Half-Intro To Python

What's A Programming Language?

- Who's written a program before?
- Series of instructions; computers are stupid. The don't know what you mean, you have to tell them exactly.

Small Bits To Get Out Of The Way

- Python is an interpreted language. If you don't know what tha means, don't worry about it. If you've written in C or Java before, you know that your program needs to be compiled, which means the program is turned into machine code. Python skips that step, which makes it easier to work with in many ways, but also makes it slower than a compiled language like C. Until you start working on high-performance code or operating systems, you don't need to worry about it.
- Two branches of Python: Python 2 and 3. We'll be working with Python 2.7 today. The future of Python is Python 3, but 2.7 is more common right now. The differences are very minor and not worth worrying about. But if you come across Python 3 while you're Googling, that's what that is about.

Basics for getting started.

In the interpreter.

- print statements
    - `print "Hello World"`
    - `print "Want to head a Yoda joke?"`
    - `print "Why was 5 afraid of 6?"`
    - `print "Because 6 7 8."`

At some level, a computer is a really fancy calculator, and a programming language is a way of talking to the calculator.

- Arithmetic
    - Add `7 + 3`
    - Subtract `7 - 3`
    - Multiply `7 * 3`
    - Exponentiate `7 ** 3`

- Divide `7 / 3`
    - WTF?
    - Try: `7/3.0`
    - What is going on??
    - Try: `8/3`, `9/3`
    - Types
        - everything is binary in computer memory
        - how does the computer interpret those numbers
        - ints vs floats vs strings vs Boolean
- Remainder (or mod): `7 % 3`
- `print '7' + '3'`
- Discussion about types, int vs float vs string
    - Point out you can cast string to an int (this will be important for summing numbers in a text file)
- Square root
    - What's a function?
    - `sqrt(9)` doesn't work
    - `import math`
    - `math.sqrt(9)` does
    - we "imported" the "math" "module", we'll get back to that later.

Variables, lists and for loops

Suppose we want to know the square root of the first 10 integers. (Just pretend you care.)

We could do this:

```
print math.sqrt(1)
print math.sqrt(2)
print math.sqrt(3)
... etc
```

That gets old fast. Especially if we want the square root of the first 100 numbers. Since we're doing the exact same thing over and over again, we can automate it with a "for loop". This is what programming is all about.

```
for i in [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]:
    print math.sqrt(i)
```

For anything but a couple commands, using the command interface (interpreter) is a pain.

- Easier to write programs in a text editor.
- Explain that Word is not a text editor.
- Mention `vi`, `emacs`, we'll use `nano`

Build a program that sums the first 10 integers.

- Start with list
- Loop over list, realize we need a sum variable
- Find out we need to initialize sum = 0
- Realize we need to print sum when finished
- Point out importance of indentation

```
numbers_to_sum = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
sum = 0
for number in numbers_to_sum:
    sum = sum + number
print sum
```

- Point out `sum += number`
- Point out syntax of for loop, and of indentation, and why indentation is so important in Python.

Lists are boss

Lists are super-useful. You can keep any sequence in a list. Numbers or strings, or anything.

```
walrus_parts = ['tusk', 'flipper', 'vibrissae']
for part in walrus_part:
    print 'walrus', part
```

You can access individual parts of a list:

- `print walrus_parts[0]`
- `print "mermaid ", walrus_parts[1]`

Draw a diagram explaining zero-based indexing for lists.

You can append to lists:

- `walrus_parts.append('blubber')`

You can "slice" lists:

- `print walrus_parts[1:3]`

Strings are kind of similar to lists

- A string is a sequence of characters. "String" is a type, like "int" or "float"
- Denoted by single- or double-quotes.
- They can be cut like lists.
    - `dna_sequence = 'AAGTACTACTAAGACTALIENSWEREHERELOLGTACTACTACT'`
    - `print dna_sequence[16:33]`
- To the interpreter
- `walrus_poem = 'Walrus flipper, walrus tusk, walrus mustache, walrus musk`
- `dismember_walrus = walrus_poem.split()`
- `print dismember_walrus[7]`
- You can "add" strings to concatenate them together
    - `print "elon " + dismember_walrus[7]`
    - To treat a string as a number, cast to `int` or `float`
- You can "join" a list into a string:
    - `'\t'.join(dismember_walrus)`

Dictionaries are great too

- Kind of like lists, except the indeces don't have to be integers. And they aren't ordered.
- `english_to_german = {'one': 'eins', 'two': 'zwei', 'three': 'drei'}`
- `print english_to_german['one']`
- english_to_german['four'] = 'vier'
- Key-value pairs
- Iterate over keys of dictionary:
    - `for key in english_to_german: print key`
    - (note, not same order)
    - `for key in english_to_german: print english_to_german[key]`

Conditionals

Explain `if-else`.

```
sequence = 'ACGU'
if 'U' in sequence:
    print "It's not DNA!"
else:
    print 'It *might* be DNA...'
```

Show ET?

File IO And Summing Walrus Vocalizations

Start showing how to create the counts, and use Unix `paste` to paste them together?

```
infile = open('walrus_sound_table.tsv', 'r')

for line in infile:
    split_line = line.split()
    total_count = 0
    for number in split_line[1:]:
        total_count += int(number)
    print split_line[0] + '\t', total_count

infile.close()
```

- Show `open`, `close`, type them out together
- Start with just input, printing each line
    - Point out that the file handle is an *iterable*, like a list is (but it's not a list)
    - Print line.split()
- Print count stupidly: with indeces 1, 2, 3
    - Make the split line a variable first
- This won't work if we add another column, and typing out all those indeces is a little weak
    - Loop over split_line[1:]
    - New variable `total_count`
    - Need to initialize `total_count`
- Print vocalization together with `total_count`

Didn't mention and should have

- More data types: Booleans, tuples, sets, many more
- Functions! Didn't even mention functions!
- Classes: whole 'nother world
- More time on imports

Other modules and resources

These are like the math module. Import them and get new functionality!

- SciPy

- matplotlib
- IPython
- pandas
- scikit-learn