Final exam review

CS106AP Lecture 29
Today’s questions

How do I prepare for the final exam?
Today’s topics

1. Data structures, comprehensions, and lambdas
2. Classes
3. Graphics
Practice Problem 1
Lambda Function

\texttt{lambda \ params: \ return\_value}

A lambda is an unnamed, one-line function that takes the above form.

- \texttt{params} is a name for the one or more parameters passed into this function
- \texttt{return\_value} is the expression that the function returns
  - usually an operation on \texttt{params}
- We use lambdas in the optional \texttt{key=} keyword argument of \texttt{min()}, \texttt{max()}, and \texttt{sorted()}

\texttt{lambda \ x: \ 2 * \ x \ # \ doubles \ the \ passed \ in \ parameter}
List Comprehensions

[expression for elem in collection if condition]

- **collection** is some collection of elements (a list, or a dict’s keys, or a tuple, or a string).
- **elem** is a temporary variable representing each element in collection.
- **expression** is a transformed version of **elem** stored in the output list.
  - It can use a function!
- **condition** is an optional condition we can use to only transform the elements in collection for which **condition** is true.
  - If you want to indiscriminately transform all the elements in collection, you can omit **if condition** from the comprehension.
Problem statement

Given a dictionary whose keys are integers and whose values are lists of integers, find a list of the squares of each key, if the sum of the elements in the key’s corresponding value is an even number.

**Input:** \( d = \{1 : [1, 2, 4], 5 : [4, 4, 8, 0], 42 : [42]\} \)

**Output:** \([25, 1764]\)
Problem approach

- Start by writing stuff you know will be there: a foreach loop over elems in list
- Then add the expression
- Then add the condition
Problem statement

- “Battleship” is a well-known kids game, where each player is responsible for placing “ships” on a square grid.

Given a dictionary that uses \((\text{row, col})\) tuples as keys and 0 or 1 as values to indicate whether or not a ship is at that location and an int indicating the grid’s width/height, return a list of lists containing ‘o’ and ‘x’ that represents a Battleship grid.

e.g.
\{ (0,0): 0, (0, 1): 0, (0, 2): 1, (1,0): 0, (1, 1): 0, (1, 2): 1, (2,0): 0, (2, 1): 0, (2, 2): 0 \}  
\[
[ ['o', 'o', 'x'],
  ['o', 'o', 'x'],
  ['o', 'o', 'o']]
\]
Problem solution

def make_board(positions, size):
    board = []
    for row in range(size):
        board_row = []
        for col in range(size):
            val = positions[(row, col)]
            if val == 0:
                board_row.append('o')
            else:
                board_row.append('x')
        board.append(board_row)
    return board
Problem approach

- Start by writing down inputs/outputs
  - What are the specific data types used?
- Write down high-level goal in your own words
  - Brainstorm how to accomplish goal
    - Break ideas into subpieces
    - Code subpieces

Adapted from Brahm Capoor
Problem statement

Given a list of words, produce a \texttt{starts\_with\_dict} count dictionary, where the keys in the dictionary are all of the prefixes contained in the list of words, and each value is a list of full-length words that start with the given prefix. Note: “sonja” is considered a prefix of “sonja”.

\textbf{Input:} [‘sonja’, ‘son’, ‘sun’]

Problem solution

def get_prefix_dict(words):
    prefix_dict = {}
    for word in words:
        for i in range(1, len(word) + 1):
            prefix = word[:i]
            if prefix not in prefix_dict:
                prefix_dict[prefix] = []
            prefix_dict[prefix].append(word)
    return prefix_dict
Core topics:

- **Data structures**
  - Lists
  - Dictionaries
  - Tuples

- **Comprehensions**
  - List, Dict

- **Lambdas**
  - Min, Max, Sorted
Practice Problem 2
You’re remodeling your house and want an easy way to keep track of items you need to get, as well as how much each item might cost (based on their different price points at different stores).

