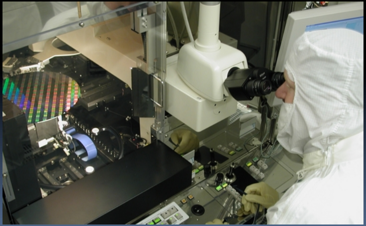
UC Berkeley EECS
Teaching Professor
Dan Garcia

The Beauty and Joy of Computing

Lecture #20
Besides Blocks II:
Python Data Structures & APIs

Moore's Law continuing for 10+ years

Approaching the 50th anniversary of Moore's Law declaration, engineers at Intel declare that don't see Moore's law slowing for another 10 years! They are working on 7nm and 5nm processes (the width between wires). One thing that will help is 3D stacking of components; heat is ever an issue!



www.v3.co.uk/v3-uk/news/2403113/intel-predicts-moores-law-to-last-another-10-years

(Cal) Admin Notes

- Final Projects start this week!
    - Use Snap! or Python
    - Individual
    - 3 weeks to work
    - Milestone at 7.5 hours with submission
- Schedule (see website)

(Cal) Lecture Overview

- Data Structures
  - Sequences
    - String
    - List
    - Tuple
    - Range
  - Dictionaries
- Higher-Order Functions
- APIs

Python Objects & Sequences

Python … everything is an object!

- Just like OOP from Programming paradigms…
- Importing a class/module that isn't built-in:
  - import <module>
  - E.g., `import math`
- Getting help
  - `help(<type>)` or `help(<value>)` or `help(<module>)`
  - E.g., `help(int)` or `help(1)` or `help(math)`
- Calling
  - <module/object>.<function>(<args>, …) or <module>.<constant> or <object>.<field>
  - E.g., `"12".isdigit()` or `math.pi` or `(1+2j).real`

Python Sequences

- Contain an ORDERED set of data
  - `str` – "text in quotes" (Snap*!* sentence)
  - `list` – ['a', 'group' , 'of', 'items']
  - `tuple` – ('a', 'group', 'of', 'items')
    - a list that can't be modified
  - range(start, stop, step) – sequence of #s
- Supports very easy iteration:

```
for item in sequence:
    print(item)
```

Python Sequence (general) Operations

- elem `in` & `not in` sequence
- `+` & `*`
- my_sequence`[START:END:STEP]`
- `len()`
- `min()` & `max()`
- Even `map() filter()` & `reduce()`!
- `count(item)`
- Many, many more: http://docs.python.org/library/stdtypes.html#typesseq

Python Strings, Lists, Tuples & Ranges

Python Strings

- Sequence (or "list" or "array") of chars
- Quoting
  - Single Quotes, Double Quotes
  - Triple Quotes (this keeps formatting and line breaks)
- Concentration, finding length, etc.
  - `help(str)` and `help("string")`
- http://docs.python.org/library/stdtypes.html#string-methods

Python Lists

- Collection of any type
  - Including itself!
- Indexing (`mylist[item]`)
  - Indexed from 0, **NOT** 1, unlike *Snap!*
- Modifying (`my_list[item] = new_item`)
- Slicing and slicing notation (i.e. [::])
  - Exactly the same as string notation!
- Operators
  - `append(x)`, `insert(i,x)`, `count(x)`, `sort()`, etc.
- http://docs.python.org/library/stdtypes.html#mutable-sequence-types

Python Tuples & Ranges

- Tuples mostly like Lists except `( )` not `[ ]`
  - Except they can't be changed (like strings)
  - This immutability will be helpful in dictionaries
- Ranges are virtual sequences of #s
  - Useful and fast
    - They don't actually exist until you need them
    - Use `list(range(<args>))` to see it
  - Would be nice to have this available in Snap*!*

Demo! ☑

(Cal) Clicker Question

What is **people** after clicking the "**replace**" command?

