Day 3: Collections

Suggested reading: *Learning Python* (3rd Ed.)

Chapter 8: Lists and Dictionaries
Chapter 9: Tuples, Files, and Everything Else
Chapter 13: while and for Loops

Turn In Homework

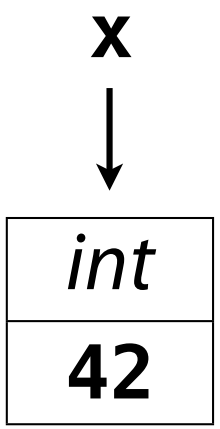
Homework Review

Will not be posted online

Write code.
At least a little.
Every day.
Play around!

Single-Value Objects

- So far: **`int`**, **`float`**, **`str`**, **`bool`**
- Objects of these types hold exactly one value



How can we have a *collection* of (related) values?

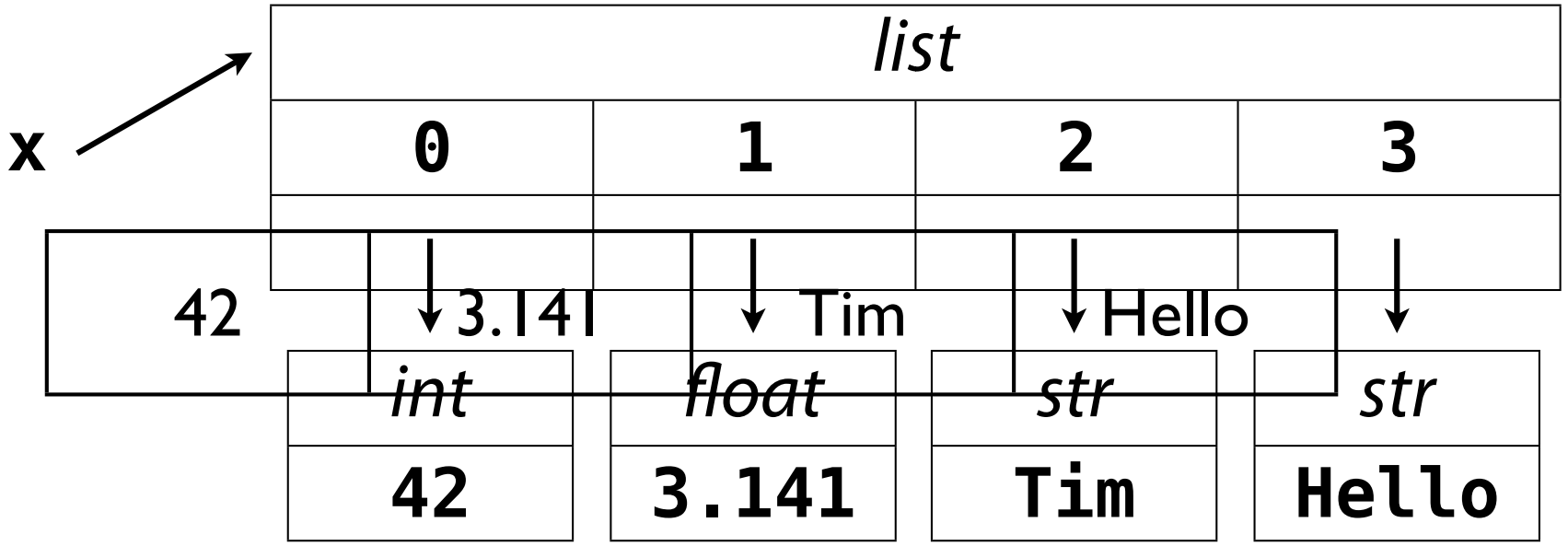
Collection Examples

- All even numbers from 40–100
- Outdoor temperatures by date/time
- List of data files to read
- Weights associated with test items
- Frequency counts of a set of tokens
- Set of all observations from an instrument

Lists

List

- Ordered
- Variable length
- Arbitrary objects and types
- Access via integer position ("index")
- Mutable



