Beyond Blocks: Python

Session #2

Data Structures

Data Structures (overview)

- Review
- Sequences
  - Iterators
  - Operators
- Sets
  - Operators
- Dictionaries
  - Hash Tables

Data Structures (overview)

- Review & some new introductions
- Sequences
  - Iterators
  - Operators
- Sets
  - Operators
- Dictionaries
  - Hash Tables

Review

- Typing, built-in types
- Variables
- Looping and conditionals
- Functions
- Recursion

Review++

- Typing, built-in types
- Variables
- Looping and conditionals
- Functions
- Recursion
- This week's content
  - Strings and string operators
  - Lists

Typing, (Some) Built-In

- Weak-typing vs. Strong-typing
  - aka - Dynamic vs. Static-typing
- Numeric types
  - <int>, <float>, <long>

Typing, (Some) Built-In

- Weak-typing vs. Strong-typing
  - aka - Dynamic vs. Static-typing
- Numeric types
  - <int>, <float>, <long>, <complex>
- Sequence types
  - <str>, <unicode>, <list>, <tuple>, <buffer>, <range>
- New: Collection types
  - <set>, <frozenset>, <dict>

Variables

- Simple assignments
- Weakly/Dynamically typed
  - type() function

Variables

- Simple assignments
- Multiple assignments
- Weakly/Dynamically typed
  - type() function
- “Mutable” vs. “Immutable”
  - We’ll see more of these as examples

Looping and Conditionals

- While loops
- If statements with boolean comparisons
  - Parenthetical evaluation
  - or, and, not, <, <=, >, >=, ==, =, is, is not

Looping and Conditionals

- While loops
- If statements with boolean comparisons
    - Parenthetical evaluation
    - or, and, not, <, <=, >, >=, ==, =, is, is not
- For loops (e.g. "for x in range(0,10):")
    - We'll talk more about ranges later...

Functions

- How to define them
- Variable scope

- Returning values

Functions

- How to define them
- Variable scope
  - Global scope & keyword
- Returning values
  - Returning multiple values

Recursion

- How to write a recursive function
- Factorial(n)?
- IsPalindrome(word)?
  - IsPalindrome is left as an exercise for the reader...

Sequences (overview)

- Str “”
  - Raw & Unicode (256 vs. 65535 chars.)
- List []
- Tuple ()
- Buffer
- Range
  - Range vs. xRange
    - We’ll talk more about xRange later...
- http://docs.python.org/library/stdtypes.html#typesseq

Sequences (overview)

- Str “” - immutable
  - Raw & Unicode (256 vs. 65535 chars.)
- List [] - mutable
- Tuple () - immutable
- Buffer
- Range - mutable-ish
  - Range vs. xRange
    - We’ll talk more about xRange later...
- http://docs.python.org/library/stdtypes.html#typesseq

Strings and String Operators

- Sequence (or “list” or “array”) of chars
- Quoting
  - Single vs. double vs. triple and mixing
- Printing
  - Formatted and unformatted
- Concatenation, finding length, etc.
  - help(“string”)
- Slicing and slicing notation [::]
- http://docs.python.org/library/stdtypes.html#string-methods

Lists

- Collection of *any* type
  - Including itself!
- Indexing (BYOB: Item () of [])
- Modifying (Replace item () of [] with ())
- 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

Tuples (|ˈtjuːp(ə)l| :)

- Immutable
  - Same as strings
- Also contains *any* type of element(s).
- Syntax ()
- What are the advantages of using them?
  - Faster and “Safer,”
  - Can be used as Dictionary keys
    - More on dictionaries later...

Buffers

"Buffer objects are not directly supported by Python syntax, but can be created by calling the built-in function buffer(). They don't support concatenation or repetition."

- http://docs.python.org/library/stdtypes.html#sequence-types-str-unicode-list-tuple-bytearray-buffer-xrange

Ranges (and xRanges)

- Range syntax (start, stop, step)
  - Results in a list []
- xRange
  - “Lazy Evaluation”
  - Results in an xrange instance...
    - More about this later...
- http://docs.python.org/library/stdtypes.html#xrange-type

Iterators

- Syntax
  - i = iter(object)
- Usage
  - i.next()
- Why does Python have them?
  - You’ll see...
- http://docs.python.org/library/stdtypes.html#iterator-types

Sequence (general) Operators

- elem in & not in sequence
- + & *
- slice [::]
- len()
- min() & max()
- even map() & reduce() !
- Many, many more:
  - http://docs.python.org/library/stdtypes.html#typesseq

Sets

- NO duplicate members (unique)
- Unordered
- Syntax: set([1,2,3,4]) or set(“blah”)
- NO array-like indexing (e.g., s[0])
  - Iterators are used instead...

