Final Review Day

Pick up a study sheet and a number
Encouraged: TA hours, office hours, Piazza for final review

Final project is due May 3 (Wednesday)

TA review session: May 3 (Wednesday), 7:30-9:30pm

Practice midterm during lab this week

Self-scheduled final exam (similar style to the midterm)

Last office hours: Monday 3-5pm (final prep only)
Outline: 5/1

+ While loops
+ Files
+ Dictionaries
+ Activity that combines while loops, files, and dictionaries
+ Wed: recursion and classes
Preparing for the Final
Study Strategies

- Use the development of your cheat sheet as a way to structure reviewing the material
- Go over lectures notes/code, homeworks, labs (your code and the solutions)
- Use the shell to test out things as much as possible
- Redo parts of in-class code or assignments on paper without looking at the solutions
- Take the practice midterm and review the feedback
- Do the practice problems in the book
Practice Midterm

+ During each lab section (you don’t have to go to your assigned section)

+ TAs will provide feedback

+ I will add **3 points to your final exam score** if you take the practice midterm during lab, give an honest effort, put your name on it, turn it in, and **PICK IT UP** after feedback has been given
While loops
Idea of while loops

while <condition>:
<code>
Idea of while loops

1) Evaluate `<condition>` -> True
Idea of while loops

while <condition>:
<code>

1) Evaluate <condition> -> True

2) Enter while loop and execute <code>
### Idea of while loops

<table>
<thead>
<tr>
<th>Line 1</th>
<th>Line 2</th>
</tr>
</thead>
</table>
| while <condition>:
  <code> |        |

1. Evaluate `<condition>` -> **True**
2. Enter while loop and execute `<code>`
3. Go back and evaluate `<condition>` -> **True**
Idea of while loops

while <condition>:
  <code>

1) Evaluate <condition> -&gt; True

2) Enter while loop and execute <code>

3) Go back and evaluate <condition> -&gt; True

4) Enter while loop and execute <code>
Idea of while loops

while <condition>: <code>

1) Evaluate <condition> -> True
2) Enter while loop and execute <code>
3) Go back and evaluate <condition> -> True
4) Enter while loop and execute <code>
5) Go back and evaluate <condition> -> True
Idea of while loops

while <condition>:  
  <code>

1) Evaluate <condition> -> True

2) Enter while loop and execute <code>

3) Go back and evaluate <condition> -> True

4) Enter while loop and execute <code>

5) Go back and evaluate <condition> -> True

6) Enter while loop and execute <code>
Idea of while loops

while <condition>:
  <code>

1) Evaluate <condition> -> True
2) Enter while loop and execute <code>
3) Go back and evaluate <condition> -> True
4) Enter while loop and execute <code>
5) Go back and evaluate <condition> -> True
6) Enter while loop and execute <code>
7) Go back and evaluate <condition> -> False
Idea of while loops

while <condition>:
  <code>

1) Evaluate <condition> -> True
2) Enter while loop and execute <code>
3) Go back and evaluate <condition> -> True
4) Enter while loop and execute <code>
5) Go back and evaluate <condition> -> True
6) Enter while loop and execute <code>
7) Go back and evaluate <condition> -> False
8) Skip over <code> inside while loop and move on
Common types of while loops

```python
while <condition>:
    <code>

<initialize x>
while x < 400:
    <do something that modifies x>

while <condition1> and <condition2>:
    <code>

while <condition1> or <condition2>:
    <code>
```
Impact of and/or on while loops

while A and B:
  print("both A and B are true")

while A or B:
  print("either A is true, B is true, or both")

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>A and B</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>True</td>
<td>True</td>
</tr>
<tr>
<td>True</td>
<td>False</td>
<td>False</td>
</tr>
<tr>
<td>False</td>
<td>True</td>
<td>False</td>
</tr>
<tr>
<td>False</td>
<td>False</td>
<td>False</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>A or B</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>True</td>
<td>True</td>
</tr>
<tr>
<td>True</td>
<td>False</td>
<td>True</td>
</tr>
<tr>
<td>False</td>
<td>True</td>
<td>True</td>
</tr>
<tr>
<td>False</td>
<td>False</td>
<td>False</td>
</tr>
</tbody>
</table>

