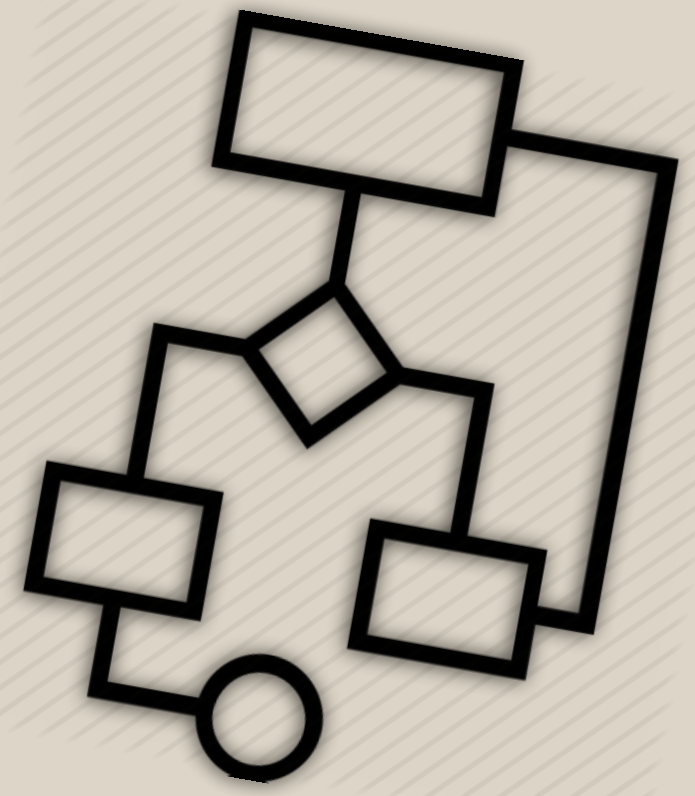
W200 - Python for Data Science

Topic 4:

More About Control and Algorithms



Course Content | First 8 Weeks - Programming

Unit 1 | Introduction, the Command Line, Source Control

Unit 2 | Starting Out with Python

Unit 3 | Sequence Types and Dictionaries

Unit 4 | More About Control and Algorithms

Unit 5 | Functions

Unit 6 | Modules and Packages

Unit 7 | Classes

Unit 8 | Object-Oriented Programming

Week 4 | Agenda

Week 2 Assignment Review

Week 3 Assignment Discussion

Pseudocoding & Algorithms - Activity 1

For Loops vs. While Loops - Activity 2

Exiting Loops Early

Comprehensions - Activity 3

Some of the main points include
(1) iterators vs. loop;
(2) the importance of pseudocoding, and
(3) planning ahead for algorithm efficiency.

Helpful Git Term | “git stash”

- If you’ve modified files in a repository but want to keep them without “committing” them, try “git stash” to stash the changes before using “`git pull`” or “`git add`”.
- **`Stash`** states away untracked changes.
- See https://gist.github.com/carnivore/997001
- https://git-scm.com/downloads

Assignment Review | Week 2

Refresher:

1. Tip calculator
2. Gasoline Conversion
3. Average Types
4. Income Tax Calculator
5. Palindrome! Names

Assignment Review | Week 2

- **Grading questions: Please email all instructors so we can discuss!**
- Test your code to avoid crashes!
- Check for incorrect math and formulas! Might need to do them by hand first and see if the code matches.
- Use string formatting to print without gaps and with a logical number of digits. You don't need to separate the formatting by a comma:

```
print("The total including tip would be ${:,.2f}".format(total))   vs:
print("The total including tip would be","${:,.2f}".format(total))
```

Assignment Review | Week 2 (cont)

- Your user experience matters - in coding and in data analysis!
- Consider number based inputs vs. text based. Sometimes it makes sense to number the options and ask the user for an integer.
- `.lower()` can be useful on input to ensure a standard string format. Always control user input where you can!
- You can use a while loop for data validation!
- You can "update" a variable each pass through a loop to build strings, lists, etc. over time. Even though strings aren't mutable! You simply are creating a new string with the same variable name each pass through the loop

Week 3 | Polls

Discuss: What was the hardest part of HW3?