In particular, you have a furniture_options.txt file (an example file is pictured below), in which each line consists of a name of a piece of furniture and how much it costs at a particular location, delineated by a space. Furniture items can be repeated on different lines.

```
mattress 499.99
chair 49.99
mattress 799.99
BED 525.00
BED 223.99
table 20.00
```
Write a class called **FurnitureCatalog** that satisfies the following requirements:

- Reads in the furniture information from a file passed into the constructor and stores it in some data structure that represents a furniture catalog for keeping track of furniture items and their relevant associated prices.
  - This should happen either inside the constructor or inside a method called by the constructor – part of what this problem is assessing is the effective use of data types and instance variables
- Has three specific case-insensitive methods:
  - `get_price_options()` – takes in a string that represents a furniture item and returns a list of all possible costs for that item if that item is in the catalog
  - `get_must_buys()` – returns a list of the items that only appear once inside the furniture catalog, since those items must be purchased at the one specific price that’s listed
  - `add_item()` – takes in a string that represents a furniture item and a float that represents a price and then adds this information to the furniture catalog data structure
How do I approach the problem?
Problem statement

You’re remodeling your house and want an easy way to keep track of items you need to get, as well as how much each item might cost (based on their different price points at different stores).

In particular, you have a `furniture_options.txt` file (an example file is pictured below), in which each line consists of a name of a piece of furniture and how much it costs at a particular location, delineated by a space. Furniture items can be repeated on different lines.

```
mattress 499.99
chAir 49.99
mattress 799.99
BEd 525.00
bED 223.99
table 20.00
```
Problem statement

You’re remodeling your house and want an easy way to keep track of items you need to get, as well as how much each item might cost (based on their different price points at different stores).

In particular, you have a furniture_options.txt file (an example file is pictured below), in which each line consists of a name of a piece of furniture and how much it costs at a particular location, delineated by a space. Furniture items can be repeated on different lines.

<table>
<thead>
<tr>
<th></th>
<th>mattress 499.99</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>chAir 49.99</td>
</tr>
<tr>
<td>2</td>
<td>mattress 799.99</td>
</tr>
<tr>
<td>3</td>
<td>BED 525.00</td>
</tr>
<tr>
<td>4</td>
<td>bED 223.99</td>
</tr>
<tr>
<td>5</td>
<td>table 20.00</td>
</tr>
</tbody>
</table>
You’re remodeling your house and want an easy way to keep track of items you need to get, as well as how much each item might cost (based on their different price points at different stores).

In particular, you have a furniture_options.txt file (an example file is pictured below), in which each line consists of a name of a piece of furniture and how much it costs at a particular location, delineated by a space. Furniture items can be repeated on different lines.

```
1 mattress 499.99
2 chAir 49.99
3 mattress 799.99
4 BEd 525.00
5 bED 223.99
6 table 20.00
```
Problem statement

You’re remodeling your house and want an easy way to keep track of items you need to get, as well as how much each item might cost (based on their different price points at different stores).

In particular, you have a `furniture_options.txt` file (an example file is pictured below), in which each line consists of a name of a piece of furniture and how much it costs at a particular location, delineated by a space. Furniture items can be repeated on different lines.

```
mattress 499.99
chair 49.99
mattress 799.99
bed 525.00
bed 223.99
table 20.00
```
You’re remodeling your house and want an easy way to keep track of items you need to get, as well as how much each item might cost (based on their different price points at different stores).

In particular, you have a furniture_options.txt file (an example file is pictured below), in which each line consists of a name of a piece of furniture and how much it costs at a particular location, delineated by a space. Furniture items can be repeated on different lines.

```
mattress 499.99
chair 49.99
mattress 799.99
bed 525.00
bed 223.99
table 20.00
```
Problem statement

You’re remodeling your house and want an easy way to keep track of items you need to get, as well as how much each item might cost (based on their different price points at different stores).

In particular, you have a furniture_options.txt file (an example file is pictured below), in which each line consists of a name of a piece of furniture and how much it costs at a particular location, delineated by a space. Furniture items can be repeated on different lines.

```
mattress 499.99
chair 49.99
mattress 799.99
bed 525.00
bed 223.99
table 20.00
```

Inconsistent casing
Problem statement

You’re remodeling your house and want an easy way to keep track of items you need to get, as well as how much each item might cost (based on their different price points at different stores).

