

Advanced Database Systems Design

Homework 2

Sarika Sadineni (ssadinen@kent.edu, 811178010)

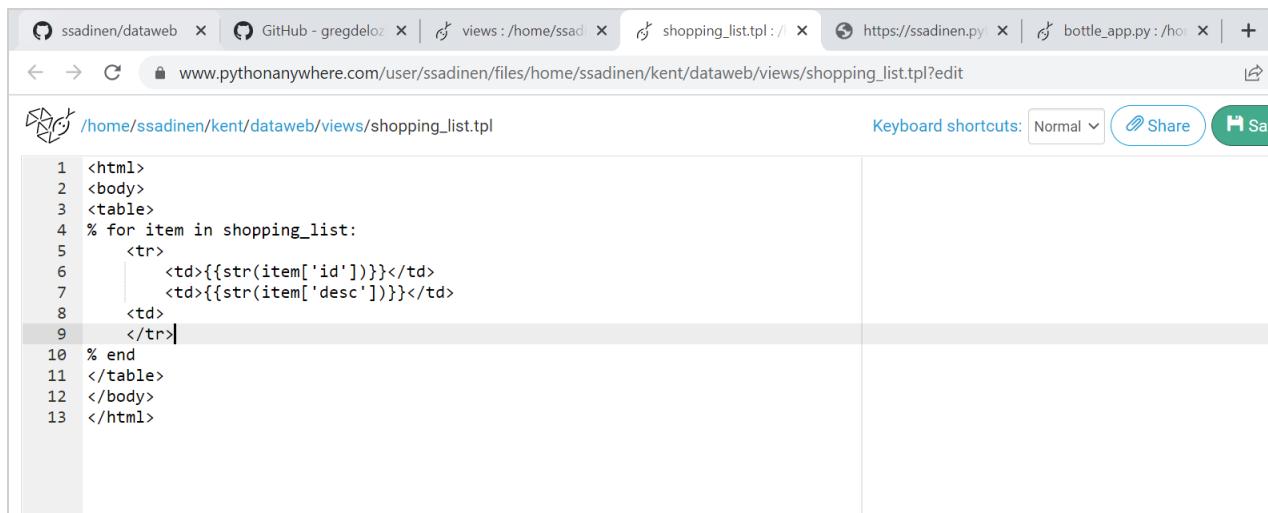
Web app - <https://ssadinen.pythonanywhere.com/>

Git repositories –

<https://github.com/ssadinen/advanced-database-ssadinen>

<https://github.com/ssadinen/dataweb>

- Created a template for Shopping list as below and added table data items – id, desc



```
1 <html>
2 <body>
3 <table>
4 % for item in shopping_list:
5   <tr>
6     <td>{{str(item['id'])}}</td>
7     <td>{{str(item['desc'])}}</td>
8   <td>
9   </td>
10 % end
11 </table>
12 </body>
13 </html>
```

- Created bottle app.py in dataweb directory and added routes hi, bye for testing and route list for the shopping list application

```

1  from bottle import default_app, route, template
2  import sqlite3
3
4  connection = sqlite3.connect("shopping_list.db")
5
6  @route('/')
7  def hello_world():
8      return 'Hello from Sarika Sadineni!'
9
10 @route('/hi')
11 def hi_world():
12     return 'Hi from Sarika Sadineni!'
13
14 @route('/bye')
15 def bye_world():
16     return 'Bye from Sarika Sadineni!'
17
18 @route('/list')
19 def get_list():
20     cursor = connection.cursor()
21     rows = cursor.execute("select id, description from list")
22     rows = list(rows)
23     rows = [ {'id':row[0] , 'desc':row[1]} for row in rows ]
24     return template("shopping_list.tpl", shopping_list=rows)
25
26
27 application = default_app()
28

```

- Below is the 'list' route of the app with the item description and id selected from shopping_list.db

```

1 apples
2 broccoli
3 pizza
4 tangerine
5 potatoes

```

- Created add_item.tpl in web app views to add new items to the shopping list

```

1 <html>
2 <body>
3 <hr/>
4 <form action="add" method="post">
5   <p>New Item:<input name="description"/></p>
6   <p><button type="submit">Submit</button></p>
7 </form>
8 <hr/>
9 </body>
10 </html>

```

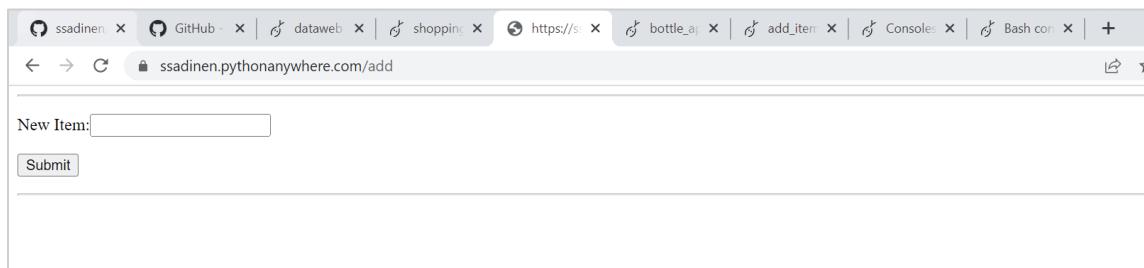
- Also added '/add' get route to the app so when user select on add item they will be taken to new route page /add of web page

```

6
7 @route('/')
8 def hello_world():
9     return 'Hello from Sarika Sadineni!'
10
11 @route('/hi')
12 def hi_world():
13     return 'Hi from Sarika Sadineni!'
14
15 @route('/bye')
16 def bye_world():
17     return 'Bye from Sarika Sadineni!'
18
19 @route('/list')
20 def get_list():
21     cursor = connection.cursor()
22     rows = cursor.execute("select id, description from list")
23     rows = list(rows)
24     rows = [ {'id':row[0], 'desc':row[1]} for row in rows ]
25     return template("shopping_list.tpl", shopping_list=rows)
26
27 @get('/add')
28 def get_add():
29     return template("add_item.tpl")
30

```

- Below is the route to /add a new item to the shopping list



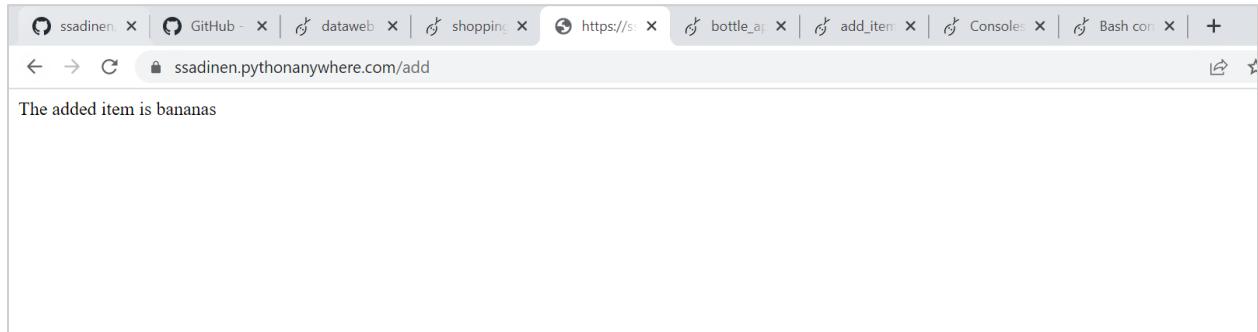
- With just get route we cannot add item to the list we need to add post route as well so the item will get added to the list as soon as user click on submit. So added below lines to /add a new item with post.

```

11 @route('/hi')
12 def hi_world():
13     return 'Hi from Sarika Sadineni!'
14
15 @route('/bye')
16 def bye_world():
17     return 'Bye from Sarika Sadineni!'
18
19 @route('/list')
20 def get_list():
21     cursor = connection.cursor()
22     rows = cursor.execute("select id, description from list")
23     rows = list(rows)
24     rows = [ {'id':row[0], 'desc':row[1]} for row in rows ]
25     return template("shopping_list.tpl", shopping_list=rows)
26
27 @get('/add')
28 def get_add():
29     return template("add_item.tpl")
30
31 @post('/add')
32 def post_add():
33     description = request.forms.get("description")
34     return "The added item is ", description
35

```

- After saving above lines and clicking on run, I've opened webapp and added new item bananas to the list.



