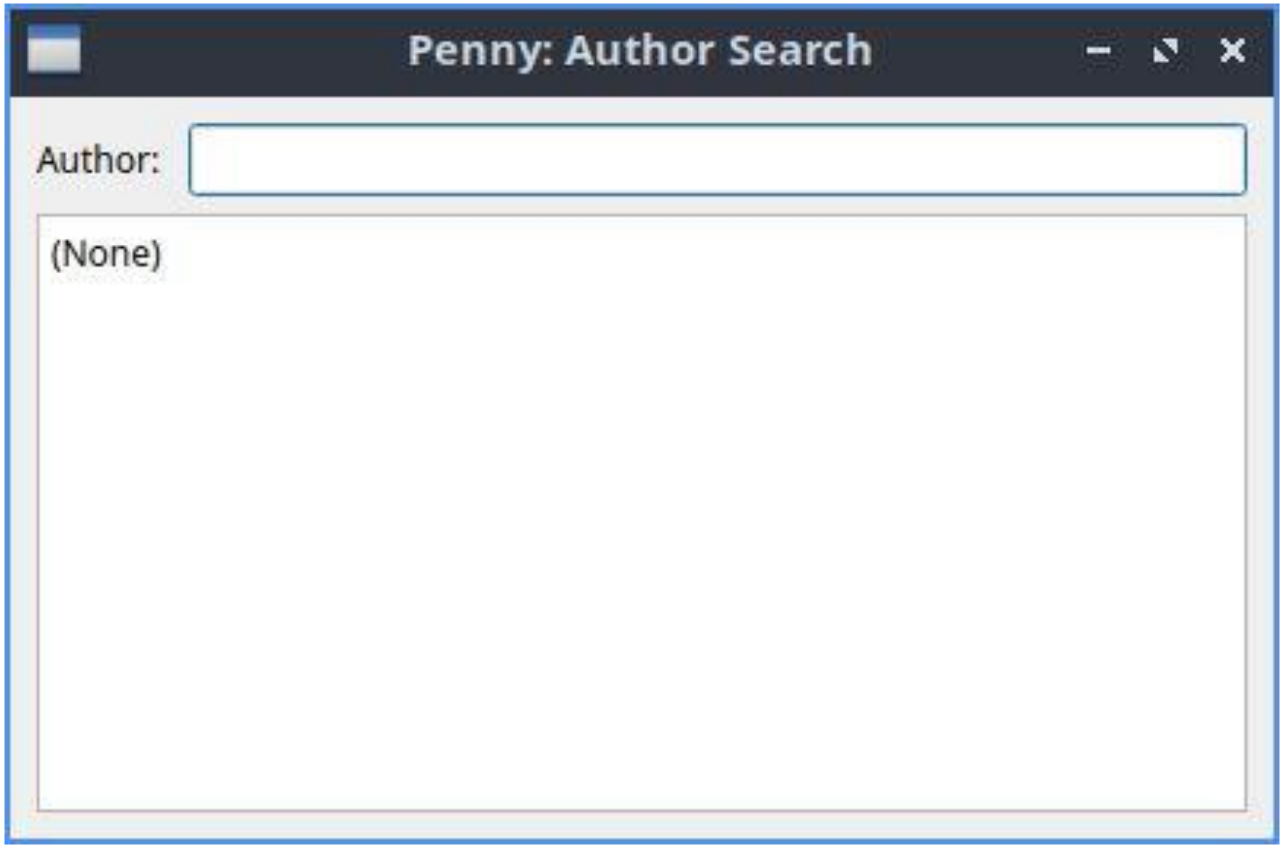
- Penny desktop v4

```
$ export IO_DELAY=1
$ python runserver.py 55555
```

```
$ python pennydesktop4threads.py https://localhost:55555
```

