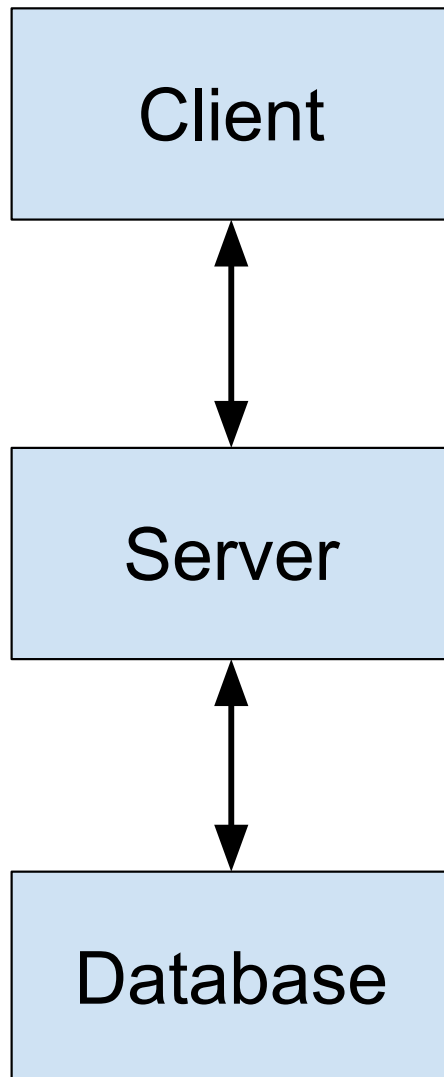


# Client-Side Options (Part 1)

Copyright © 2024 by  
Robert M. Dondero, Ph.D.  
Princeton University

# 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
<b>GTK</b>	C, C++	Linux, Windows, macOS
<b>Qt</b>	<b>C, C++</b>	<b>Linux, Windows, macOS</b>
<b>wxWidgets</b>	C, C++	Linux, Windows, macOS
<b>Flutter</b>	C, C++, Dart	Linux, Windows, macOS, Android, iOS
<b>Kivy</b>	Python	Linux, Windows, macOS, Android, iOS
<b>Swing</b>	Java	Linux, Windows, macOS
<b>Tck/Tk</b>	Tcl	Linux, Windows, macOS, Android, iOS
...	...	...
<b>PyQt</b>	<b>Python</b>	<b>Linux, Windows, macOS</b>

# Qt and PyQt5

- *Qt*
  - Haavard Nord and Eirik Chambe-Eng
  - Trolltech, 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

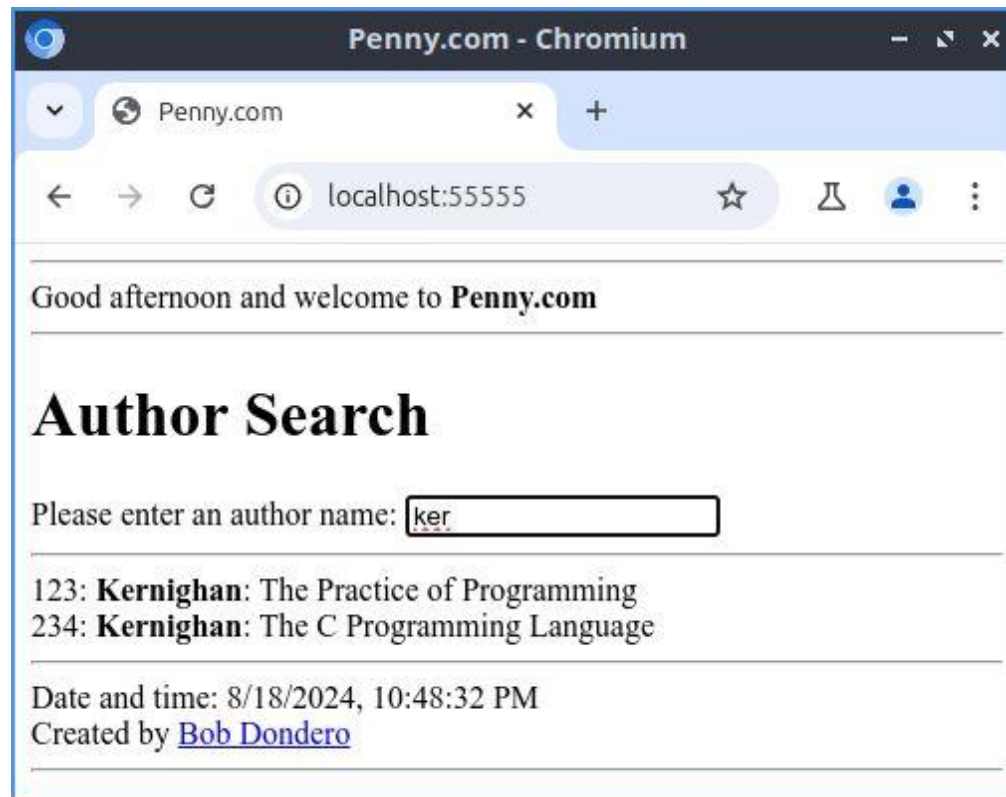
- **Question:** Why study PyQt5?
  - Instead of some other GUI library?
- **Answers:**
  - Convenient way to learn the fundamentals of Qt!

# 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 **PennyJson** app
  - (Almost) same as **PennyAjax4** app from previous lecture



# PennyJson Server

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

**PennyJson/penny.py (Page 1 of 1)**

```

1: #!/usr/bin/env python
2:
3: #-----
4: # penny.py
5: # Author: Bob Dondero
6: #-----
7:
8: import os
9: import time
10: import json
11: import flask
12: import dotenv
13: import database
14:
15: dotenv.load_dotenv()
16: _IO_DELAY = int(os.getenv('IO_DELAY', '0'))
17:
18: #-----
19:
20: app = flask.Flask(__name__)
21:
22: #-----
23:
24: @app.route('/', methods=['GET'])
25: @app.route('/index', methods=['GET'])
26: def index():
27:
28:     return flask.send_file('index.html')
29:
30: #-----
31:
32: @app.route('/searchresults', methods=['GET'])
33: def search_results():
34:
35:     author = flask.request.args.get('author')
36:     if author is None:
37:         author = ''
38:     author = author.strip()
39:
40:     # Simulate a slow database.
41:     time.sleep(_IO_DELAY)
42:
43:     if author == '':
44:         books = []
45:     else:
46:         books = database.get_books(author) # Exception handling omitted
47:
48:     json_doc = json.dumps(books)
49:     response = flask.make_response(json_doc)
50:     response.headers['Content-Type'] = 'application/json'
51:     return response

```

**blank (Page 1 of 1)**

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



The image shows a desktop application window titled "Penny: Author Search". The window has a dark blue title bar with standard window controls (minimize, maximize, close) on the right. Below the title bar, there is a light gray panel containing the text "Author:" followed by a white text input field. To the right of the input field is a gray button labeled "Submit". Below this panel is a large white rectangular area, which currently displays the text "(None)".

