

**books.json (Page 1 of 1)**

```
1: [  
2:   { "author": "Kernighan",  
3:     "title": "The Practice of Programming",  
4:     "price": 40.74,  
5:     "currency": "dollars"  
6:   },  
7:   { "author": "Kernighan",  
8:     "title": "The C Programming Language",  
9:     "price": 24.99,  
10:    "currency": "dollars"  
11:  },  
12:  { "author": "Sedgewick",  
13:    "title": "Algorithms in C",  
14:    "price": 61.59,  
15:    "currency": "dollars"  
16:  }  
17: ]
```

**blank (Page 1 of 1)**

```
1: This page is intentionally blank.
```

## writebooksjson.py (Page 1 of 1)

```

1: #!/usr/bin/env python
2:
3: #-----
4: # writebooksjson.py
5: # Author: Bob Dondero
6: #-----
7:
8: import sys
9: import json
10:
11: #-----
12:
13: def main():
14:
15:     try:
16:         with open('books.json', 'r', encoding='UTF-8') as json_file:
17:             json_doc = json_file.read()
18:
19:             books = json.loads(json_doc)
20:
21:             for book in books:
22:                 print('Author:', book['author'])
23:                 print('Title:', book['title'])
24:                 print('Price:', book['price'], book['currency'])
25:                 print()
26:
27:     except Exception as ex:
28:         print(ex, file=sys.stderr)
29:         sys.exit(1)
30:
31: #-----
32:
33: if __name__ == '__main__':
34:     main()

```

## writebooksjson.js (Page 1 of 1)

```

1: //-----
2: // writebooksjson.js
3: // Author: Bob Dondero
4: //-----
5:
6: 'use strict';
7:
8: let fs = require('fs');
9:
10: //-----
11:
12: function parseFile(err, jsonDoc) {
13:     if (err) {
14:         process.stderr.write(err.message + '\n');
15:         process.exit(1);
16:     }
17:
18:     // JSON to JavaScript:
19:     let books = JSON.parse(jsonDoc);
20:
21:     for (let book of books) {
22:         process.stdout.write('Author: ' + book.author + '\n');
23:         process.stdout.write('Title: ' + book.title + '\n');
24:         process.stdout.write('Price: ' + book.price + ' ' +
25:             book.currency + '\n');
26:         process.stdout.write('\n');
27:     }
28: }
29:
30: //-----
31:
32: function main() {
33:     fs.readFile('books.json', 'utf-8', parseFile);
34: }
35:
36: //-----
37:
38: if (require.main === module)
39:     main();

```

## roundtripjson.py (Page 1 of 1)

```

1: #!/usr/bin/env python
2:
3: #-----
4: # roundtripjson.py
5: # Author: Bob Dondero
6: #-----
7:
8: import sys
9: import json
10:
11: #-----
12:
13: def main():
14:
15:     if len(sys.argv) != 2:
16:         print('Usage: ' + sys.argv[0] + ' file', file=sys.stderr)
17:         sys.exit(1)
18:
19:     json_file_name = sys.argv[1]
20:
21:     try:
22:         with open(json_file_name, 'r', encoding='UTF-8') as json_file:
23:             json_doc = json_file.read()
24:
25:             # JSON to Python object tree:
26:             object_tree = json.loads(json_doc)
27:
28:             # Python object tree to JSON:
29:             json_doc = json.dumps(object_tree)
30:
31:             # Show that it worked:
32:             print(json_doc)
33:
34:     except Exception as ex:
35:         print(ex, file=sys.stderr)
36:         sys.exit(1)
37:
38: #-----
39:
40: if __name__ == '__main__':
41:     main()

```

## roundtripjson.js (Page 1 of 1)

```

1: //-----
2: // roundtripjson.js
3: // Author: Bob Dondero
4: //-----
5:
6: 'use strict';
7:
8: const fs = require('fs');
9:
10: //-----
11:
12: function parseFile(err, jsonDoc) {
13:     if (err) {
14:         process.stderr.write(err.message + '\n');
15:         process.exit(1);
16:     }
17:
18:     // JSON to JavaScript object tree:
19:     let object_tree = JSON.parse(jsonDoc);
20:
21:     // JavaScript object tree to JSON:
22:     jsonDoc = JSON.stringify(object_tree);
23:
24:     // Show that it worked:
25:     process.stderr.write(jsonDoc + '\n');
26: }
27:
28: //-----
29:
30: function main() {
31:     if (process.argv.length !== 3) {
32:         process.stderr.write('Usage: ' + process.argv[0] + ' '
33:             + process.argv[1] + ' file\n');
34:         process.exit(1);
35:     }
36:
37:     let jsonFileName = process.argv[2];
38:
39:     fs.readFile(jsonFileName, 'utf-8', parseFile);
40: }
41:
42: if (require.main === module)
43:     main();