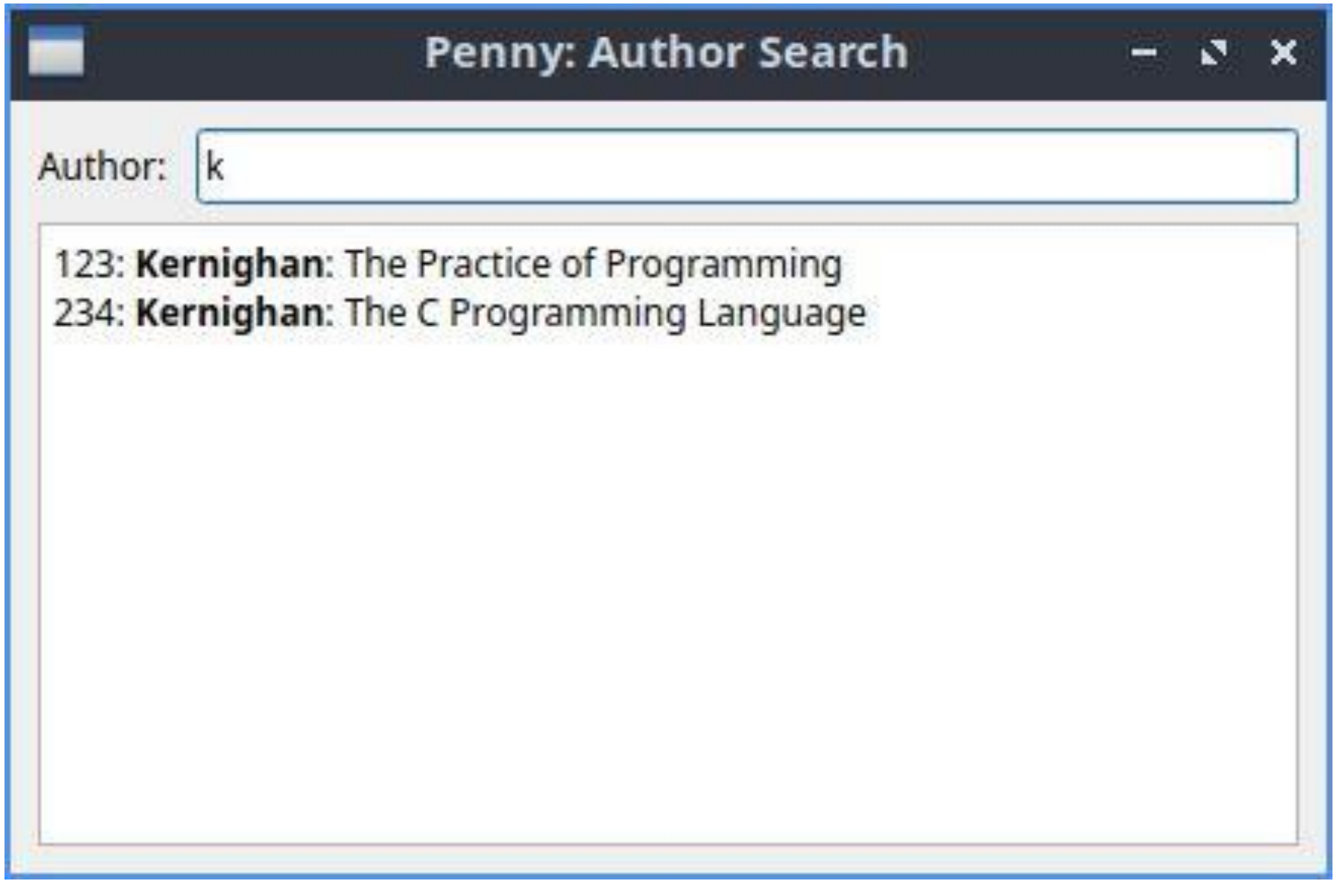
Penny Desktop v4

- Penny desktop v4 (cont.)



