Jonathan McKinsey

The Beauty and Joy of Computing

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

(GUI) Text Editors vs IDEs

Python Objects & Sequences

Object-Oriented Programming in Python

- A class defines the blueprint of an object and can contain
  - Properties = values
  - Functions = actions
  - E.g. A Rabbit might have
    properties: name, age
    functions: eat, sleep
- An instance of an object is what you get after you build a blueprint.
  - E.g Petey is a Rabbit that is 8 months old (8/12 = 0.67 years).
    >>> petey = Rabbit("Petey", 0.67)
- To access a property or call a function, use . ← period
  - E.g. >>> petey.eat("hay")
    >>> Petey eats a lot of hay! Nom nom nom…

Test your understanding

```
class Rabbit:
    def __init__(self, name, age):
        self.name = name
        self.age = age

    def eat(self, food):
        print(self.name + " eats a lot of " + food + "! Nom nom nom...")

    def sleep(self):
        print(self.name + " sleeps on your shoes.")
```

What happens if we call: `petey.sleep()`?

a) self.name + " sleeps on yours shoes."
b) Petey sleeps on yours shoes.
c) Petey eats a lot of hay! Nom nom nom…
d) 0.67

Importing Modules and Getting Help

- 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)`
- Treating everything as an object
  - <module/object>.<function>(<args>, ...) or <module>.<constant> or <object>.<field>
  - E.g. `"12".isdigit()` or `math.pi` or `(1+2j).real`

Python Sequences

- `str` "text in quotes"
- `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

- `in` & `not in`
- `+` & `*`
- `SEQUENCE[START:END:STEP]`
- `len()`
- `min()` & `max()`
- `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
- Indexing `mylist[item]`
  - Indices start at 0, **NOT** 1
- 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

Brief Tangent on Variables

Clicker Question

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

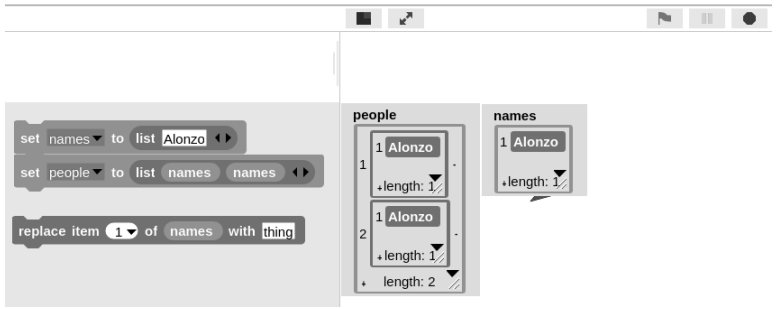
a) ((Alonzo) (Alonzo))

b) ((Alonzo) (thing))

c) ((thing) (Alonzo))

d) ((thing) (thing))

e) Error



Clicker Question

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

a) [["Alonzo"], ["Alonzo"]]

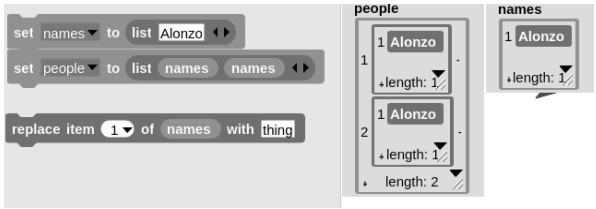
b) [["Alonzo"], ["thing"]]

c) [["thing"], ["Alonzo"]]

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

e) Error

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



Python Dictionaries

Python Dictionaries (`dict`)

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

Python 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: Missing Persons API
  - `find-us.herokuapp.com`

Demo (reference)

- Code files are all on the class website
- `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