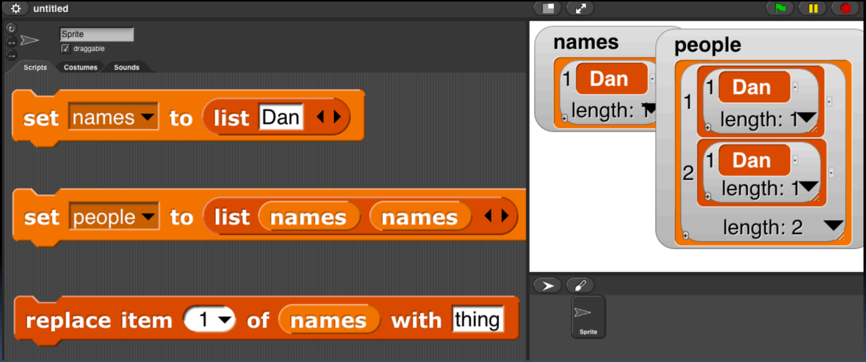
a) ((Dan) (Dan))

b) ((Dan) (thing))

c) ((thing) (Dan))

d) ((thing) (thing))

e) Error



```
set names to list Dan
set people to list names names
replace item 1 of names with thing
```

names: 1 Dan, length: 1

people: 1 (1 Dan, length: 1), 2 (1 Dan, length: 1), length: 2

(Cal) Clicker Question

What is **`people`** after clicking the "**`replace`**" command?

```
>>> names = ["Dan"]
>>> people = [names, names]
>>> people
[['Dan'], ['Dan']]
>>> names[0] = "thing"
```

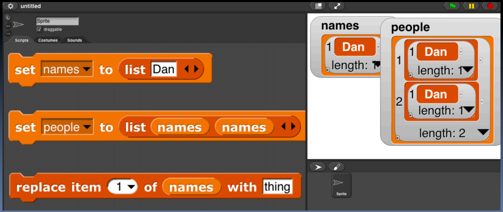
a) [["Dan"], ["Dan"]]

b) [["Dan"], ["thing"]]

c) [["thing"], ["Dan"]]

d) [["thing"], ["thing"]]

e) Error



```
set names to list Dan
set people to list names names
replace item 1 of names with thing
```

```
names
1 Dan
length: 1

people
1  1 Dan
     length: 1
2  1 Dan
     length: 1
length: 2
```

Garcia

Python Dictionaries

Python Dictionaries (`dict`)

- Very fast access (by **key**, not number)
- "Map" from a key to a value
- Syntax
  - `{ key1 : value1, key2 : value2, … }`
- Adding elements
  - `dict[key] = value`
- Accessing elements (just like sequences):
  - `dict[key]`
- Keys
  - Looking for specific keys ("`in`")
  - Iterating over (`iterkeys()`)

Demo! ☑

Garcia

(Cal) Clicker Question

We saw a series of "is a particular student here" problems when the students were unsorted, sorted, & given-months-in-advance. What running time did a dictionary-like structure help us attain?

a) Exponential

b) Quadratic

c) Linear

d) Logarithmic

e) Constant

Python
HOFs & APIs

Python APIs

- "Application Programming Interface"
  - Set of agreements for sharing information
- Programming APIs (i.e., how to use modules)
  - E.g., Building Blocks for common elements such as Open or Save prompts
- Web APIs
  - "Special" URLs for accessing data directly
- Example: Jeopardy API
  - `http://jservice.io/api/random`
- Example: Solved Game APIs (e.g., Tic Tac Toe)
  - `getNextMoveValues`

Demo (reference)

- Code files are all on the website
- `hof.py`
  - Some Higher Order Functions in Python we've seen in Snap*!*
- `fractals.py`
  - Some fractals in Turtle Graphics
- `jeopardyAPI.py`
  - Standalone text-based Jeopardy game
- `tttAPI.py`
  - Tic-Tac-Toe in Python
  - Games Crafters API for information about best moves

More Information

- Online Python Tutor (invaluable!!)
  - http://www.pythontutor.com/
- Sequences & Methods
  - http://docs.python.org/library/stdtypes.html
- Coding Bat (***Great*** practice!)
  - http://codingbat.com/python
- Google's Python Class
  - http://code.google.com/edu/languages/google-python-class/
- Exercises (More practice!)
  - http://code.google.com/edu/languages/google-python-class/exercises/basic.html

Garcia