Lab for Week 11

Data Structures and MYSQL

Chaufournier & Wood
CSCI 2541

Lists

```
myList = ["Apples", "Oranges", "Bananas", "Grapes"]
fruit3 = myList[2]
myList[1] = “Pineapple”
myList.append(“Kiwi”)
# ["Apples", "Pineapple", "Bananas", “Grapes","Kiwi"]
```

- Lists store multiple items in a single value.
- Ordered and accessible by index.
- Adding an item, adds it to the end of the list.
- Mutable: you can add, remove, update the values of a list.

Tuples

```
myTuple = ("Apples", "Oranges", "Bananas", “Grapes")
fruit1 = myTuple[0]
```

- Lists store multiple items in a single value.
- Ordered and accessible by index.
- Immutable: you can’t modify a tuples values once its created.

Tuples

```
user = “Dan”

c.execute(“select * from users where username=?,user)

c.execute("select * from users where username=?”,(user,))
```

Gets treated as D,a,n ✗

Gets treated as Dan ✓

- Be careful with Tuples and iterators.
- A lot of functions and libraries expect a tuple.
- If you provide a string it’s treated as a tuple and you get weird results

Dictionaries

```
myDict = {
  "name": "Maya",
  "address": "156 East 24th street",
  "city": "New York",
  "state":"New York",
 "cars": ["Ford","Honda"]
}
myDict[“name”] = “Bob”
myDict[“age”] = 36
```

Dictionaries are structured data which makes it easy for us to convert it between different data structures and formats as needed. Ex. Json

- Dictionaries allow you to store key value pairs
- Store arbitrary amounts of structured data.
- Mutable: You can add, remove, and update values.
- You can nest data structures inside of dictionaries.

Dictionaries

```
myDict = {}
myDict["color"]  = "red"
myDict["day"]    = "Monday"
myDict["months"] = ["Jan","Apr","May"]
myDict["user"]   = {
                  "Name":"Alice"
                  "Username": "a123"
                  }
```

```
{
"color" : "red",
"day" : "Monday"
"months": ["Jan","Apr","May"]
"user": {"Name":"Alice",
          "Username":"a123"
        }
}
```

- Dictionaries allow you to store key value pairs
- Store arbitrary amounts of structured data.
- Mutable: You can add, remove, and update values.
- You can nest data structures inside of dictionaries.

Dictionaries resemble a table structure

```
myDict = {
  "name": "Maya",
  "address": "156 East 24th street",
  "city": "New York",
  "state":"New York",
}
```

Name	Address	City	State
Maya	156 East 24th street	New York	New York

So it stands to reason we should be able to use dictionaries with mysql.

Previously we would do the following:

Import the mysql.connector library

```
import mysql.connector
```

Create a new connection to the database

```
mydb = mysql.connector.connect(
    host="10.0.12.12",
    user="student",
    password="seas",
    database="dev"
  )
```

Create a cursor. It’s a pointer to the database and tracks the location of operations.

```
c = mydb.cursor()
```

The query we want to execute on the database.

```
c.execute('''Select * from users''')
```

Fetch the results from the cursor

```
results = c.fetchall()
```

Close the cursor to save memory

```
c.close()
```

Now we just make one small modification

Import the mysql.connector library

```
import mysql.connector
```

Create a new connection to the database

```
mydb = mysql.connector.connect(
    host="10.0.12.12",
    user="student",
    password="seas",
    database="dev"
  )
```

Create a cursor. It's a pointer to the database and tracks the location of operations.

```
c = mydb.cursor(dictionary=True)
```

Enable dictionary results

The query we want to execute on the database.

```
c.execute('''Select * from users''')
```

Fetch the results from the cursor

```
results = c.fetchall()
```

Close the cursor to save memory

```
c.close()
```

Now results are returned back as dictionaries

```
c = mydb.cursor(dictionary=True)
```