# Penny Desktop v1

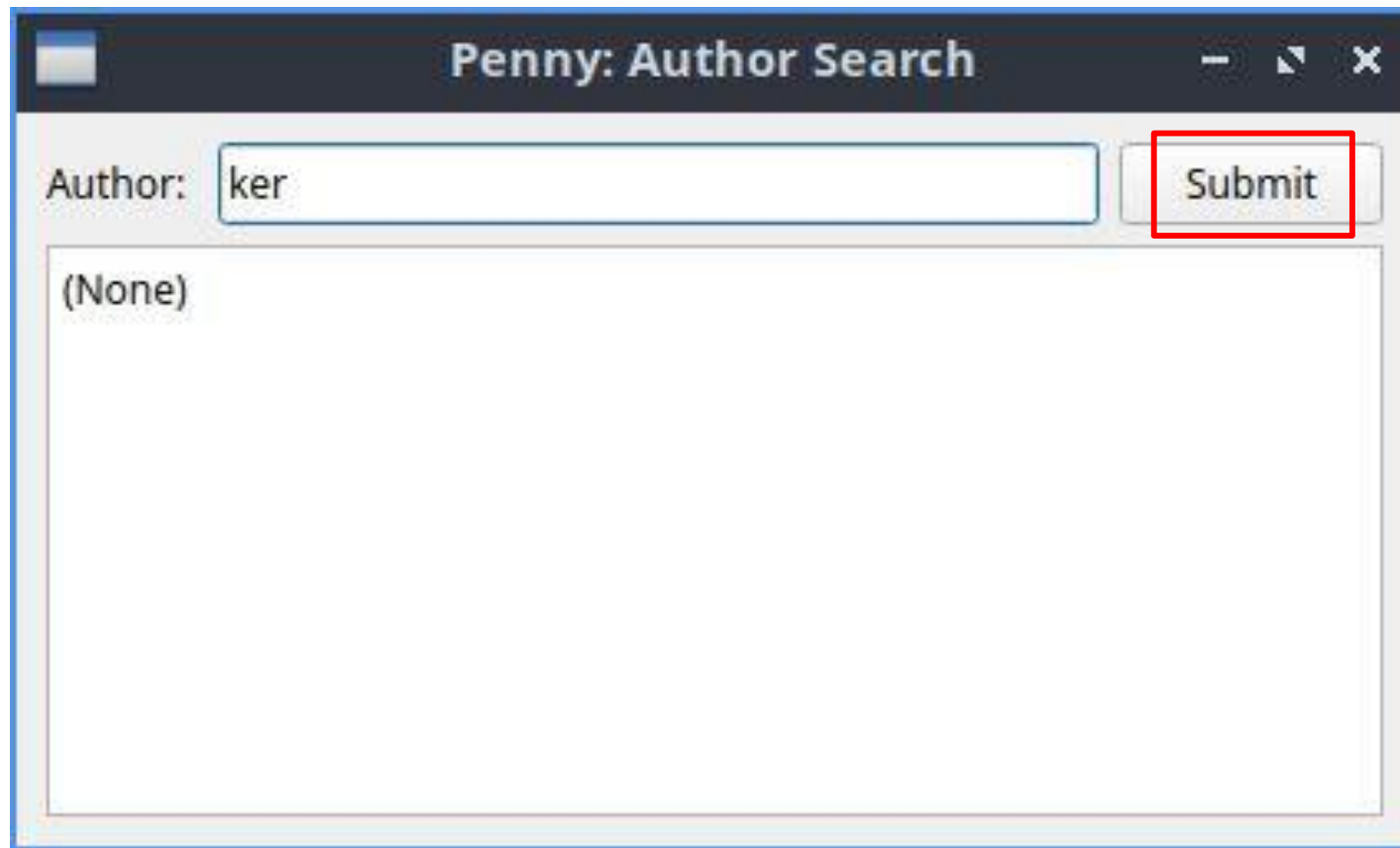
- Penny desktop v1



A screenshot of a web application window titled "Penny: Author Search". The window has a dark blue header bar with the title and standard window controls (minimize, maximize, close). Below the header, there is a search form. The form consists of a label "Author:" followed by a text input field containing the text "ker". The input field is highlighted with a red rectangular border. To the right of the input field is a "Submit" button. Below the input field is a large, empty rectangular area, likely for displaying search results. The text "(None)" is visible in the top-left corner of this area.

# Penny Desktop v1

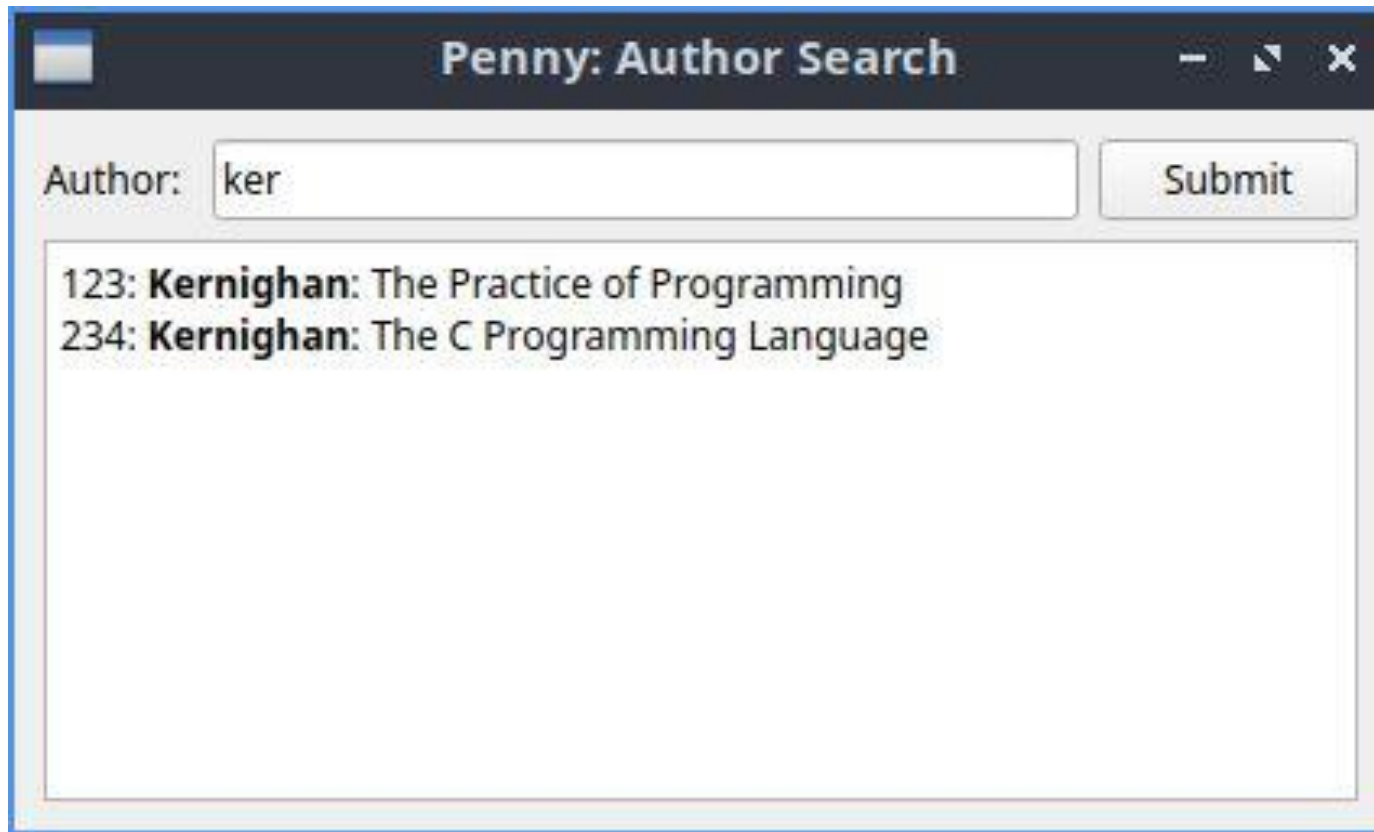
- Penny desktop v1



The image shows a desktop application window titled "Penny: Author Search". The window has a dark blue title bar with standard window controls (minimize, maximize, close) on the right. Below the title bar, there is a search form. On the left, the label "Author:" is followed by a text input field containing the text "ker". To the right of the input field is a button labeled "Submit", which is highlighted with a red rectangular border. Below the input field and button is a large, empty rectangular area, likely a list or display for search results. The text "(None)" is visible in the top-left corner of this area, indicating no results were found.