Penny Desktop v4

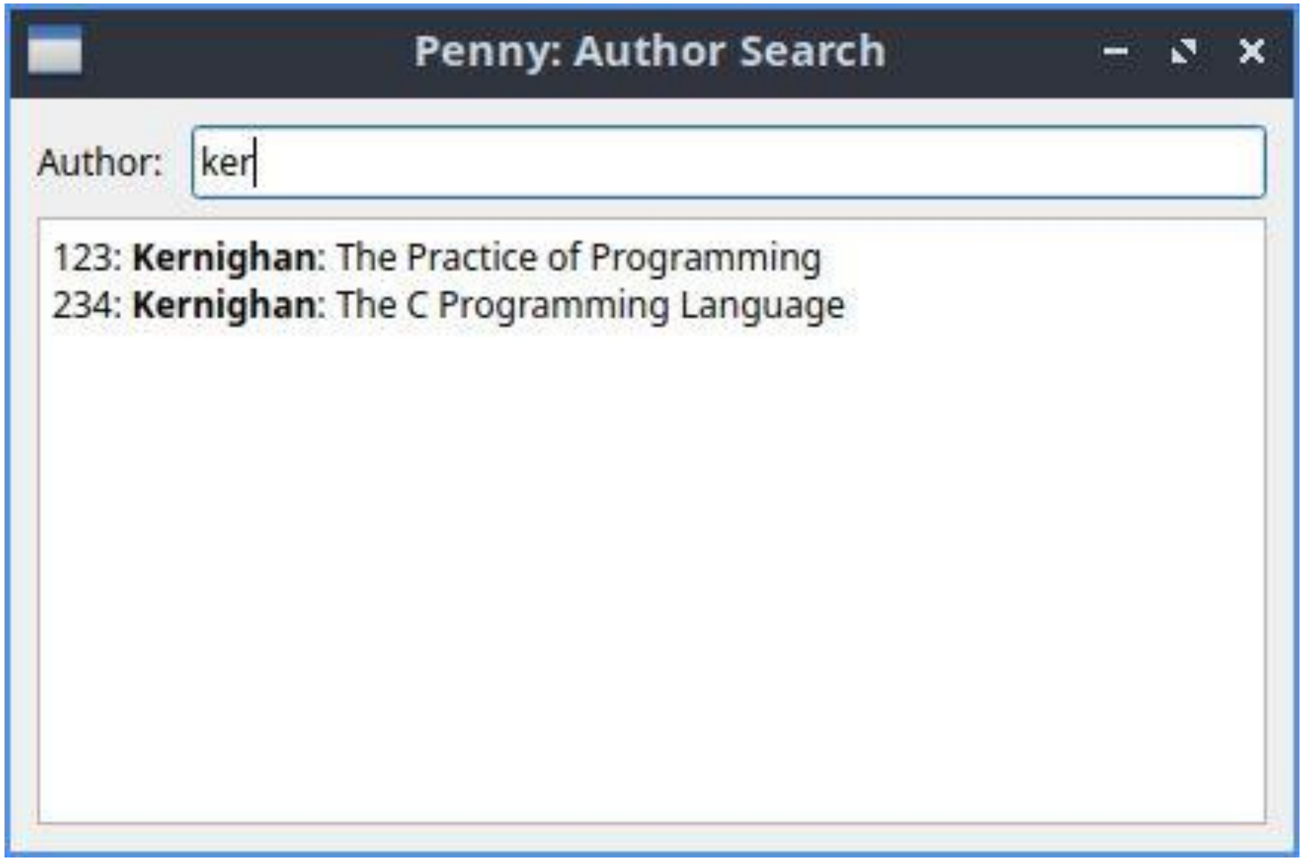
- Penny desktop v4 (cont.)



123: **Kernighan**: The Practice of Programming
234: **Kernighan**: The C Programming Language

Penny Desktop v4

- Penny desktop v4 (cont.)

