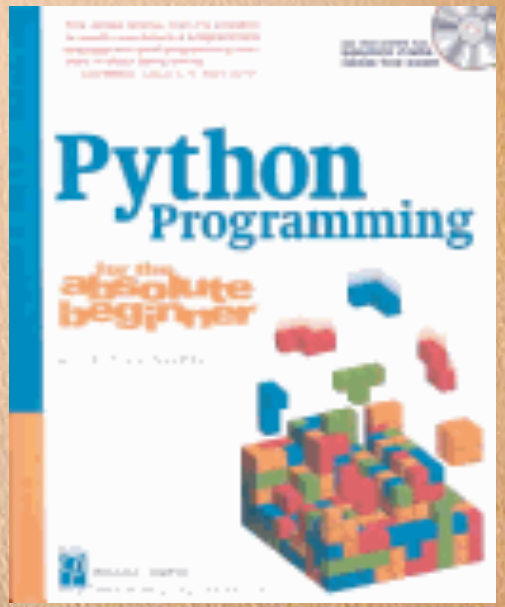
Python Programming: An Introduction to Computer Science

Dictionaries

Dan Fleck



Coming up: Data Structures

Data Structures

- Tuple – immutable, sequential
- List – mutable, sequential
- String – immutable, sequential

- sequential: items can be indexed by a number, they are in order (in a sequence), duplicate values allowed

- Dictionary – mutable, non-sequential!

Dictionary Mapping

- Creates a mapping between keys and values. So I can lookup values by key

Keys	Values
Carl	CS
Mary	Math
Jin	EE
Alice	Pysics

```
# Key is student's name
# Value is their major
studentMajors = {"Carl":"CS","Mary":"Math","Jin":"EE", \
                 "Alice":"Physics"}
```

What is a dictionary?

- Allows you to lookup items based on keys, not based on index.

dictionaryEx1.py – /Users/dfleck/Documents/gmuwebsite/classes/cs112/spring09/samplecode/dictionar...

```
# Dictionary example 1

def main():

    # Create a dictionary of majors

    # Key is student's name
    # Value is their major
    studentMajors = {"Carl":"CS","Mary":"Math","Jin":"EE", \
                     "Alice":"Physics"}

    print studentMajors

    # Find a specific student's major:
    jinMajor = studentMajors["Jin"] # Lookup the key: "Jin"
    print "\nJin's major is: %s " %(jinMajor)



main()
```

Ln: 16 Col: 16

Coming up: Modify

Modify or Add an entry

- Assign the key to a new value (mutable!)

```
def modifyEntry(students):
    # Change the value of "Carl"'s entry
    # to ArtHistory
    students["Carl"] = "ArtHistory"
    print students
```

So, can a dictionary have duplicate keys?

- If the key isn't present… creates a new entry!

```
def addEntry(students):
    # Add a new key
    students["Harold"] = "SWE"
    print students
```

Dictionaries are not sequential

- You cannot say “give me the first entry”… the dictionary has no defined order (or sequence)! Python can do whatever it wants.

```
studentMajors = {"Carl":"CS","Mary":"Math","Jin":"EE", \
                 "Alice":"Physics"}
```

- Print it:

```
=======
>>>
{'Jin': 'EE', 'Mary': 'Math', 'Carl': 'CS', 'Alice': 'Physics'}
```

Remove Entry

- Remove an entry using “del”

```
def removeEntry(students):
    # Remove a key
    del students["Carl"]
    print students
```

- Watch out, error if key isn’t present:

```
Traceback (most recent call last):
  File "/Users/dfleck/Documents/gmuwebsite/classes/cs112/spring09/samp
lecode/dictionaries/dictionaryEx1.py", line 43, in <module>
    main()
  File "/Users/dfleck/Documents/gmuwebsite/classes/cs112/spring09/samp
lecode/dictionaries/dictionaryEx1.py", line 41, in main
    removeEntry(studentMajors)
  File "/Users/dfleck/Documents/gmuwebsite/classes/cs112/spring09/samp
lecode/dictionaries/dictionaryEx1.py", line 16, in removeEntry
    del students["CAAAarl"]
KeyError: 'CAAAarl'
>>>
```

Coming up: How do I tell if a key is present?

How do I tell if a key is present?

Check the Python Library Reference!