# Penny Desktop v1

- Penny desktop v1



The screenshot shows a window titled "Penny: Author Search". It has a search input field labeled "Author:" containing the text "ker" and a "Submit" button. Below the input field, there is a list of search results:

- 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)

```

1: #!/usr/bin/env python
2:
3: #-----
4: # pennydesktop1base.py
5: # Author: Bob Dondero
6: #-----
7:
8: import sys
9: import urllib.parse
10: import urllib.request
11: import json
12: import PyQt5.QtWidgets
13:
14: #-----
15:
16: def create_widgets():
17:
18:     author_label = PyQt5.QtWidgets.QLabel('Author: ')
19:     author_lineedit = PyQt5.QtWidgets.QLineEdit()
20:     submit_button = PyQt5.QtWidgets.QPushButton('Submit')
21:     books_textedit = PyQt5.QtWidgets.QTextEdit()
22:     books_textedit.setReadOnly(True)
23:
24:     layout = PyQt5.QtWidgets.QGridLayout()
25:     layout.addWidget(author_label, 0, 0)
26:     layout.addWidget(author_lineedit, 0, 1)
27:     layout.addWidget(submit_button, 0, 2)
28:     layout.addWidget(books_textedit, 1, 0, 1, 3)
29:     layout.setRowStretch(0, 0)
30:     layout.setRowStretch(1, 1)
31:     layout.setColumnStretch(0, 0)
32:     layout.setColumnStretch(1, 1)
33:     layout.setColumnStretch(2, 0)
34:
35:     frame = PyQt5.QtWidgets.QFrame()
36:     frame.setLayout(layout)
37:
38:     window = PyQt5.QtWidgets.QMainWindow()
39:     window.setWindowTitle('Penny: Author Search')
40:     window.setCentralWidget(frame)
41:     screen_size = PyQt5.QtWidgets.QDesktopWidget().screenGeometry()
42:     window.resize(screen_size.width()/2, screen_size.height()/2)
43:
44:     return (window, author_lineedit, submit_button, books_textedit)
45:
46: #-----
47:
48: def author_slot_helper(server_url, author_lineedit, books_textedit):
49:
50:     author = author_lineedit.text()
51:     encoded_author = urllib.parse.quote_plus(author)
52:     url = server_url + '/searchresults?author=' + encoded_author
53:
54:     books_textedit.clear()
55:
56:     try:
57:         with urllib.request.urlopen(url) as in_flo:
58:             response = in_flo.read()
59:             json_doc = response.decode('utf-8')
60:             books = json.loads(json_doc)
61:
62:             if len(books) == 0:
63:                 books_textedit.insertPlainText(' (None) ')
64:             else:
65:                 pattern = '%s: <strong>%s</strong>: %s<br>'

```

## PennyPyqt/pennydesktop1base.py (Page 2 of 2)

```

66:         for book in books:
67:             books_textedit.insertHtml(pattern %
68:                                     (book['isbn'], book['author'], book['title']))
69:
70:     except Exception as ex:
71:         books_textedit.insertPlainText(str(ex))
72:
73:     books_textedit.repaint() # Required on Mac.
74:
75: #-----
76:
77: def main():
78:
79:     if len(sys.argv) != 2:
80:         print('Usage: penny serverURL', file=sys.stderr)
81:         sys.exit(1)
82:
83:     server_url = sys.argv[1]
84:
85:     # Create and lay out the widgets.
86:
87:     app = PyQt5.QtWidgets.QApplication(sys.argv)
88:     window, author_lineedit, submit_button, books_textedit = (
89:         create_widgets())
90:
91:     # Handle signals.
92:
93:     def author_slot():
94:         author_slot_helper(server_url, author_lineedit, books_textedit)
95:     submit_button.clicked.connect(author_slot)
96:
97:     # Start up.
98:
99:     window.show()
100:     author_slot() # Populate books_textedit initially.
101:     sys.exit(app.exec_())
102:
103: if __name__ == '__main__':
104:     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
```

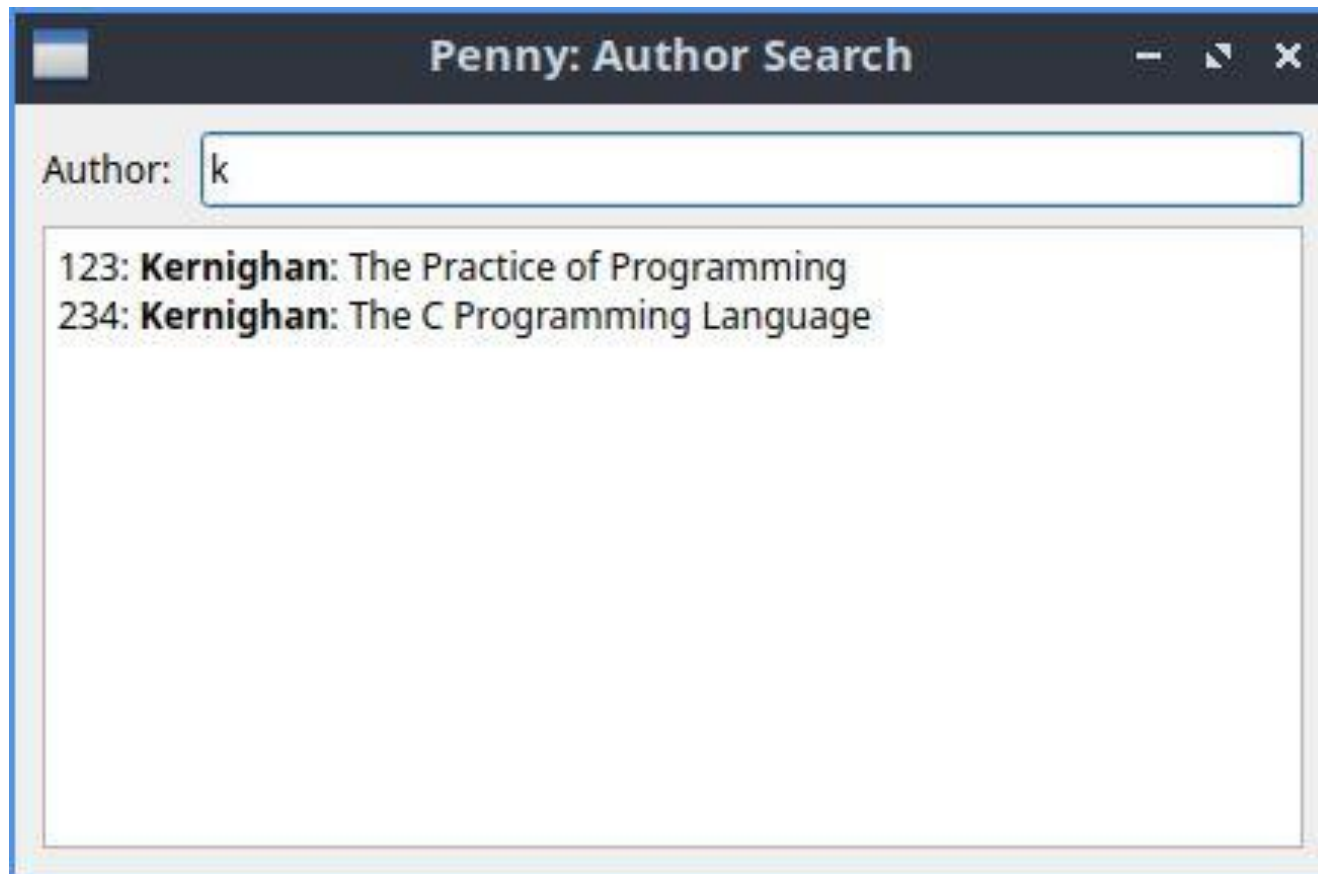
# Penny Desktop v2

- Penny desktop v2 (cont.)