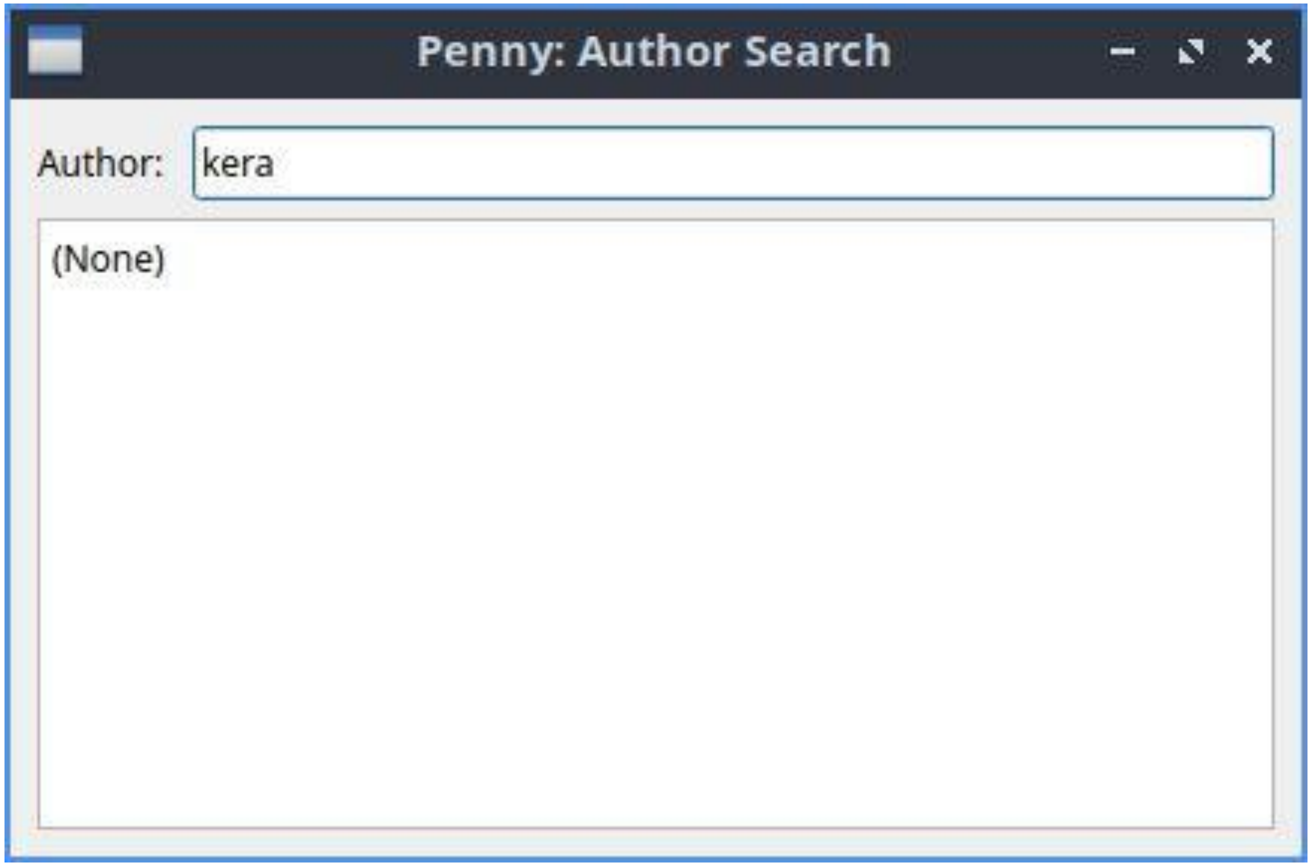


ker

123: **Kernighan**: The Practice of Programming
234: **Kernighan**: The C Programming Language

Penny Desktop v4

- Penny desktop v4 (cont.)



kera

(None)

