İSTANBUL ŞEHİR ÜNİVERSİTY

ENGR 212: Programming Practice

Week 5

İSTANBUL ŞEHİR ÜNİVERSİTY

Part 2: Making Recommendations

Finding Similar Users

İSTANBUL ŞEHİR ÜNİVERSİTY

- After collecting data about the things people like, you need a way to determine how similar people are in their tastes.
- Compare each person with every other person and calculate a similarity score.

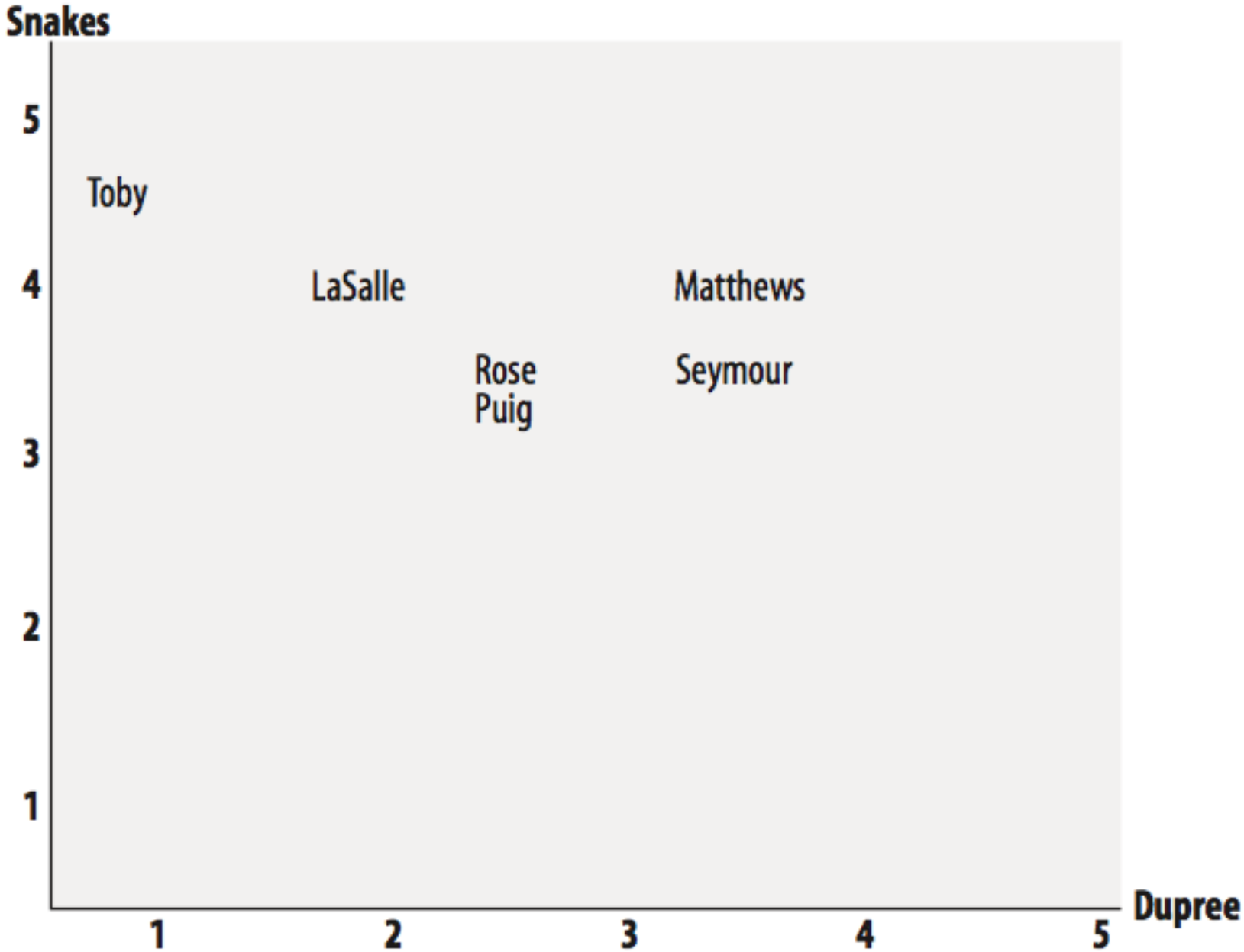
*Different ways of deciding which people are **similar***