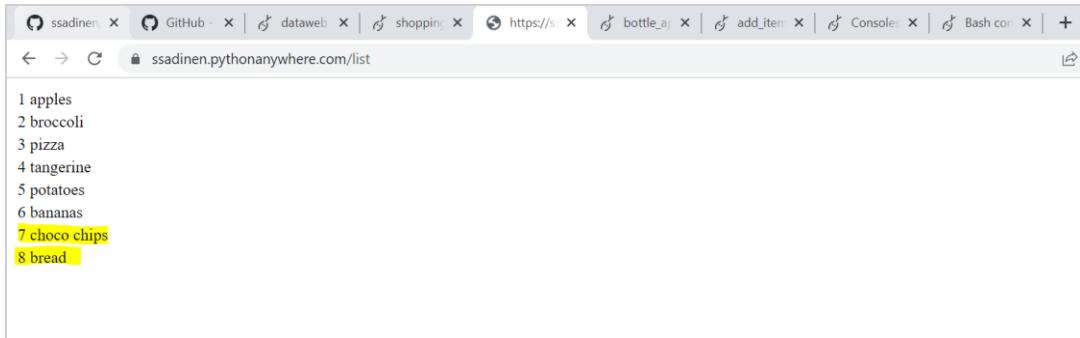
- Added below cursor.execute to insert the new items to the shopping list. Also added redirect, so when we open the webpage without any route we will be default redirected to the /list route

```

14
15 @route('/bye')
16 def bye_world():
17     return 'Bye from Sarika Sadineni!'
18
19 @route('/list')
20 def get_list():
21     cursor = connection.cursor()
22     rows = cursor.execute("select id, description from list")
23     rows = list(rows)
24     rows = [ {'id':row[0] , 'desc':row[1]} for row in rows ]
25     return template("shopping_list.tpl", shopping_list=rows)
26
27 @get('/add')
28 def get_add():
29     return template("add_item.tpl")
30
31 @post('/add')
32 def post_add():
33     description = request.forms.get("description")
34     cursor = connection.cursor()
35     cursor.execute(f"insert into list (description) values ('{description}')")
36     connection.commit()
37     redirect('/list')
38

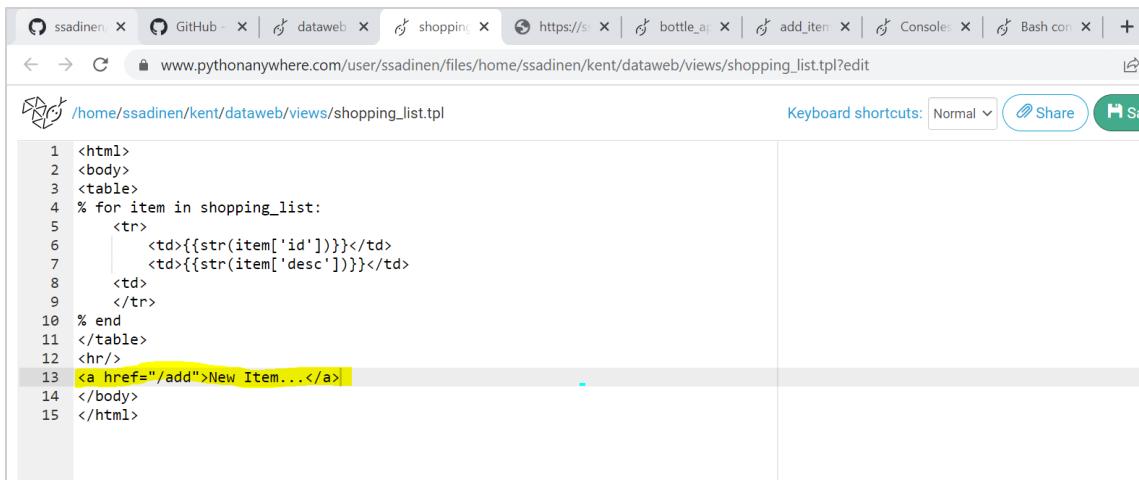
```

- After saving the above lines of code in bottle_app.py and reloading the webpage I have added chocochips and bread to the list as highlighted in below screenshot.



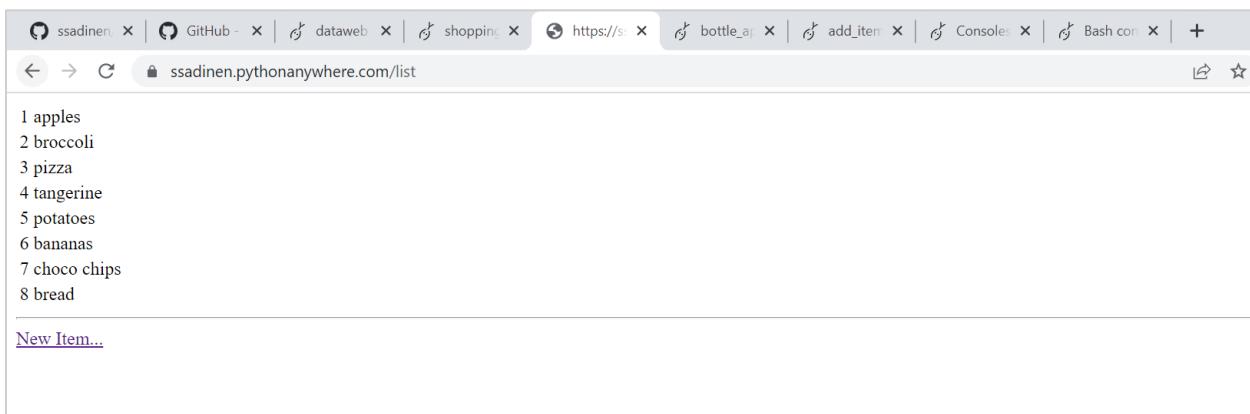
```
1 apples
2 broccoli
3 pizza
4 tangerine
5 potatoes
6 bananas
7 choco chips
8 bread
```

- In the previous step we got to go to /add route page in order to add the new item. In order to avoid going to new page by changing the route and add new item with a link in the same web page as list, I've added href in shopping_list.tpl for new item.



```
1 <html>
2 <body>
3 <table>
4 % for item in shopping_list:
5   <tr>
6     <td>{{str(item['id'])}}</td>
7     <td>{{str(item['desc'])}}</td>
8   <td>
9   </tr>
10 % end
11 </table>
12 <hr/>
13 <a href="/add">New Item...</a>
14 </body>
15 </html>
```

- After saving above template and reloading the bottle_app.py, New Item appears in the Shopping list page itself as shown in below screenshot.



```
1 apples
2 broccoli
3 pizza
4 tangerine
5 potatoes
6 bananas
7 choco chips
8 bread
```

[New Item...](#)

- Created a new route to delete the item from list where id = {id}

```

21     cursor = connection.cursor()
22     rows = cursor.execute("select id, description from list")
23     rows = list(rows)
24     rows = [ {'id':row[0] , 'desc':row[1]} for row in rows ]
25     return template("shopping_list.tpl", shopping_list=rows)
26
27 @get('/add')
28 def get_add():
29     return template("add_item.tpl")
30
31 @post('/add')
32 def post_add():
33     description = request.forms.get("description")
34     cursor = connection.cursor()
35     cursor.execute(f"insert into list (description) values ('{description}')")
36     connection.commit()
37     redirect('/list')
38
39 @route('/delete/<id>')
40 def get_delete(id):
41     cursor = connection.cursor()
42     cursor.execute(f"delete from list where id = {id}")
43     connection.commit()
44     redirect('/list')

```

- Now I went to route /delete/8 to delete item with id 8. As per below screenshot I have deleted item 8 which was bread.

1 apples
2 broccoli
3 pizza
4 tangerine
5 potatoes
6 bananas
7 choco chips

[New Item...](#)

- Now rather than deleting an item by changing the route in web url, I have added href shopping list template to add delete in the shopping list web page itself

```

1 <html>
2 <body>
3 <table>
4 % for item in shopping_list:
5     <tr>
6         <td>{{str(item['id'])}}</td>
7         <td>{{str(item['desc'])}}</td>
8         <td><a href="">del</a></td>
9     <td>
10    </tr>
11 % end
12 </table>
13 <hr/>
14 <a href="/add">New Item...</a>
15 </body>
16 </html>

```

- After saving the above template and reloading the page I ran the web app again and could see delete option across each of the items however this delete is not associated with respective id's of items.

The screenshot shows a web browser window with several tabs open. The active tab is 'ssadinen.pythonanywhere.com/list'. The content of the page is a table with the following data:

1 apples	del
2 broccoli	del
3 pizza	del
4 tangerine	del
5 potatoes	del
6 bananas	del
7 choco chips	del

Below the table, there is a link labeled 'New Item...'. The URL in the address bar is 'ssadinen.pythonanywhere.com/list'.

- To associate the respective id's of items in delete, I've modified the href of delete in shopping list template as below –

The screenshot shows a code editor displaying the file '/home/ssadinen/kent/dataweb/views/shopping_list.tpl'. The content of the file is a Python template for generating an HTML shopping list. The relevant part of the code is:

```

1 <html>
2 <body>
3 <table>
4 % for item in shopping_list:
5   <tr>
6     <td>{{str(item['id'])}}</td>
7     <td>{{str(item['desc'])}}</td>
8     <td><a href="/delete/{{str(item['id'])}}">del</a></td>
9   </td>
10 </tr>
11 % end
12 </table>
13 <hr/>
14 <a href="/add">New Item...</a>
15 </body>
16 </html>

```

A line of code containing '<td>del</td>' is highlighted with a yellow box.

- I then saved the above template and reloaded the web page. After which I could see the id's associated for each of the items in the shopping list. I then deleted bananas which is of id 6.

