DSC 201: Data Analysis & Visualization

Dictionaries

Dr. David Koop
Python Containers

• Container: store more than one value
• Mutable versus immutable: Can we update the container?
  - Yes $\rightarrow$ mutable
  - No $\rightarrow$ immutable
  - Lists are mutable, tuples are immutable
• Lists and tuples may contain values of different types:
  • List: $l = [1, "abc", 12.34]$
  • Tuple: $t = (1, "abc", 12.34)$
• You can also put functions in containers!
  • `len` function: number of items: `len(l)`
• `+` serves as a concatenate operation, creates a new list/tuple
Lists

- New list: \( l = [] \) or \( l = \text{list}() \)
- Add to a list \( l \):
  - \( l.append(v) \): add one value \((v)\) to the end of the list
  - \( l.extend(vlist) \): add multiple values \((vlist)\) to the end of \( l \)
  - \( l.insert(i, v) \): add one value \((v)\) at index \(i\)
- Remove from a list \( l \):
  - \( \text{del } l[i] \): deletes the value at index \(i\)
  - \( l.pop(i) \): removes the value at index \(i\) (and returns it)
  - \( l.remove(v) \): removes the first occurrence of value \(v\) (careful!)
- Changing an entry:
  - \( l[i] = v \): changes the value at index \(i\) to \(v\)
For loops

• Used much more frequently than while loops
• Is actually a "for-each" type of loop
  
  ```python
  for item in someList:
    print(item)
  ```

• Grabs each element of `someList` in order and puts it into `item`

• To count, use `range`:
  
  ```python
  for i in range(100):
    print(i)
  ```

• Break: terminate the loop
• Continue: stop processing the current item, but continue the loop
Assignment 1

• [http://www.cis.umassd.edu/~dkoop/dsc201/assignment1.html](http://www.cis.umassd.edu/~dkoop/dsc201/assignment1.html)

• Populations of New England towns

• Due next Wednesday

• Start now!

• Lists, loops, counting, and math
Dictionaries

• One of the most useful features of Python
• Also known as associative arrays
• Exist in other languages but a core feature in Python
• Associate a key with a value
• When I want to find a value, I give the dictionary a key, and it returns the value
• Example: InspectionID (key) → InspectionRecord (value)
• Keys must be immutable (technically, hashable):
  - Normal types like numbers, strings are fine
  - Tuples work, but lists do not (TypeError: unhashable type: 'list')
• There is only one value per key!
Dictionaries

• Defining a dictionary: curly braces
  states = {'MA': 'Massachusetts, 'RI': 'Road Island', 'CT': 'Connecticut'}

• Accessing a value: use brackets!
  states['MA']

• Adding a value:
  states['NH'] = 'New Hampshire'

• Checking for a key:
  'ME' in states → returns True or False

• Removing a value: del states['CT']

• Changing a value: states['RI'] = 'Rhode Island'
Dictionaries

• Combine dictionaries: `d1.update(d2)`
  - `update` overwrites any key-value pairs in `d1` when the same key appears in `d2`

• `len(d)` is the number of entries in `d`
Extracting Parts of a Dictionary

- `d.keys()`: the keys only
- `d.values()`: the values only
- `d.items()`: key-value pairs as a collection of tuples: `[(k1, v1), (k2, v2), ...]`

- Unpacking a tuple or list
  - `t = (1,2)`
    - `a, b = t`

- Iterating through a dictionary:
  - `for (k,v) in d.items():`
    - `if k % 2 == 0:`
      - `print(v)`

- Important: keys, values, and items are not in any specific order!
Example: Counting Letters

• Write code that takes a string s and creates a dictionary with that counts how often each letter appears in s

• `count_letters("Mississippi") →
  {'s': 4, 'i': 4, 'p': 2, ...}`
Example: Counting Letters

• def count_letters(s):
  letter_counts = {}
  for ch in s:
    if ch not in letter_counts:
      letter_counts[ch] = 1
    else:
      letter_counts[ch] += 1

• Cases? s.upper, s.lower

• get method
Sets

- Sets are like dictionaries but without any values:
  - `s = {'MA', 'RI', 'CT', 'NH'}`
  - `{}` is an empty dictionary, `set()` is an empty set
- Adding values: `s.add('ME')`
- Removing values: `s.discard('CT')`
Nesting Containers

- Can have lists inside of lists, tuples inside of tuples, dictionaries inside of dictionaries
- Can also have dictionaries inside of lists, tuples inside of dictionaries, …
- d = {"Brady": [(2015, 4770, 36), (2014, 4109, 33)],
  "Luck": [(2015, 1881, 15), (2014, 4761, 40)],
  …
}

- JavaScript Object Notation (JSON) looks very similar for literal values; Python allows variables in these types of structures
Nesting Code

• Can have loops inside of loops, if statements inside of if statements

• Careful with variable names:
  
  ```python
  l = {1: 3, 4: 5, 9: 12}
  for i in range(100):
      square = i ** 2
      max_val = l[square]
      for i in range(max_val):
          print(i)
  ```

• Strange behavior, likely unintended, but Python won't complain!
None

- Like null in other languages
- Used as a placeholder when no value exists
- The value returned from a function that doesn't return a value

```python
def f(name):
    print("Hello,", name)
    v = f("Patricia")  # v will have the value None
```

- Also used when you need to create a new list or dictionary:

```python
def add_letters(s, d=None):
    if d is None:
        d = {}
    d.update(count_letters(s))
```

- Looks like `d={}` would make more sense, but that causes issues

- `None` serves as a sentinel value in `add_letters`
is and ==

- == does a normal equality comparison
- is checks to see if the object is the exact same object
- Common style to write statements like `if d is None: ...`
- Weird behavior:
  - `a = 4 - 3`
    - `a is 1` # True
  - `a = 10 ** 3`
    - `a is 1000` # False
  - `a = 10 ** 3`
    - `a == 1000` # True
- Python caches common integer objects
- Generally, avoid is unless writing `is None`
Objects

- \( d = \text{dict()} \) # construct an empty dictionary object
- \( l = \text{list()} \) # construct an empty list object
- \( s = \text{set()} \) # construct an empty set object
- \( s = \text{set([1,2,3,4])} \) # construct a set with 4 numbers

- Calling methods:
  - \( l.\text{append('abc')} \)
  - \( d.\text{update({'a': 'b'})} \)
  - \( s.\text{add(3)} \)

- The method is tied to the object preceding the dot (e.g. append modifies \( l \) to add \('abc'\))
Python Modules

- Python module: a file containing definitions and statements
- Import statement: like Java, get a module that isn't a Python builtin

```python
import collections
d = collections.defaultdict(list)
d[3].append(1)
```

- From...import...: don't need to refer to the module

```python
from collections import defaultdict
d = defaultdict(list)
d[3].append(1)
```
Other Collections

- `collections.defaultdict`: specify a default value for any item in the dictionary (instead of `KeyError`)
- `collections.OrderedDict`: keep entries ordered according to when the key was inserted
  - `dict` objects will be ordered in future versions of Python so this will be obsolete then
- `collections.Counter`: counts hashable objects, has a `most_common` method