Set Operators

- len(s)
- s.add(elem)
- elem in & not in s
- remove & pop & -
- Iteration
- Union, intersection, isdisjoint, etc.
- Much, much more:
    - help("set")
    - http://docs.python.org/library/stdtypes.html#set

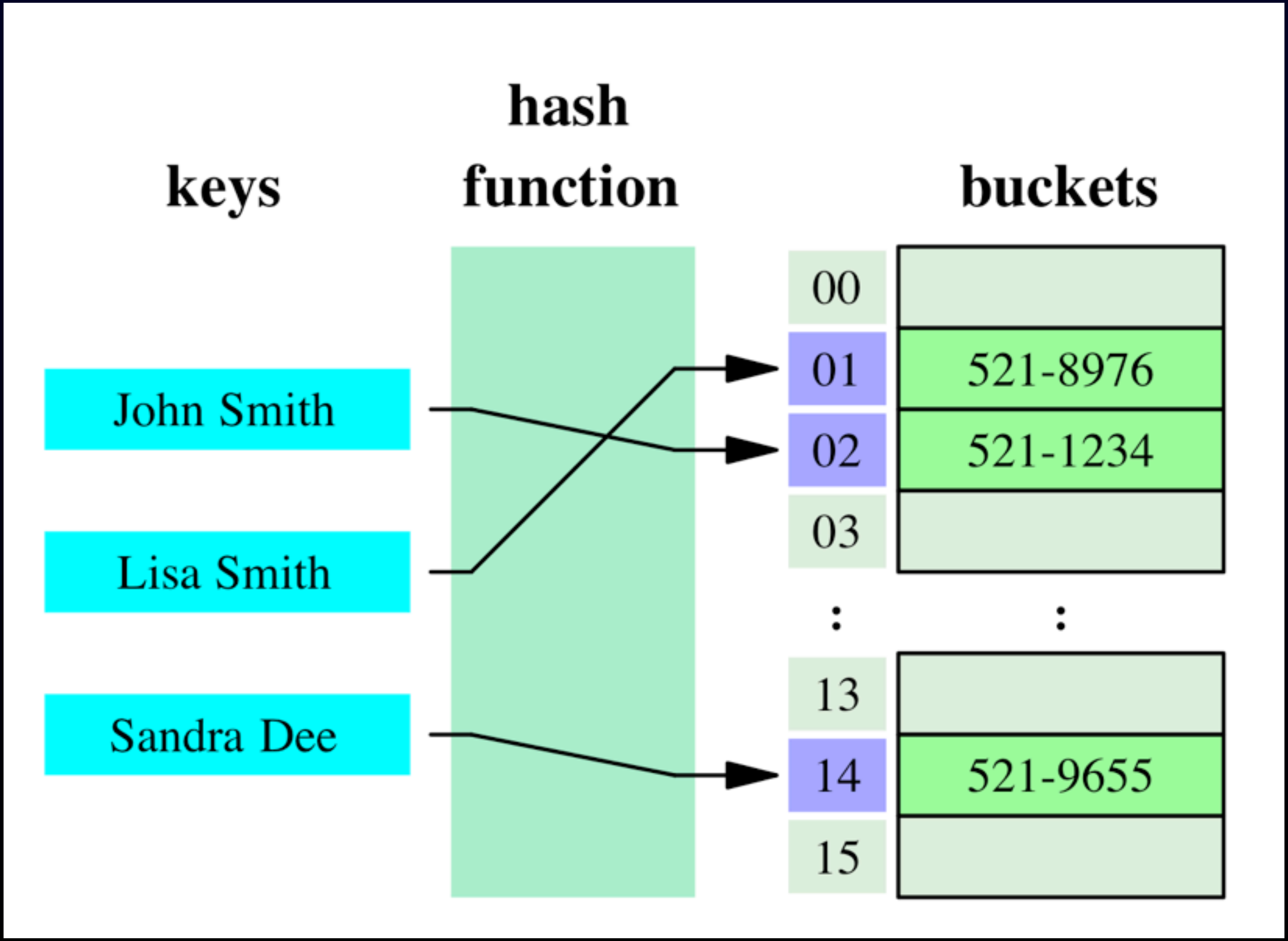
Dictionaries

- Syntax
  - {key:value}
- Adding elements
  - dict[key]=value
- Printing contents
- Accessing elements
  - dict[key]
- Keys
  - Looking for specific keys (has_key() & “in”)
  - Iterating over (iterkeys())
- http://docs.python.org/library/stdtypes.html#dict

How Do Dictionaries Work, and Why Use Them?

- Hash table based
  - Hash codes & array indexes
- Very fast look-up time (i.e., O(C))
- Classic trade-off:
  - Speed and space

Dictionaries = Hash Tables



http://en.wikipedia.org/wiki/File:Hash_table_3_1_1_0_1_0_0_SP.svg

“Class()?”

- http://docs.python.org/library/stdtypes.html
- You may have noticed that these containers are defined as “class()”es...
  - *class* **`dict`**([*arg*])
  - *class* **`set`**([*iterable*])
- ...stay tuned for Session #3:

Class()!

- http://docs.python.org/library/stdtypes.html
- You may have noticed that these containers are defined as "class()"es...
  - *class* **`dict`**([*arg*])
  - *class* **`set`**([*iterable*])
- ...stay tuned for Session #3:
  - Object Oriented Programming!

More Information

- 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