The screenshot shows a web browser window with multiple tabs open. The active tab is titled "ssadinen.pyt...". The URL in the address bar is ssadinen.pythonanywhere.com/list. The page content displays a list of items with their descriptions and a "del" link next to each item:

Item	Description	Action
1 apples		del
2 broccoli		del
3 pizza		del
4 tangerine		del
5 potatoes		del
7 choco chips		del

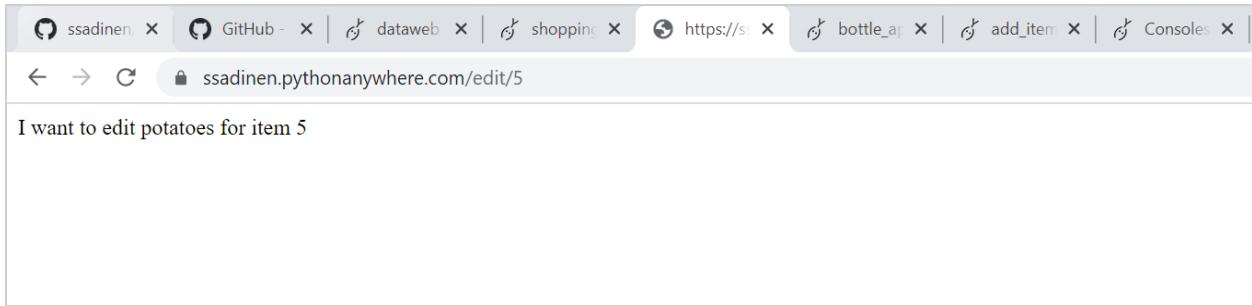
Below the list is a link labeled "New Item...".

- In `bottle_app.py`, I have then added a /get route for edit in order to edit the items description in shopping list.

The screenshot shows a code editor displaying the `bottle_app.py` file. The code contains several routes and database operations:

```
32 - def post_add():
33     description = request.forms.get("description")
34     cursor = connection.cursor()
35     cursor.execute(f"insert into list (description) values ('{description}')")
36     connection.commit()
37     redirect('/list')
38
39 @route('/delete/<id>')
40 - def get_delete(id):
41     cursor = connection.cursor()
42     cursor.execute(f"delete from list where id = {id}")
43     connection.commit()
44     redirect('/list')
45
46 @get('/edit/<id>')
47 - def get_edit(id):
48     cursor = connection.cursor()
49     items = cursor.execute(f"select description from list where id={id}")
50     items = list(items)
51 -     if len(items) != 1:
52 -         redirect('/list')
53     description = items[0][0]
54     return f"I want to edit {description} for item {id}"
55
```

- After saving the above and reloading the web app, I accessed the web page with route edit/5 in order to edit item 5.



- I then modified the shopping list template to directly edit items from shopping list page rather than going to new route.

```
ssadinen GitHub - dataweb shopping https:// bottle_ap add_item Consoles

/home/ssadinen/kent/dataweb/views/shopping_list.tpl

1 <html>
2 <body>
3 <table>
4 % for item in shopping_list:
5     <tr>
6         <td>{{str(item['id'])}}</td>
7         <td>{{str(item['desc'])}}</td>
8         <td><a href="/edit/{{str(item['id'])}}">edit</a></td>
9         <td><a href="/delete/{{str(item['id'])}}">del</a></td>
10        <td>
11    </tr>
12 % end
13 </table>
14 <hr/>
15 <a href="/add">New Item...</a>
16 </body>
17 </html>
```

- After saving above template and reloading the web app, I could see the edit option across each of the shopping list items in the web app.

1	apples	edit del
2	broccoli	edit del
3	pizza	edit del
4	tangerine	edit del
5	potatoes	edit del
7	choco chips	edit del

New Item...

- In order to edit item in the list completely we need to add post method. Hence created a new template of edit_item and included the form action.

- After saving the edit_item template, I have then added post route in bottle_app.py inoder to post the items edited in the shopping list web app.

```

38
39 @route('/delete/<id>')
40 def get_delete(id):
41     cursor = connection.cursor()
42     cursor.execute(f"delete from list where id = {id}")
43     connection.commit()
44     redirect('/list')
45
46 @get('/edit/<id>')
47 def get_edit(id):
48     cursor = connection.cursor()
49     items = cursor.execute(f"select description from list where id={id}")
50     items = list(items)
51     if len(items) != 1:
52         redirect('/list')
53     description = items[0][0]
54     return template("edit_item.tpl", id=id, description=description)
55
56 @post('/edit/<id>')
57 def post_edit(id):
58     description = request.forms.get("description")
59     cursor = connection.cursor()
60     cursor.execute(f"update list set description = '{description}' where id={id}")
61     connection.commit()
62     redirect('/list')
63

```

- After saving the bottle_app.py, I have reloaded the web app and then edited the item broccoli to broccoli florets and clicked on submit

Edit this item..

Edit Item:

[Cancel](#)

- As we can see below I can edit the items description in the shopping list.

1 apples	edit del
2 broccoli florets	edit del
3 pizza	edit del
4 tangerine	edit del
5 potatoes	edit del
7 choco chips	edit del

[New Item...](#)

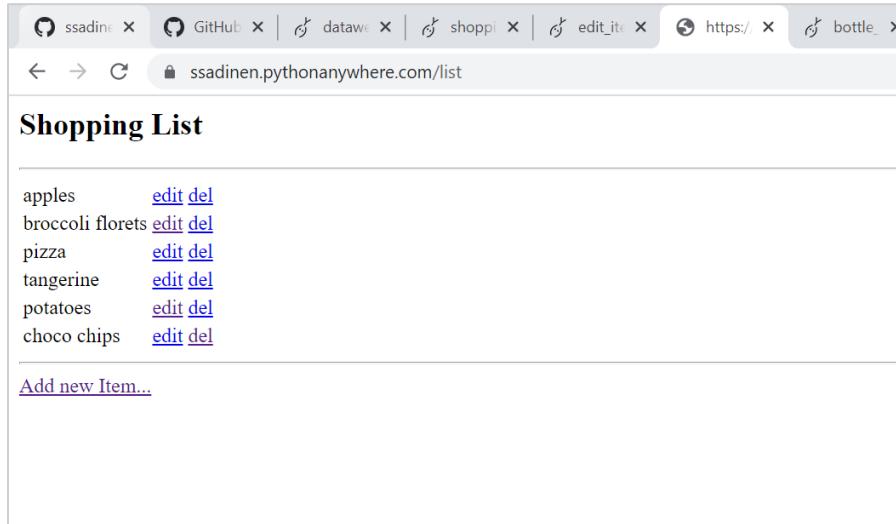
- I then added heading Shopping List and change Add item to Add new item in shopping list template

```

1 <html>
2 <body>
3 <h2>Shopping List</h2>
4 <hr/>
5 <table>
6 % for item in shopping_list:
7   <tr>
8     <td>{{str(item['desc'])}}</td>
9     <td><a href="/edit/{{str(item['id'])}}">edit</a></td>
10    <td><a href="/delete/{{str(item['id'])}}">del</a></td>
11  </td>
12 </tr>
13 % end
14 </table>
15 <hr/>
16 <a href="/add">Add new Item...</a>
17 </body>
18 </html>

```

- After saving the above template and reloading the web page, I can see the heading of and the change to Add New item in the web page.

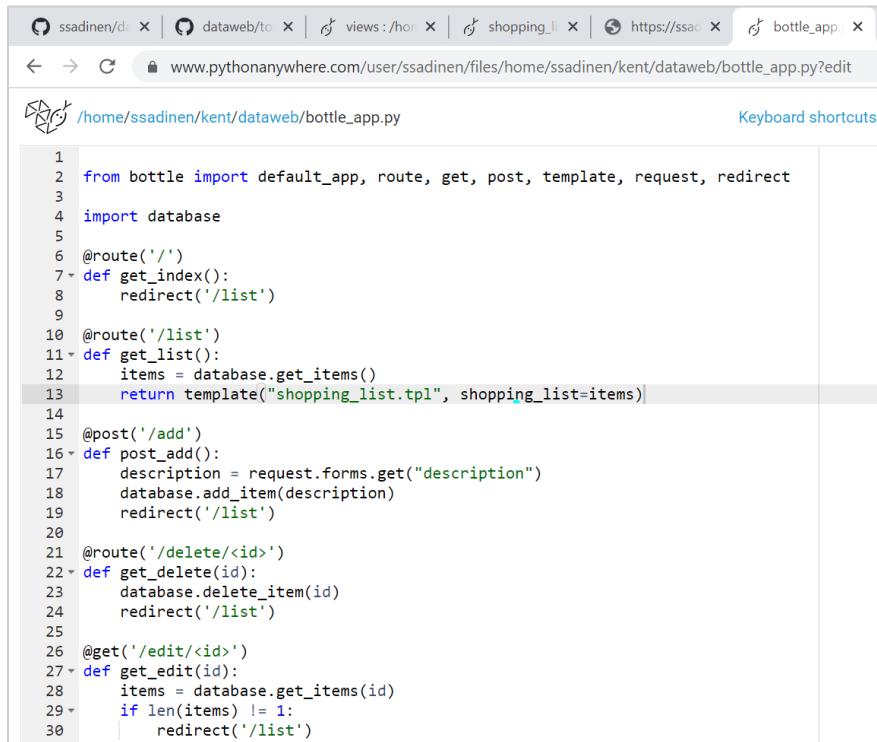


The screenshot shows a web browser window with multiple tabs open. The active tab is titled "edit_it" and displays a URL "https://ssadinen.pythonanywhere.com/list". The content of the page is a "Shopping List" with the following items:

Item	Action
apples	edit del
broccoli florets	edit del
pizza	edit del
tangerine	edit del
potatoes	edit del
choco chips	edit del

At the bottom of the list, there is a link: [Add new Item...](#).