Truth table

Truth table
Files
Assignments to review

- **Lab 5**: letter frequencies
- **Lab 7**: decoding hidden message
- **Lab 10**: comparing different species
- **Homework 5**: analyzing tweets from the Twitter file
- **Homework 8**: Smith College map digitization
Writing the file of coordinates

```python
# write the (x,y) positions to a file
for point in building_lst:
    file.write(str(point.getX()) + " " + str(point.getY()) + " ")
file.write("\n")
```

def main():
    """Read a text file of coordinates and draw buildings. Each line of the text file is a separate building. The (x,y) coordinates are read one right after the other, in pairs. """

    # set up our graphics window
    width = 760
    height = 620
    win = GraphWin("Digital Map", width, height)

    # open the file of building coordinates
    file = open("buildings.txt", "r")

    # read each line of the file (one building)
    for line in file:
        tokens = line.strip().split() # split the line

        # read in pairs to get (x,y) coordinates
        point_lst = []
        for i in range(0, len(tokens), 2):
            p = Point(tokens[i], tokens[i+1])
            point_lst.append(p)

        # after building the list of points, create a Polygon
        p = Polygon(point_lst)
        p.setFill("lightgreen")
        p.draw(win)

    # close the file
    file.close()
```
Dictionaries
Dictionaries

- Created using: {}

- To add pairs right away: {0: “A”, 1: “B”}
Dictionaries

- Created using: {}
- To add pairs right away: {0: “A”, 1: “B”}
- Think about dictionaries like lists with a special index
Dictionaries

- Created using: {}
- To add pairs right away: {0: “A”, 1: “B”}
- Think about dictionaries like lists with a special index
- Special index is the “key” (unique), element at that key is the “value”
Dictionaries

- Created using: {}
- To add pairs right away: {0: “A”, 1: “B”}
- Think about dictionaries like lists with a special index
- Special index is the “key” (unique), element at that key is the “value”
- Unlike lists: not ordered
Dictionaries

- Created using: {}
- To add pairs right away: {0: “A”, 1: “B”}
- Think about dictionaries like lists with a special index
- Special index is the “key” (unique), element at that key is the “value”
- Unlike lists: not ordered
- Like lists: can get (lookup), set (assign, mutate)
Dictionaries

- Created using: `{}`
- To add pairs right away: `{0: “A”, 1: “B”}`
- Think about dictionaries like lists with a special index
- Special index is the “key” (unique), element at that key is the “value”
- Unlike lists: not ordered
- Like lists: can get (lookup), set (assign, mutate)
- To add more pairs later: `dictionary[2] = “C”`
Combine dictionaries, files, and while loops
Step 1: make the dictionary

- Find your random partner and introduce yourselves.
- In main, write some code that will ask the user for their 99 number and their name (two questions).
- Use the first two and last two digits for speed and privacy.
- Add the 99 number (key) and name (value) to a dictionary that will keep track of individuals using their 99 numbers.

```python
>>> Enter your 99 number: 9995
>>> Enter your name: Sara Mathieson
>>> {9995: 'Sara Mathieson'}
```
Step 2: use a while loop

+ Create a while loop that will keep asking for more 99 numbers and student names (use you and your partner’s info)

+ Create a way to stop the while loop (i.e. user enters -1 for their number or “stop” for their name)

```python
>>> Enter your 99 number: 9995
Enter your name: Sara Mathieson
Enter your 99 number: 9921
Enter your name: Alan Turing
Enter your 99 number: -1
{9921: 'Alan Turing', 9995: 'Sara Mathieson'}
```
Step 3: write dictionary data to a file

After the while loop is over, write each number and name to a file using a loop over the keys of the dictionary.