Penny Desktop v4

- Penny desktop v4 (cont.)
  - See **pennydesktop4threads.py**

PennyPyqt/pennydesktop4threads.py (Page 1 of 2)

```python
#!/usr/bin/env python

#-----------------------------------------------------------------------
# pennydesktop4threads.py
# Author: Bob Dondero
#-----------------------------------------------------------------------

import sys
import threading
import urllib.parse
import urllib.request
import json
import queue
import PyQt5.QtWidgets
import PyQt5.QtCore

#-----------------------------------------------------------------------

def create_widgets():

    author_label = PyQt5.QtWidgets.QLabel('Author: ')
    author_lineedit = PyQt5.QtWidgets.QLineEdit()
    books_textedit = PyQt5.QtWidgets.QTextEdit()
    books_textedit.setReadOnly(True)

    layout = PyQt5.QtWidgets.QGridLayout()
    layout.addWidget(author_label, 0, 0)
    layout.addWidget(author_lineedit, 0, 1)
    layout.addWidget(books_textedit, 1, 0, 1, 2)
    layout.setRowStretch(0, 0)
    layout.setRowStretch(1, 1)
    layout.setColumnStretch(0, 0)
    layout.setColumnStretch(1, 1)
    layout.setColumnStretch(2, 0)

    frame = PyQt5.QtWidgets.QFrame()
    frame.setLayout(layout)

    window = PyQt5.QtWidgets.QMainWindow()
    window.setWindowTitle('Penny: Author Search')
    window.setCentralWidget(frame)
    screen_size = PyQt5.QtWidgets.QDesktopWidget().screenGeometry()
    window.resize(screen_size.width()//2, screen_size.height()//2)

    return (window, author_lineedit, books_textedit)

#-----------------------------------------------------------------------

class WorkerThread (PyQt5.QtCore.QThread):

    _response_signal = PyQt5.QtCore.pyqtSignal(bool, object)

    def __init__(self, server_url, author, handle_response):
        super().__init__()
        self._server_url = server_url
        self._author = author
        self._response_signal.connect(handle_response)

    def run(self):
        encoded_author = urllib.parse.quote_plus(self._author)
        url = self._server_url
        url += '/searchresults?author=' + encoded_author
        try:
            with urllib.request.urlopen(url) as in_flo:
                response = in_flo.read()
```

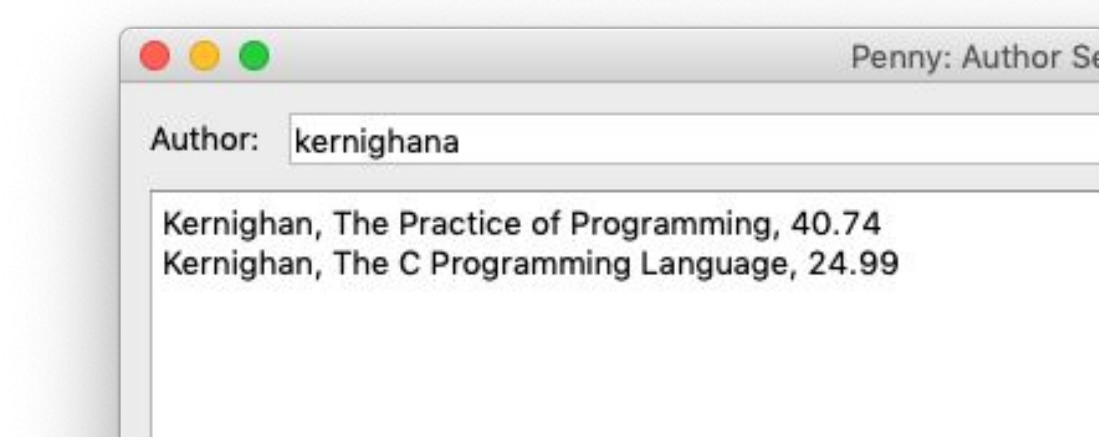
PennyPyqt/pennydesktop4threads.py (Page 2 of 2)

```python
                json_doc = response.decode('utf-8')
                books = json.loads(json_doc)
            self._response_signal.emit(True, books)
        except Exception as ex:
            self._response_signal.emit(False, str(ex))

#-----------------------------------------------------------------------

def main():

    if len(sys.argv) != 2:
        print('Usage: penny serverURL', file=sys.stderr)
        sys.exit(1)

    server_url = sys.argv[1]

    # Create and lay out the widgets.

    app = PyQt5.QtWidgets.QApplication(sys.argv)
    window, author_lineedit, books_textedit = create_widgets()

    # Handle signals.

    def handle_response(successful, data):
        books_textedit.clear()
        if successful:
            books = data
            if len(books) == 0:
                books_textedit.insertPlainText('(None)')
            else:
                pattern = '%s: <strong>%s</strong>: %s<br>'
                for book in books:
                    books_textedit.insertHtml(pattern %
                        (book['isbn'], book['author'], book['title']))
        else:
            ex = data
            books_textedit.insertPlainText(ex)
        books_textedit.repaint()

    worker_thread = None
    def author_slot():
        nonlocal worker_thread
        author = author_lineedit.text()
        worker_thread = WorkerThread(server_url, author,
            handle_response)
        worker_thread.start()

    author_lineedit.textChanged.connect(author_slot)

    # Start up.

    window.show()
    author_slot()  # Populate books_textedit initially.
    sys.exit(app.exec_())

if __name__ == '__main__':
    main()
```