```

## PennyjQuery3/book.py (Page 1 of 1)

```
1: #!/usr/bin/env python
2:
3: #-----
4: # book.py
5: # Author: Bob Dondero
6: #-----
7:
8: class Book:
9:
10:     def __init__(self, author, title, price):
11:         self._author = author
12:         self._title = title
13:         self._price = price
14:
15:     def get_author(self):
16:         return self._author
17:
18:     def get_title(self):
19:         return self._title
20:
21:     def get_price(self):
22:         return self._price
23:
24:     def to_tuple(self):
25:         return (self._author, self._title, self._price)
26:
27: #-----
28:
29: def _test():
30:     book = Book('Kernighan', 'The Practice of Programming', 40.74)
31:     print(book.get_author())
32:     print(book.get_title())
33:     print(book.get_price())
34:     print(book.to_tuple())
35:
36: if __name__ == '__main__':
37:     _test()
```

## blank (Page 1 of 1)

```
1: This page is intentionally blank.
```

## PennyJQuery3/penny.py (Page 1 of 1)

```

1: #!/usr/bin/env python
2:
3: #-----
4: # penny.py
5: # Author: Bob Dondero
6: #-----
7:
8: import flask
9: import database
10:
11: #-----
12:
13: app = flask.Flask(__name__, template_folder='.')
14:
15: #-----
16:
17: @app.route('/', methods=['GET'])
18: @app.route('/index', methods=['GET'])
19: def index():
20:
21:     html_code = flask.render_template('index.html')
22:     response = flask.make_response(html_code)
23:     return response
24:
25: #-----
26:
27: @app.route('/searchresults', methods=['GET'])
28: def search_results():
29:
30:     author = flask.request.args.get('author')
31:     if author is None:
32:         author = ''
33:     author = author.strip()
34:
35:     if author == '':
36:         books = []
37:     else:
38:         books = database.get_books(author) # Exception handling omitted
39:
40:     html_code = flask.render_template('books.html', books=books)
41:     response = flask.make_response(html_code)
42:     return response

```

## PennyJQuery3/books.html (Page 1 of 1)

```

1: {% for book in books: %}
2:     <strong>{{book.get_author()}}</strong>:
3:     {{book.get_title()}}
4:     (${{'%'.2f' % book.get_price()}})
5:     <br>
6: {% endfor %}

```

## PennyjQuery3/index.html (Page 1 of 2)

```

1: <!DOCTYPE html>
2: <html>
3:   <head>
4:     <title>Penny.com</title>
5:   </head>
6:
7:   <body>
8:     <hr>
9:     Good <span id="ampmSpan"></span> and welcome to
10:    <strong>Penny.com</strong>
11:    <hr>
12:
13:    <h1>Author Search</h1>
14:    Please enter an author name:
15:    <input type="text" id="authorInput" autoFocus>
16:    <hr>
17:    <div id="resultsDiv"></div>
18:
19:    <hr>
20:    Date and time: <span id="datetimeSpan"></span><br>
21:    Created by <a href="https://www.cs.princeton.edu/~rdondero">
22:    Bob Dondero</a>
23:    <hr>
24:
25:    <script src=
26:    "https://ajax.googleapis.com/ajax/libs/jquery/3.7.1/jquery.min.js">
27:    </script>
28:
29:    <!-- <script src="static/jquery-3.7.1.min.js"></script> -->
30:
31:    <script>
32:
33:      'use strict';
34:
35:      function getAmPm() {
36:        let dateTime = new Date();
37:        let hours = dateTime.getHours();
38:        let amPm = 'morning';
39:        if (hours >= 12)
40:          amPm = 'afternoon';
41:        $('#ampmSpan').html(amPm);
42:      }
43:
44:      function getDateTime() {
45:        let dateTime = new Date();
46:        $('#datetimeSpan').html(dateTime.toLocaleString());
47:      }
48:
49:      function handleResponse(data) {
50:        $('#resultsDiv').html(data);
51:      }
52:
53:      function handleError(request) {
54:        if (request.statusText !== 'abort')
55:          alert('Error: Failed to fetch data from server');
56:      }
57:
58:      let request = null;
59:
60:      function getResults() {
61:        let author = $('#authorInput').val();
62:        let encodedAuthor = encodeURIComponent(author);
63:        let url = '/searchresults?author=' + encodedAuthor;
64:        if (request != null)
65:          request.abort();