Select all users from the table

```
c.execute('''Select * from users''')
```

Fetch all the results from the cursor

The results are a list of a dictionaries, one per row of results

```
results = c.fetchall()
```

Access the firstname and address of the first result

We get back results that look like the following when we call fetchall:

```
[
{"Name": "Bob Smith", "Address": "1st street", City: "Washington","State": "DC" },
{"Name": "Alice White", "Address": "2nd street", City": "New York", "State": "NY"},
{"Name": "Dana Rice", "Address": "3rd street", City": "New York", "State": "NY"}
]
```

Essentially a list of dictionary objects. One for each row in your result set.

Now results are returned back as dictionaries

```
c = mydb.cursor(dictionary=True)
```

Select all users from the table

```
c.execute('''Select * from users''')
```

Fetch all the results from the cursor

The results are a list of a dictionaries, one per row of results

```
results = c.fetchall()
```

Access the firstname and address of the first result

```
firstname = results[0]["Name"]
address = results[0]["Address"]

c.close()
```

Using this pattern we can iterate through the results and use the data in our logic.

How can we take advantage of this when returning data to users?

We can pass the length and list of users to our template

```
c = mydb.cursor(dictionary=True)

c.execute('''Select * from users''')


results = c.fetchall()

c.close()

return render_template("index.html",len=len(results), users=results)
```

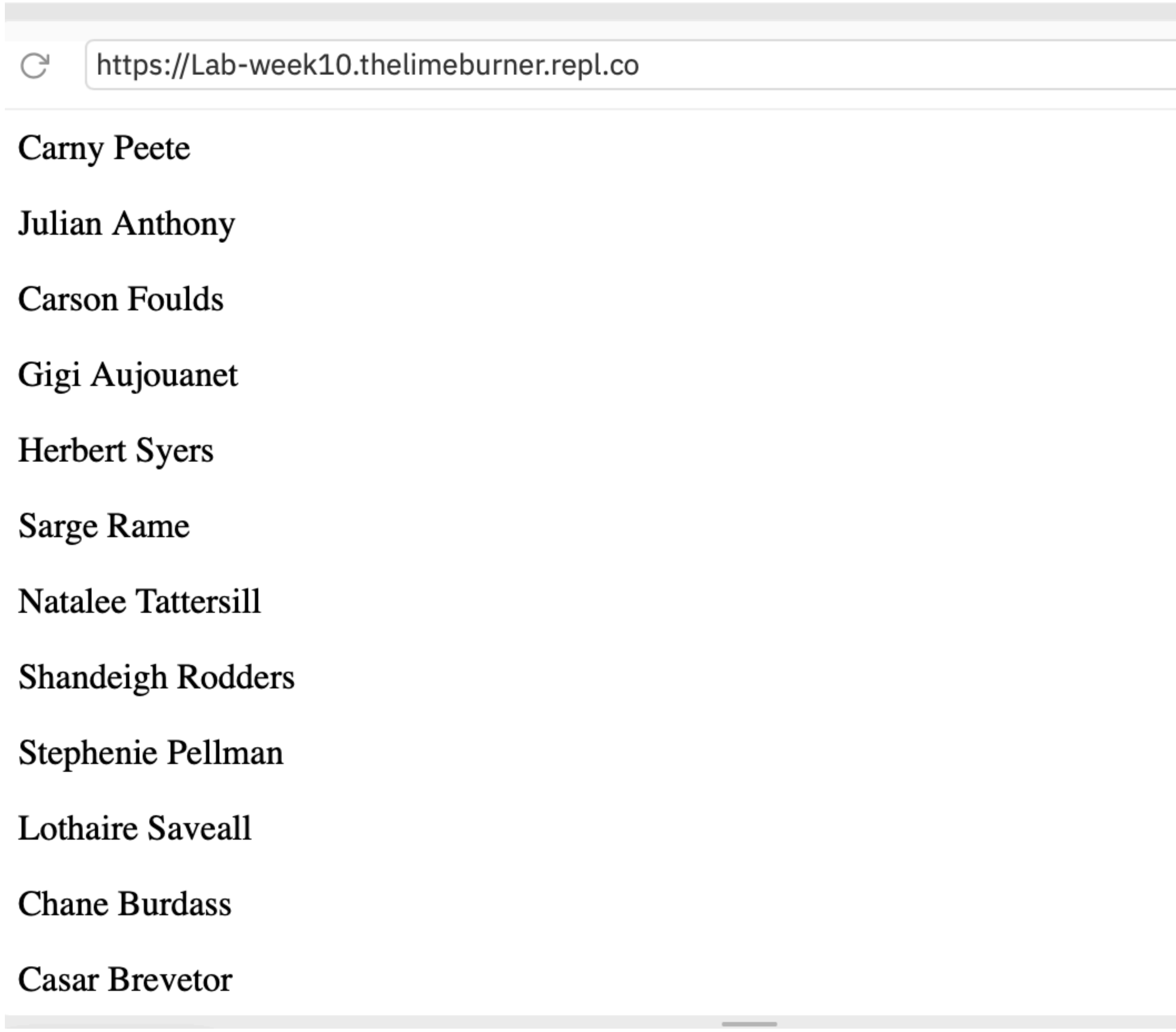
We pass results as a list so our template can pull the needed data.