Penny Desktop v4

- **Problem:**
  - Server will respond to requests in arbitrary order

old
screen
shot



kernighana

```
Kernighan, The Practice of Programming, 40.74
Kernighan, The C Programming Language, 24.99
```

- **Solution:**
  - Abort previous request

Agenda

- Qt and PyQt5
- PennyJson server
- **Penny desktop client**
  - Version 1: Baseline
  - Version 2: Sequential
  - Version 3: Bad
  - Version 4: Multi-threaded
  - **Version 5: Stop**
  - Version 6: Debouncing

Penny Desktop v5

- Penny desktop v5

```
$ export IO_DELAY=0
$ python runserver.py 55555
```

```
$ python pennydesktop5stop.py https://localhost:55555
```

[Same screen images]

Penny Desktop v5

- Penny desktop v5 (cont.)
  - See **<u>pennydesktop5stop.py</u>**

PennyPyqt/pennydesktop5stop.py (Page 1 of 2)

```python
#!/usr/bin/env python

#-----------------------------------------------------------------------
# pennydesktop5stop.py
# Author: Bob Dondero
#-----------------------------------------------------------------------

import sys
import threading
import urllib.parse
import urllib.request
import json
import queue
import PyQt5.QtWidgets
import PyQt5.QtCore

#-----------------------------------------------------------------------

def create_widgets():

    author_label = PyQt5.QtWidgets.QLabel('Author: ')
    author_lineedit = PyQt5.QtWidgets.QLineEdit()
    books_textedit = PyQt5.QtWidgets.QTextEdit()
    books_textedit.setReadOnly(True)

    layout = PyQt5.QtWidgets.QGridLayout()
    layout.addWidget(author_label, 0, 0)
    layout.addWidget(author_lineedit, 0, 1)
    layout.addWidget(books_textedit, 1, 0, 1, 2)
    layout.setRowStretch(0, 0)
    layout.setRowStretch(1, 1)
    layout.setColumnStretch(0, 0)
    layout.setColumnStretch(1, 1)
    layout.setColumnStretch(2, 0)

    frame = PyQt5.QtWidgets.QFrame()
    frame.setLayout(layout)

    window = PyQt5.QtWidgets.QMainWindow()
    window.setWindowTitle('Penny: Author Search')
    window.setCentralWidget(frame)
    screen_size = PyQt5.QtWidgets.QDesktopWidget().screenGeometry()
    window.resize(screen_size.width()//2, screen_size.height()//2)

    return (window, author_lineedit, books_textedit)

#-----------------------------------------------------------------------

class WorkerThread (PyQt5.QtCore.QThread):

    _response_signal = PyQt5.QtCore.pyqtSignal(bool, object)

    def __init__(self, server_url, author, handle_response):
        super().__init__()
        self._server_url = server_url
        self._author = author
        self._response_signal.connect(handle_response)
        self._should_stop = False

    def stop(self):
        self._should_stop = True

    def run(self):
        encoded_author = urllib.parse.quote_plus(self._author)
        url = self._server_url
```