=

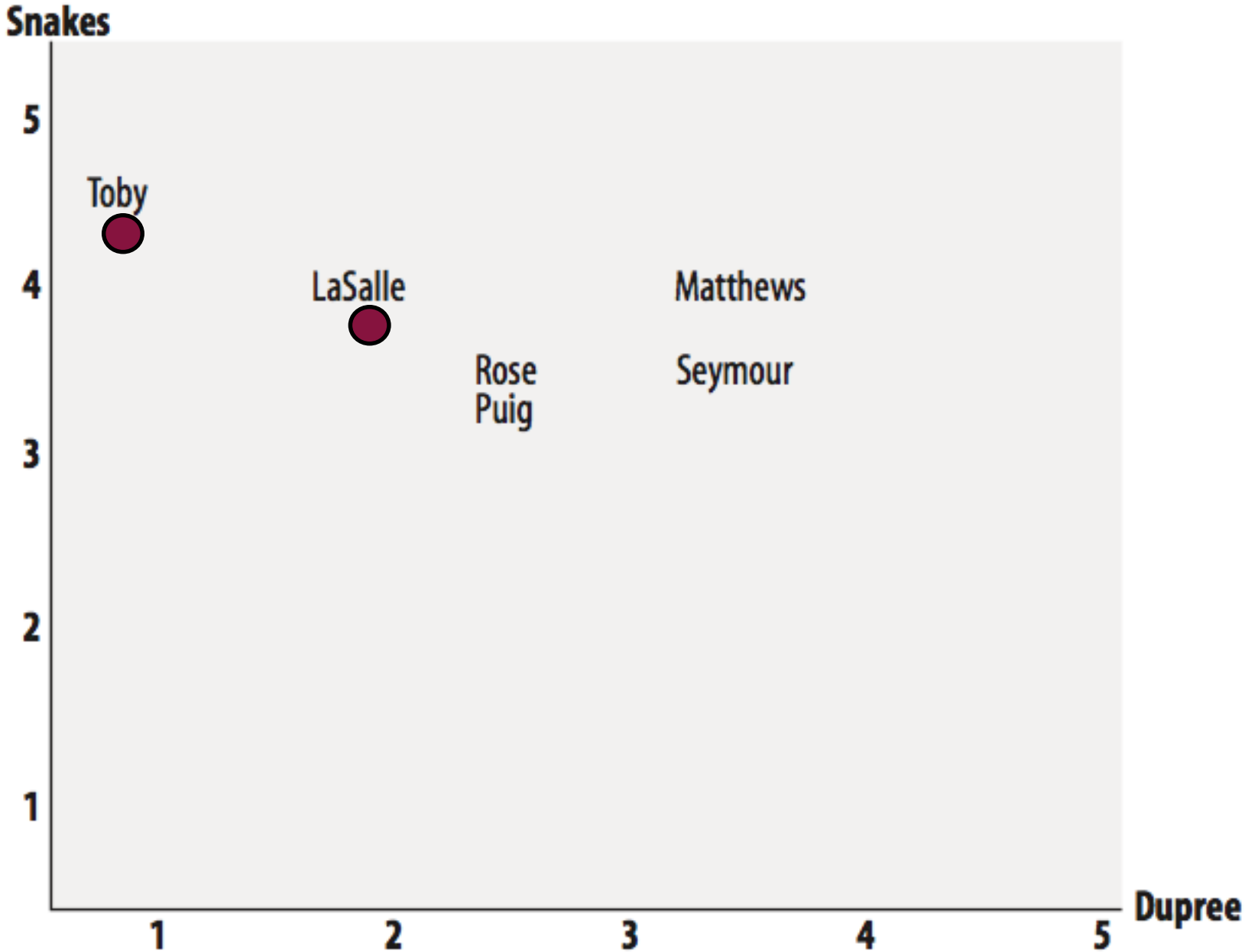
Different Algorithms

- Similarity scores:
  - Euclidean distance,
  - Pearson correlation,
  - etc.