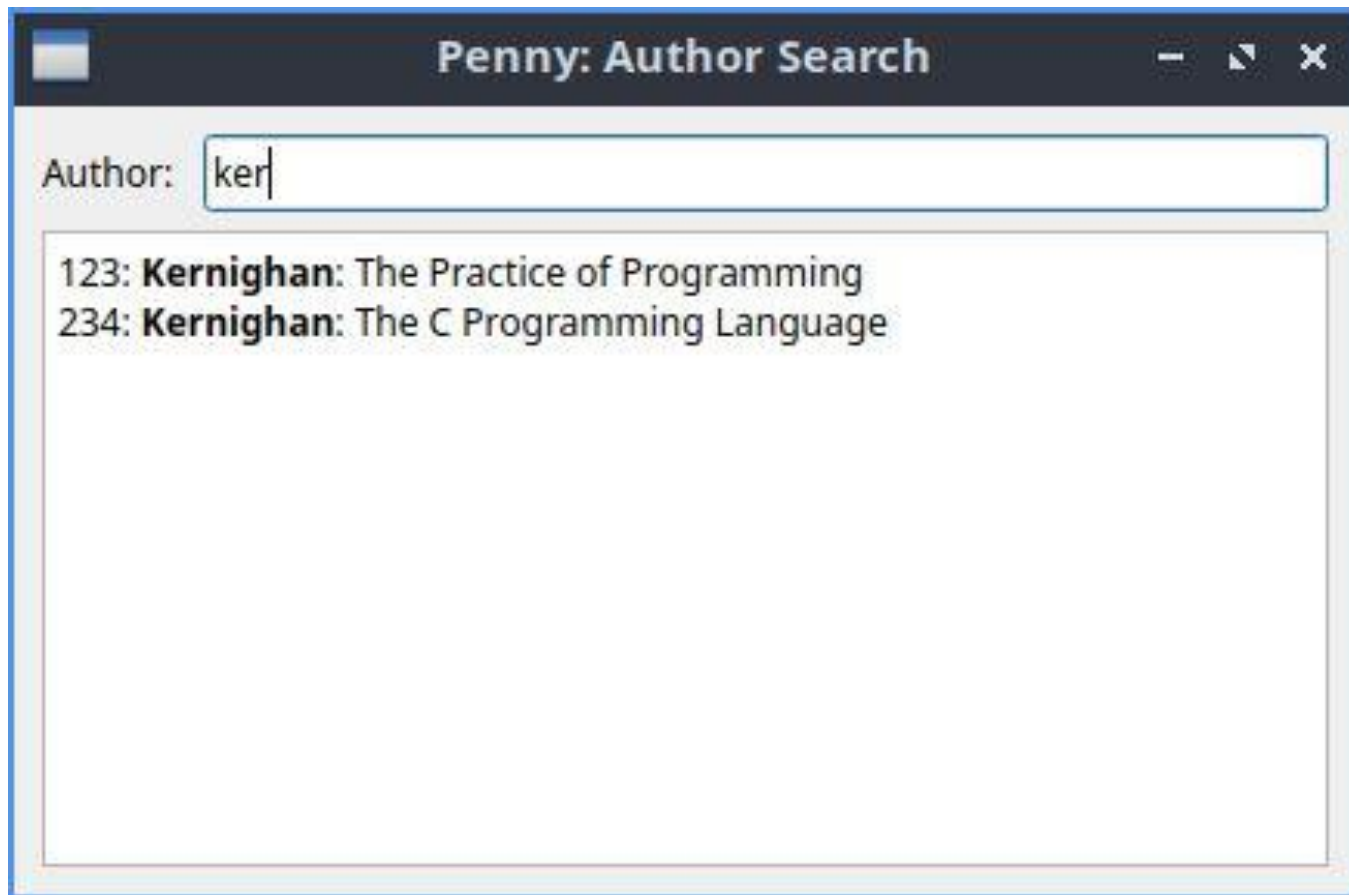
# Penny Desktop v2

- Penny desktop v2 (cont.)



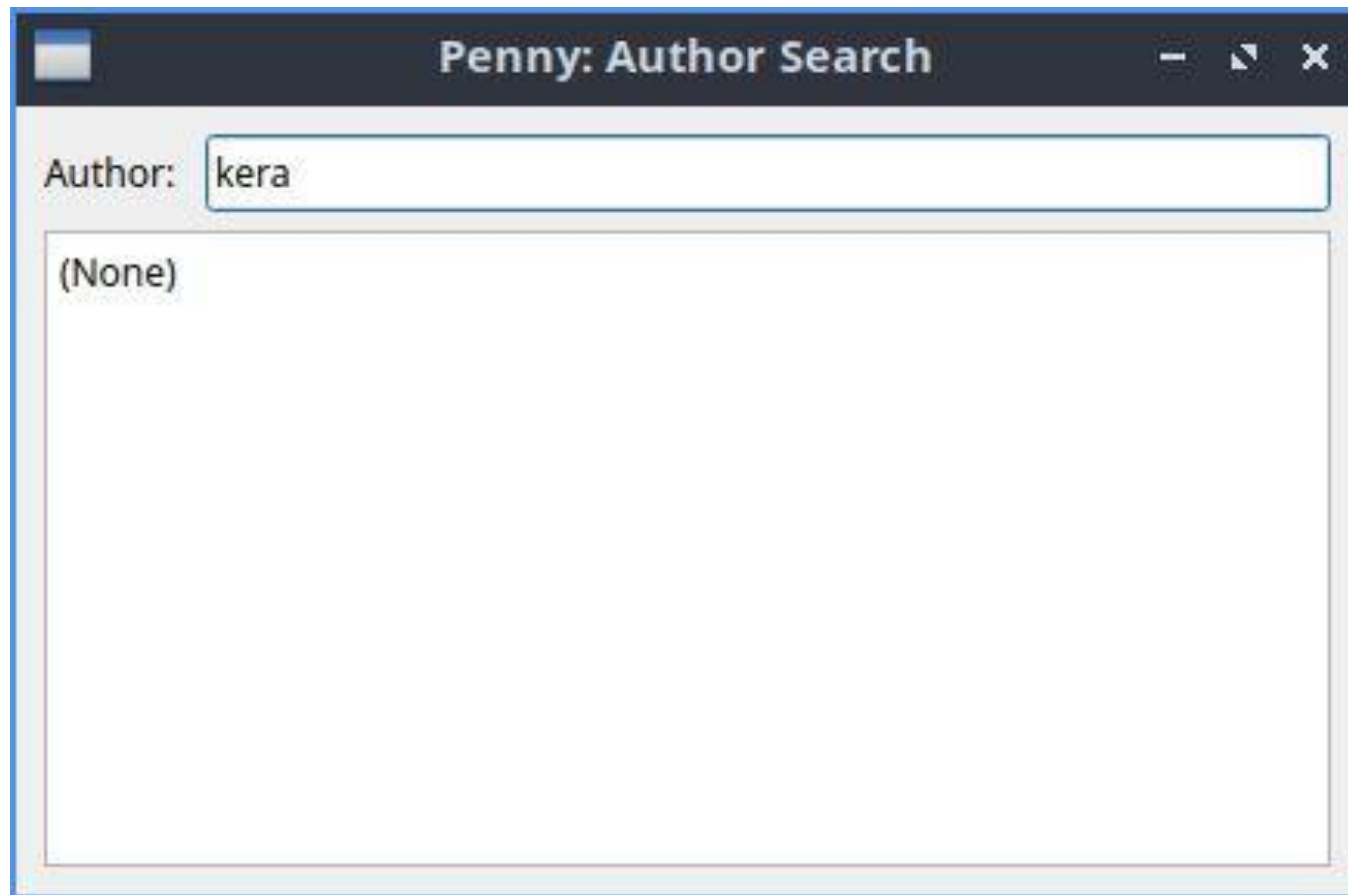
# Penny Desktop v2

- Penny desktop v2 (cont.)