Poll: How long did you spend on this week's assignment?

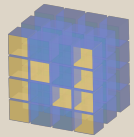
Look Forward - Arrays | NumPy

We will come back to the topic of *arrays* when we discuss NumPy.

Other programming languages use "*arrays*" and "*matrices*" frequently. In Python, a lot can be done with *lists* instead.

When working with large vectors or matrices of numeric calculations, NumPy arrays are much, much faster.

NumPy also has some useful data generation functions, such as those in the "random" library, that can be leveraged in code. There is also the regular "random" library. https://docs.python.org/3.6/library/random.html

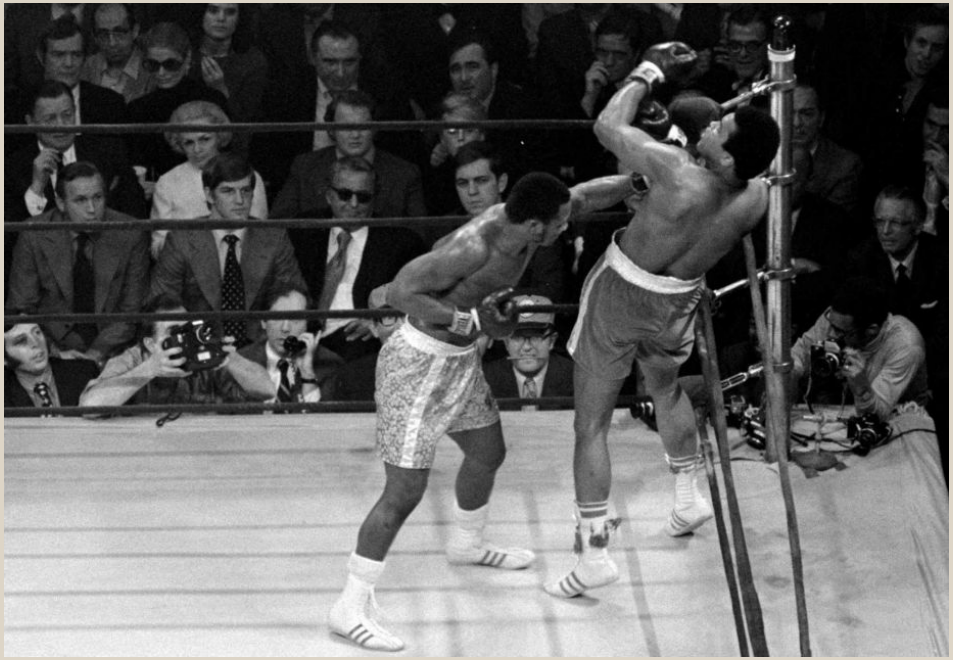


NumPy

Soft skills; stamina = energy and focus

Coding can be strenuous and tiring and confusing

- Breaks
- Polmodoros (25on:5off)
- Time tracking
- Psuedocoding



- While coding - take a break! Polmodors (try 25 min on, 5 min off). Keep track of your time while working.
- Pseudocoding (aka "Structured English") to clarify your goals; the logic of the program … *modularize the code's activities* … very important.
- Very easy to get lost in complicated tasks - use the resources *you can* understand.
- Control the scope of your vars.
- Less, simpler, clarity … but balance…

https://en.wikipedia.org/wiki/Pseudocode
https://en.wikipedia.org/wiki/Pomodoro_Technique

Soft skills - More answers than questions

Don't get lost in the forest

Use resources that **you can** understand.

Identify your specific question.

Control scope of your vars.

Eliminate more than you keep

Pseudocoding & Algorithms | Pseudocoding

What is Pseudocoding?

Who used it in Homework 3?

Pseudocoding & Algorithms | Pseudocoding

What is *Pseudocoding*?

