ENGR 102 PROGRAMMING PRACTICE

WEEK 5

Graphical User Interface (GUI) Programming

Events

- An ***event*** is some occurrence that your application needs to know about.
- An ***event handler*** is a function in your application that gets called when an event occurs.
- We call it ***binding*** when your application sets up an event handler that gets called when an event happens to a widget.

Events

- Event Sources:
  - Mouse operations by user
  - Key presses by user
  - Redraw events by window manager

- Capturing and handling events
- If an event matching the *event* description occurs on the widget, *handler* is called with an event object

```
widget.bind(event_type, handler_function)
```

Events – Capturing Clicks

```
from tkinter import *

class MyApp(Frame):
    def __init__(self, parent):
        Frame.__init__(self, parent)
        self.initUI()

    def initUI(self):
        self.pack(fill=BOTH, expand=True)
        label = Label(self, bg="yellow")
        label.pack(fill=BOTH, expand=True)

        label.bind("<Button-1>", self.on_click)

    def on_click(self, event):
        print("clicked at", event.x, event.y)

def main():
    root = Tk()
    root.geometry('300x300+200+200')
    app = MyApp(root)
    root.mainloop()

main()
```

Events – Capturing Clicks

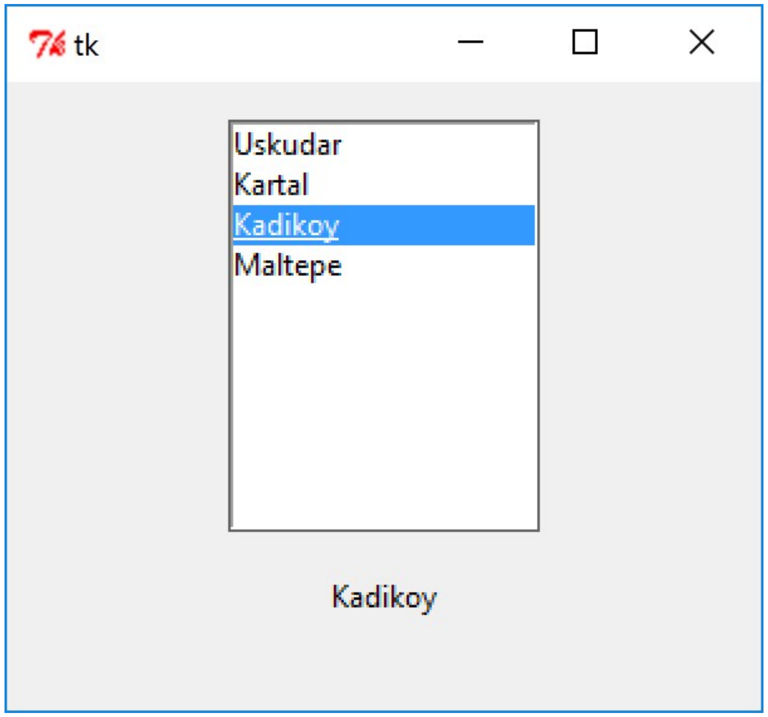
```
from tkinter import *
class MyApp(Frame):
    def __init__(self, parent):
        Frame.__init__(self, parent)
        self.colors = ['red', 'yellow', 'green']
        self.color_index = 0
        self.initUI()

    def initUI(self):
        self.pack(fill=BOTH, expand=True)
        label = Label(self, text = 'Click me to change my color!')
        label.pack(fill=BOTH, expand=True)

        label.bind("<Button-3>", self.on_click)

    def on_click(self, event):
        widget = event.widget
        self.color_index = (self.color_index + 1) % len(self.colors)
        widget.config(bg=self.colors[self.color_index])
        print("clicked at", event.x, event.y)

def main():
    root = Tk()
    root.geometry('300x300+200+200')
    app = MyApp(root)
    root.mainloop()

main()
```