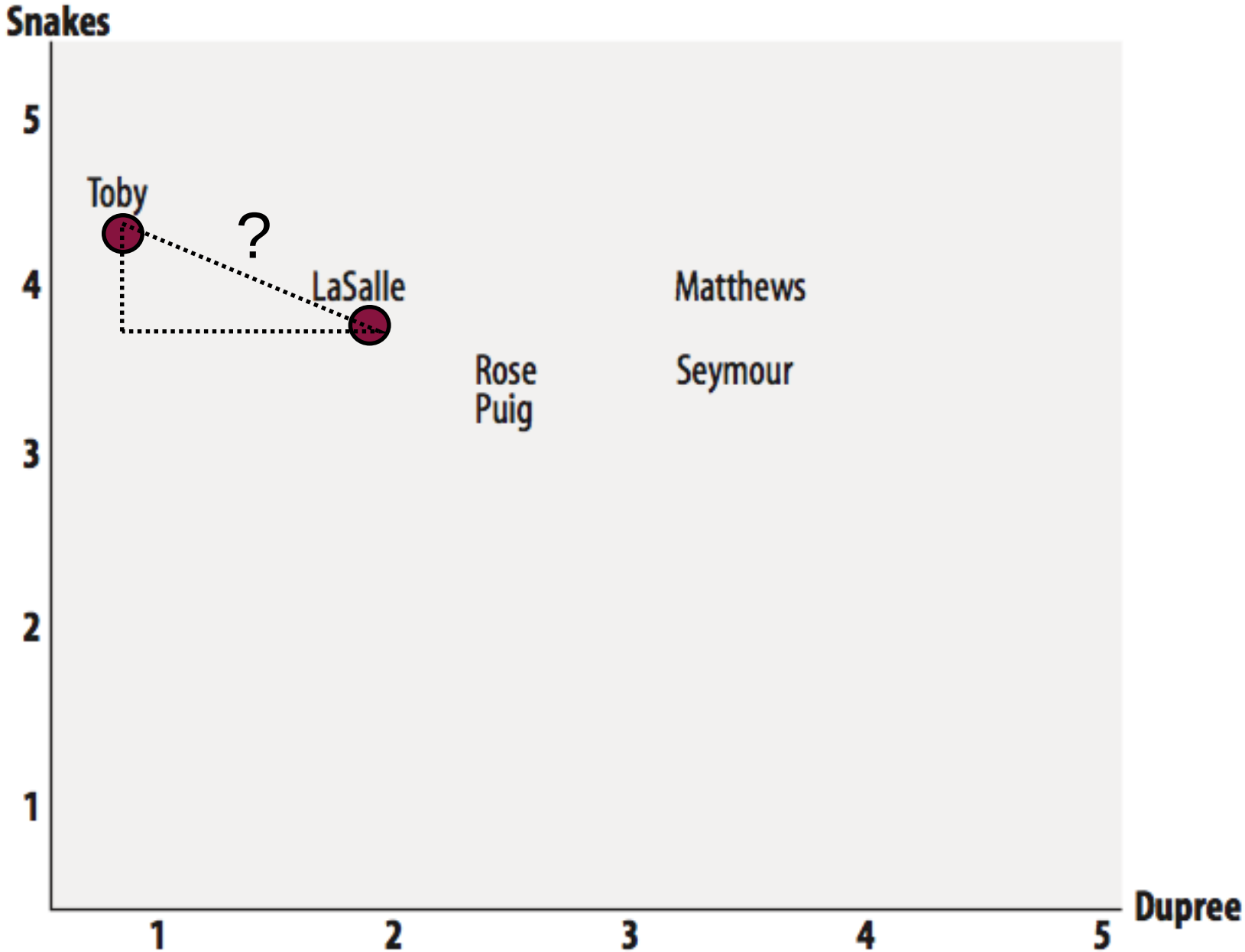
Euclidean Distance



Euclidean Distance



Euclidean Distance



Python Tip: List Comprehension

İSTANBUL ŞEHİR ÜNİVERSİTY

[*expression* **for** *variable* **in** *list*]

or

[*expression* **for** *variable* **in** *list* **if** *condition*]

```
>> l1 = [1, 2, 3, 4, 5, 6]
>> l2 = [v*10 for v in l1]
>> print l2
```

