Ranges

- Range syntax `(start, stop, step)`
  - Start: Inclusive; stop: exclusive
  - “Lazy Evaluation”
  - Results in an iterable object
- `list(range(x))` is a list.
  - `range(start, stop)` or `range(stop)` also work.
  - Default start is 0, Default step is 1.
- [http://docs.python.org/library/stdtypes.html#xrange-type](http://docs.python.org/library/stdtypes.html#xrange-type)
Iterators

- Syntax
  - \( i = \text{iter}(\text{object}) \)

- Usage
  - \( \text{next}(i) \) #In Python3!
  - Python 2.x: \( i.\text{next}() \)

- Why does Python have them?
  - You’ll see...

- [http://docs.python.org/library/stdtypes.html#iterator-types](http://docs.python.org/library/stdtypes.html#iterator-types)
Sequence (general) Operators

- `elem in & not in sequence`
- `+ & *`
- `slice [:]`
- `len()`
- `min() & max()`
- `even map() filter() & reduce()`
- Many, many more:
  - [http://docs.python.org/library/stdtypes.html#typesseq](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...
• Faster (for large number of entries)
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](http://docs.python.org/library/stdtypes.html#set)
Dictionaries

• Syntax
  • {key:value}

• Adding elements
  • dict[key]=value

• 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(1) )
• Classic trade-off:
  • Speed and space
Dictionaries = Hash

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