# Penny Desktop v2

- Penny desktop v2 (cont.)



# Penny Desktop v2

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



# Penny Desktop v2

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

## PennyPyqt/pennydesktop2seq.py (Page 1 of 2)

```

1: #!/usr/bin/env python
2:
3: #-----
4: # pennydesktop1seq.py
5: # Author: Bob Dondero
6: #-----
7:
8: import sys
9: import urllib.parse
10: import urllib.request
11: import json
12: import PyQt5.QtWidgets
13:
14: #-----
15:
16: def create_widgets():
17:
18:     author_label = PyQt5.QtWidgets.QLabel('Author: ')
19:     author_lineedit = PyQt5.QtWidgets.QLineEdit()
20:     books_textedit = PyQt5.QtWidgets.QTextEdit()
21:     books_textedit.setReadOnly(True)
22:
23:     layout = PyQt5.QtWidgets.QGridLayout()
24:     layout.addWidget(author_label, 0, 0)
25:     layout.addWidget(author_lineedit, 0, 1)
26:     layout.addWidget(books_textedit, 1, 0, 1, 2)
27:     layout.setRowStretch(0, 0)
28:     layout.setRowStretch(1, 1)
29:     layout.setColumnStretch(0, 0)
30:     layout.setColumnStretch(1, 1)
31:     layout.setColumnStretch(2, 0)
32:
33:     frame = PyQt5.QtWidgets.QFrame()
34:     frame.setLayout(layout)
35:
36:     window = PyQt5.QtWidgets.QMainWindow()
37:     window.setWindowTitle('Penny: Author Search')
38:     window.setCentralWidget(frame)
39:     screen_size = PyQt5.QtWidgets.QDesktopWidget().screenGeometry()
40:     window.resize(screen_size.width()/2, screen_size.height()/2)
41:
42:     return (window, author_lineedit, books_textedit)
43:
44: #-----
45:
46: def author_slot_helper(server_url, author_lineedit, books_textedit):
47:
48:     author = author_lineedit.text()
49:     encoded_author = urllib.parse.quote_plus(author)
50:     url = server_url + '/searchresults?author=' + encoded_author
51:
52:     books_textedit.clear()
53:
54:     try:
55:         with urllib.request.urlopen(url) as in_flo:
56:             response = in_flo.read()
57:             json_doc = response.decode('utf-8')
58:             books = json.loads(json_doc)
59:
60:             if len(books) == 0:
61:                 books_textedit.insertPlainText('(None)')
62:             else:
63:                 pattern = '%s: <strong>%s</strong>: %s<br>'
64:                 for book in books:
65:                     books_textedit.insertHtml(pattern %

```

## PennyPyqt/pennydesktop2seq.py (Page 2 of 2)

```

66:                     (book['isbn'], book['author'], book['title']))
67:
68:     except Exception as ex:
69:         books_textedit.insertPlainText(str(ex))
70:
71:     books_textedit.repaint() # Required on Mac.
72:
73: #-----
74:
75: def main():
76:
77:     if len(sys.argv) != 2:
78:         print('Usage: penny serverURL', file=sys.stderr)
79:         sys.exit(1)
80:
81:     server_url = sys.argv[1]
82:
83:     # Create and lay out the widgets.
84:
85:     app = PyQt5.QtWidgets.QApplication(sys.argv)
86:     window, author_lineedit, books_textedit = create_widgets()
87:
88:     # Handle signals.
89:
90:     def author_slot():
91:         author_slot_helper(server_url, author_lineedit, books_textedit)
92:     author_lineedit.textChanged.connect(author_slot)
93:
94:     # Start up.
95:
96:     window.show()
97:     author_slot() # Populate books_textedit initially.
98:     sys.exit(app.exec_())
99:
100: if __name__ == '__main__':
101:     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 **pennyclient3bad.py**

## PennyPyqt/pennydesktop3bad.py (Page 1 of 2)

```

1: #!/usr/bin/env python
2:
3: #-----
4: # pennydesktop3bad.py
5: # Author: Bob Dondero
6: #-----
7:
8: import sys
9: import threading
10: import urllib.parse
11: import urllib.request
12: import json
13: import PyQt5.QtWidgets
14:
15: #-----
16:
17: def create_widgets():
18:
19:     author_label = PyQt5.QtWidgets.QLabel('Author: ')
20:     author_lineedit = PyQt5.QtWidgets.QLineEdit()
21:     books_textedit = PyQt5.QtWidgets.QTextEdit()
22:     books_textedit.setReadOnly(True)
23:
24:     layout = PyQt5.QtWidgets.QGridLayout()
25:     layout.addWidget(author_label, 0, 0)
26:     layout.addWidget(author_lineedit, 0, 1)
27:     layout.addWidget(books_textedit, 1, 0, 1, 2)
28:     layout.setRowStretch(0, 0)
29:     layout.setRowStretch(1, 1)
30:     layout.setColumnStretch(0, 0)
31:     layout.setColumnStretch(1, 1)
32:     layout.setColumnStretch(2, 0)
33:
34:     frame = PyQt5.QtWidgets.QFrame()
35:     frame.setLayout(layout)
36:
37:     window = PyQt5.QtWidgets.QMainWindow()
38:     window.setWindowTitle('Penny: Author Search')
39:     window.setCentralWidget(frame)
40:     screen_size = PyQt5.QtWidgets.QDesktopWidget().screenGeometry()
41:     window.resize(screen_size.width()//2, screen_size.height()//2)
42:
43:     return (window, author_lineedit, books_textedit)
44:
45: #-----
46:
47: class WorkerThread (threading.Thread):
48:
49:     def __init__(self, server_url, author, books_textedit):
50:         threading.Thread.__init__(self)
51:         self._server_url = server_url
52:         self._author = author
53:         self._books_textedit = books_textedit
54:
55:     def run(self):
56:         encoded_author = urllib.parse.quote_plus(self._author)
57:         url = self._server_url
58:         url += '/searchresults?author=' + encoded_author
59:
60:         self._books_textedit.clear()
61:
62:         try:
63:             with urllib.request.urlopen(url) as in_flo:
64:                 response = in_flo.read()
65:                 json_doc = response.decode('utf-8')

```

## PennyPyqt/pennydesktop3bad.py (Page 2 of 2)