We pass the length so we can iterate through the list of users.

Our template can then iterate through the dictionary

index.html

```
<html>
<body>
{%for i in range(0, len)%}
  <p>{{users[i]["Name"]}}</p>
{%endfor%}
</body>
</html>
```



https://Lab-week10.thelimeburner.repl.co

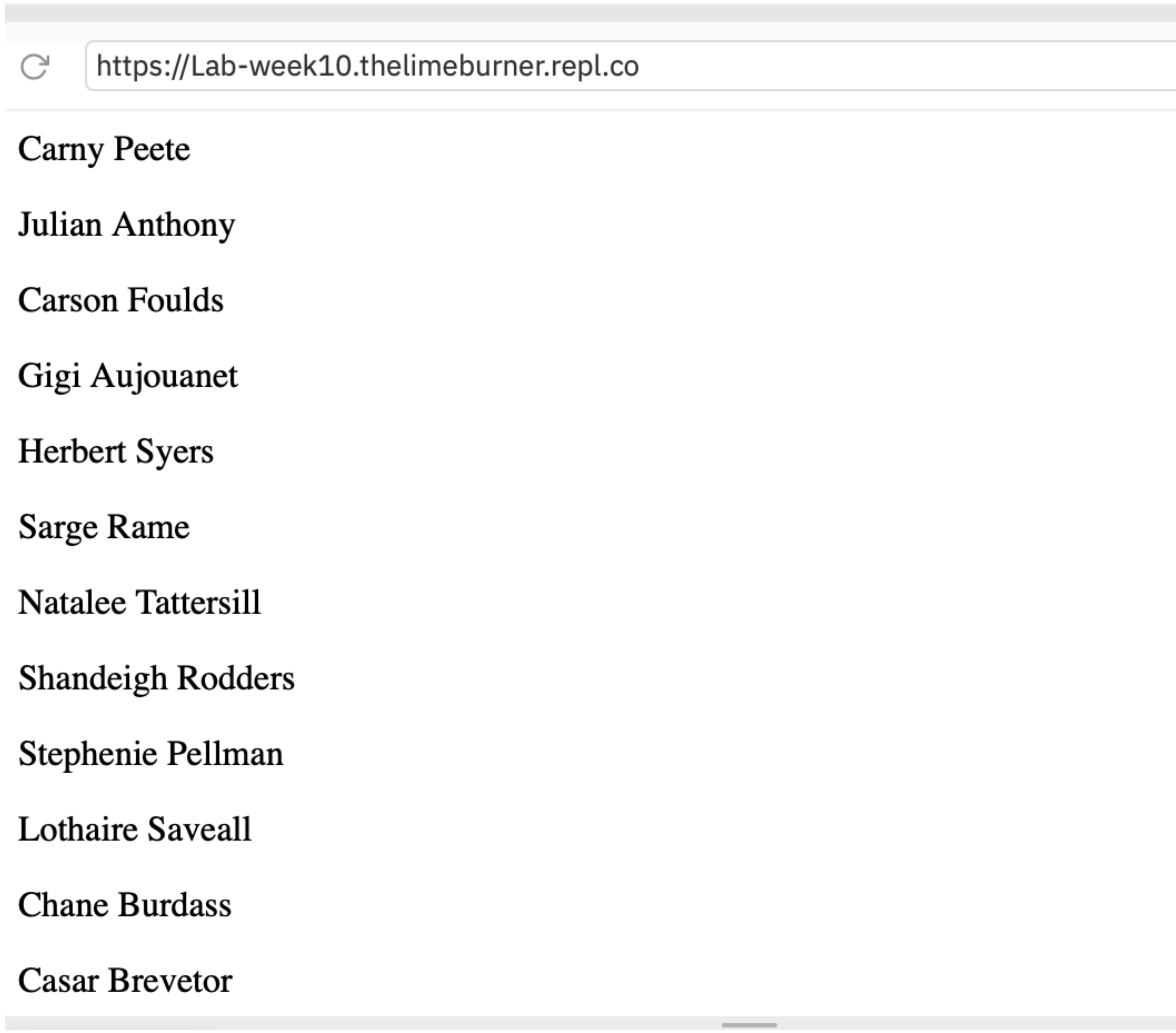
```
Carny Peete

Julian Anthony

Carson Foulds

Gigi Aujouanet

Herbert Syers

Sarge Rame

Natalee Tattersill

Shandeigh Rodders

Stephenie Pellman

Lothaire Saveall

Chane Burdass

Casar Brevetor
```

