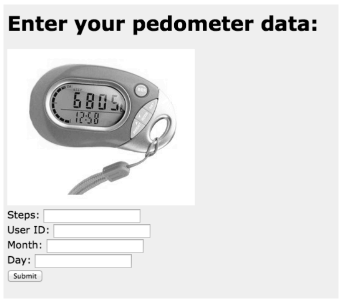
ECS 89

5/7

Announcements

- Checkpoint on Proj3 due Tuesday
- Make sure you have set up model for Django database for pedometer project, making sure the user ID part is to the level of the tutorial.
- Final part will be to read in pedometer data, add to database, and make some Web pages that let users look up combinations of data items.

Pedometer data entry



Enter your pedometer data:

Steps:
User ID:
Month:
Day:
Submit

Pedometer data

- Using Alisha's app
- Up at pc110.cs.ucdavis.edu:10002/hw2/index.html
- Make up a UID, four letters/digits
- Use it consistently, please; we are not checking

Last time

- Django's Model class represents database table
- We create a child class for every table in our database, that inherits from Model
- Let's continue with tutorial…adding Choice

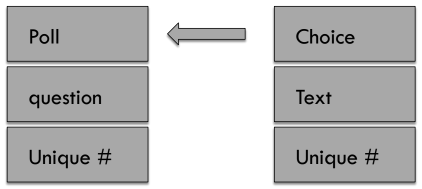
Talking to databases

- The database code (sqlite3) does not understand Python objects
- Standard interface to database code is SQL

```
python manage.py synchdb
```

- Generates SQL commands to make all current models, adds corresponding rows and columns to database if they are not already there

Relation between model elements



- ForeignKey is a method of Model. We inherited it.
- It connects Poll to Choice; that arrow in the picture
- It makes a relation (as in "relational database")
- Poll has many choices but each choice has one Poll

Looking at model in the shell

- This is a tool to help develop the Django app, not generally for users.
- Some Model methods:
  - __init__ with keyword parameters for all attributes:

    p = Poll(question="Did you reach your goal today?", pub_date=timezone.now())

  - p.save() - puts an object into database.

Model methods for extracting data

- Polls.objects.all() – gets everything

```
queryObj = Poll.objects.all()
for o in queryObj:
    print o.question
```

- Variable queryObj contains a QuerySet object, which includes a list of objects of type Poll, and other stuff
- Polls.objects.filter(question__contains="goal")

  Produces a QuerySet containing only some of the objects, those that have the string "goal" in their questions

Methods using relation b/w tables

Choice.objects.filter(poll__question__contains="goal")

# returns a QuerySet containing all choices belonging to any poll question that contains the word "goal"

Poll.objects.filter(choice__choice_text__contains="Y")

# returns a QuerySet containing all polls that have a choice whose choice_text contains a "Y"

Views and URLs

- Views are the functions that produce the output string that gets returned in a HTTP response, usually HTML (what else might it be?)
- We connect views to URLs in the urls.py file.
- The urls.py file in mysite/mysite sends urls to the right apps.
- The urls.py file within each app sends urls to the right views.
- URLs are specified by regular expressions.

Regular expressions

- Where did they come up before?

```
[0-9]    # a digit between 0 and 9
[0-9]+   # one or more digits
[0-9]*   # zero or more digits

^a       # "a" at the beginning of line
a$       # "a" at end of line
^$       # empty string
```

Example

url(r'^[0-9]+/results/$’

- The r before the string stands for “raw”; tells Python not to over-think things like “\n”, and just pass a “\” and an “n” to whatever function is going to take this string as an argument.
- And the rest?

Example in tutorial

r’(?P<poll_id>\d+)/results/$’

- Captures the digit and sticks it into parameter poll_id, which is then sent to the results function.
- We are not going to have to do something like this, so we can stick to regular expressions like the ones on the previous slide.