```

## PennyjQuery3/index.html (Page 2 of 2)

```

66:        let requestData = {
67:          type: 'GET',
68:          url: url,
69:          success: handleResponse,
70:          error: handleError
71:        };
72:        request = $.ajax(requestData);
73:      }
74:
75:      let timer = null;
76:
77:      function debouncedGetResults() {
78:        clearTimeout(timer);
79:        timer = setTimeout(getResults, 500);
80:      }
81:
82:      function setup() {
83:        getAmPm();
84:        window.setInterval(getAmPm, 1000);
85:        getDateTime();
86:        window.setInterval(getDateTime, 1000);
87:        $('#authorInput').on('input', debouncedGetResults);
88:      }
89:
90:      $('document').ready(setup);
91:
92:    </script>
93:  </body>
94: </html>

```

## PennyJQueryXml/book.py (Page 1 of 1)

```

1: #!/usr/bin/env python
2:
3: #-----
4: # book.py
5: # Author: Bob Dondero
6: #-----
7:
8: class Book:
9:
10:     def __init__(self, author, title, price):
11:         self._author = author
12:         self._title = title
13:         self._price = price
14:
15:     def get_author(self):
16:         return self._author
17:
18:     def get_title(self):
19:         return self._title
20:
21:     def get_price(self):
22:         return self._price
23:
24:     def to_xml(self):
25:         pattern = '<book>'
26:         pattern += '<author><![CDATA[%s]]></author>'
27:         pattern += '<title><![CDATA[%s]]></title>'
28:         pattern += '<price><![CDATA[%f]]></price>'
29:         pattern += '</book>'
30:         return pattern % (self._author, self._title, self._price)
31:
32: #-----
33:
34: def _test():
35:     book = Book('Kernighan', 'The Practice of Programming', 40.74)
36:     print(book.to_xml())
37:
38: if __name__ == '__main__':
39:     _test()

```

## PennyJQueryXml/penny.py (Page 1 of 1)

```

1: #!/usr/bin/env python
2:
3: #-----
4: # penny.py
5: # Author: Bob Dondero
6: #-----
7:
8: import flask
9: import database
10:
11: #-----
12:
13: app = flask.Flask(__name__, template_folder='.')
14:
15: #-----
16:
17: @app.route('/', methods=['GET'])
18: @app.route('/index', methods=['GET'])
19: def index():
20:
21:     html_code = flask.render_template('index.html')
22:     response = flask.make_response(html_code)
23:     return response
24:
25: #-----
26:
27: @app.route('/searchresults', methods=['GET'])
28: def search_results():
29:
30:     xml_doc = '<?xml version="1.0"?>'
31:
32:     author = flask.request.args.get('author')
33:     if author is None:
34:         author = ''
35:     author = author.strip()
36:
37:     if author == '':
38:         books = []
39:     else:
40:         books = database.get_books(author) # Exception handling omitted
41:
42:     xml_doc += '<books>'
43:     for book in books:
44:         xml_doc += book.to_xml()
45:     xml_doc += '</books>'
46:
47:     response = flask.make_response(xml_doc)
48:     response.headers['Content-Type'] = 'application/xml'
49:     return response

