Decomposition

Modular Design

Separation of Concerns

A design principle: Isolate different parts of a program that address different concerns
A modular component can be developed and tested independently

Game rules
• Ordering of events
• State tracking to determine the winner

Game commentary
• Event descriptions
• State tracking to generate commentary

Player strategies
• Decision rules
• Strategy parameters (e.g., margins & number of dice)

Hog Game
Simulator

Hog
- Game rules
- Ordering of events
- State tracking to determine the winner

Ants Game
Simulator

Ants
- Order of actions
- Food tracking
- Game ending conditions

Examples
Restaurant Search

Restaurant Search Data

Given the following data, look up a restaurant by name and show related restaurants.

```
{"business_id": "gclB3ED6uk6viWlolSb_uA", "name": "Cafe 3", "stars": 2.0, "price": 1, ...
{"business_id": "WXKx2I2SEzBpeUGtDMCS8A", "name": "La Cascada Taqueria", "stars": 3.0, "price": 2, ...
...
{"business_id": "gclB3ED6uk6viWlolSb_uA", "user_id": "xVocUszkZtAqCxgWak3xVQ", "stars": 1, "text": "Cafe 3 (or Cafe Tre, as I like to say) used to be the bomb diggity when I first lived in the dorms but sadly, quality has dramatically decreased over the years....", "date": "2012-01-19", ...
...
```
(Demo)

Example: Restaurant Search

Discussion Question: Most Similar Restaurants

Implement similar, a Restaurant method that takes a positive integer k and a function similarity that takes two restaurants as arguments and returns a number. Higher similarity values indicate more similar restaurants. The similar method returns a list containing the k most similar restaurants according to the similarity function, but not containing self.

```
def similar(self, k, similarity):
    """Return the K most similar restaurants to SELF, using SIMILARITY for comparison.""
    others = list(Restaurant.all)
    others.remove(self)
    return sorted(others, key=lambda r: -similarity(self, r) if r != self else float('inf'))[:k]
```

Restaurant Search Data

Given the following data, look up a restaurant by name and show related restaurants.

```
{"business_id": "gclB3ED6uk6viWlolSb_uA", "name": "Cafe 3", "stars": 2.0, "price": 1, ...
{"business_id": "WXKx2I2SEzBpeUGtDMCS8A", "name": "La Cascada Taqueria", "stars": 3.0, "price": 2, ...
...
{"business_id": "gclB3ED6uk6viWlolSb_uA", "user_id": "xVocUszkZtAqCxgWak3xVQ", "stars": 1, "text": "Cafe 3 (or Cafe Tre, as I like to say) used to be the bomb diggity when I first lived in the dorms but sadly, quality has dramatically decreased over the years....", "date": "2012-01-19", ...
...
```
(Demo)

Example: Restaurant Search

Discussion Question: Most Similar Restaurants

Implement similar, a Restaurant method that takes a positive integer k and a function similarity that takes two restaurants as arguments and returns a number. Higher similarity values indicate more similar restaurants. The similar method returns a list containing the k most similar restaurants according to the similarity function, but not containing self.

```
def similar(self, k, similarity):
    """Return the K most similar restaurants to SELF, using SIMILARITY for comparison.""
    others = list(Restaurant.all)
    others.remove(self)
    return sorted(others, key=lambda r: -similarity(self, r) if r != self else float('inf'))[:k]
```

Example: Similar Restaurants

Restaurant Search Data

Given the following data, look up a restaurant by name and show related restaurants.

```
{"business_id": "gclB3ED6uk6viWlolSb_uA", "name": "Cafe 3", "stars": 2.0, "price": 1, ...
{"business_id": "WXKx2I2SEzBpeUGtDMCS8A", "name": "La Cascada Taqueria", "stars": 3.0, "price": 2, ...
...
{"business_id": "gclB3ED6uk6viWlolSb_uA", "user_id": "xVocUszkZtAqCxgWak3xVQ", "stars": 1, "text": "Cafe 3 (or Cafe Tre, as I like to say) used to be the bomb diggity when I first lived in the dorms but sadly, quality has dramatically decreased over the years....", "date": "2012-01-19", ...
...
```
(Demo)

Example: Restaurant Search

Discussion Question: Most Similar Restaurants

Implement similar, a Restaurant method that takes a positive integer k and a function similarity that takes two restaurants as arguments and returns a number. Higher similarity values indicate more similar restaurants. The similar method returns a list containing the k most similar restaurants according to the similarity function, but not containing self.

```
def similar(self, k, similarity):
    """Return the K most similar restaurants to SELF, using SIMILARITY for comparison.""
    others = list(Restaurant.all)
    others.remove(self)
    return sorted(others, key=lambda r: -similarity(self, r) if r != self else float('inf'))[:k]
```

Example: Similar Restaurants

Restaurant Search Data

Given the following data, look up a restaurant by name and show related restaurants.

```
{"business_id": "gclB3ED6uk6viWlolSb_uA", "name": "Cafe 3", "stars": 2.0, "price": 1, ...
{"business_id": "WXKx2I2SEzBpeUGtDMCS8A", "name": "La Cascada Taqueria", "stars": 3.0, "price": 2, ...
...
{"business_id": "gclB3ED6uk6viWlolSb_uA", "user_id": "xVocUszkZtAqCxgWak3xVQ", "stars": 1, "text": "Cafe 3 (or Cafe Tre, as I like to say) used to be the bomb diggity when I first lived in the dorms but sadly, quality has dramatically decreased over the years....", "date": "2012-01-19", ...
...
```
(Demo)

Example: Restaurant Search

Discussion Question: Most Similar Restaurants

Implement similar, a Restaurant method that takes a positive integer k and a function similarity that takes two restaurants as arguments and returns a number. Higher similarity values indicate more similar restaurants. The similar method returns a list containing the k most similar restaurants according to the similarity function, but not containing self.

```
def similar(self, k, similarity):
    """Return the K most similar restaurants to SELF, using SIMILARITY for comparison.""
    others = list(Restaurant.all)
    others.remove(self)
    return sorted(others, key=lambda r: -similarity(self, r) if r != self else float('inf'))[:k]
```

Example: Similar Restaurants

Restaurant Search Data

Given the following data, look up a restaurant by name and show related restaurants.

```
{"business_id": "gclB3ED6uk6viWlolSb_uA", "name": "Cafe 3", "stars": 2.0, "price": 1, ...
{"business_id": "WXKx2I2SEzBpeUGtDMCS8A", "name": "La Cascada Taqueria", "stars": 3.0, "price": 2, ...
...
{"business_id": "gclB3ED6uk6viWlolSb_uA", "user_id": "xVocUszkZtAqCxgWak3xVQ", "stars": 1, "text": "Cafe 3 (or Cafe Tre, as I like to say) used to be the bomb diggity when I first lived in the dorms but sadly, quality has dramatically decreased over the years....", "date": "2012-01-19", ...
...
```
(Demo)
Sets

One more built-in Python container type

- Set literals are enclosed in braces
- Duplicate elements are removed on construction
- Sets have arbitrary order

```python
>>> s = {'one', 'two', 'three', 'four', 'four'}
>>> s
{'three', 'one', 'four', 'two'}
>>> 'three' in s
True
>>> len(s)
4
>>> s.union({'one', 'five'})
{'three', 'five', 'one', 'two'}
>>> s.intersection({'six', 'five', 'one', 'four', 'three'})
{'three', 'four'}
>>> s
{'three', 'one', 'four', 'two'}
```

Sets