```

66:         books = json.loads(json_doc)
67:
68:         if len(books) == 0:
69:             self._books_textedit.insertPlainText(' (None)')
70:         else:
71:             pattern = '%s: <strong>%s</strong>: %s<br>'
72:             for book in books:
73:                 self._books_textedit.insertHtml(pattern %
74:                     (book['isbn'], book['author'],
75:                      book['title']))
76:
77:         except Exception as ex:
78:             self._books_textedit.insertPlainText(str(ex))
79:
80:         self._books_textedit.repaint() # Required on Mac.
81:
82: #-----
83:
84: def main():
85:
86:     if len(sys.argv) != 2:
87:         print('Usage: penny serverURL', file=sys.stderr)
88:         sys.exit(1)
89:
90:     server_url = sys.argv[1]
91:
92:     # Create and lay out the widgets.
93:
94:     app = PyQt5.QtWidgets.QApplication(sys.argv)
95:     window, author_lineedit, books_textedit = create_widgets()
96:
97:     # Handle signals.
98:
99:     def author_slot():
100:         author = author_lineedit.text()
101:         worker_thread = WorkerThread(server_url, author, books_textedit)
102:         worker_thread.start()
103:         author_lineedit.textChanged.connect(author_slot)
104:
105:     # Start up.
106:
107:     window.show()
108:     author_slot() # Populate books_textedit initially.
109:     sys.exit(app.exec_())
110:
111: if __name__ == '__main__':
112:     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
```

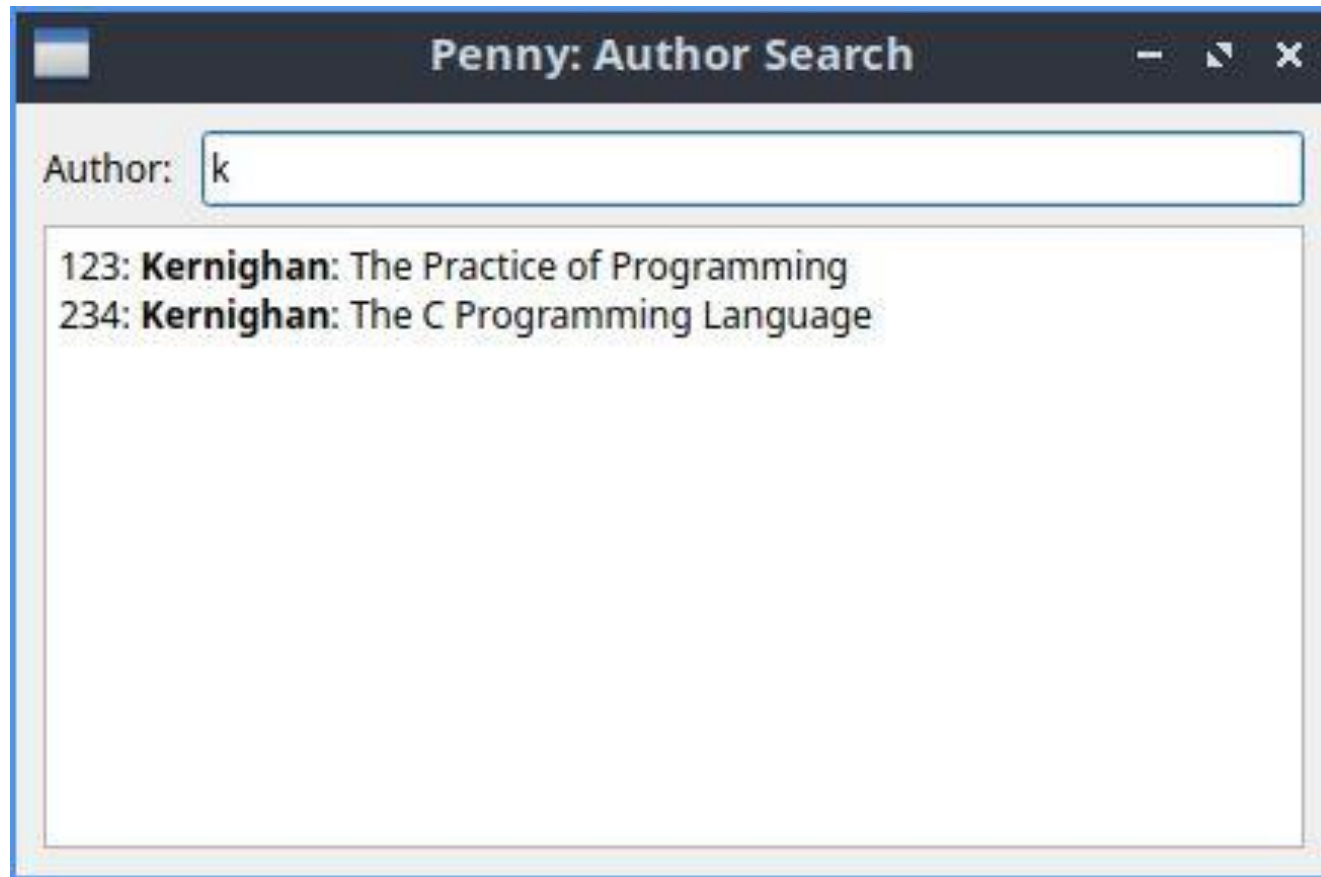
# Penny Desktop v4

- Penny desktop v4 (cont.)



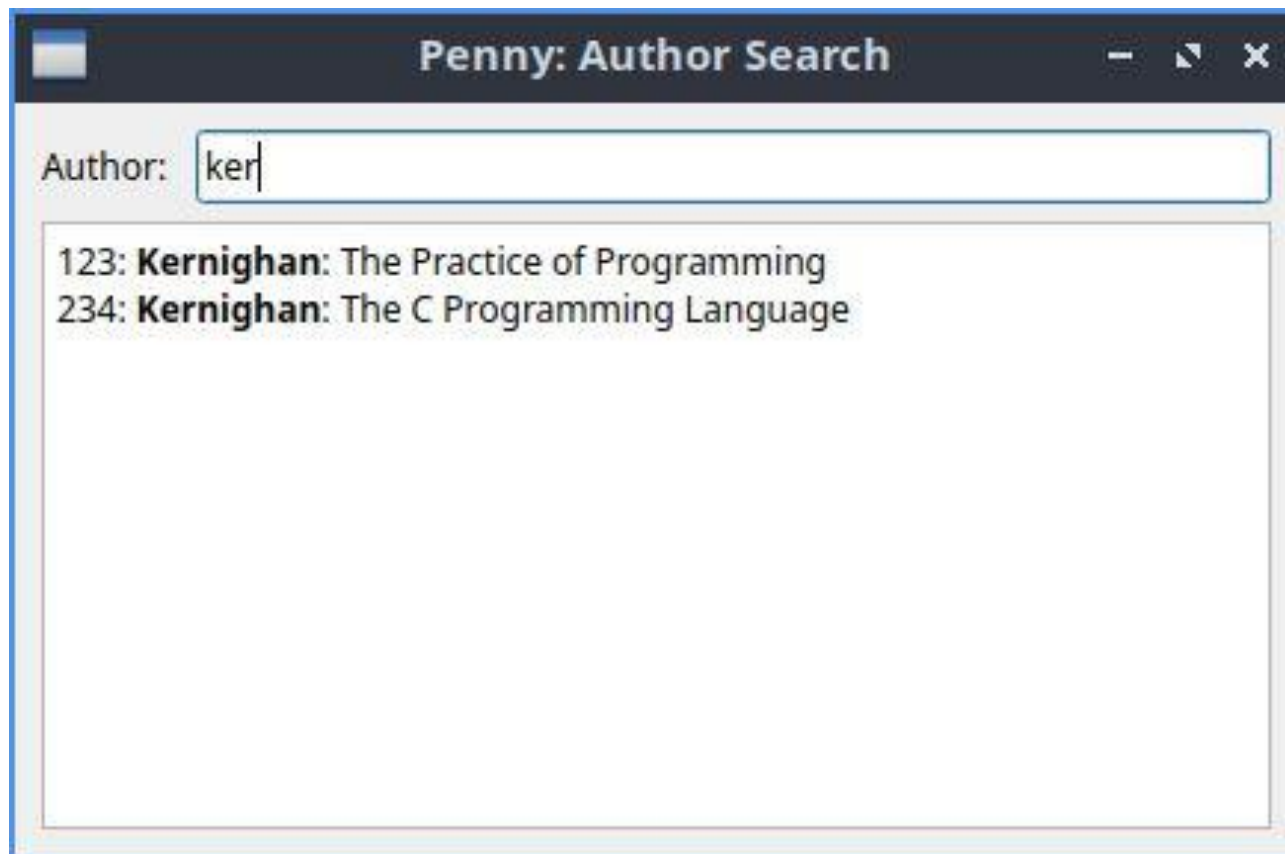
# Penny Desktop v4

- Penny desktop v4 (cont.)



# Penny Desktop v4

- Penny desktop v4 (cont.)



# Penny Desktop v4

- Penny desktop v4 (cont.)



# Penny Desktop v4

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

## PennyPyqt/pennydesktop4threads.py (Page 1 of 2)