Pseudocode is an informal high-level description of the operating principle of a computer program or other algorithm. It uses the structural conventions of a normal programming language, but is intended for human reading rather than machine reading (Wikipedia).

Planning leads to smart coding!

Modularization.

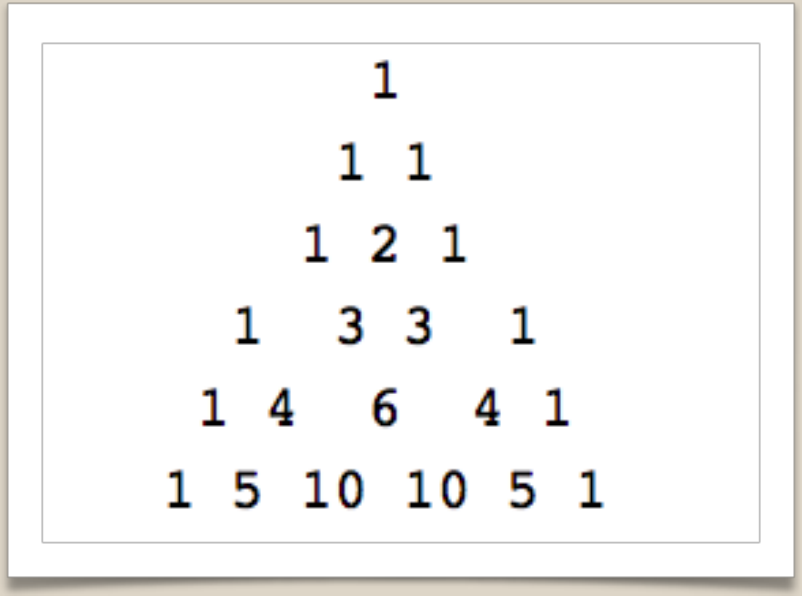
Saves time, and in the real world, saves money.

https://en.wikipedia.org/wiki/Software_design

Pseudocoding & Algorithms | Pseudocoding

Pause – Draw it out

- These statements should be distilled to **simplicity**
- There are 1s flanking each row
- Adding each possible pair of previous row makes next row without the flanking 1s



Pseudocoding & Algorithms | Pseudocoding

We will use two arrays, one for the "last row" one for "current row"
Insight: Each pass through the loop, we can set "current row" to "last row". Then we can build a new "current row".

1. *Initialize a* list *with the number 1.*
2. *Then, make a new list in each iteration.*
3. *In that list, put in the first element of the old list.*
4. *Then, go through every 2 elements in the list (firstelement + secondele, secondele + thirdele) till you are at the last element, and append every addition to the newList.*
5. *Finally, append the last element of the oldList to the new list.*
6. *Print out the list sometime*

We will use two arrays, one for the "last row" one for "current row"
Insight: Each pass through the loop, we can set "current row" to "last row". Then we can build a new "current row".

```
# pascal.py
list = [1]
for i in range(10):
    print(list)
    newList = []
    newList.append(list[0])
    for i in range(len(list)-1):
        newList.append(list[i]+list[i+1])
    newList.append(list[-1])
    list = newList
```

```
>>> for i in range(10):
...     print(list)
...     newList = []
...     newList.append(list[0])
...     for i in range(len(list)-1):
...         newList.append(list[i]+list[i+1])
...     newList.append(list[-1])
...     list = newList
...
[1]
[1, 1]
[1, 2, 1]
[1, 3, 3, 1]
[1, 4, 6, 4, 1]
[1, 5, 10, 10, 5, 1]
[1, 6, 15, 20, 15, 6, 1]
[1, 7, 21, 35, 35, 21, 7, 1]
[1, 8, 28, 56, 70, 56, 28, 8, 1]
[1, 9, 36, 84, 126, 126, 84, 36, 9, 1]
>>>
```

Pseudocoding & Algorithms | Pseudocoding

We will use two arrays, one for the "last row" one for "current row"
Insight: Each pass through the loop, we can set "current row" to "last row". Then we can build a new "current row".