```

## PennyjQueryXml/index.html (Page 1 of 2)

```

1: <!DOCTYPE html>
2: <html>
3:   <head>
4:     <title>Penny.com</title>
5:   </head>
6:
7:   <body>
8:     <hr>
9:     Good <span id="ampmSpan"></span> and welcome to
10:    <strong>Penny.com</strong>
11:    <hr>
12:
13:    <h1>Author Search</h1>
14:    Please enter an author name:
15:    <input type="text" id="authorInput" autoFocus>
16:    <hr>
17:    <div id="resultsDiv"></div>
18:
19:    <hr>
20:    Date and time: <span id="datetimeSpan"></span><br>
21:    Created by <a href="https://www.cs.princeton.edu/~rdondero">
22:    Bob Dondero</a>
23:    <hr>
24:
25:    <script src=
26:    "https://ajax.googleapis.com/ajax/libs/jquery/3.7.1/jquery.min.js">
27:    </script>
28:
29:    <!-- <script src="static/jquery-3.7.1.min.js"></script> -->
30:
31:    <script>
32:
33:      'use strict';
34:
35:      function getAmPm() {
36:        let dateTime = new Date();
37:        let hours = dateTime.getHours();
38:        let amPm = 'morning';
39:        if (hours >= 12)
40:          amPm = 'afternoon';
41:        $('#ampmSpan').html(amPm);
42:      }
43:
44:      function getDateTime() {
45:        let dateTime = new Date();
46:        $('#datetimeSpan').html(dateTime.toLocaleString());
47:      }
48:
49:      function handleResponse(xmlDomTree) {
50:        try {
51:          let resultsNode = $('#resultsDiv');
52:          resultsNode.text('');
53:
54:          let books = xmlDomTree.getElementsByTagName('book');
55:          for (let book of books) {
56:            let authorNode = book.childNodes[0];
57:            let author = authorNode.childNodes[0].nodeValue
58:
59:            let titleNode = book.childNodes[1];
60:            let title = titleNode.childNodes[0].nodeValue
61:
62:            let priceNode = book.childNodes[2];
63:            let price = priceNode.childNodes[0].nodeValue;
64:            price = parseFloat(price).toFixed(2);
65:

```

## PennyjQueryXml/index.html (Page 2 of 2)

```

66:          let strongNode = $('<strong>', {text: author});
67:          let textNode = document.createTextNode(': ' + title
68:          + ' ($' + price + ')');
69:          let brNode = $('<br>');
70:
71:          resultsNode.append(strongNode);
72:          resultsNode.append(textNode);
73:          resultsNode.append(brNode);
74:        }
75:      }
76:    } catch (e) {
77:      $('#resultsDiv').text(e);
78:    }
79:  }
80:
81:  function handleError(request) {
82:    if (request.statusText !== 'abort')
83:      alert('Error: Failed to fetch data from server');
84:  }
85:
86:  let request = null;
87:
88:  function getResults() {
89:    let author = $('#authorInput').val();
90:    let encodedAuthor = encodeURIComponent(author);
91:    let url = '/searchresults?author=' + encodedAuthor;
92:    if (request !== null)
93:      request.abort();
94:    let requestData = {
95:      type: 'GET',
96:      url: url,
97:      success: handleResponse,
98:      error: handleError
99:    };
100:    request = $.ajax(requestData);
101:  }
102:
103:  let timer = null;
104:
105:  function debouncedGetResults() {
106:    clearTimeout(timer);
107:    timer = setTimeout(getResults, 500);
108:  }
109:
110:  function setup() {
111:    getAmPm();
112:    window.setInterval(getAmPm, 1000);
113:    getDateTime();
114:    window.setInterval(getDateTime, 1000);
115:    $('#authorInput').on('input', debouncedGetResults);
116:  }
117:
118:  $('document').ready(setup);
119:
120: </script>
121: </body>
122: </html>

```

## PennyjQueryJson/book.py (Page 1 of 1)

```

1: #!/usr/bin/env python
2:
3: #-----
4: # book.py
5: # Author: Bob Dondero
6: #-----
7:
8: class Book:
9:
10:     def __init__(self, author, title, price):
11:         self._author = author
12:         self._title = title
13:         self._price = price
14:
15:     def get_author(self):
16:         return self._author
17:
18:     def get_title(self):
19:         return self._title
20:
21:     def get_price(self):
22:         return self._price
23:
24:     def to_dict(self):
25:         return {'author': self._author, 'title': self._title,
26:               'price': self._price}
27:
28: #-----
29:
30: def _test():
31:     book = Book('Kernighan', 'The Practice of Programming', 40.74)
32:     print(book.to_dict())
33:
34: if __name__ == '__main__':
35:     _test()