Our template can then iterate through the dictionary

index.html

```
<html>
<body>
{%for i in range(0, len)%}
  <p>{{users[i]["Name"]}}</p>
{%endfor%}
</body>
</html>
```

This looks a little messy. Is there a cleaner way to do this?



https://Lab-week10.thelimeburner.repl.co

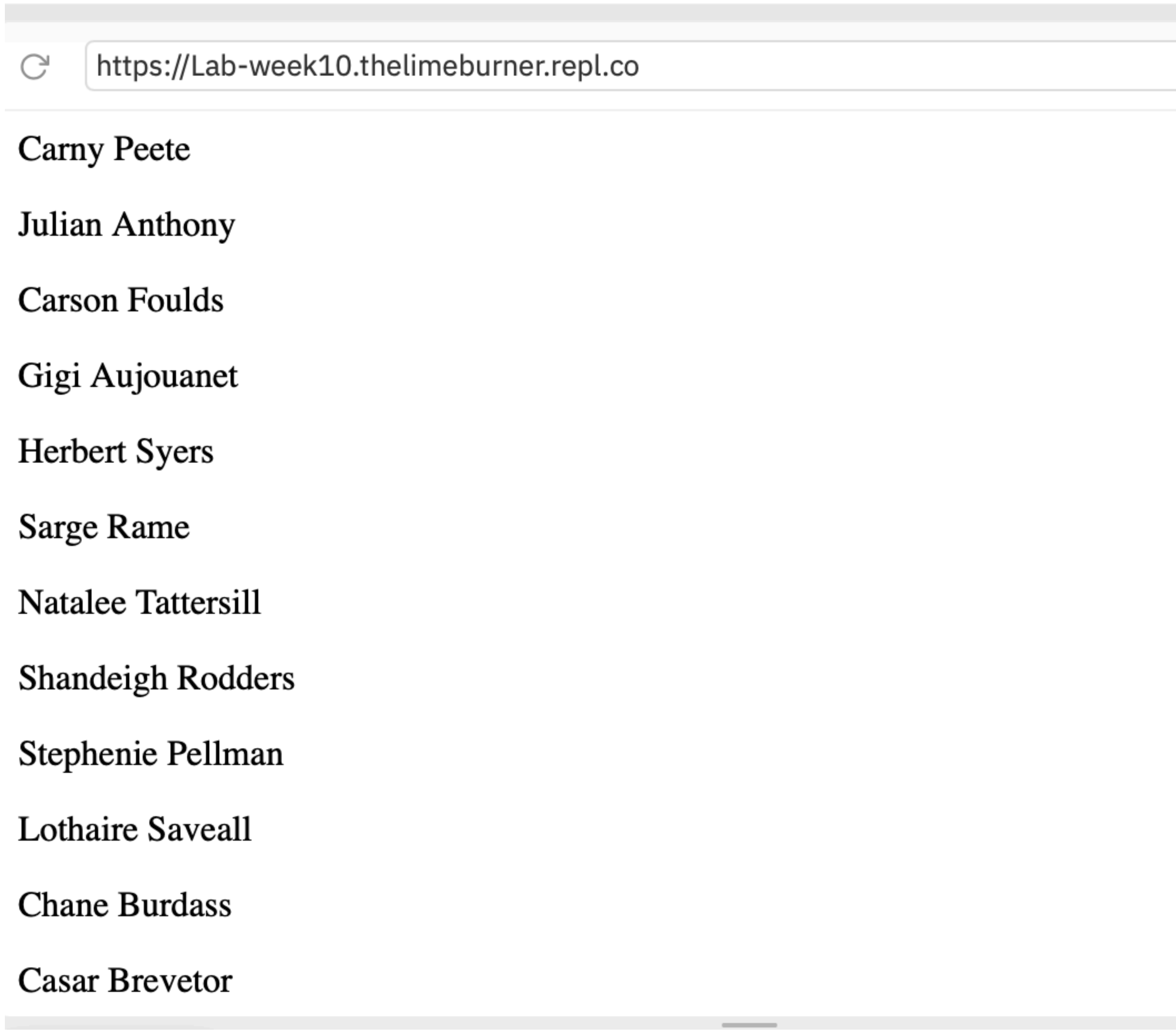
```
Carny Peete
Julian Anthony
Carson Foulds
Gigi Aujouanet
Herbert Syers
Sarge Rame
Natalee Tattersill
Shandeigh Rodders
Stephenie Pellman
Lothaire Saveall
Chane Burdass
Casar Brevetor
```

Our template can then iterate through the dictionary

index.html

```
<html>
<body>
{%for user in users%}
  <p>{{user["Name"]}}</p>
{%endfor%}
</body>
</html>
```

This is much neater but is there still a better way?