??? [10, 20, 30, 40, 50, 60]

```
>> l3 = [v*10 for v in l1 if v>3]
>> print l3
```

??? [40, 50, 60]

Euclidean Distance

İSTANBUL ŞEHİR ÜNİVERSİTY

```
def sim_distance(prefs,person1,person2):
  # Get the list of shared_items
  si={}
  for item in prefs[person1]:
    if item in prefs[person2]: si[item]=1

  # if they have no ratings in common, return 0
  if len(si)== 0: return 0

  # Add up the squares of all the differences
  sum_of_squares=sum([pow(prefs[person1][item]-
                          prefs[person2][item],2)
                      for item in si])

  return 1/(1+sqrt(sum_of_squares))
```

Week 5/code.zip

İSTANBUL ŞEHİR ÜNİVERSİTY

- load the Python module recommendations.py

```
from recommendations import *
```

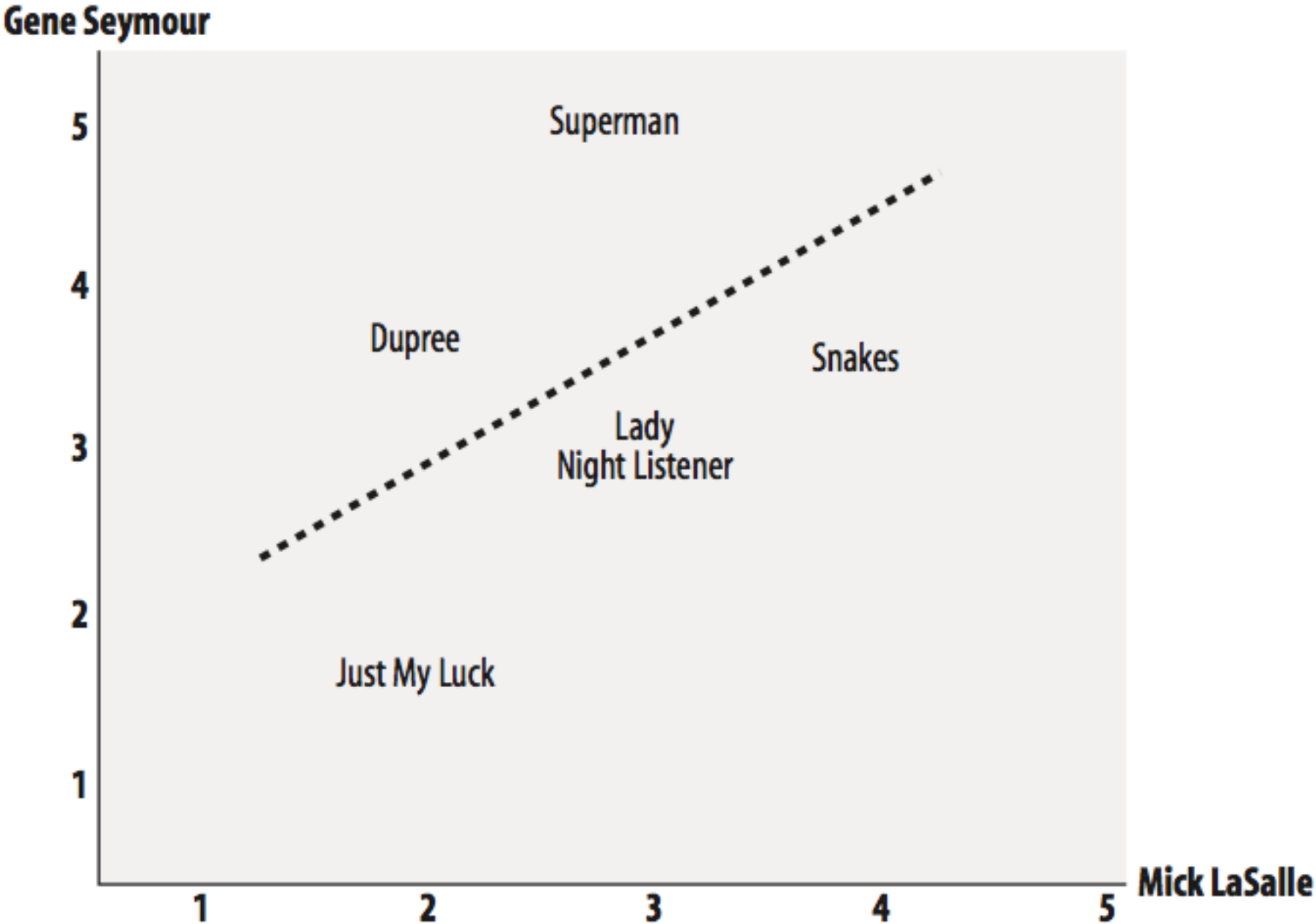
- Play around with sim_distance function, which uses Euclidean distance as a similarity measure.