We first initialize the two arrays to [1] and [1,1] for rows 1 and 2
We then loop through an algorithm as follows:

Pseudocoding & Algorithms | Pseudocoding

We will use two arrays, one for the "last row" one for "current row"
Insight: Each pass through the loop, we can set "current row" to "last row". Then we can build a new "current row".

We first initialize the two arrays to [1] and [1,1] for rows 1 and 2
We then loop through an algorithm as follows:
1. Replace "last_row" with the contents of "current_row" so we can create a new "current_row".

Pseudocoding & Algorithms | Pseudocoding

We will use two arrays, one for the "last row" one for "current row"
Insight: Each pass through the loop, we can set "current row" to "last row". Then we can build a new "current row".

We first initialize the two arrays to [1] and [1,1] for rows 1 and 2
We then loop through an algorithm as follows:
1. Replace "last_row" with the contents of "current_row" so we can create a new "current_row".
2. Clear out "current_row" to have nothing in it.

Pseudocoding & Algorithms | Pseudocoding

We will use two arrays, one for the "last row" one for "current row"
Insight: Each pass through the loop, we can set "current row" to "last row". Then we can build a new "current row".

We first initialize the two arrays to [1] and [1,1] for rows 1 and 2
We then loop through an algorithm as follows:
1. Replace "last_row" with the contents of "current_row" so we can create a new "current_row".
2. Clear out "current_row" to have nothing in it.
3. The 0 digit in current_row is 1

Pseudocoding & Algorithms | Pseudocoding

We will use two arrays, one for the "last row" one for "current row"
Insight: Each pass through the loop, we can set "current row" to "last row". Then we can build a new "current row".

We first initialize the two arrays to [1] and [1,1] for rows 1 and 2
We then loop through an algorithm as follows:
1. Replace "last_row" with the contents of "current_row" so we can create a new "current_row".
2. Clear out "current_row" to have nothing in it.
3. The 0 digit in current_row is 1
4. Start a "While" loop that iterates until len(current_row) == len(last_row)

Pseudocoding & Algorithms | Pseudocoding

We will use two arrays, one for the "last row" one for "current row"
Insight: Each pass through the loop, we can set "current row" to "last row". Then we can build a new "current row".

We first initialize the two arrays to [1] and [1,1] for rows 1 and 2
We then loop through an algorithm as follows:
1. Replace "last_row" with the contents of "current_row" so we can create a new "current_row".
2. Clear out "current_row" to have nothing in it.
3. The 0 digit in current_row is 1
4. Start a "While" loop that iterates until len(current_row) == len(last_row)
  ## 5. Use a counter (n) that starts at 1 and increments by 1 each time through the loop

Pseudocoding & Algorithms | Pseudocoding

We will use two arrays, one for the "last row" one for "current row"
Insight: Each pass through the loop, we can set "current row" to "last row". Then we can build a new "current row".

We first initialize the two arrays to [1] and [1,1] for rows 1 and 2
We then loop through an algorithm as follows:
1. Replace "last_row" with the contents of "current_row" so we can create a new "current_row".
2. Clear out "current_row" to have nothing in it.
3. The 0 digit in current_row is 1
4. Start a "While" loop that iterates until len(current_row) == len(last_row)
  ## 5. Use a counter (n) that starts at 1 and increments by 1 each time through the loop
  ## 6. Use "append" to set current_row[n] = last_row[n] + last_row[n-1]

Pseudocoding & Algorithms | Pseudocoding

We will use two arrays, one for the "last row" one for "current row"
Insight: Each pass through the loop, we can set "current row" to "last row". Then we can build a new "current row".