Creating a List

```
[item0, item1, ..., item-2, item-1]
```

```
names = ['Tim', 'Scot', 'Alan']

empty = []

mixed = [42, 3.14159, 'hello', True]

long = [925, 161, 164, 529,
        168, 208, 896, 531,
        747, 932]

some_ints = range(2, 100, 2)
```

Using a List

```
list[index]
```

courses →

0	1	2
CS 302	CS 367	CS 368

```
courses = ['CS 302', 'CS 367', 'CS 368']
print courses
courses[0] = 'CS 302 - Intro to Progr.'
courses[1] += ' - Data Structures'
print 'What is %s?' % (courses[2])
```

Other List Operations

```
x = [42, 't', 1.3]
```

length	`len(x)`	`2`
concatenate	`[1, 2] + x`	`[1, 2, 42, 't', 1.3]`
membership	`42 in x`	`True`
slice	`x[0:2]`	`[42, 't']`
append	`x.append(3)`	`x: [42, 't', 1.3, 3]`
extend	`x += [3, 1]`	`x: [42, 't', 1.3, 3, 1]`
insert	`x.insert(1, 'a')`	`x: [42, 'a', 't', 1.3]`
delete	`del x[1]`	`x: [42, 1.3]`
remove	`x.pop(1)`	`'t'  x: [42, 1.3]`

List Bounds

- valid index: **`int`** from **`0`** to (***`length - 1`***)
- lists can grow and shrink (**`append`**, **`insert`**, …)
- limited only by memory
- going out of bounds is run-time error:

```
>>> x = ['a', 'b', 'c']
>>> x[3]
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
IndexError: list index out of range
```

Perplexing Python

Mutable vs. Immutable

- ***Mutable*** types allow changes to objects in memory
  - Examples: **`list`**, **`dict`**
- ***Immutable*** types do not
  - Examples: **`int`**, **`float`**, **`str`**, **`bool`**

```
>>> x = 42
>>> y = x
>>> x += 1
>>> print x
43
>>> print y    # ??
```

```
>>> x = []
>>> y = x
>>> x += [1]
>>> print x
[1]
>>> print y    # ??
```

Back to Collections

Tuples

```
(item0, item1, ..., item-2, item-1)
tuple[n]
```

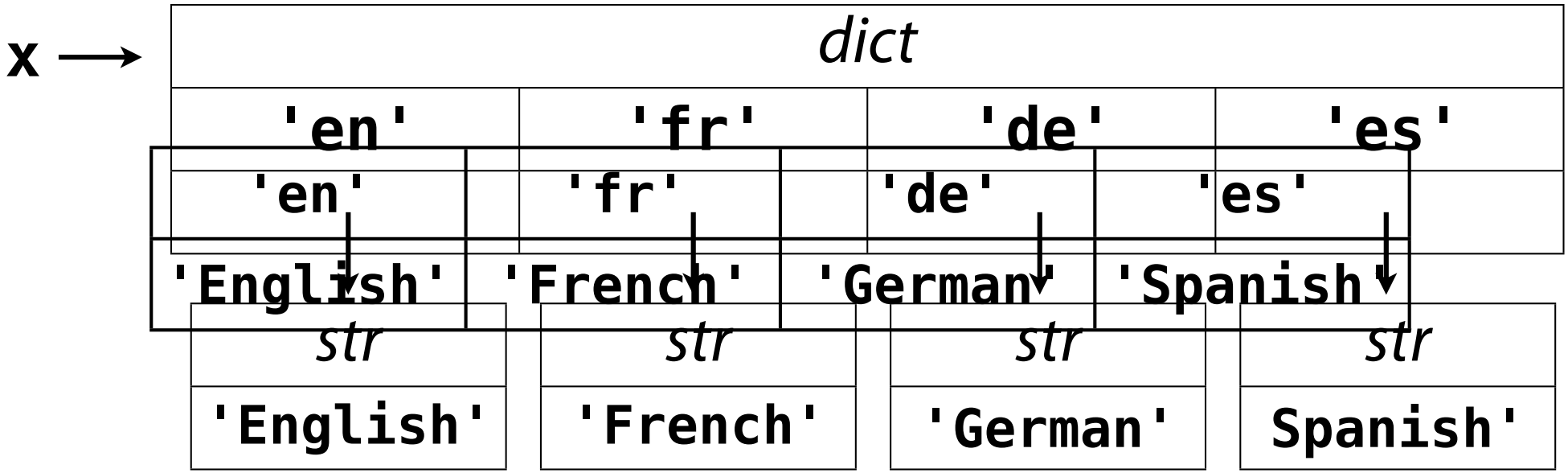
- Immutable lists
- List operators work, methods do not

```
t = (1, 2)
t += (3, 4)    # wait, what?!?
if 3 in t:
    print t[1:3]
```