Pearson Correlation Score

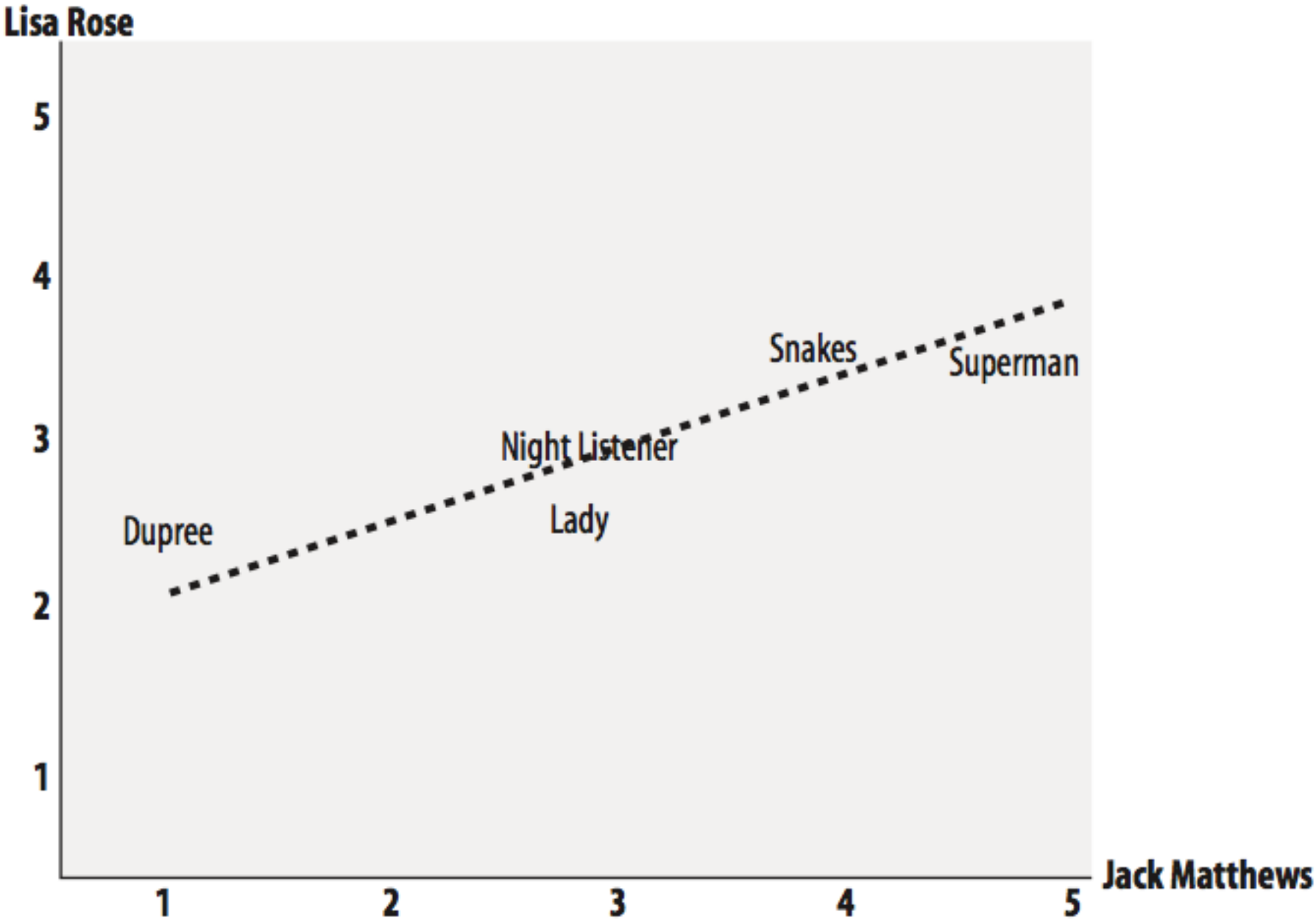
İSTANBUL ŞEHİR ÜNİVERSİTY

- The correlation coefficient is a measure of how well two sets of data fit on a straight line.
- The formula for this is more complicated than the Euclidean distance score
- It tends to give better results in situations where the data isn't well normalized
  - for example, if some critics' movie rankings are routinely more harsh than average.

OK correlation!



Good correlation!



Week 5/code.zip

İSTANBUL ŞEHİR ÜNİVERSİTY

- load the Python module recommendations.py

```
from recommendations import *
```

- Play around with sim_pearson function, which uses Correlation coefficient as a similarity measure.

Ranking the critics

İSTANBUL ŞEHİR ÜNİVERSİTY

"Which movie critics have tastes similar to mine so that I know whose advice I should take when deciding on a movie!"

- Create a function that ***scores everyone against a given person*** and finds the closest matches.

Ranking the critics

İSTANBUL ŞEHİR ÜNİVERSİTY

```
# Returns the best matches for person from the prefs dictionary.
# Number of results and similarity function are optional params.

def topMatches(prefs,person,n=5,similarity=sim_pearson):

  scores=[(similarity(prefs,person,other),other)

                  for other in prefs if other!=person]

  scores.sort()

  scores.reverse()

  return scores[0:n]
```

Week 5/code

İSTANBUL ŞEHİR ÜNİVERSİTY

- load the Python module recommendations.py

```
from recommendations import *
```

- Play around with topMatches function.

Recommending Items

İSTANBUL ŞEHİR ÜNİVERSİTY

- Finding a good critic to read is great, but what I really want is a movie recommendation.
- Simple solution:
  - look at the person who has tastes most similar to mine, and
  - look for a movie he likes that I haven't seen yet.