Locating Event Widget

```
from tkinter import *

class MyApp(Frame):
    def __init__(self, parent):
        Frame.__init__(self, parent)
        self.initUI()

    def initUI(self):
        self.pack(fill=BOTH, expand=True)
        for i in range(10):
            button = Button(self, text=str(i))
            button.pack(fill=BOTH, expand=True)
            button.bind("<Button-1>", self.on_click)

        self.label = Label(self)
        self.label.pack()

    def on_click(self, event):
        widget = event.widget
        self.label["text"] = "You clicked on button " + widget["text"]

def main():
    root = Tk()
    root.geometry('300x300+200+200')
    app = MyApp(root)
    root.mainloop()

main()
```

Events – Capturing ListBox Selections

```python
from tkinter import *
class MyApp(Frame):
    def __init__(self, parent):
        Frame.__init__(self, parent)
        self.parent = parent
        self.initUI()

    def initUI(self):
        self.pack(fill=BOTH, expand=1)
        acts = ['Uskudar', 'Kartal', 'Kadikoy', 'Maltepe']
        lb = Listbox(self)
        for i in acts:
            lb.insert(END, i)

        lb.bind("<<ListboxSelect>>", self.on_select)
        lb.pack(pady=15)
        self.var = StringVar()
        self.label = Label(self, text=0, textvariable=self.var)
        self.label.pack()

    def on_select(self, val):
        sender = val.widget
        idx = sender.curselection()
        value = sender.get(idx)
        self.var.set(value)

def main():
    root = Tk()
    ex = MyApp(root)
    root.geometry("300x250+300+300")
    root.mainloop()

main()
```



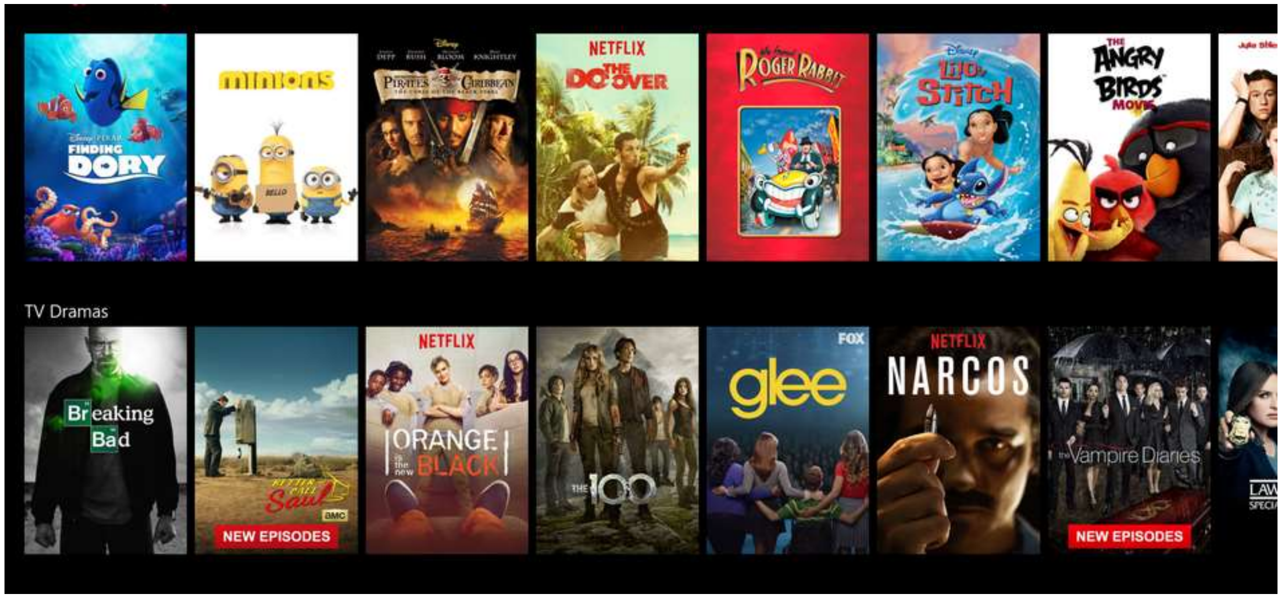
List of Events

- <Button-1> <Button-2> <Button-3> <Button-4> <Button-5>
- <Motion>
- <ButtonRelease-1> <ButtonRelease-2> <ButtonRelease-3>
- <Double-Button-1> <Double-Button-2> <Double-Button-3>
- <Enter>
- <Leave>
- <FocusIn>
- <FocusOut>
- <Return>
- <Key>
- a
- <Shift-Up>
- <Configure>

Ref: http://effbot.org/tkinterbook/tkinter-events-and-bindings.htm

Collective Intelligence

Introduction



- By using data about which movies each customer enjoyed, Netflix is able to recommend movies to other customers that they may never have even heard of and keep them coming back for more.
- Any way to improve its recommendation system is worth a lot of money to Netflix.