Dictionaries

Dictionary

- Unordered
- Variable length
- Arbitrary objects and types
- Access via arbitrary, unique *immutable* object ("key")
- Mutable



Creating a Dictionary

```
{keyA: valueA, keyB: valueB, ...}
```

```
languages = {'en': 'English',
             'fr': 'French',
             'de': 'German',
             'es': 'Spanish'}
# print languages
# {'fr': 'French', 'en': 'English', ...}

readings = {'TI3a': 43.23, 'TF3a': 47.09,
            'TI3b': 38.22, 'TF3b': 42.96,
            'TI4a': 42.98, 'TF4a': 47.00,
            ...}
```

Using a Dictionary

dictionary[key]

```
languages = {'en': 'English', ...}
print languages['de']
languages['es'] = 'Espanol'
languages['de'] = 'Deutsche'
languages['de'] += ' (German)'
print "'es' => '%s'" % (languages['es'])

len(languages)                  # 4
languages['ja'] = 'Japanese'    # new entry
len(languages)                  # 5
```

Other Dictionary Operations

```
c = {'a': 5, 'the': 7, 'an': 2}
```

length	`len(c)`	`3`
membership	`'an' in c`	`True`
safe lookup	`c.get('an')` `c.get('no')`	`2` `None`
keys	`c.keys()`	`['the', 'an', 'a']`
items	`c.items()`	`[('the', 7), ('an', 2) …]`
delete	`del c['the']`	`c: {'an': 2, 'a': 5}`
remove	`c.pop('the')`	`7  c: {'an': 2, 'a': 5}`

Sets

- models mathematical concept of a set
- unordered, variable-length, mutable collection
- somewhat between a list and a dictionary

```
a = set([1, 2, 3])
b = set([2, 3, 4])

1 in a      # True
1 in b      # False

a & b       # set([2, 3])
a | b       # set([1, 2, 3, 4])
a - b       # set([1])
```

Sequences and Loops

```
for item in seq:    # list, tuple, set, str
    print item
```

```
sum = 0
for n in xrange(1, 11):
    sum += n
print sum
```

```
s = 'Hello, world!'
print 'Char  ASCII'
print '----  -----'
for char in s:
    print '%4s  %5s' % (char, ord(char))
```

Dictionaries and Loops

```
for key in dict.keys():
    print '%s => %s' % (key, dict[key])
```

```
for pair in dict.items():
    print '%s => %s' % pair
```

```
for key, value in dict.items():
    print '%s => %s' % (key, value)
```

```
valid_tests = []
for key, value in readings.items():
    if value > 0:
        valid_tests.append(key)
```

Phew!

Other Scripting Languages

- All have arrays and associative arrays
- Check for different or additional:
  - **Terminology** (list, array; hash, map, dictionary, …)
  - **Syntax** (`[]` vs. `{}`, `len(array)` vs. `array.length`)
  - **Operations** (sort, unique elements, flatten, shuffle)
  - **Collections** (e.g., set)

Homework

- Implement a simple data analysis tool
  - Collect data observations
  - Display the items and their count, sum, mean(, …)
- BE SURE TO LABEL YOUR PRINTOUT!!!

```
#!/usr/bin/env python

"""Homework for CS 368-2 (2012 Spring)
Assigned on Day 03, 2012-03-20
Written by <Your Name>
"""
```