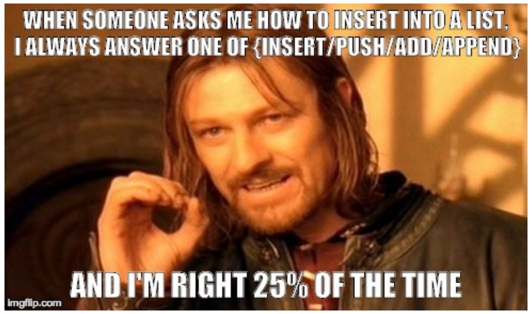
# Python Data Structures

Quick recap

# Lists


WHEN SOMEONE ASKS ME HOW TO INSERT INTO A LIST,
I ALWAYS ANSWER ONE OF {INSERT/PUSH/ADD/APPEND}
AND I'M RIGHT 25% OF THE TIME
imgflip.com

- append(x)
- extend(L)
- insert(i,x)
- remove(x)
- pop([i])
- index(x)
- count(x)
- reverse()
- sort(cmp=None,key=None, reverse=False)
- zip(L1, L2)

## Dictionaries

- Unordered set of *key: value* pairs
  - tel = {'jack': 4098, 'sape': 4139}
  - tel['cesar'] = 2656
  - del tel['sape']
  - dict([ ('jack', 4098), ('sape', 4139) ])
  - dict(jack=4098, sape=4139)
- Iteration:
  - for k, v in tel.iteritems(): …
  - for k in tel: …
  - for k in tel.keys(): …
  - For v in tel.values(): …
- Can be used as sets:
  - names = set(['jack', 'sape'])

# List comprehension

- Convenient to create lists/dictionaries with data
  - L = [x for x in range(10)]
  - D = {x: x*x for x in L}
  - L2 = [D[x] for x in D]
  - L3 = [D[x] for x in D if x % 2 == 0]
  - S = {x for x in L}
  - L4 = [x for x in S if x % 3 == 0]
  - Z = [x * y for x, y in zip([0, 1, 2], [3, 4, 5])]

# Sorting

- sorted([5, 2, 3, 1, 4]    *or*
  - A = [5, 2, 3, 1, 4]
  - A.sort()
- sorted("This is a test string from Andrew".split())
- sorted("This is a test string from Andrew".split(), key=str.lower)
  - A = [('a', 2), ('c', 1), ('b', 0)]
  - sorted(A, key=lambda elm:elm[0])
  - sorted(A, key=lambda elm:elm[1])
  - sorted(A, key=lambda elm:elm[1], reverse=True)

- Note: sorting is **stable**: same keys, preserve order of appearance
  - sorted([('red', 1), ('blue', 1), ('red', 2), ('blue', 2)], key=lambda elm: elm[0])