- Deleted the routes for hi, bye. Below shows the lines in bottle_app.py for routes - /list, /add, /delete



```

1  from bottle import default_app, route, get, post, template, request, redirect
2
3  import database
4
5  @route('/')
6  def get_index():
7      redirect('/list')
8
9  @route('/list')
10 def get_list():
11     items = database.get_items()
12     return template("shopping_list.tpl", shopping_list=items)
13
14
15 @post('/add')
16 def post_add():
17     description = request.forms.get("description")
18     database.add_item(description)
19     redirect('/list')
20
21 @route('/delete/<id>')
22 def get_delete(id):
23     database.delete_item(id)
24     redirect('/list')
25
26 @get('/edit/<id>')
27 def get_edit(id):
28     items = database.get_items(id)
29     if len(items) != 1:
30         redirect('/list')

```

- Below screenshot shows lines in bottle_app.py for /update.

```

13     return template("shopping_list.tpl", shopping_list=items)
14
15 @post('/add')
16 def post_add():
17     description = request.forms.get("description")
18     database.add_item(description)
19     redirect('/list')
20
21 @route('/delete<id>')
22 def get_delete(id):
23     database.delete_item(id)
24     redirect('/list')
25
26 @get('/edit<id>')
27 def get_edit(id):
28     items = database.get_items(id)
29     if len(items) != 1:
30         redirect('/list')
31     item_id, description = items[0]['id'], items[0]['desc']
32     assert item_id == int(id)
33     return template("edit_item.tpl", id=id, description=description)
34
35 @post('/edit<id>')
36 def post_edit(id):
37     description = request.forms.get("description")
38     database.update_item(id, description)
39     redirect('/list')
40
41 application = default_app()
42

```

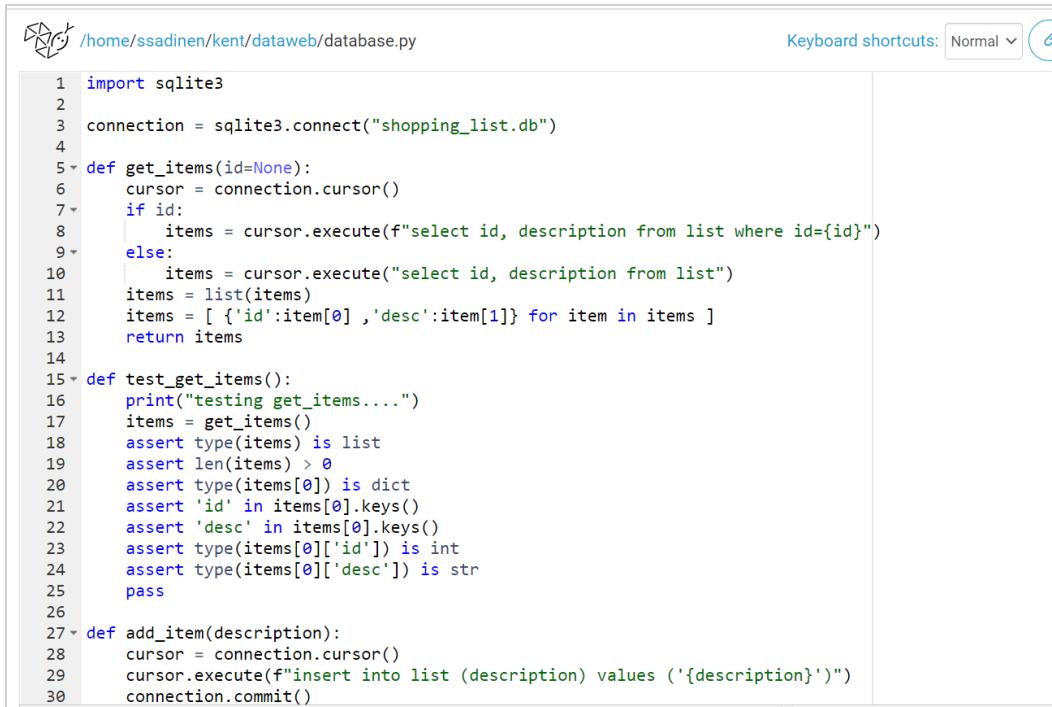
- I then updated shopping_list.tpl to include add new item in the same page as the shopping list.

```

1 <html>
2 <body>
3 <h2>Shopping List</h2>
4 <hr/>
5 <table>
6 % for item in shopping_list:
7     <tr>
8         <td>{{str(item['desc'])}}</td>
9         <td><a href="/edit/{{str(item['id'])}}">edit</a></td>
10        <td><a href="/delete/{{str(item['id'])}}">del</a></td>
11        <td>
12    </tr>
13 % end
14 </table>
15 <hr/>
16 <form action="add" method="post">
17     <p>Add new item: <input name="description"/></p>
18     <p><button type="submit">Submit</button></p>
19 </form>
20 </body>
21 </html>

```

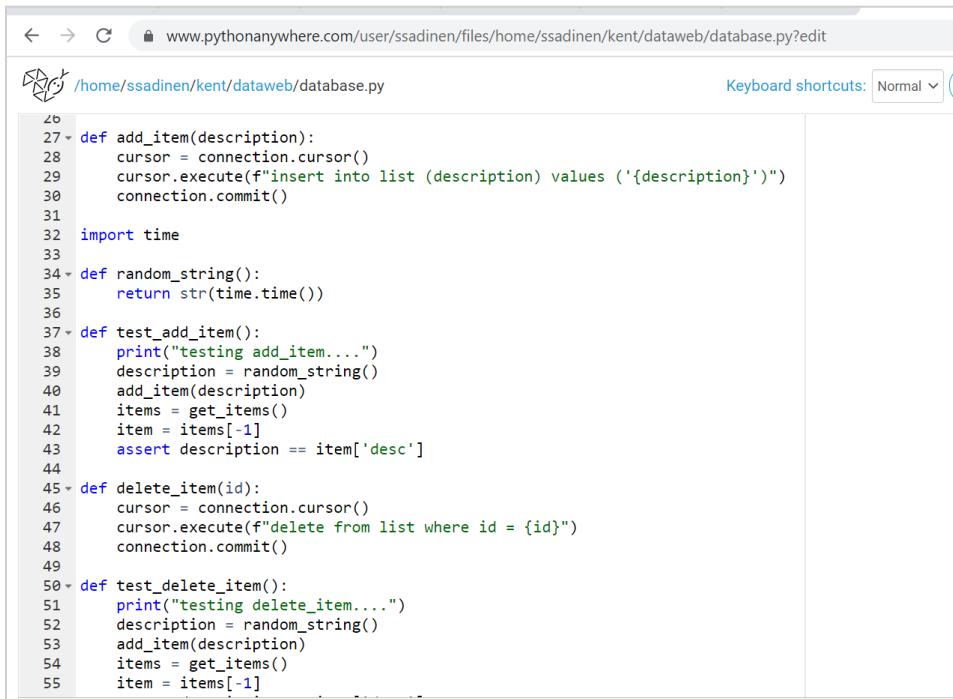
- I've created database.py to separate the database related code from app related code. To test if each of get, add, delete, update are working fine, I've included below test_get_items() to test if get items is working fine.



A screenshot of a Python code editor showing a file named `database.py`. The code defines a class with methods for getting items from a SQLite database and testing those methods. It also includes methods for adding and deleting items.

```
1 import sqlite3
2
3 connection = sqlite3.connect("shopping_list.db")
4
5 def get_items(id=None):
6     cursor = connection.cursor()
7     if id:
8         items = cursor.execute(f"select id, description from list where id={id}")
9     else:
10        items = cursor.execute("select id, description from list")
11    items = list(items)
12    items = [ {'id':item[0] , 'desc':item[1]} for item in items ]
13    return items
14
15 def test_get_items():
16     print("testing get_items....")
17     items = get_items()
18     assert type(items) is list
19     assert len(items) > 0
20     assert type(items[0]) is dict
21     assert 'id' in items[0].keys()
22     assert 'desc' in items[0].keys()
23     assert type(items[0]['id']) is int
24     assert type(items[0]['desc']) is str
25     pass
26
27 def add_item(description):
28     cursor = connection.cursor()
29     cursor.execute(f"insert into list (description) values ('{description}')")
30     connection.commit()
31
32 import time
33
34 def random_string():
35     return str(time.time())
36
37 def test_add_item():
38     print("testing add_item....")
39     description = random_string()
40     add_item(description)
41     items = get_items()
42     item = items[-1]
43     assert description == item['desc']
44
45 def delete_item(id):
46     cursor = connection.cursor()
47     cursor.execute(f"delete from list where id = {id}")
48     connection.commit()
49
50 def test_delete_item():
51     print("testing delete_item....")
52     description = random_string()
53     add_item(description)
54     items = get_items()
55     item = items[-1]
```

- I've included below `test_add_item()` to test if add item is working fine.