https://Lab-week10.thelimeburner.repl.co

Carny Peete

Julian Anthony

Carson Foulds

Gigi Aujouanet

Herbert Syers

Sarge Rame

Natalee Tattersill

Shandeigh Rodders

Stephenie Pellman

Lothaire Saveall

Chane Burdass

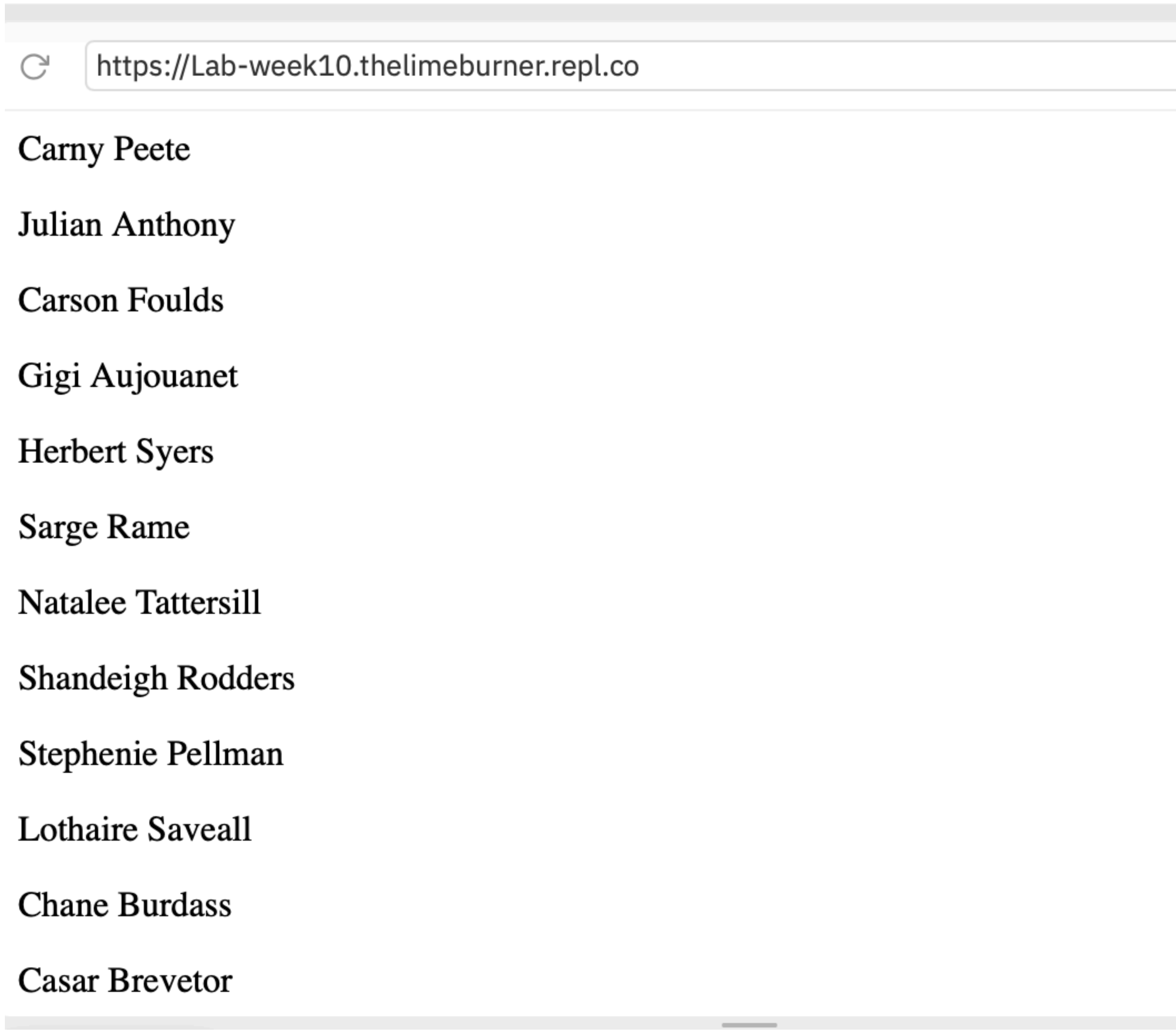
Casar Brevetor

Our template can then iterate through the dictionary

index.html

```
<html>
<body>
{%for user in users%}
  <p>{{user.Name}}</p>
{%endfor%}
</body>
</html>
```

This is much cleaner and allows you to directly access members of an object.



https://Lab-week10.thelimeburner.repl.co

```
Carny Peete

Julian Anthony

Carson Foulds

Gigi Aujouanet

Herbert Syers

Sarge Rame

Natalee Tattersill

Shandeigh Rodders

Stephenie Pellman

Lothaire Saveall

Chane Burdass

Casar Brevetor
```