In particular, you have a .txt file (an example file is pictured below), in which each line consists of a name of a piece of furniture and how much it costs at a particular location, delineated by a space. Furniture items can be repeated on different lines.

```
1 mattress 499.99
2 chAir 49.99
3 mattress 799.99
4 BEd 525.00
5 bED 223.99
6 table 20.00
```
Problem statement

Write a class called **FurnitureCatalog** that satisfies the following requirements:

- Reads in the furniture information from a file passed into the constructor and stores it in some data structure that represents a furniture catalog for keeping track of furniture items and their relevant associated prices.
  - This should happen either inside the constructor or inside a method called by the constructor – part of what this problem is assessing is the effective use of data types and instance variables
- Has three specific case-insensitive methods:
  - `get_price_options()` – takes in a string that represents a furniture item and returns a list of all possible costs for that item if that item is in the catalog
  - `get_must_buys()` – returns a list of the items that only appear once inside the furniture catalog, since those items must be purchased at the one specific price that’s listed
  - `add_item()` – takes in a string that represents a furniture item and a float that represents a price and then adds this information to the furniture catalog data structure
Write a class called **FurnitureCatalog** that satisfies the following requirements:

- Reads in the furniture information from a file passed into the constructor and stores it in some data structure that represents a furniture catalog for keeping track of furniture items and their relevant associated prices.
  - This should happen either inside the constructor or inside a method called by the constructor – part of what this problem is assessing is the effective use of data types and instance variables
- Has three specific case-insensitive methods:
  - `get_price_options()` – takes in a string that represents a furniture item and returns a list of all possible costs for that item if that item is in the catalog
  - `get_must_buys()` – returns a list of the items that only appear once inside the furniture catalog, since those items must be purchased at the one specific price that’s listed
  - `add_item()` – takes in a string that represents a furniture item and a float that represents a price and then adds this information to the furniture catalog data structure
Problem statement

Write a class called `FurnitureCatalog` that satisfies the following requirements:

- **Reads in the furniture information from a file passed into the constructor** and stores it in some data structure that represents a furniture catalog for keeping track of furniture items and their relevant associated prices.
  - This should happen either inside the constructor or inside a method called by the constructor – part of what this problem is assessing is the effective use of data types and instance variables

- **Has three specific case-insensitive methods:**
  - `get_price_options()` – takes in a string that represents a furniture item and returns a list of all possible costs for that item if that item is in the catalog
  - `get_must_buys()` – returns a list of the items that only appear once inside the furniture catalog, since those items must be purchased at the one specific price that’s listed
  - `add_item()` – takes in a string that represents a furniture item and a float that represents a price and then adds this information to the furniture catalog data structure
Problem statement

Write a class called `FurnitureCatalog` that satisfies the following requirements:

- **Reads in the furniture information from a file passed into the constructor** and stores it in some data structure that represents a furniture catalog for keeping track of furniture items and their relevant associated prices.
  - This should happen either inside the constructor or inside a method called by the constructor — part of what this problem is assessing is the effective use of data types and instance variables.

- **Has three specific case-insensitive methods:**
  - `get_price_options()` — takes in a string that represents a furniture item and returns a list of all possible costs for that item if that item is in the catalog.
  - `get_must_buys()` — returns a list of the items that only appear once inside the furniture catalog, since those items must be purchased at the one specific price that’s listed.
  - `add_item()` — takes in a string that represents a furniture item and a float that represents a price and then adds this information to the furniture catalog data structure.
Problem statement

Write a class called FurnitureCatalog that satisfies the following requirements:

● **Reads in the furniture information from a file passed into the constructor** and stores it in some data structure that represents a furniture catalog for keeping track of furniture items and their relevant associated prices.
  ○ This should happen either inside the constructor or inside a method called by the constructor – part of what this problem is assessing is the effective use of data types and instance variables
● **Has three specific case-insensitive methods:**
  ○ `get_price_options()` – takes in a string that represents a furniture item and returns a list of all possible costs for that item if that item is in the catalog
  ○ `get_must_buys()` – returns a list of the items that only appear once inside the furniture catalog, since those items must be purchased at the one specific price that’s listed
  ○ `add_item()` – takes in a string that represents a furniture item and a float that represents a price and then adds this information to the furniture catalog data structure.
Core topics:

- File reading
  - Common paradigm: read data in from a file (often line-by-line) and populate a data structure
- Classes
  - Identify the three main parts.
    - Constructor: What happens when you make a new instance of this type?
    - Attributes: What subvariables make up this new variable type?
    - Methods: What functions can you call on a variable of this type?
  - Think about any error checking you’ll need to do!
  - Go back and double-check type conversions
Practice Problem 3
Problem statement

While on an alien planet, Karel figures it’s good to collect samples of space rocks. Help Karel by writing a complete graphical program to help organize and discard space rock samples.