PennyPyqt/pennydesktop5stop.py (Page 2 of 2)

```python
        url += '/searchresults?author=' + encoded_author
        try:
            with urllib.request.urlopen(url) as in_flo:
                response = in_flo.read()
                json_doc = response.decode('utf-8')
                books = json.loads(json_doc)
            if not self._should_stop:
                self._response_signal.emit(True, books)
        except Exception as ex:
            if not self._should_stop:
                self._response_signal.emit(False, str(ex))

#-----------------------------------------------------------------------

def main():

    if len(sys.argv) != 2:
        print('Usage: penny serverURL', file=sys.stderr)
        sys.exit(1)

    server_url = sys.argv[1]

    # Create and lay out the widgets.

    app = PyQt5.QtWidgets.QApplication(sys.argv)
    window, author_lineedit, books_textedit = create_widgets()

    # Handle signals.

    def handle_response(successful, data):
        books_textedit.clear()
        if successful:
            books = data
            if len(books) == 0:
                books_textedit.insertPlainText('(None)')
            else:
                pattern = '%s: <strong>%s</strong>: %s<br>'
                for book in books:
                    books_textedit.insertHtml(pattern %
                        (book['isbn'], book['author'], book['title']))
        else:
            ex = data
            books_textedit.insertPlainText(ex)
        books_textedit.repaint()

    worker_thread = None
    def author_slot():
        nonlocal worker_thread
        author = author_lineedit.text()
        if worker_thread is not None:
            worker_thread.stop()
        worker_thread = WorkerThread(server_url, author,
            handle_response)
        worker_thread.start()

    author_lineedit.textChanged.connect(author_slot)

    # Start up.

    window.show()
    author_slot()  # Populate books_textedit initially.
    sys.exit(app.exec_())

if __name__ == '__main__':
    main()
```

Penny Desktop v5

- **Problem**:
  - Server could be overwhelmed with requests
- **Solution:**
  - ***Debounce*** the requests
- **Bonus:**
  - Reduces (but does not eliminate) the need to abort old requests!

Agenda

- Qt and PyQt5
- PennyJson server
- **Penny desktop client**
  - Version 1: Baseline
  - Version 2: Sequential
  - Version 3: Bad
  - Version 4: Multi-threaded
  - Version 5: Stop
  - **Version 6: Debouncing**

Penny Desktop v6

- Penny desktop v6

```
$ export IO_DELAY=0
$ python runserver.py 55555
```

```
$ python pennydesktop6debounce.py https://localhost:55555
```

[Same screen images]