Recommending Items

İSTANBUL ŞEHİR ÜNİVERSİTY

- Simple approach could turn up reviewers who haven't reviewed some of the movies that I might like.
- Simple approach may return a reviewer who strangely liked a movie that got bad reviews from all the other critics returned by topMatches.
- Solution:
  - ***Take the votes of all the other critics and multiply how similar they are to me by the score they gave each movie.***

Recommending Items

İSTANBUL ŞEHİR ÜNİVERSİTY

Critic	Similarity	Night	S.xNight	Lady	S.xLady	Luck	S.xLuck
Rose	0.99	3.0	2.97	2.5	2.48	3.0	2.97
Seymour	0.38	3.0	1.14	3.0	1.14	1.5	0.57
Puig	0.89	4.5	4.02			3.0	2.68
LaSalle	0.92	3.0	2.77	3.0	2.77	2.0	1.85
Matthews	0.66	3.0	1.99	3.0	1.99		
Total			12.89		8.38		8.07
Sim. Sum			3.84		2.95		3.18
Total/Sim. Sum			3.35		2.83		2.53

```
def getRecommendations(prefs,person,similarity=sim_pearson):
  totals={}
  simSums={}
  for other in prefs:
    # don't compare me to myself
    if other==person: continue
    sim=similarity(prefs,person,other)

    # ignore scores of zero or lower
    if sim<=0: continue

    for item in prefs[other]:
      # only score movies I haven't seen yet
      if item not in prefs[person]:
        # Similarity * Score
        totals.setdefault(item,0)
        totals[item]+=prefs[other][item]*sim
        # Sum of similarities
        simSums.setdefault(item,0)
        simSums[item]+=sim

  # Create the normalized list
  rankings=[(total/simSums[item],item) for item,total in totals.items()]

  # Return the sorted list
  rankings.sort()
  rankings.reverse()
  return rankings
```

İSTANBUL ŞEHİR ÜNİVERSİTY

Week 5/code

İSTANBUL ŞEHİR ÜNİVERSİTY

- Play around with getRecommendations function.

```
from recommendations import *
print getRecommendations(critics, ‘Ali’)
```