```

## PennyjQueryJson/penny.py (Page 1 of 1)

```

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

```

## PennyjQueryJson/index.html (Page 1 of 2)

```

1: <!DOCTYPE html>
2: <html>
3:   <head>
4:     <title>Penny.com</title>
5:   </head>
6:   <body>
7:     <hr>
8:     Good <span id="ampmSpan"></span> and welcome to
9:     <strong>Penny.com</strong>
10:    <hr>
11:
12:    <h1>Author Search</h1>
13:    Please enter an author name:
14:    <input type="text" id="authorInput" autoFocus>
15:    <hr>
16:    <div id="resultsDiv"></div>
17:
18:    <hr>
19:    Date and time: <span id="datetimeSpan"></span><br>
20:    Created by <a href="https://www.cs.princeton.edu/~rdondero">
21:    Bob Dondero</a>
22:    <hr>
23:
24:    <script src=
25:    "https://ajax.googleapis.com/ajax/libs/jquery/3.7.1/jquery.min.js">
26:    </script>
27:
28:    <!-- <script src="static/jquery-3.7.1.min.js"></script> -->
29:
30:    <script>
31:
32:      'use strict';
33:
34:      function getAmPm() {
35:        let dateTime = new Date();
36:        let hours = dateTime.getHours();
37:        let amPm = 'morning';
38:        if (hours >= 12)
39:          amPm = 'afternoon';
40:        $('#ampmSpan').html(amPm);
41:      }
42:
43:      function getDateTime() {
44:        let dateTime = new Date();
45:        $('#datetimeSpan').html(dateTime.toLocaleString());
46:      }
47:
48:      function handleResponse(books) {
49:        try {
50:          let resultsNode = $('#resultsDiv');
51:          resultsNode.text('');
52:
53:          for (let book of books) {
54:            let author = book['author'];
55:            let title = book['title'];
56:            let price = book['price'].toFixed(2);
57:
58:            let strongNode = $('<strong>', {text: author});
59:            let textNode = document.createTextNode(': ' + title
60:            + ' ($' + price + ')');
61:            let brNode = $('<br>');
62:
63:            resultsNode.append(strongNode);
64:            resultsNode.append(textNode);
65:

```

## PennyjQueryJson/index.html (Page 2 of 2)

```

66:          resultsNode.append(brNode);
67:        }
68:      }
69:      catch (e) {
70:        $('#resultsDiv').text(e);
71:      }
72:    }
73:
74:    function handleError(request) {
75:      if (request.statusText !== 'abort')
76:        alert('Error: Failed to fetch data from server');
77:    }
78:
79:    let request = null;
80:
81:    function getResults() {
82:      let author = $('#authorInput').val();
83:      let encodedAuthor = encodeURIComponent(author);
84:      let url = '/searchresults?author=' + encodedAuthor;
85:      if (request != null)
86:        request.abort();
87:      let requestData = {
88:        type: 'GET',
89:        url: url,
90:        success: handleResponse,
91:        error: handleError
92:      };
93:      request = $.ajax(requestData);
94:    }
95:
96:    let timer = null;
97:
98:    function debouncedGetResults() {
99:      clearTimeout(timer);
100:      timer = setTimeout(getResults, 500);
101:    }
102:
103:    function setup() {
104:      getAmPm();
105:      window.setInterval(getAmPm, 1000);
106:      getDateTime();
107:      window.setInterval(getDateTime, 1000);
108:      $('#authorInput').on('input', debouncedGetResults);
109:    }
110:
111:    $('document').ready(setup);
112:
113:    </script>
114:  </body>
115: </html>

```

**booksbyauthorjson.py (Page 1 of 1)**

```

1: #!/usr/bin/env python
2:
3: #-----
4: # booksbyauthorjson.py
5: # Author: Bob Dondero
6: #-----
7:
8: import sys
9: import urllib.request
10: import json
11:
12: def main():
13:
14:     if len(sys.argv) != 4:
15:         print('Usage: python %s host port author' % sys.argv[0])
16:         sys.exit(1)
17:
18:     try:
19:         host = sys.argv[1]
20:         port = sys.argv[2]
21:         author = sys.argv[3]
22:
23:         url = 'http://' + host + ':' + port
24:         url += '/searchresults?author=' + author
25:         with urllib.request.urlopen(url) as in_flo:
26:             json_bytes = in_flo.read()
27:
28:             json_doc = json_bytes.decode('utf-8')
29:             books = json.loads(json_doc)
30:             for book in books:
31:                 print(book['author'])
32:                 print(book['title'])
33:                 print(book['price'])
34:                 print()
35:
36:     except Exception as ex:
37:         print(ex, file=sys.stderr)
38:         sys.exit(1)
39:
40: if __name__ == '__main__':
41:     main()