```

1: #!/usr/bin/env python
2:
3: #-----
4: # pennydesktop4threads.py
5: # Author: Bob Dondero
6: #-----
7:
8: import sys
9: import threading
10: import urllib.parse
11: import urllib.request
12: import json
13: import queue
14: import PyQt5.QtWidgets
15: import PyQt5.QtCore
16:
17: #-----
18:
19: def create_widgets():
20:
21:     author_label = PyQt5.QtWidgets.QLabel('Author: ')
22:     author_lineedit = PyQt5.QtWidgets.QLineEdit()
23:     books_textedit = PyQt5.QtWidgets.QTextEdit()
24:     books_textedit.setReadOnly(True)
25:
26:     layout = PyQt5.QtWidgets.QGridLayout()
27:     layout.addWidget(author_label, 0, 0)
28:     layout.addWidget(author_lineedit, 0, 1)
29:     layout.addWidget(books_textedit, 1, 0, 1, 2)
30:     layout.setRowStretch(0, 0)
31:     layout.setRowStretch(1, 1)
32:     layout.setColumnStretch(0, 0)
33:     layout.setColumnStretch(1, 1)
34:     layout.setColumnStretch(2, 0)
35:
36:     frame = PyQt5.QtWidgets.QFrame()
37:     frame.setLayout(layout)
38:
39:     window = PyQt5.QtWidgets.QMainWindow()
40:     window.setWindowTitle('Penny: Author Search')
41:     window.setCentralWidget(frame)
42:     screen_size = PyQt5.QtWidgets.QDesktopWidget().screenGeometry()
43:     window.resize(screen_size.width()/2, screen_size.height()/2)
44:
45:     return (window, author_lineedit, books_textedit)
46:
47: #-----
48:
49: class WorkerThread (PyQt5.QtCore.QThread):
50:
51:     _response_signal = PyQt5.QtCore.pyqtSignal(bool, object)
52:
53:     def __init__(self, server_url, author, handle_response):
54:         super().__init__()
55:         self._server_url = server_url
56:         self._author = author
57:         self._response_signal.connect(handle_response)
58:
59:     def run(self):
60:         encoded_author = urllib.parse.quote_plus(self._author)
61:         url = self._server_url
62:         url += '/searchresults?author=' + encoded_author
63:         try:
64:             with urllib.request.urlopen(url) as in_flo:
65:                 response = in_flo.read()

```

## PennyPyqt/pennydesktop4threads.py (Page 2 of 2)

```

66:         json_doc = response.decode('utf-8')
67:         books = json.loads(json_doc)
68:         self._response_signal.emit(True, books)
69:     except Exception as ex:
70:         self._response_signal.emit(False, str(ex))
71:
72: #-----
73:
74: def main():
75:
76:     if len(sys.argv) != 2:
77:         print('Usage: penny serverURL', file=sys.stderr)
78:         sys.exit(1)
79:
80:     server_url = sys.argv[1]
81:
82:     # Create and lay out the widgets.
83:
84:     app = PyQt5.QtWidgets.QApplication(sys.argv)
85:     window, author_lineedit, books_textedit = create_widgets()
86:
87:     # Handle signals.
88:
89:     def handle_response(successful, data):
90:         books_textedit.clear()
91:         if successful:
92:             books = data
93:             if len(books) == 0:
94:                 books_textedit.insertPlainText('(None)')
95:             else:
96:                 pattern = '%s: <strong>%s</strong>: %s<br>'
97:                 for book in books:
98:                     books_textedit.insertHtml(pattern %
99:                                                  (book['isbn'], book['author'], book['title']))
100:         else:
101:             ex = data
102:             books_textedit.insertPlainText(ex)
103:             books_textedit.repaint()
104:
105:     worker_thread = None
106:     def author_slot():
107:         nonlocal worker_thread
108:         author = author_lineedit.text()
109:         worker_thread = WorkerThread(server_url, author,
110:                                       handle_response)
111:         worker_thread.start()
112:
113:     author_lineedit.textChanged.connect(author_slot)
114:
115:     # Start up.
116:
117:     window.show()
118:     author_slot() # Populate books_textedit initially.
119:     sys.exit(app.exec_())
120:
121: if __name__ == '__main__':
122:     main()

```

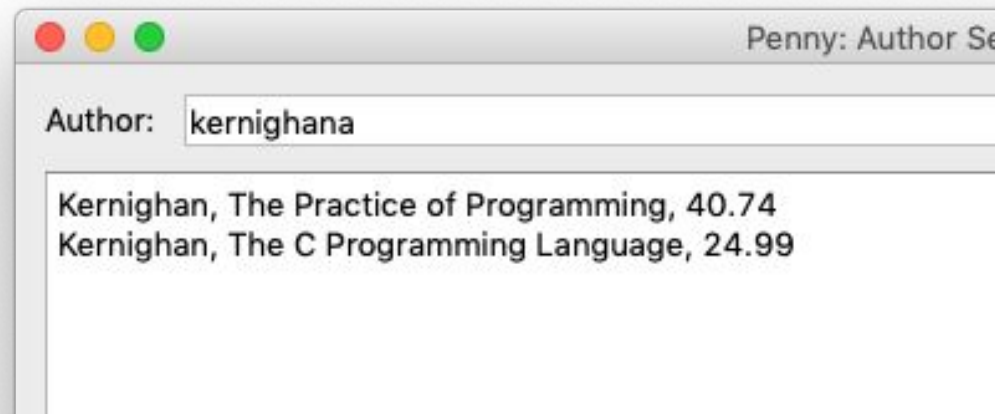


# Penny Desktop v4

- **Problem:**

- Server will respond to requests in arbitrary order

old  
screen  
shot



- **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 **pennydesktop5stop.py**

## PennyPyqt/pennydesktop5stop.py (Page 1 of 2)

```

1: #!/usr/bin/env python
2:
3: #-----
4: # pennydesktop5stop.py
5: # Author: Bob Dondero
6: #-----
7:
8: import sys
9: import threading
10: import urllib.parse
11: import urllib.request
12: import json
13: import queue
14: import PyQt5.QtWidgets
15: import PyQt5.QtCore
16:
17: #-----
18:
19: def create_widgets():
20:
21:     author_label = PyQt5.QtWidgets.QLabel('Author: ')
22:     author_lineedit = PyQt5.QtWidgets.QLineEdit()
23:     books_textedit = PyQt5.QtWidgets.QTextEdit()
24:     books_textedit.setReadOnly(True)
25:
26:     layout = PyQt5.QtWidgets.QGridLayout()
27:     layout.addWidget(author_label, 0, 0)
28:     layout.addWidget(author_lineedit, 0, 1)
29:     layout.addWidget(books_textedit, 1, 0, 1, 2)
30:     layout.setRowStretch(0, 0)
31:     layout.setRowStretch(1, 1)
32:     layout.setColumnStretch(0, 0)
33:     layout.setColumnStretch(1, 1)
34:     layout.setColumnStretch(2, 0)
35:
36:     frame = PyQt5.QtWidgets.QFrame()
37:     frame.setLayout(layout)
38:
39:     window = PyQt5.QtWidgets.QMainWindow()
40:     window.setWindowTitle('Penny: Author Search')
41:     window.setCentralWidget(frame)
42:     screen_size = PyQt5.QtWidgets.QDesktopWidget().screenGeometry()
43:     window.resize(screen_size.width()/2, screen_size.height()/2)
44:
45:     return (window, author_lineedit, books_textedit)
46:
47: #-----
48:
49: class WorkerThread (PyQt5.QtCore.QThread):
50:
51:     _response_signal = PyQt5.QtCore.pyqtSignal(bool, object)
52:
53:     def __init__(self, server_url, author, handle_response):
54:         super().__init__()
55:         self._server_url = server_url
56:         self._author = author
57:         self._response_signal.connect(handle_response)
58:         self._should_stop = False
59:
60:     def stop(self):
61:         self._should_stop = True
62:
63:     def run(self):
64:         encoded_author = urllib.parse.quote_plus(self._author)
65:         url = self._server_url

```

## PennyPyqt/pennydesktop5stop.py (Page 2 of 2)