A screenshot of a Python code editor showing the same `database.py` file with additional test functions for adding and deleting items. The new code includes imports for `time` and a `random_string` helper function, along with the `test_add_item` and `test_delete_item` methods.

```
26
27 def add_item(description):
28     cursor = connection.cursor()
29     cursor.execute(f"insert into list (description) values ('{description}')")
30     connection.commit()
31
32 import time
33
34 def random_string():
35     return str(time.time())
36
37 def test_add_item():
38     print("testing add_item....")
39     description = random_string()
40     add_item(description)
41     items = get_items()
42     item = items[-1]
43     assert description == item['desc']
44
45 def delete_item(id):
46     cursor = connection.cursor()
47     cursor.execute(f"delete from list where id = {id}")
48     connection.commit()
49
50 def test_delete_item():
51     print("testing delete_item....")
52     description = random_string()
53     add_item(description)
54     items = get_items()
55     item = items[-1]
```

- I've included below `test_delete_item()` to test if delete item is working fine

```
 49
50 def test_delete_item():
51     print("testing delete_item....")
52     description = random_string()
53     add_item(description)
54     items = get_items()
55     item = items[-1]
56     assert description == item['desc']
57     delete_item(item['id'])
58     items = get_items()
59     for item in items:
60         assert description != item['desc']
61
62 def update_item(id, description):
63     cursor = connection.cursor()
64     cursor.execute(f"update list set description = '{description}' where id={id}")
65     connection.commit()
66
67 def test_update_item():
68     print("testing update_item....")
69     description = random_string()
70     add_item(description)
71     items = get_items()
72     item = items[-1]
73     id = str(item['id'])
74     description = item['desc']
75     new_description = description.replace("1","9").replace(".",":")
76     update_item(id, new_description)
77     items = get_items()
78     new_found = False
79     for item in items:
```

- I've included `test_update_item()` to test if update item is working fine.

ssadinen/d | dataweb/tc | views:/hor | shopping_l | https://ssadinen.pythonanywhere.com/ | bottle_app | database.py

```
 63     cursor = connection.cursor()
64     cursor.execute(f"update list set description = '{description}' where id={id}")
65     connection.commit()
66
67 def test_update_item():
68     print("testing update_item....")
69     description = random_string()
70     add_item(description)
71     items = get_items()
72     item = items[-1]
73     id = str(item['id'])
74     description = item['desc']
75     new_description = description.replace("1","9").replace(".",":")
76     update_item(id, new_description)
77     items = get_items()
78     new_found = False
79     for item in items:
80         if item['id'] == int(id):
81             assert item['desc'] == new_description
82             new_found = True
83         assert item['desc'] != description
84     assert new_found
85
86
87 if __name__ == "__main__":
88     test_get_items()
89     test_add_item()
90     test_delete_item()
91     test_update_item()
92     print("done.")
```

- I then opened the bash console in `database.py` and ran `python3 database.py` to check if the testing rows are printed.

```

63     cursor = connection.cursor()
64     cursor.execute(f"update list set description = '{description}' where id={id}")
65     connection.commit()
66
67 def test_update_item():
68     print("testing update_item....")
69     description = random_string()
70     add_item(description)
71     items = get_items()
72     item = items[-1]
73     id = str(item['id'])
74     description = item['desc']
75     new_description = description.replace("1","9").replace(".","");
76     update_item(id, new_description)
77     items = get_items()
78     new_found = False
79     for item in items:
80         if item['id'] == int(id):
81             assert item['desc'] == new_description
82             new_found = True
83     assert item['desc'] != description
84
85 testing get_items....
86 testing add_item....
87 testing delete_item....
88 done.
89 21:13 ~/kent/dataweb (main)$ python3 database.py
90 testing get_items....
91 testing add_item....
92 testing delete_item....
93 testing update_item....
94 done.
95 21:21 ~/kent/dataweb (main)$

```

- Now I've saved the above database.py, bottle_app.py and the reloaded the webpage to check if the web app is displaying all the shopping list items, form correctly. I've added granola bar, salad bowls, tomatoes, peanuts for testing at various levels changes of get, add, update, delete functions.

The screenshot shows a web browser window with the URL ssadinen.pythonanywhere.com/list. The page title is "Shopping List". Below the title, there is a table-like list of items:

apples	edit	del
broccoli florets	edit	del
tangerine	edit	del
chocochips	edit	del
granola bar	edit	del
salad bowls	edit	del
tomatoes	edit	del
peanuts	edit	del

Below the table, there is a text input field labeled "Add new item:" and a "Submit" button.

- I then copied all the files to a new directory 02-DB-Extraction.

```

1:16 ~ $ ls
README.txt kent mysite
15:22 ~ $ cd kent
15:23 ~/kent $ ls
advanced-database-ssadinien dataweb
15:24 ~/kent $ cd dataweb
15:24 ~/kent/dataweb (main)$ ls
01-web-Intro README.md __pycache__ bottle_app.py database.py list.db query.py setup.py shopping_list.db views
15:24 ~/kent/dataweb (main)$ mkdir 02-DB-Abstraction
15:24 ~/kent/dataweb (main)$ cd *DB*
15:25 ~/kent/dataweb/02-DB-Abstraction (main)$ cd ..
bash: cd.: command not found
15:25 ~/kent/dataweb/02-DB-Abstraction (main)$ cd
15:25 ~/kent/dataweb/
bash: cd: /kent/dataweb/: No such file or directory
15:26 ~/kent/dataweb/
15:26 ~/kent/dataweb (main)$ ls
01-web-Intro 02-DB-Abstraction README.md __pycache__ bottle_app.py database.py list.db query.py setup.py shopping_list.db views
15:26 ~/kent/dataweb (main)$ cp -r RE* *.py *.db views 01-DB-Abstraction
cp: target '01-DB-Abstraction' is not a directory
15:27 ~/kent/dataweb (main)$ cp -r RE* *.py *.db views 02-DB-Abstraction
15:27 ~/kent/dataweb (main)$ ls
01-web-Intro 02-DB-Abstraction README.md __pycache__ bottle_app.py database.py list.db query.py setup.py shopping_list.db views
15:28 ~/kent/dataweb/02-DB-Abstraction (main)$ ls
README.md bottle_app.py database.py list.db query.py setup.py shopping_list.db views
15:28 ~/kent/dataweb/02-DB-Abstraction/views (main)$ cd views
15:28 ~/kent/dataweb/02-DB-Abstraction/views (main)$ ls
edit_item.tpl shopping_list.tpl
15:28 ~/kent/dataweb/02-DB-Abstraction/views (main)$

```

- I then copied database.py into 3 different files as shown in below screenshot.

```

22:44 ~/kent/dataweb/02-DB-Abstraction/views (main)$ cd
22:44 ~ $ cd kent/dataweb
22:44 ~/kent/dataweb (main)$ ls
01-web-Intro 02-DB-Abstraction README.md __pycache__ bottle_app.py database.py list.db query.py setup.py shopping_list.db views
22:44 ~/kent/dataweb (main)$ cp database.py dataset_db.py
22:45 ~/kent/dataweb (main)$ cp database.py testing_db.py
22:53 ~/kent/dataweb (main)$ cp database.py dataset_setup.py
23:06 ~/kent/dataweb (main)$

```

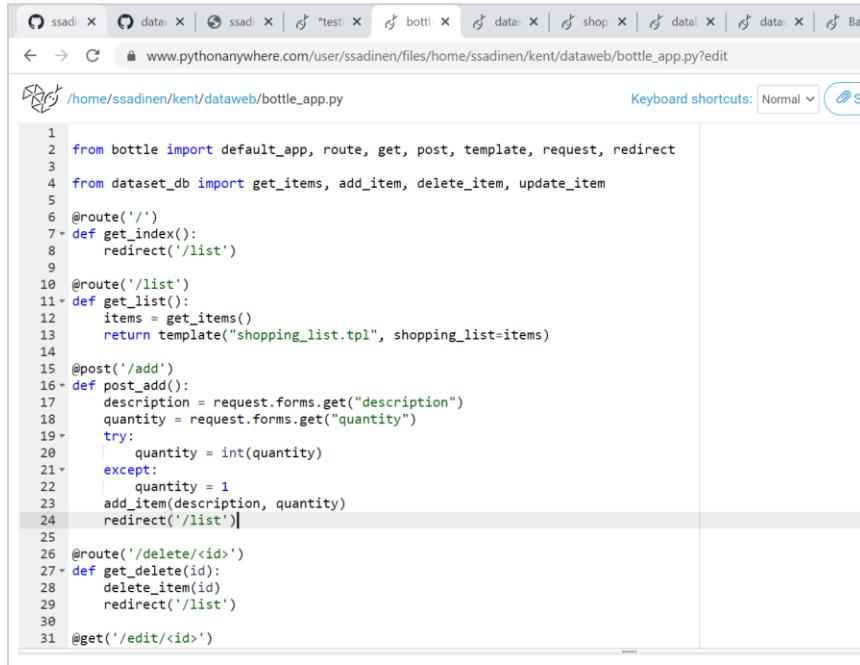
- I then moved all the testing re. code from bottle_app.py into testing_db.py file as shown in below screenshot by importing get_items, add_item, update_item, delete_item from dataset_db

```

1 from dataset_db import get_items, add_item, delete_item, update_item
2
3 def test_get_items():
4     print("testing get_items...")
5     items = get_items()
6     assert type(items) is list
7     assert len(items) > 0
8     assert type(items[0]) is dict
9     assert 'id' in items[0].keys()
10    assert 'description' in items[0].keys()
11    assert type(items[0]['id']) is int
12    assert type(items[0]['description']) is str
13    items = get_items(id=1)
14    assert type(items) is list
15    assert len(items) == 1
16    assert type(items[0]) is dict
17    assert 'id' in items[0].keys()
18    assert 'description' in items[0].keys()
19    assert type(items[0]['id']) is int
20    assert type(items[0]['description']) is str
21
22 import time
23
24 def random_string():
25     return str(time.time())
26
27 def test_add_item():
28     print("testing add_item...")
29     description = random_string()
30     add_item(description)
31     items = get_items()

```

- In bottle_app.py I've then imported get_items, add_item, update_item, delete_item from dataset_db.