```

**blank (Page 1 of 1)**

```
1: This page is intentionally blank.
```

## booksmalformed.json (Page 1 of 1)

```

1: [
2:   { "author": "Kernighan",
3:     "title": "The Practice of Programming",
4:     "price": 40.74,
5:     "currency": "dollars"
6:   },
7:   { "author": "Kernighan",
8:     "title": "The C Programming Language",
9:     "price": 24.99,
10:    "currency": "dollars"
11:  },
12:  { "author": "Sedgewick",
13:    "title": "Algorithms in C",
14:    "price": 61.59
15:    "currency": "dollars"
16:  }
17: ]

```

## checkjson.py (Page 1 of 1)

```

1: #!/usr/bin/env python
2:
3: #-----
4: # checkjson.py
5: # Author: Bob Dondero
6: #-----
7:
8: import sys
9: import json
10:
11: #-----
12:
13: def main():
14:
15:     if len(sys.argv) != 2:
16:         print('Usage: ' + sys.argv[0] + ' file', file=sys.stderr)
17:         sys.exit(1)
18:
19:     json_file_name = sys.argv[1]
20:
21:     try:
22:         with open(json_file_name, 'r', encoding='UTF-8') as json_file:
23:             json_doc = json_file.read()
24:
25:             json.loads(json_doc)
26:
27:             print('The document is well-formed.')
28:
29:     except Exception as ex:
30:         print(ex, file=sys.stderr)
31:         sys.exit(1)
32:
33: #-----
34:
35: if __name__ == '__main__':
36:     main()

```

**books.schema.json (Page 1 of 1)**

```
1: {
2:   "type": "array",
3:   "items": { "$ref": "#/definitions/book" },
4:
5:   "definitions":
6:   {
7:     "book":
8:     {
9:       "type": "object",
10:      "required": [ "author", "title", "price", "currency" ],
11:      "properties":
12:      {
13:        "author": { "type": "string"},
14:        "title": { "type": "string" },
15:        "price": { "type": "number" },
16:        "currency": { "type": "string" }
17:      }
18:    }
19:  }
20: }
```

**booksinvalid.json (Page 1 of 1)**

```
1: [
2:   { "author": "Kernighan",
3:     "title": "The Practice of Programming",
4:     "price": 40.74,
5:     "currency": "dollars"
6:   },
7:   { "author": "Kernighan",
8:     "title": "The C Programming Language",
9:     "price": 24.99,
10:    "currency": "dollars"
11:  },
12:  { "author": "Sedgewick",
13:    "title": "Algorithms in C",
14:    "price": 61.59
15:  }
16: ]
```

## checkjsonusingschema.py (Page 1 of 1)

```
1: #!/usr/bin/env python
2:
3: #-----
4: # checkjsonusingschema.py
5: # Author: Bob Dondero
6: #-----
7:
8: import sys
9: import json
10: import jsonschema
11:
12: def main():
13:
14:     if len(sys.argv) != 3:
15:         print('Usage: ' + sys.argv[0] + ' json_file schemaFile',
16:               file=sys.stderr)
17:         sys.exit(1)
18:
19:     json_file_name = sys.argv[1]
20:     schema_file_name = sys.argv[2]
21:
22:     try:
23:         with open(json_file_name, mode='r', encoding='utf-8') \
24:             as json_file:
25:             json_doc = json_file.read()
26:
27:             object_tree = json.loads(json_doc)
28:             print('The document is well-formed.')
29:
30:         with open(schema_file_name, mode='r', encoding='utf-8') \
31:             as schema_file:
32:             schema_doc = schema_file.read()
33:
34:             schema = json.loads(schema_doc)
35:
36:             jsonschema.validate(instance=object_tree, schema=schema)
37:             print('The document is valid.')
38:
39:     except Exception as ex:
40:         print(ex, file=sys.stderr)
41:         sys.exit(1)
42:
43: if __name__ == '__main__':
44:     main()
```