Penny Desktop v6

- Penny desktop v6 (cont.)
  - See **pennydesktop6debounce.py**

PennyPyqt/pennydesktop6debounce.py (Page 1 of 3)

```python
#!/usr/bin/env python

#-----------------------------------------------------------------------
# pennydesktop6debounce.py
# Author: Bob Dondero
#-----------------------------------------------------------------------

import sys
import threading
import urllib.parse
import urllib.request
import json
import queue
import PyQt5.QtWidgets
import PyQt5.QtCore

#-----------------------------------------------------------------------

def create_widgets():

    author_label = PyQt5.QtWidgets.QLabel('Author: ')
    author_lineedit = PyQt5.QtWidgets.QLineEdit()
    books_textedit = PyQt5.QtWidgets.QTextEdit()
    books_textedit.setReadOnly(True)

    layout = PyQt5.QtWidgets.QGridLayout()
    layout.addWidget(author_label, 0, 0)
    layout.addWidget(author_lineedit, 0, 1)
    layout.addWidget(books_textedit, 1, 0, 1, 2)
    layout.setRowStretch(0, 0)
    layout.setRowStretch(1, 1)
    layout.setColumnStretch(0, 0)
    layout.setColumnStretch(1, 1)
    layout.setColumnStretch(2, 0)

    frame = PyQt5.QtWidgets.QFrame()
    frame.setLayout(layout)

    window = PyQt5.QtWidgets.QMainWindow()
    window.setWindowTitle('Penny: Author Search')
    window.setCentralWidget(frame)
    screen_size = PyQt5.QtWidgets.QDesktopWidget().screenGeometry()
    window.resize(screen_size.width()//2, screen_size.height()//2)

    return (window, author_lineedit, books_textedit)

#-----------------------------------------------------------------------

class WorkerThread (PyQt5.QtCore.QThread):

    _response_signal = PyQt5.QtCore.pyqtSignal(bool, object)

    def __init__(self, server_url, author, handle_response):
        super().__init__()
        self._server_url = server_url
        self._author = author
        self._response_signal.connect(handle_response)
        self._should_stop = False

    def stop(self):
        self._should_stop = True

    def run(self):
        encoded_author = urllib.parse.quote_plus(self._author)
        url = self._server_url
```

PennyPyqt/pennydesktop6debounce.py (Page 2 of 3)

```python
        url += '/searchresults?author=' + encoded_author
        try:
            with urllib.request.urlopen(url) as in_flo:
                response = in_flo.read()
                json_doc = response.decode('utf-8')
                books = json.loads(json_doc)
            if not self._should_stop:
                self._response_signal.emit(True, books)
        except Exception as ex:
            if not self._should_stop:
                self._response_signal.emit(False, str(ex))

#-----------------------------------------------------------------------

def main():

    if len(sys.argv) != 2:
        print('Usage: penny serverURL', file=sys.stderr)
        sys.exit(1)

    server_url = sys.argv[1]

    # Create and lay out the widgets.

    app = PyQt5.QtWidgets.QApplication(sys.argv)
    window, author_lineedit, books_textedit = create_widgets()

    # Handle signals.

    def handle_response(successful, data):
        books_textedit.clear()
        if successful:
            books = data
            if len(books) == 0:
                books_textedit.insertPlainText('(None)')
            else:
                pattern = '%s: <strong>%s</strong>: %s<br>'
                for book in books:
                    books_textedit.insertHtml(pattern %
                        (book['isbn'], book['author'], book['title']))
        else:
            ex = data
            books_textedit.insertPlainText(ex)
        books_textedit.repaint()

    worker_thread = None
    def author_slot():
        nonlocal worker_thread
        author = author_lineedit.text()
        if worker_thread is not None:
            worker_thread.stop()
        worker_thread = WorkerThread(server_url, author,
            handle_response)
        worker_thread.start()

    debounce_timer = None
    def debounced_author_slot():
        nonlocal debounce_timer
        if debounce_timer is not None:
            debounce_timer.cancel()
        debounce_timer = threading.Timer(0.5, author_slot)
        debounce_timer.start()

    author_lineedit.textChanged.connect(debounced_author_slot)

```

PennyPyqt/pennydesktop6debounce.py (Page 3 of 3)

```python
    # Start up.

    window.show()
    author_slot()  # Populate books_textedit initially.
    sys.exit(app.exec_())

if __name__ == '__main__':
    main()
```

Summary

- Note:
  - If a web application delivers JSON (instead of HTML), then...
  - The client reasonably can be:
    - A browser (given HTML/JavaScript code from the server)
    - A desktop client

Summary

- We have covered:
  - Desktop programming
  - A Penny **desktop** client
    - A desktop client that works with the Penny server