Some quick logistics

- August 12th, 8:30am-10:30am
- NVIDIA Auditorium (same room as lecture)
- Open Note (but don’t rely on your notes because of timing)
Let’s Review!
Loops

- **for i in range( start, end, increment )**
  - “end” is the only mandatory one, and is EXCLUSIVE (up to but not including)
  - Example: for i in range( 5 )
  - Example: for i in range( 5, 2, -1 )

- **for elem in structure**
  - Example: for line in file
  - Example: for ch in str

- **while ________**
  - Example: while True
  - Example: while count < 10
Lists

- **LISTS CHEAT SHEET**
- A linear collection of any type of Python value
- Declare an empty list like this: `things = []`
- Use `len(things)` to get the length (# of items)
- 0-indexed, use `things[i]` to access the elements
- Lists are MUTABLE—they can be changed
- `things.append(item)` to add something to a list
- `things.index(target)` to find the index of the target in a list
  - Only works if target is IN the list! Error otherwise :(
Images

```python
image = SimpleImage(filename)
out = SimpleImage.blank( image.width, image.height )
for y in range( image.height ):
    for x in range( image.width ):
        pixel = image.get_pixel(x, y)
        new_pix = out.get_pixel(x, y)
        new_pix.red = pixel.red
        new_pix.green = 0
        new_pix.blue = 0
```
Grids

```python
grid = Grid(3, 2)
grid.width  # returns 3
grid.set(2, 0, 'a')
grid.set(2, 1, 'b')
```
Dictionaries

- [DICTIONARIES CHEAT SHEET](#)
- Declare empty dictionary with curly braces {}
- `dict_name[ key ] = value`
Maps/Lambdas

- **LAMBDAS CHEAT SHEET**
- Runs a lambda function over a list of values

```python
>>> list(map(lambda n: n * 2, [1, 2, 3, 4, 5]))
[2, 4, 6, 8, 10]
```
String Parsing Practice Problem

You shall not parse.
String Parsing Practice

Given a string s, look for a '(____)' within s.

Look for the first '(' in s, then the first ')' after the '('. If both parens are found, return the chars between them.

So 'xxx(abc)xxx' returns 'abc'.

If no such pair of parens is found, return the empty string.

Thinking about this input: ')))(abc)' → how can we find the ')’ without pointing us to the first char?
def parens(s):
    left = s.find('(')
    if left == -1:
        return ''
    right = s.find(')', left + 1)
    if right == -1:
        return ''
    return s[left + 1:right]
Nested Structures

Practice Problem
Nested Structures Practice

You’re hosting a huge dinner party, and you need to cook for a bunch of your friends. They’ve all sent you their allergies, but you need a good way to organize this information...

Let’s use a dictionary! The keys will be the names of your friends, and the values will be lists of their respective allergies.
Nested Structures Practice

We’re going to be given a dictionary that looks like this:

```python
result = {"Grant": [ "peanuts", "kiwis" ],
          "Emily": [ "grapes", "chocolate" ],
          "Jonathan": [ "cheese" ]}
```

Given this existing dictionary, the name of a person, and an allergy that they have, update the dictionary with the person and their allergy.
def update_allergies( allergy_dict, name, allergy):
    if name not in allergy_dict:
        allergy_dict[name] = []
        allergy_dict[name].append( allergy )
    return allergy_dict
Nested Structures Practice (2)

Now that we have our dictionary built, let’s put it to use!

Let’s say we want to know how many allergies a given person has.

Task: given a dictionary *allergies* and a string *name* return the number of allergies the given person has

```python
def num_allergies(allergies, name):
    allergies_list = allergies[name]
    return len(allergies_list)
```
Nested Dictionaries

Practice Problem

Once you’ve read the dictionary, every other book you read is just a remix.
Avatar the Last Airbender Characters

Let’s say we are given a text file containing information about each character of the ATLA universe.

The information includes each character’s name, their age, and the nation they belong to. See below for an example of the inputted file:

Katara-14-Water
Sokka-15-Water
Toph-12-Earth
Bumi-112-Earth
Zuko-16-Fire
Iroh-40-Fire
Azula-14-Fire
Aang-112-Air
Our task is to read (parse) through the inputted file and return a nested dictionary that sorts each character along with their age into their appropriate nation.

result = {Nation: {Name: Age}}
ATLA Problem: Our Task

Katara-14-Water
Sokka-15-Water
Toph-12-Earth
Bumi-112-Earth
Zuko-16-Fire
Iroh-40-Fire
Azula-14-Fire
Aang-112-Air

parse_characters('character_info.txt')

{ 'Water': {'Katara': 14, 'Sokka': 15},
  'Earth': {'Toph': 12, 'Bumi': 112},
  'Fire': {'Zuko': 16, 'Iroh': 40, 'Azula': 14},
  'Air': {'Aang': 112} }
ATLA Problem: Solution?

We can break down our task into three steps:

1. Opening/Reading the file
2. Parsing the information from each line
3. Adding to our result dictionary
Opening Files in Python

```python
code
filename = 'character_info.txt'
with open(filename) as f:
    # CODE HERE
    # In this case we want to deal with each line from our file,
    # so we can say `for line in f:` to iterate over each line of the file
```

FILE READING CHEAT SHEET
Parsing Information From Each Line

```python
filename = 'character_info.txt'
with open(filename) as f:
    for line in f:
        line = line.strip()  # strips any whitespace and newline character
        parts = line.split('-')  # separates line into a list (called parts) by the '-' delimiter
        name = parts[0]  # parts = ['Katara', '14', 'Water'], we can grab each part by indexing into the list
        age = parts[1]
        nation = parts[2]
```
Time to Nest the Dictionaries!

```python
cfilename = 'character_info.txt'
with open(filename) as f:
    result = {}  # We need to first declare our result dictionary that we will be returning!
    for line in f:
        line = line.strip()
        parts = line.split('-')
        name = parts[0]
        age = parts[1]
        nation = parts[2]
        if nation not in result:  # We first must check if the key is in our outer dictionary or not
            result[nation] = {}  # So we add the nation as a new key with a default value (another dictionary!)
        inner_dict = result[nation]  # Let's pull the inner dictionary from our outer one so we can better access it
        inner_dict[name] = int(age)  # Add the age (as a number!) to our inner dictionary
```
Time to Nest the Dictionaries!

```python
filename = 'character_info.txt'
with open(filename) as f:
    result = {}  # We need to first declare our result dictionary that we will be returning!
    for line in f:
        line = line.strip()
        parts = line.split('-')
        name = parts[0]
        age = parts[1]
        nation = parts[2]
        if nation not in result:  # We first must check if the key is in our outer dictionary or not
            result[nation] = {}  # So we add the nation as a new key with a default value (another dictionary!)
            result[nation][name] = int(age)  # This also works!! We grab the nested dict with result[nation] first
    return result
```
And that’s all!

Congrats! You are now the next Mark Zuckerberg!
Questions?