We first initialize the two arrays to [1] and [1,1] for rows 1 and 2
We then loop through an algorithm as follows:
1. Replace "last_row" with the contents of "current_row" so we can create a new "current_row".
2. Clear out "current_row" to have nothing in it.
3. The 0 digit in current_row is 1
4. Start a "While" loop that iterates until len(current_row) == len(last_row)
  ## 5. Use a counter (n) that starts at 1 and increments by 1 each time through the loop
  ## 6. Use "append" to set current_row[n] = last_row[n] + last_row[n-1]
7. After exiting the loop, add the final "1"

Does this fully solve the problem?

Pseudocoding & Algorithms | Pseudocoding

We will use two arrays, one for the "last row" one for "current row"
Insight: Each pass through the loop, we can set "current row" to "last row". Then we can build a new "current row".

We first initialize the two arrays to [1] and [1,1] for rows 1 and 2
We then loop through an algorithm as follows:
1. Replace "last_row" with the contents of "current_row" so we can create a new "current_row".
2. Clear out "current_row" to have nothing in it.
3. The 0 digit in current_row is 1
4. Start a "While" loop that iterates until len(current_row) == len(last_row)
  ## 5. Use a counter (n) that starts at 1 and increments by 1 each time through the loop
  ## 6. Use "append" to set current_row[n] = last_row[n] + last_row[n-1]
7. After exiting the loop, add the final "1"

Now wait, what is the problem? This only builds one row. Go into breakouts and, starting with
  ## this pseudocode, update the pseudocode to find the "N"th line, as required in the problem.

Pseudocoding & Algorithms | Activity 1

Activity 1 - Finish the Pseudocode then implement it!

Pseudocoding & Algorithms | Algorithms

Discuss: What is the underlying purpose of teaching you Search algorithms?

Pseudocoding & Algorithms | Algorithms

Discuss: What is the underlying purpose of teaching you Search algorithms?

Answer: To show that creative code solutions can drastically improve runtimes.

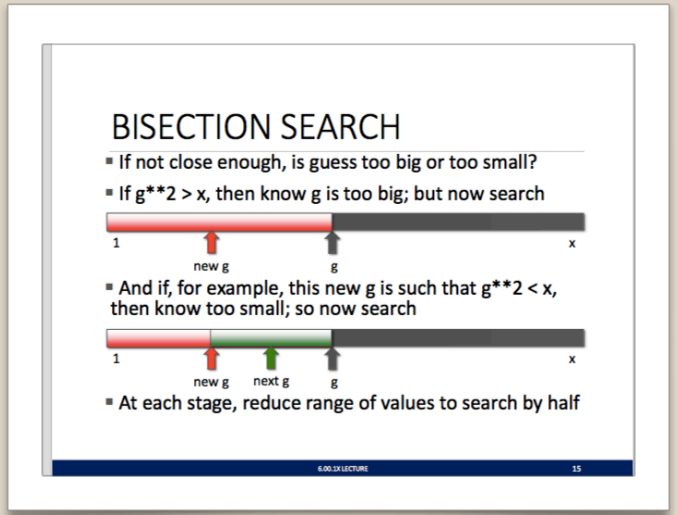
As you take courses on data at scale, it's not enough for your code to work. It has to work, efficiently. Planning and pseudocoding help!

Unit 6 will explore this further.

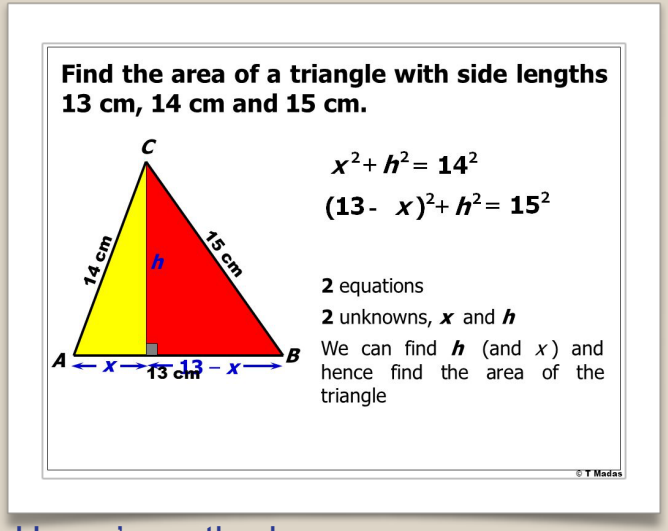
Pseudocoding & Algorithms | Algorithms

Discuss: What is the difference between the three methods?