```

66:         url += '/searchresults?author=' + encoded_author
67:         try:
68:             with urllib.request.urlopen(url) as in_flo:
69:                 response = in_flo.read()
70:                 json_doc = response.decode('utf-8')
71:                 books = json.loads(json_doc)
72:                 if not self._should_stop:
73:                     self._response_signal.emit(True, books)
74:         except Exception as ex:
75:             if not self._should_stop:
76:                 self._response_signal.emit(False, str(ex))
77:
78: #-----
79:
80: def main():
81:
82:     if len(sys.argv) != 2:
83:         print('Usage: penny serverURL', file=sys.stderr)
84:         sys.exit(1)
85:
86:     server_url = sys.argv[1]
87:
88:     # Create and lay out the widgets.
89:
90:     app = PyQt5.QtWidgets.QApplication(sys.argv)
91:     window, author_lineedit, books_textedit = create_widgets()
92:
93:     # Handle signals.
94:
95:     def handle_response(successful, data):
96:         books_textedit.clear()
97:         if successful:
98:             books = data
99:             if len(books) == 0:
100:                 books_textedit.insertPlainText(' (None)')
101:             else:
102:                 pattern = '%s: <strong>%s</strong>: %s<br>'
103:                 for book in books:
104:                     books_textedit.insertHtml(pattern %
105:                                                  (book['isbn'], book['author'], book['title']))
106:         else:
107:             ex = data
108:             books_textedit.insertPlainText(ex)
109:             books_textedit.repaint()
110:
111:     worker_thread = None
112:     def author_slot():
113:         nonlocal worker_thread
114:         author = author_lineedit.text()
115:         if worker_thread is not None:
116:             worker_thread.stop()
117:         worker_thread = WorkerThread(server_url, author,
118:                                     handle_response)
119:         worker_thread.start()
120:
121:     author_lineedit.textChanged.connect(author_slot)
122:
123:     # Start up.
124:
125:     window.show()
126:     author_slot() # Populate books_textedit initially.
127:     sys.exit(app.exec_())
128:
129: if __name__ == '__main__':
130:     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)

```

1: #!/usr/bin/env python
2:
3: #-----
4: # pennydesktop6debounce.py
5: # Author: Bob Dondero
6: #-----
7:
8: import sys
9: import threading
10: import urllib.parse
11: import urllib.request
12: import json
13: import queue
14: import PyQt5.QtWidgets
15: import PyQt5.QtCore
16:
17: #-----
18:
19: def create_widgets():
20:
21:     author_label = PyQt5.QtWidgets.QLabel('Author: ')
22:     author_lineedit = PyQt5.QtWidgets.QLineEdit()
23:     books_textedit = PyQt5.QtWidgets.QTextEdit()
24:     books_textedit.setReadOnly(True)
25:
26:     layout = PyQt5.QtWidgets.QGridLayout()
27:     layout.addWidget(author_label, 0, 0)
28:     layout.addWidget(author_lineedit, 0, 1)
29:     layout.addWidget(books_textedit, 1, 0, 1, 2)
30:     layout.setRowStretch(0, 0)
31:     layout.setRowStretch(1, 1)
32:     layout.setColumnStretch(0, 0)
33:     layout.setColumnStretch(1, 1)
34:     layout.setColumnStretch(2, 0)
35:
36:     frame = PyQt5.QtWidgets.QFrame()
37:     frame.setLayout(layout)
38:
39:     window = PyQt5.QtWidgets.QMainWindow()
40:     window.setWindowTitle('Penny: Author Search')
41:     window.setCentralWidget(frame)
42:     screen_size = PyQt5.QtWidgets.QDesktopWidget().screenGeometry()
43:     window.resize(screen_size.width()/2, screen_size.height()/2)
44:
45:     return (window, author_lineedit, books_textedit)
46:
47: #-----
48:
49: class WorkerThread (PyQt5.QtCore.QThread):
50:
51:     _response_signal = PyQt5.QtCore.pyqtSignal(bool, object)
52:
53:     def __init__(self, server_url, author, handle_response):
54:         super().__init__()
55:         self._server_url = server_url
56:         self._author = author
57:         self._response_signal.connect(handle_response)
58:         self._should_stop = False
59:
60:     def stop(self):
61:         self._should_stop = True
62:
63:     def run(self):
64:         encoded_author = urllib.parse.quote_plus(self._author)
65:         url = self._server_url

```

## PennyPyqt/pennydesktop6debounce.py (Page 2 of 3)

```

66:         url += '/searchresults?author=' + encoded_author
67:         try:
68:             with urllib.request.urlopen(url) as in_flo:
69:                 response = in_flo.read()
70:                 json_doc = response.decode('utf-8')
71:                 books = json.loads(json_doc)
72:                 if not self._should_stop:
73:                     self._response_signal.emit(True, books)
74:         except Exception as ex:
75:             if not self._should_stop:
76:                 self._response_signal.emit(False, str(ex))
77:
78: #-----
79:
80: def main():
81:
82:     if len(sys.argv) != 2:
83:         print('Usage: penny serverURL', file=sys.stderr)
84:         sys.exit(1)
85:
86:     server_url = sys.argv[1]
87:
88:     # Create and lay out the widgets.
89:
90:     app = PyQt5.QtWidgets.QApplication(sys.argv)
91:     window, author_lineedit, books_textedit = create_widgets()
92:
93:     # Handle signals.
94:
95:     def handle_response(successful, data):
96:         books_textedit.clear()
97:         if successful:
98:             books = data
99:             if len(books) == 0:
100:                 books_textedit.insertPlainText(' (None)')
101:             else:
102:                 pattern = '%s: <strong>%s</strong>: %s<br>'
103:                 for book in books:
104:                     books_textedit.insertHtml(pattern %
105:                                                 (book['isbn'], book['author'], book['title']))
106:         else:
107:             ex = data
108:             books_textedit.insertPlainText(ex)
109:             books_textedit.repaint()
110:
111:     worker_thread = None
112:     def author_slot():
113:         nonlocal worker_thread
114:         author = author_lineedit.text()
115:         if worker_thread is not None:
116:             worker_thread.stop()
117:         worker_thread = WorkerThread(server_url, author,
118:                                     handle_response)
119:         worker_thread.start()
120:
121:     debounce_timer = None
122:     def debounced_author_slot():
123:         nonlocal debounce_timer
124:         if debounce_timer is not None:
125:             debounce_timer.cancel()
126:         debounce_timer = threading.Timer(0.5, author_slot)
127:         debounce_timer.start()
128:
129:     author_lineedit.textChanged.connect(debounced_author_slot)
130:

```

**PennyPyqt/pennydesktop6debounce.py (Page 3 of 3)**

```
131:     # Start up.
132:
133:     window.show()
134:     author_slot() # Populate books_textedit initially.
135:     sys.exit(app.exec_())
136:
137: if __name__ == '__main__':
138:     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