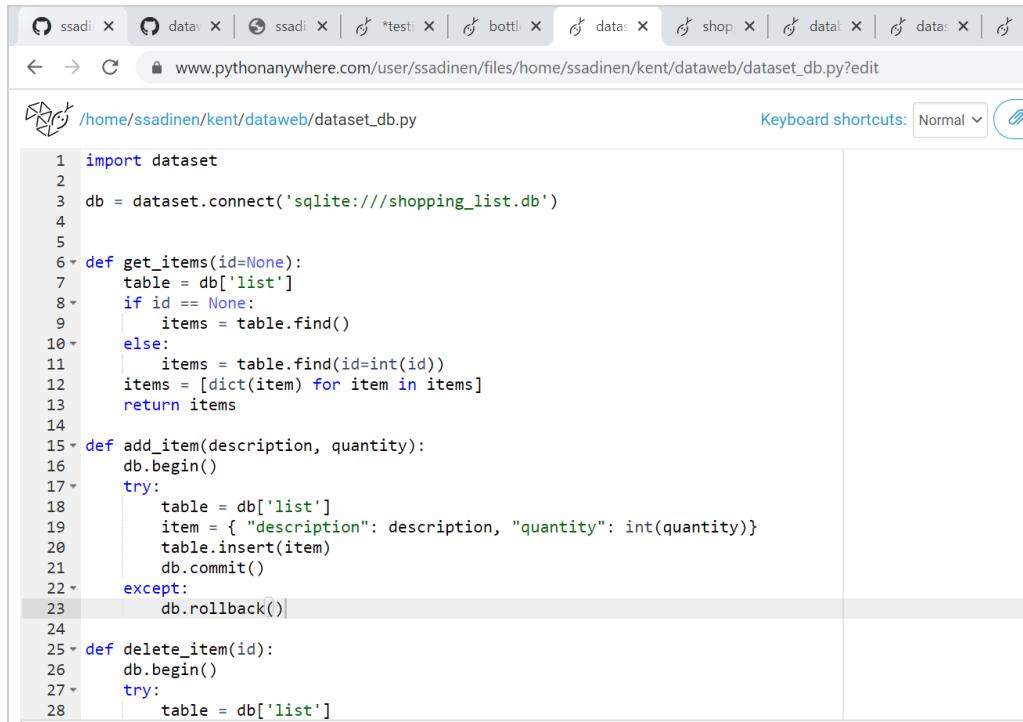
The screenshot shows a PythonAnywhere code editor window with the file `bottle_app.py` open. The code implements a simple web application using the Bottle framework. It defines routes for listing items, adding items, and deleting items from a shopping list database. The code uses form data to handle item addition and database operations to manage the list.

```

1  from bottle import default_app, route, get, post, template, request, redirect
2
3  from dataset_db import get_items, add_item, delete_item, update_item
4
5  @route('/')
6  def get_index():
7      redirect('/list')
8
9
10 @route('/list')
11 def get_list():
12     items = get_items()
13     return template("shopping_list.tpl", shopping_list=items)
14
15 @post('/add')
16 def post_add():
17     description = request.forms.get("description")
18     quantity = request.forms.get("quantity")
19     try:
20         quantity = int(quantity)
21     except:
22         quantity = 1
23     add_item(description, quantity)
24     redirect('/list')
25
26 @route('/delete/<id>')
27 def get_delete(id):
28     delete_item(id)
29     redirect('/list')
30
31 @get('/edit/<id>')

```

- In dataset_db.py, I've added dataset connection and modified old references.



The screenshot shows a PythonAnywhere code editor window with the file `dataset_db.py` open. This script connects to a SQLite database named `shopping_list.db` and provides functions for managing shopping list items. It uses the `dataset` library to interact with the database, specifically the `list` table.

```

1  import dataset
2
3  db = dataset.connect('sqlite:///shopping_list.db')
4
5
6  def get_items(id=None):
7      table = db['list']
8      if id == None:
9          items = table.find()
10     else:
11         items = table.find(id=int(id))
12     items = [dict(item) for item in items]
13     return items
14
15 def add_item(description, quantity):
16     db.begin()
17     try:
18         table = db['list']
19         item = { "description": description, "quantity": int(quantity) }
20         table.insert(item)
21         db.commit()
22     except:
23         db.rollback()
24
25 def delete_item(id):
26     db.begin()
27     try:
28         table = db['list']

```

- In shopping list template I've added the data item and form action for quantity.

The screenshot shows a web-based code editor with multiple tabs at the top. The active tab is titled "shopping_list.tpl". The code in the editor is as follows:

```
1 <html>
2 <body>
3 <h2>Shopping List</h2>
4 <hr/>
5 <table>
6 % for item in shopping_list:
7   <tr>
8     <td>{{str(item['description'])}}</td>
9     <td>{{str(item['quantity'])}}</td>
10    <td><a href="/edit/{{str(item['id'])}}">edit</a></td>
11    <td><a href="/delete/{{str(item['id'])}}">del</a></td>
12  </tr>
13</table>
14 % end
15 <hr/>
16 <form action="add" method="post">
17   <p>Add new item: <input name="description"/></p>
18   <p>Quantity: <input name="quantity"/></p>
19   <p><button type="submit">Submit</button></p>
20 </form>
21 </body>
22 </html>
```

- In dataset_setup.py, I've imported the dataset, added dataset connection.

The screenshot shows a web-based code editor with multiple tabs at the top. The active tab is titled "dataset_setup.py". The code in the editor is as follows:

```
1 import dataset
2
3 db = dataset.connect('sqlite:///shopping_list.db')
4
5 try:
6     db['list'].drop()
7 except:
8     pass
9
10 db.begin()
11 try:
12     table = db['list']
13     items = [
14         { "description": 'apples' },
15         { "description": 'broccoli' },
16         { "description": 'pizza' },
17         { "description": 'tangerine' },
18         { "description": 'potatoes' }
19     ]
20     table.insert_many(items)
21     db.commit()
22 except:
23     db.rollback()
24
```

- After saving the above files, I've reloaded the webpage and I could add the quantity for new items as shown in below screenshot.

Shopping List

apples	None	edit	del
broccoli florets	None	edit	del
tangerine	None	edit	del
choco chips	None	edit	del
granola bar	None	edit	del
salad bowls	None	edit	del
tomatoes	None	edit	del
peanuts	None	edit	del
Frozen Peas	1	edit	del
Watermelons	5	edit	del
Spices	10	edit	del

Add new item:

Quantity:

- I've copied all the files edited to a new directory 03-Datasets

```
cp: cannot create 'views': No such file or directory
03:13 ~/kent/dataweb (main)$ cp -r *.md *.py sho* views 03-Datasets
03:13 ~/kent/dataweb (main)$ cd 03*
03:14 ~/kent/dataweb/03-Datasets (main)$ ls
README.md      database.py    dataset_setup.py    setup.py     shopping_list.db-shm  testing_db.py
bottle_app.py  dataset_db.py  query.py        shopping_list.db  shopping_list.db-wal  views
03:14 ~/kent/dataweb/03-Datasets (main)$ █
```

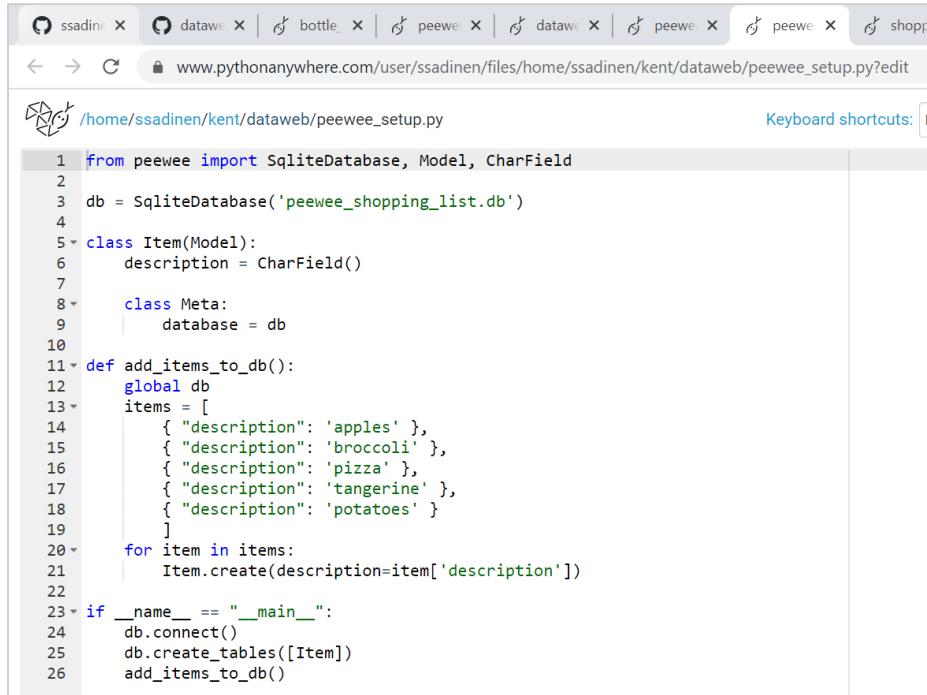
- I then created peewee_database.py file and modified the code as shown below –

```

from peewee import SqliteDatabase, Model, CharField
db = SqliteDatabase('peewee_shopping_list.db')
class Item(Model):
    description = CharField()
    class Meta:
        database = db
def get_items(id=None):
    if id == None:
        items = Item.select()
    else:
        items = Item.select().where(Item.id == int(id))
    items = [{id:item.id, 'description':item.description} for item in items]
    return items
def add_item(description):
    Item.create(description=description)
def delete_item(id):
    q = Item.delete().where(Item.id == int(id))
    q.execute()
def update_item(id, description):
    q = Item.update({Item.description: description}).where(Item.id == int(id))
    q.execute()

```

- I then created peewee_setup.py file and modified the code by importing required contents from peewee.