Introduction

- 2006: announcement of a prize of $1 million to the first person to improve the accuracy of its recommendation system by 10 percent.
- Thousands of teams from all over the world.
- The leading team improved 7 percent within 1st year.

Introduction

- Google started in 1998.
  - There were Netscape, AOL, Yahoo, MSN.
- Google took a completely new approach.
  - Rank search results by using the links on millions of web sites to decide which pages were most relevant.
- Google's search results were so much better than those of the other players
  - By 2004, it handled 85 percent of searches on the Web.

What do these companies have in common?

- Please elaborate...

What do these companies have in common?

- The ability to collect information
  - Sophisticated algorithms: ***combine data collected from many different sources***.
- The computational power to interpret it
  - enabling great collaboration opportunities and a better understanding of users/customers (**Data Mining**).
  - Everyone wants to understand their customers better in order to create more targeted advertising.

Making Recommendations

Amazon.com

- Amazon tracks the purchasing habits of all its shoppers
  - when you log onto the site, it uses this information to suggest products you might like.
- Amazon can even suggest *movies* you might like, even if you've only bought **books** from it before.

Collaborative Filtering

- A collaborative filtering algorithm usually works by
  - searching a large group of people, and
  - finding a smaller set with tastes **similar** to yours.
  - looking at other things they like and combining them to create a ranked list of suggestions.

Collaborative Filtering

- A collaborative filtering algorithm usually works by
  - **searching a large group of people**, and
  - finding a smaller set with tastes **similar** to yours.
  - looking at other things they like and combining them to create a ranked list of suggestions.

Preferences

critics dict.

- # Dictionary of movie critics and their ratings of movies

```
critics={
    'Lisa Rose': { 'Lady in the Water': 2.5,
                   'Snakes on a Plane': 3.5,
                   'Just My Luck': 3.0,
                   'Superman Returns': 3.5,
                   'You, Me and Dupree': 2.5,
                   'The Night Listener': 3.0},
    'Gene Seymour': { 'Lady in the Water': 3.0,
                                    ...},
    ...
```

Preferences

critics dict.

- # Dictionary of movie critics and their ratings of movies

```
critics={
    'Lisa Rose': { 'Lady in the Water': 2.5,
                   'Snakes on a Plane': 3.5,
                   'Just My Luck': 3.0,
                   'Superman Returns': 3.5,
                   'You, Me and Dupree': 2.5,
                   'The Night Listener': 3.0},
    'Gene Seymour': { 'Lady in the Water': 3.0,
                                    ...},
    ...
```

Critic

Preferences

critics dict.

- # Dictionary of movie critics and their ratings of movies

```
critics={
    'Lisa Rose': { 'Lady in the Water': 2.5,
                   'Snakes on a Plane': 3.5,
                   'Just My Luck': 3.0,
                   'Superman Returns': 3.5,
                   'You, Me and Dupree': 2.5,
                   'The Night Listener': 3.0},
    'Gene Seymour': { 'Lady in the Water': 3.0,
                                    ...},
    ...
```

Movie

Preferences

critics dict.

- # Dictionary of movie critics and their ratings of movies

```
critics={
    'Lisa Rose': { 'Lady in the Water': 2.5,
                   'Snakes on a Plane': 3.5,
                   'Just My Luck': 3.0,
                   'Superman Returns': 3.5,
                   'You, Me and Dupree': 2.5,
                   'The Night Listener': 3.0},
    'Gene Seymour': { 'Lady in the Water': 3.0,
                                  ...},
    ...
```

Rating: 1-5

recommendations.py
`critics dict.`

- Go to LMS and download recommendations.py (/this week/recommendations.py)
- Create a new Python file in the same directory
- Load the Python module recommendations.py

```
from recommendations import *
```

- and play around with `critics` dictionary (i.e., dataset).