Operation	Result
`len(a)`	the number of items in *a*
`a[k]`	the item of *a* with key *k*
`a[k] = v`	set `a[k]` to *v*
`del a[k]`	remove `a[k]` from *a*
`a.clear()`	remove all items from `a`
`a.copy()`	a (shallow) copy of `a`
`k in a`	`True` if *a* has a key *k*, else `False`
`k not in a`	Equivalent to `not k in a`
`a.has_key(k)`	Equivalent to `k in a`, use that form in new code
`a.items()`	a copy of *a*'s list of (*key*, *value*) pairs
`a.keys()`	a copy of *a*'s list of keys
`a.update([b])`	updates *a* with key/value pairs from *b*, overwriting existing keys, returns `None`
`a.fromkeys(seq[, value])`	Creates a new dictionary with keys from *seq* and values set to *value*
`a.values()`	a copy of *a*'s list of values
`a.get(k[, x])`	`a[k]` if *k* `in` *a*, else *x*
`a.setdefault(k[, x])`	`a[k]` if *k* `in` *a*, else *x* (also setting it)
`a.pop(k[, x])`	`a[k]` if *k* `in` *a*, else *x* (and remove k)
`a.popitem()`	remove and return an arbitrary (*key*, *value*) pair
`a.iteritems()`	return an iterator over (*key*, *value*) pairs
`a.iterkeys()`	return an iterator over the mapping's keys
`a.itervalues()`	return an iterator over the mapping's values

Coming up: How do I get the length of the dictionary?

How do I get the length of the dictionary?

Check the Python Library Reference!

Operation	Result
`len(a)`	the number of items in *a*
`a[k]`	the item of *a* with key *k*
`a[k] = v`	set `a[k]` to *v*
`del a[k]`	remove `a[k]` from *a*
`a.clear()`	remove all items from `a`
`a.copy()`	a (shallow) copy of `a`
`k in a`	`True` if *a* has a key *k*, else `False`
`k not in a`	Equivalent to `not` *k* `in` *a*
`a.has_key(k)`	Equivalent to *k* `in` *a*, use that form in new code
`a.items()`	a copy of *a*'s list of (*key*, *value*) pairs
`a.keys()`	a copy of *a*'s list of keys
`a.update([b])`	updates *a* with key/value pairs from *b*, overwriting existing keys, returns `None`
`a.fromkeys(seq[, value])`	Creates a new dictionary with keys from *seq* and values set to *value*
`a.values()`	a copy of *a*'s list of values
`a.get(k[, x])`	`a[k]` if *k* `in` *a*, else *x*
`a.setdefault(k[, x])`	`a[k]` if *k* `in` *a*, else *x* (also setting it)
`a.pop(k[, x])`	`a[k]` if *k* `in` *a*, else *x* (and remove k)
`a.popitem()`	remove and return an arbitrary (*key*, *value*) pair
`a.iteritems()`	return an iterator over (*key*, *value*) pairs
`a.iterkeys()`	return an iterator over the mapping's keys
`a.itervalues()`	return an iterator over the mapping's values

Other options

- Creation
  - aDict = { } # Creates an empty dictionary
  - aDict = {“Jon”:”CS”, “Carl”:”EE”}
    - same as
  - aDict = dict(“Jon”=“CS”, “Carl”=“EE”)
  - x = aDict[“Jon”] same as x = aDict.get(“Jon”)

Valid Keys

- Any immutable type is a valid key: int, float, string, even tuple.
  - aDict = { 12:”square”, 3:”circle”, “HHH”:”unknown”}
- Python uses “==“ to compare keys.
  Is 0xC == 12 ?
  - aDict[0xC] = “sq” # So what does this do?
- Because floating point numbers are not exact, we do NOT recommend using them as keys!

```
>>> 2.3 * 6
13.799999999999999
```

Valid Values

- Any Python type is a valid value
  - aDict = {“a”:(1,2,3), “b”:1.9, “c”:[33,3,3]}

Iteration

- Iterating over the dictionary

```
def iteration(students):
    # Loop over each key in the dictionary
    # Note: the order is undefined!
    for eachKey in students:
        print "Key:%s  Value: %s " %(eachKey, students[eachKey])
    # How we would show the dictionary with keys in
    # alphabetical order?
```

```
Key:Jin  Value: EE
Key:Harold  Value: SWE
Key:Mary  Value: Math
Key:Alice  Value: Physics
>>>
```

Data Structures Summary

Type			Create Using	Access
List	mutable	sequential	[ ]	[index], [start:stop]
String	immutable	sequential	“ “	[index], [start:stop]
Tuple	immutable	sequential	(,)	[index], [start:stop]
Dictionary	mutable	not sequential	{ : , }	[key], get(key)

Terminology / Concepts

- Key / Value Pairs
- Dictionaries
- Mapping

End of presentation