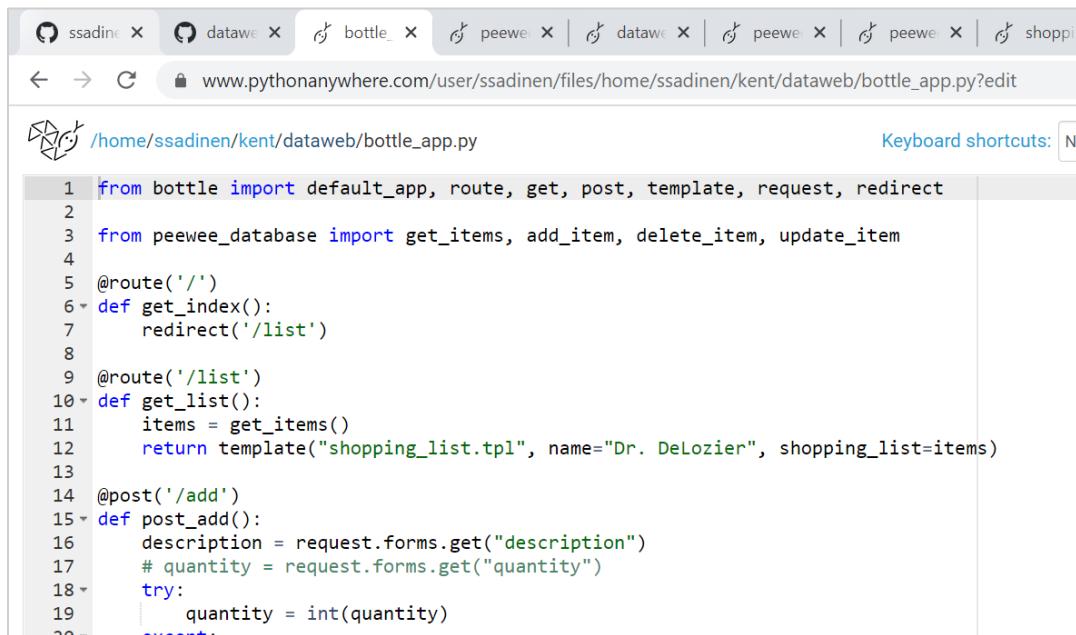


```

1  from peewee import SqliteDatabase, Model, CharField
2
3  db = SqliteDatabase('peewee_shopping_list.db')
4
5  class Item(Model):
6      description = CharField()
7
8      class Meta:
9          database = db
10
11     def add_items_to_db():
12         global db
13         items = [
14             { "description": 'apples' },
15             { "description": 'broccoli' },
16             { "description": 'pizza' },
17             { "description": 'tangerine' },
18             { "description": 'potatoes' }
19         ]
20         for item in items:
21             Item.create(description=item['description'])
22
23     if __name__ == "__main__":
24         db.connect()
25         db.create_tables([Item])
26         add_items_to_db()

```

- I then updated bottle_app.py and imported get_items, add_item, delete_item, update_item from peewee_database.

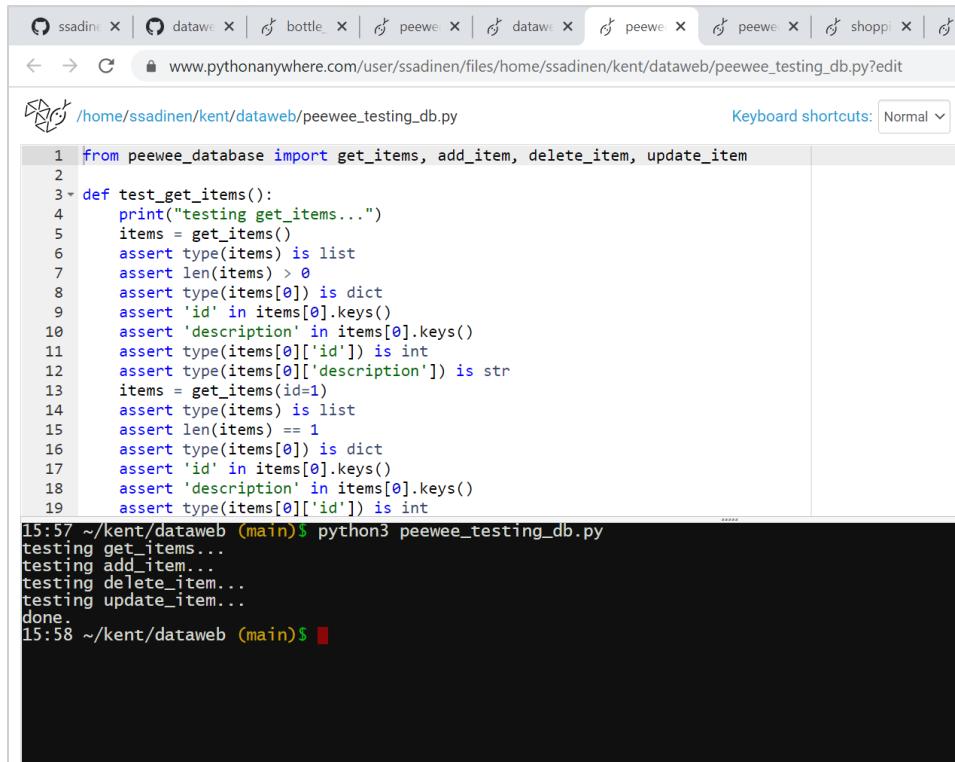


```

1  from bottle import default_app, route, get, post, template, request, redirect
2
3  from peewee_database import get_items, add_item, delete_item, update_item
4
5  @route('/')
6  def get_index():
7      redirect('/list')
8
9  @route('/list')
10 def get_list():
11     items = get_items()
12     return template("shopping_list.tpl", name="Dr. DeLozier", shopping_list=items)
13
14 @post('/add')
15 def post_add():
16     description = request.forms.get("description")
17     # quantity = request.forms.get("quantity")
18     try:
19         quantity = int(quantity)
20     except:

```

- Copied testing_db.py to peewee_testing_db.py and modified to import get_items, add_item, delete_item, update_item from peewee_database.
- I then saved the file after making required changes and opened bash console. I then ran python3 peewee_testing.db command to test get_items, add_item, delete_item, update_item.

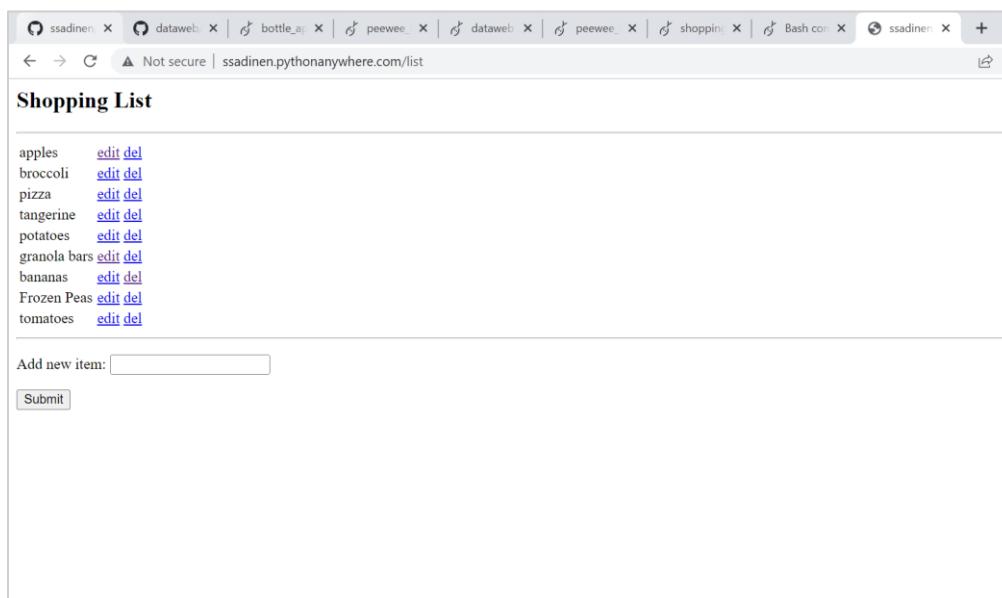


```

from peewee_database import get_items, add_item, delete_item, update_item
def test_get_items():
    print("testing get_items...")
    items = get_items()
    assert type(items) is list
    assert len(items) > 0
    assert type(items[0]) is dict
    assert 'id' in items[0].keys()
    assert 'description' in items[0].keys()
    assert type(items[0]['id']) is int
    assert type(items[0]['description']) is str
    items = get_items(id=1)
    assert type(items) is list
    assert len(items) == 1
    assert type(items[0]) is dict
    assert 'id' in items[0].keys()
    assert 'description' in items[0].keys()
    assert type(items[0]['id']) is int
15:57 ~/kent/dataweb (main)$ python3 peewee_testing_db.py
testing get_items...
testing add_item...
testing delete_item...
testing update_item...
done.
15:58 ~/kent/dataweb (main)$

```

- I then saved all the above files and reloaded the website. I could see all the contents in web page as expected.