Collaborative Filtering

- A collaborative filtering algorithm usually works by
  - searching a large group of people, and
  - **finding a smaller set with tastes similar to yours**.
  - looking at other things they like and combining them to create a ranked list of suggestions.

Finding Similar Users

- 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.

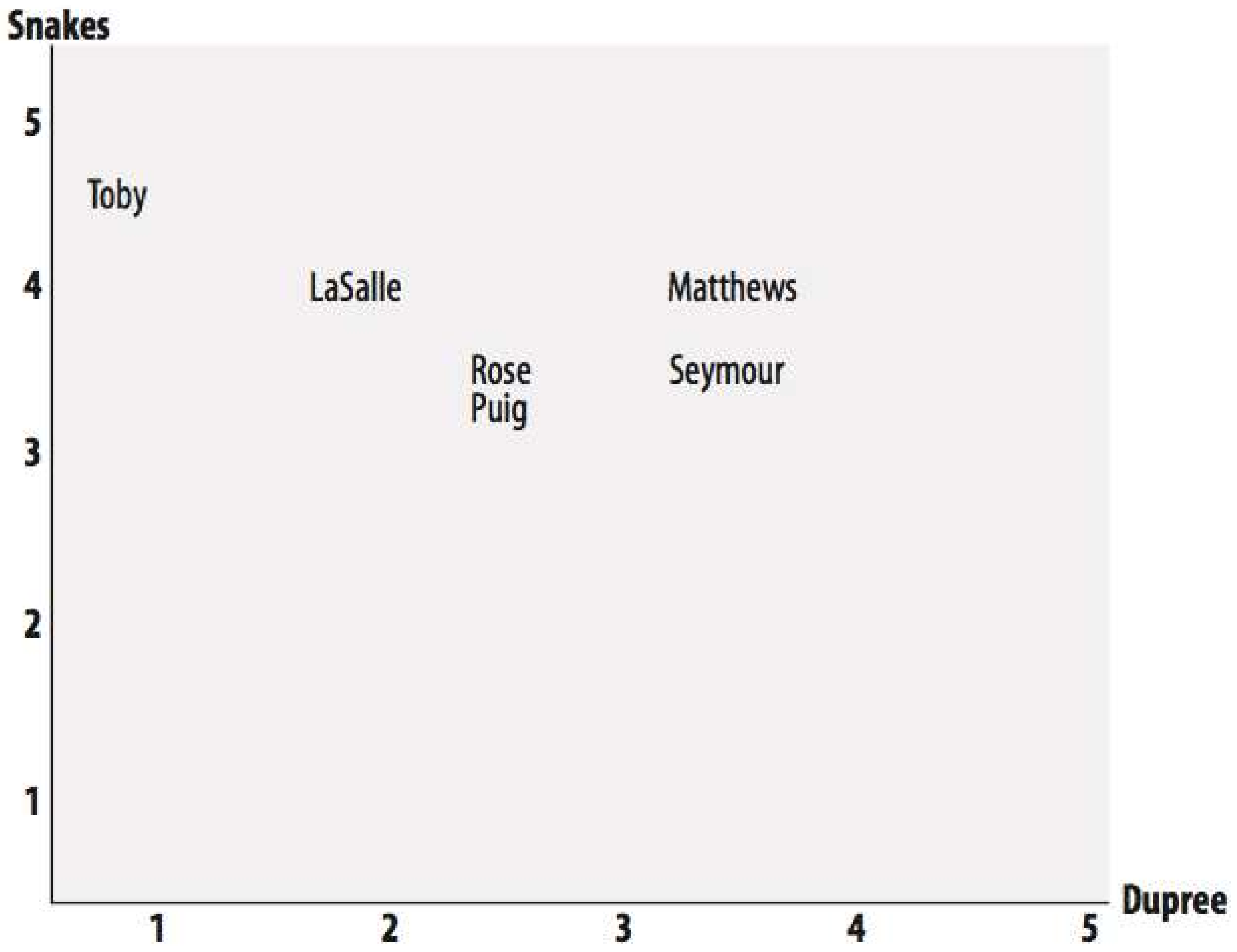
Different ways of deciding which people are ***similar***