When you start the program, the canvas should display the space rocks as GOvals with random colors. The rocks should be positioned starting at the left of the canvas, with the first rock positioned at (0, 0). You can assume that the window height and width have been provided. When your program first runs, it should look something like this:
Problem statement

If the user clicks on one space rock, followed by another space rock, those two space rocks should swap positions in the row of rocks. For instance, if the user clicks on the first (blue) rock, followed by the last (gray) rock, they would swap and the display would look like so:
Problem statement

If the user clicks on one space rock, and again on the same space rock, that rock should be removed, and all rocks to its right should shift to the left so that the rocks are still correctly padded. For instance, if the user clicks twice on the 4th rock from the left (yellow), you should remove that rock and shift to the left the three rocks to its right to fill the empty space, like so:
Problem statement

Assumptions and Specifications

- You may assume that the user will only click on the rocks (and not any blank spaces).
- You may not change the color of any rocks after they are initialized to a random color.

Hint: To make the program simpler, we recommend writing a method like `reposition_rocks` that updates the position of all rocks still present.
You should define a class called `SpaceRocksGraphics` to achieve the desired functionality. Your class will be constructed as follows and no further method calls will be made. Do not modify the contents of the `main()` function.

```python
def main():
    space_rocks = SpaceRocksGraphics(width=800, height=80)
```

When writing this program, you should make use of the following provided constants (you do not need to define these in your answer):

- `NUM_ROCKS = 8`  # number of space rocks to display
- `ROCK_DIAMETER = 80`  # width/height of a space rock
- `ROCK_SPACING = 20`  # pixels between each space rock
How do I approach the problem?
Problem statement

While on an alien planet, Karel figures it’s good to collect samples of space rocks. Help Karel by writing a complete graphical program to help organize and discard space rock samples.

When you start the program, the canvas should display the space rocks as GOvals with random colors. The rocks should be positioned starting at the left of the canvas, with the first rock positioned at (0, 0). You can assume that the window height and width have been provided. When your program first runs, it should look something like this:
Problem statement

While on an alien planet, Karel figures it’s good to collect samples of space rocks. Help Karel by writing a complete graphical program to help organize and discard space rock samples.

When you start the program, the canvas should display the space rocks as GOvals with random colors. The rocks should be positioned starting at the left of the canvas, with the first rock positioned at (0, 0). You can assume that the window height and width have been provided. When your program first runs, it should look something like this:
Problem statement

While on an alien planet, Karel figures it’s good to collect samples of space rocks. Help Karel by writing a complete graphical program to help organize and discard space rock samples.

When you start the program, the canvas should display the space rocks as GOvals with random colors. The rocks should be positioned starting at the left of the canvas, with the first rock positioned at (0, 0). You can assume that the window height and width have been provided. When your program first runs, it should look something like this:
Problem statement

While on an alien planet, Karel figures it’s good to collect samples of space rocks. Help Karel by writing a complete graphical program to help organize and discard space rock samples.

When you start the program, the canvas should display the space rocks as GOvals with random colors. The rocks should be positioned starting at the left of the canvas, with the first rock positioned at (0, 0). You can assume that the window height and width have been provided. When your program first runs, it should look something like this:

![Space Rocks!](image-url)
Problem statement

While on an alien planet, Karel figures it’s good to collect samples of space rocks. Help Karel by writing a complete graphical program to help organize and discard space rock samples.