Shopping List

apples	edit del
broccoli	edit del
pizza	edit del
tangerine	edit del
potatoes	edit del
granola bars	edit del
bananas	edit del
Frozen Peas	edit del
tomatoes	edit del

Add new item:

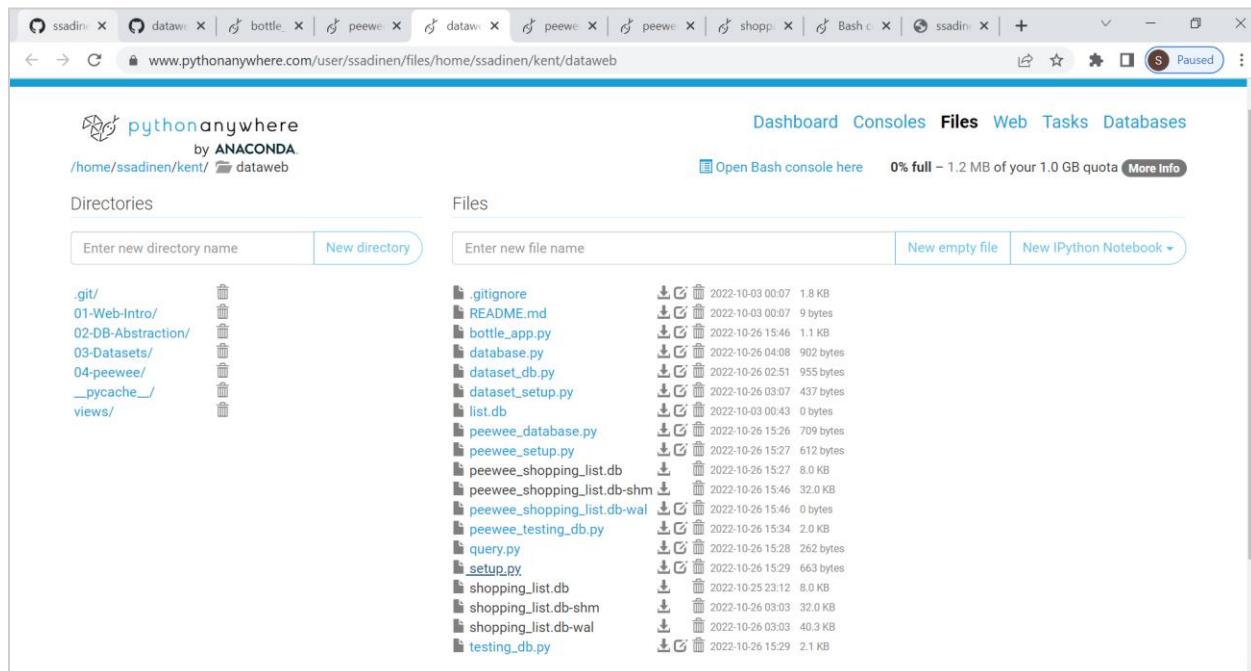
- I've then copied all the new files into a new directory 04-peewee

```

04:09 ~/kent/dataweb (main)$ cp shopping_list.db peewee_shopping_list.db
15:27 ~/kent/dataweb (main)$ ls
01-Web-Intro 04-peewee bottle_app.py dataset_setup.py peewee_setup.py setup.py shopping_list.db testi
02-DB-Abstraction README.md database.py list.db peewee_shopping_list.db shopping_list.db-shm views
03-Datasets __pycache__ dataset_db.py peewee_database.py query.py
15:27 ~/kent/dataweb (main)$ ls
01-Web-Intro 04-peewee bottle_app.py dataset_setup.py peewee_setup.py query.py shopping_
02-DB-Abstraction README.md database.py list.db peewee_shopping_list.db setup.py shopping_
03-Datasets __pycache__ dataset_db.py peewee_database.py peewee_testing_db.py shopping_list.db testing_
15:35 ~/kent/dataweb (main)$ cp -r *.md bottle_app.py database.py peew*.py peew*.db 04-peewee
15:37 ~/kent/dataweb (main)$ cd 04*
15:37 ~/kent/dataweb/04-peewee (main)$ ls
README.md bottle_app.py database.py peewee_database.py peewee_setup.py peewee_shopping_list.db peewee_testing_db.
15:37 ~/kent/dataweb/04-peewee (main)$ cd kent/dataweb
bash: cd: kent/dataweb: No such file or directory
15:37 ~/kent/dataweb/04-peewee (main)$ cd
15:37 ~/kent/dataweb (main)$ cd kent/dataweb
15:37 ~/kent/dataweb (main)$ cp query.py setup.py 04-peewee
15:37 ~/kent/dataweb (main)$ cd 04*
15:38 ~/kent/dataweb/04-peewee (main)$ ls
README.md database.py peewee_setup.py peewee_testing_db.py setup.py
bottle_app.py peewee_database.py peewee_shopping_list.db query.py
15:38 ~/kent/dataweb/04-peewee (main)$

```

- Below are the files and directories I've created for my practice of first 4 topics of dataweb.



- I have then committed and pushed all contents to my github by generating ssh key and adding it into my git.

Bash console 25750590

```
(use "git restore <file>..." to discard changes in working directory)
 modified:   Sarika-intro01/vegetables.db

Untracked files:
 (use "git add <file>..." to include in what will be committed)
   Sarika-intro01/query.py

21:47 ~/kent/advanced-database-ssadinien (main)$ git add --all
21:47 ~/kent/advanced-database-ssadinien (main)$ git status
On branch main
Your branch is up to date with 'origin/main'.

Changes to be committed:
 (use "git restore --staged <file>..." to unstage)
   new file:   Sarika-intro01/query.py
   new file:   Sarika-intro01/setup.py
   new file:   Sarika-intro01/vegetables.db

21:47 ~/kent/advanced-database-ssadinien (main)$ git commit -m "Advanced DB Practice"
[main d9654b1] Advanced DB Practice
 3 files changed, 63 insertions(+)
  create mode 100644 Sarika-intro01/query.py
  create mode 100644 Sarika-intro01/setup.py
  create mode 100644 Sarika-intro01/vegetables.db
21:48 ~/kent/advanced-database-ssadinien (main)$ git push git@github.com:ssadinien/advanced-database-ssadinien.git
Enter passphrase for key '/home/ssadinien/.ssh/id_rsa':
Enumerating objects: 7, done.
Counting objects: 100% (7/7), done.
Delta compression using up to 4 threads
Compressing objects: 100% (6/6), done.
Writing objects: 100% (6/6), 1.23 KiB | 314.00 KiB/s, done.
Total 6 (delta 0), reused 0 (delta 0)
To github.com:ssadinien/advanced-database-ssadinien.git
  3ea328a..d9654b1  main -> main
21:48 ~/kent/advanced-database-ssadinien (main)$
```

Bash console 25750590

```
21:42 ~/kent/dataweb (main)$ git push
Username for 'https://github.com': ^[
Password for 'https://github.com':
remote: Support for password authentication was removed on August 13, 2021.
remote: Please see https://docs.github.com/en/get-started/getting-started-with-git/about-remote-repositories#cloning-with-https-urls for information on currently recommended modes of authentication.
fatal: Authentication failed for 'https://github.com/ssadinien/dataweb.git/'
21:43 ~/kent/dataweb (main)$ git push
Username for 'https://github.com': ssadinen@kent.edu
Password for 'https://ssadinen@kent.edu@github.com':
remote: Support for password authentication was removed on August 13, 2021.
remote: Please see https://docs.github.com/en/get-started/getting-started-with-git/about-remote-repositories#cloning-with-https-urls for information on currently recommended modes of authentication.
fatal: Authentication failed for 'https://github.com/ssadinien/dataweb.git/'
21:44 ~/kent/dataweb (main)$ git push
Username for 'https://github.com': ssadinen
Password for 'https://ssadinen@github.com':
remote: Support for password authentication was removed on August 13, 2021.
remote: Please see https://docs.github.com/en/get-started/getting-started-with-git/about-remote-repositories#cloning-with-https-urls for information on currently recommended modes of authentication.
fatal: Authentication failed for 'https://github.com/ssadinien/dataweb.git/'
21:44 ~/kent/dataweb (main)$ git push git@github.com:ssadinien/dataweb.git
warning: Permanently added the ECDSA host key for IP address '140.82.113.4' to the list of known hosts.
Enter passphrase for key '/home/ssadinien/.ssh/id_rsa':
Enumerating objects: 47, done.
Counting objects: 100% (47/47), done.
Delta compression using up to 4 threads
Compressing objects: 100% (45/45), done.
Writing objects: 100% (46/46), 10.52 KiB | 598.00 KiB/s, done.
Total 46 (delta 20), reused 0 (delta 0)
remote: Resolving deltas: 100% (20/20), done.
To github.com:ssadinien/dataweb.git
  4590486..13fe358  main -> main
21:44 ~/kent/dataweb (main)$
21:45 ~/kent/dataweb (main)$
```