=

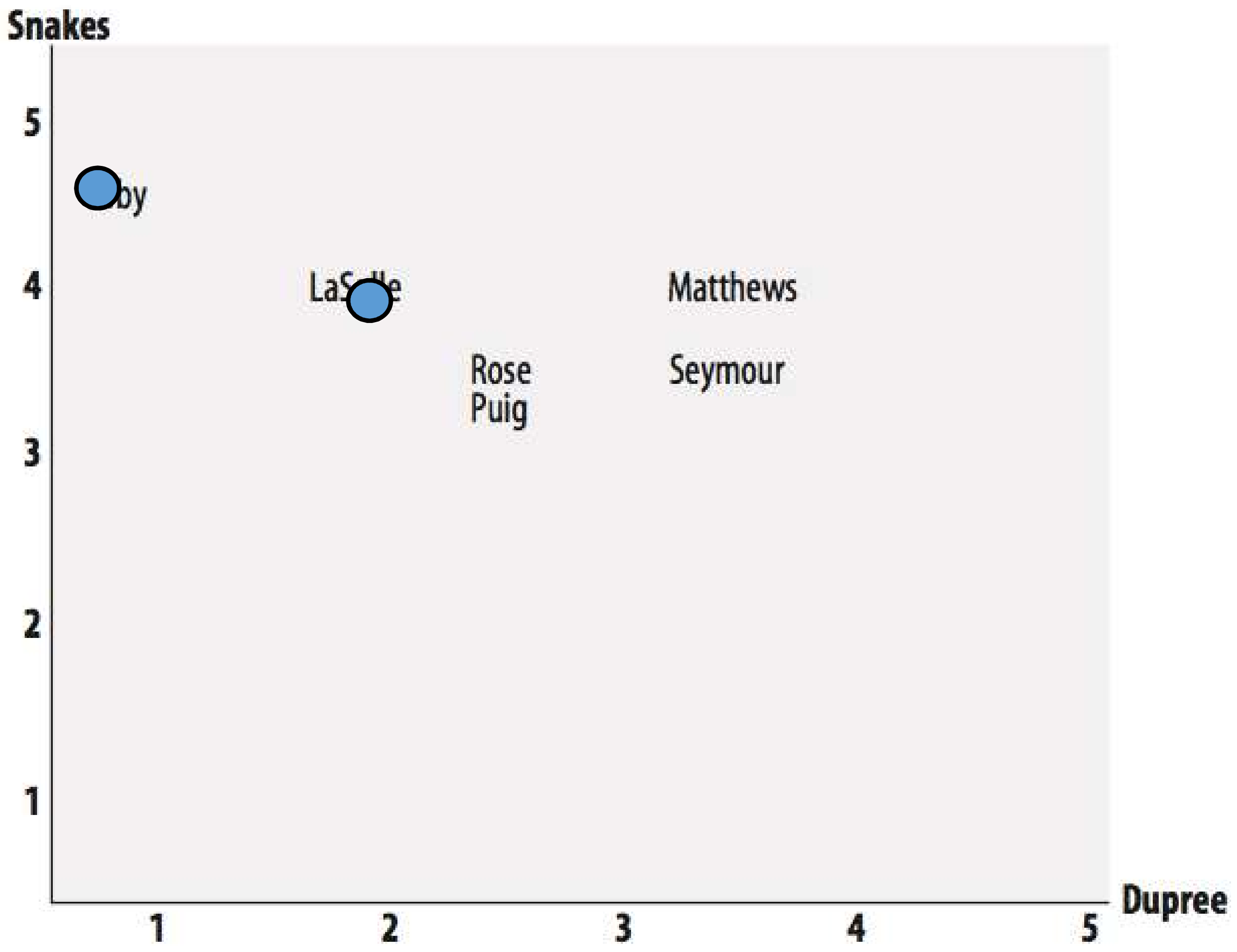
Different Algorithms

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