When you start the program, the canvas should display the space rocks as GOvals with random colors. The rocks should be positioned starting at the left of the canvas, with the first rock positioned at (0, 0). You can assume that the window height and width have been provided. When your program first runs, it should look something like this:
Problem statement

If the user clicks on one space rock, followed by another space rock, those two space rocks should swap positions in the row of rocks. For instance, if the user clicks on the first (blue) rock, followed by the last (gray) rock, they would swap and the display would look like so:

![Space Rocks!](image)
Problem statement

If the user **clicks** on one space rock, followed by another space rock, those two space rocks **should swap positions in the row of rocks**. For instance, if the user clicks on the first (blue) rock, followed by the last (gray) rock, they would swap and the display would look like so:

![Space Rocks!](image_url)
If the user clicks on one space rock, and again on the same space rock, that rock should be removed, and all rocks to its right should shift to the left so that the rocks are still correctly padded. For instance, if the user clicks twice on the 4th rock from the left (yellow), you should remove that rock and shift to the left the three rocks to its right to fill the empty space, like so:
Problem statement

If the user **clicks** on one space rock, and **again on the same space rock**, that *rock should be removed*, and all *rocks to its right should shift to the left* so that the *rocks are still correctly padded*. For instance, if the user clicks twice on the 4th rock from the left (yellow), you should remove that rock and shift to the left the three rocks to its right to fill the empty space, like so:
Problem statement

If the user **clicks** on one space rock, and again **on the same space rock**, that rock **should be removed**, and all rocks to its right should shift to the left so that the rocks are still correctly padded. For instance, if the user clicks twice on the 4th rock from the left (yellow), you should remove that rock and shift to the left the three rocks to its right to fill the empty space, like so:
Problem statement

If the user **clicks** on one space rock, and **again on the same space rock**, that **rock should be removed**, and **all rocks to its right should shift to the left** so that the **rocks are still correctly padded**. For instance, if the user clicks twice on the 4th rock from the left (yellow), you should remove that rock and shift to the left the three rocks to its right to fill the empty space, like so:
Problem statement

Assumptions and Specifications

- You may assume that the user will only click on the rocks (and not any blank spaces).
- You may not change the color of any rocks after they are initialized to a random color.

Hint: To make the program simpler, we recommend writing a method like `reposition_rocks` that updates the position of all rocks still present.
Problem statement

Assumptions and Specifications

- You may assume that the user will only click on the rocks (and not any blank spaces).
- You may not change the color of any rocks after they are initialized to a random color.

Hint: To make the program simpler, we recommend writing a method like `reposition_rocks` that updates the position of all rocks still present.
Problem statement

You should define a class called SpaceRocksGraphics to achieve the desired functionality. Your class will be constructed as follows and no further method calls will be made. Do not modify the contents of the main() function.

```python
def main():
    space_rocks = SpaceRocksGraphics(width=800, height=80)
```

When writing this program, you should make use of the following provided constants (you do not need to define these in your answer):

```
NUM_ROCKS = 8  # number of space rocks to display
ROCK_DIAMETER = 80  # width/height of a space rock
ROCK_SPACING = 20  # pixels between each space rock
```
You should define a **class** called `SpaceRocksGraphics` to achieve the desired functionality. Your class will be constructed as follows and no further method calls will be made. Do not modify the contents of the `main()` function.

```python
def main():
    space_rocks = SpaceRocksGraphics(width=800, height=80)
```

When writing this program, you should make use of the **following provided constants** (you do not need to define these in your answer):

```python
NUM_ROCKS = 8  # number of space rocks to display
ROCK_DIAMETER = 80  # width/height of a space rock
ROCK_SPACING = 20  # pixels between each space rock
```
Implementing the solution

[space_rocks.py]
Core topics:

● Campy
  ○ GOBjects
    ■ Think about the different GOBjects you might need to construct. What are their dimensions, colors, and other properties? How do you set these properties?
  ○ Mouse events
    ■ Think about the different mouse events you need to handle. Are there multiple different actions that might need to be taken for a single event? In what way are the coordinates of the mouse click useful (if at all)?

● Classes
  ○ Identify the three main parts.
    ■ Constructor: What is the initial state of my graphics program?
    ■ Attributes: What pieces of information do I need to persist across the lifetime of my program?
    ■ Methods: What functions will be helpful to accomplish the goals of the program?
What’s next?