- Brute force
- Bisection search
- Heron's method



Bisection search



Heron's method

Looping | For vs. While

Question: What is the difference between a “for” loop and a “while” loop?

Looping | For *vs.* While

Question: What is the difference between a "for" loop and a "while" loop?

Answer: A "while" loop iterates until a condition is met.
The programmer may not know in advance how many iterations are needed.

A "for" loop iterates over a predefined group of objects, acting once on each one.

Looping | Examples

```
countdown = 5
while countdown > 0:
    print(countdown)
    countdown -= 1
print("Blast off!")
```

```
5
4
3
2
1
Blast off!
```

```
x = ["Paul", "Bill", "Kay"]
for name in x:
    print(name + ", as himself")
```

```
Paul, as himself
Bill, as himself
Kay, as himself
```

Looping | Code Design

Discuss: How should you select what type of loop to use?

Is one more "general" than the other?

Looping | Code Design

Discuss: How should you select what type of loop to use?

Is one more "general" than the other?

Answer: Generally, use a "for" loop when you have a known set of items to iterate over. Use a "while" loop when you are not sure how many iterations are needed in advance.

All "for" loops can be re-written as "while" loops. In fact, in the first weeks of this course we had you practice implementing several items as "while" loops that are easier to implement as "for" loops.

Looping | Breakout Activity 2

Exiting Loops | Purpose

Question: Why might we want to exit a loop early?

Exiting Loops | Three Tools

Question: Why might we want to exit a loop early?

1. Why "break" a loop?
2. Why use "continue"?
3. Why use "else" after a loop?

Exiting Loops | Explanations

Question: Why might we want to exit a loop early?

Break exits the entire loop early, preventing our code for running needlessly.

Continue skips the rest of the iteration, and starts the next part of the loop. There is no need to continue processing a row.

Else can be used *after* a loop completes, and triggers if the loop never "breaks".

Exiting Loops | Break

Question: When would we use "break"?

In the prime checker code, we do not need to check any values after we find the first non-prime value.

- * What is a **boolean flag**

```
x = int(input("Enter a number:"))
prime = True
for i in range(2,x):
    print("checking potential factor:",i)
    if x % i == 0:
        prime = False
        break
if prime == True:
    print(x,"is prime.")
else:
    print(x,"is not prime.")
```

```
Enter a number:25
checking potential factor: 2
checking potential factor: 3
checking potential factor: 4
checking potential factor: 5
25 is not prime.
```

Exiting Loops | Continue

Question: When would we use "`continue`"?

Use of "continue" early in a loop can save you processing time on many iterations, or when only some values need to be processed (e.g., vowels).

```
word = input("Enter a word: ").lower()
last = "a"

for letter in word:
    if letter not in "aeiou":
        continue
    if letter < last:
        print("The vowels in", word, "are not ordered.")
        break
    last = letter
```

See whether the vowels in the input are ordered

Exiting Loops | Break/Else

Question: When would we use “else”?

“Else” can be used to save coding effort by processing the data in a certain way if your entire loop runs.

```
word = input("Enter a word: ").lower()
last = "a"

for letter in word:
    if letter not in "aeiou":
        continue
    if letter < last:
        print("The vowels in", word, "are not ordered.")
        break
    last = letter
else:
    print("The vowels in", word, "are ordered.")
```

```
Enter a word: Pual
The vowels in pual are not ordered.
```

Comprehensions | Purpose

Question: What is a “list comprehension”?

Question: Why would we want to use one?

Comprehensions | Purpose

Question: What is a "list comprehension"?