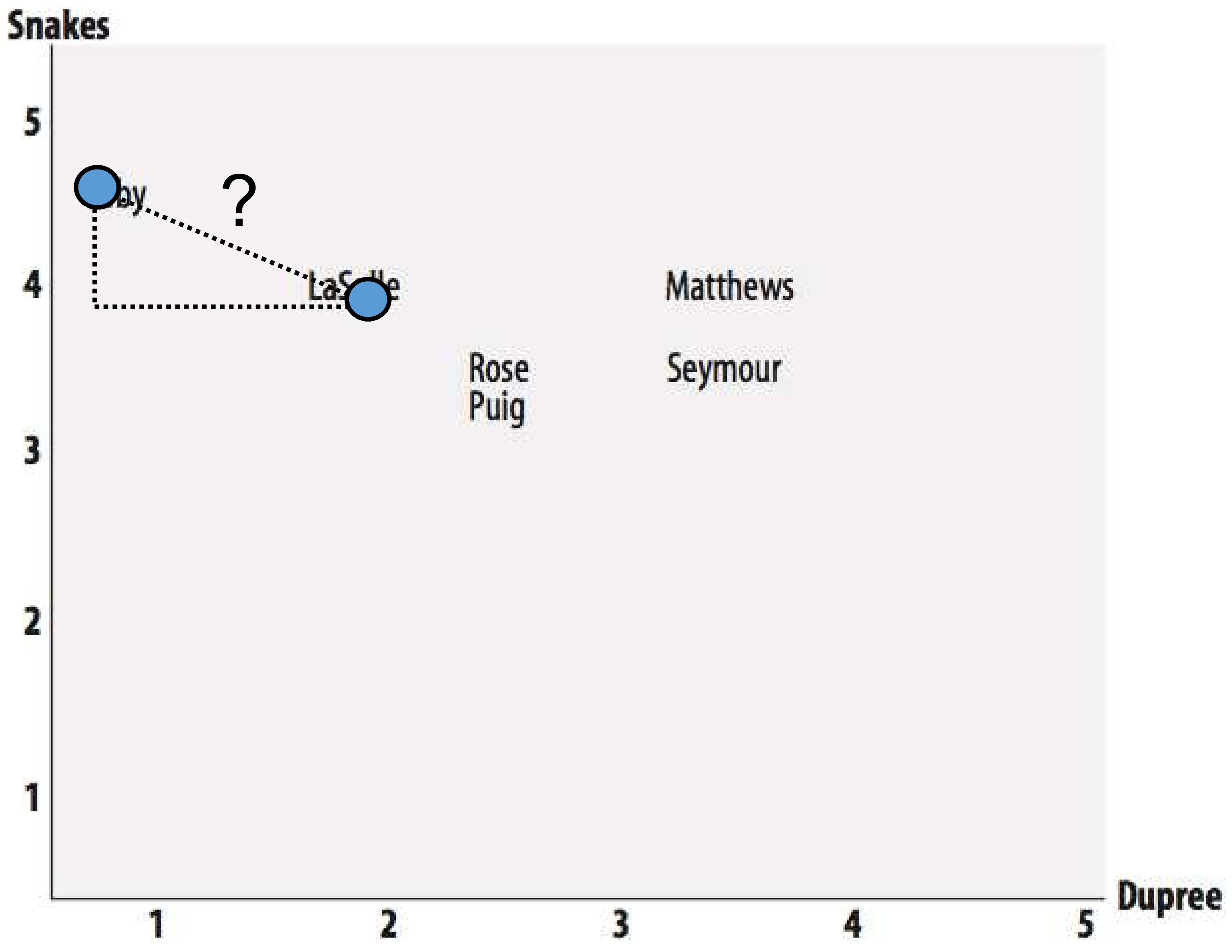
Euclidean Distance



Euclidean Distance



Euclidean Distance


Snakes
by
?
LaSalle
Matthews
Rose
Puig
Seymour
Dupree