Answer: A comprehension is a single-line "for loop" that enables you to quickly build a list. You can also build sets or dictionaries.

Question: Why would we want to use one?

Answer: They are fast, efficient, and fun ways to build sequences out of other sequences!

List Comprehensions | a compact for loop

List comprehension implicitly appends items resulting from the "Expression"

Structure :[Expression(item) *for* **Item** *in* **Iterable]**

For loop:

```
squares = []
for i in range(1,11):
    squares.append(i**2)
print(squares)
```

```
[1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
```

List comp:

```
[i**2 for i in range(1,11)]
```

```
[1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
```

List Comprehensions | loops with conditionals

List comprehension is run on items also in the second iterable

Structure :[Expression(item) *for* **Item** *in* **Iterable** *if* **Item** *in* **container]**

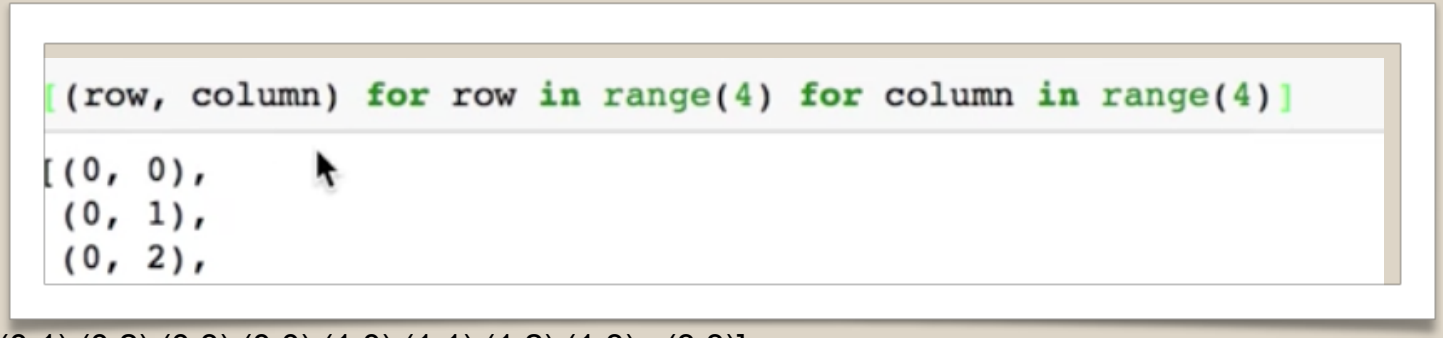
```
import string

word = '2345-Nessie ate-YOU_TOO!!@#'.lower()

[i + ' vowel' for i in 'aeiou' if i in word]
```

```
['a vowel', 'e vowel', 'i vowel', 'o vowel', 'u vowel']
```

List Comprehensions | loops + loops

List comprehension produces tuples resulting from NESTED loops

Structure :[(Expr1(item1), Expr2(item2)) *for* Item1 *in* Iter1 *for* Item2 *in* Iter2]



[(0,0),(0,1),(0,2),(0,3),(0,0),(1,0),(1,1),(1,2),(1,3)...(3,3)]

Try this on pythontutor.org http://pythontutor.com/visualize.html#togetherjs=6mOAavcRgK

Other Comprehensions | Key:value pairs

Curly {} braces and key value pairs define this one

Structure :{Item : Expression(item) ***for*** **Item** ***in*** **Iterable}**

```
text = "Here is some example text."
frequencies = {letter : text.count(letter) for letter in text}
print(frequencies)
```

```
{'t': 2, 'o': 1, 'e': 6, 'l': 1, 'H': 1, 'i': 1, 'r': 1, 'x': 2, '.': 1, 'm': 2, 'p': 1, ' ': 4
, 'a': 1, 's': 2}
```

Comprehensions | Activity 3

What is JSON data?

Similar to a dictionary.

Fun Fact: Your entire Jupyter Notebook is actually saved as JSON. You will see this if you ever need to resolve a Jupyter "merge conflict".