LING 388: Computers and Language

Lecture 6
Today's Topics

• Reminder Homework 3 on Summarization and Clickbait due tomorrow (midnight)
• Python introduction contd.
Python

- [https://docs.python.org/3/tutorial/introduction.html](https://docs.python.org/3/tutorial/introduction.html)
- Numbers
- Strings
- Lists
- Dictionaries
Python Lists

- Lists as stacks
- Lists as queues
- List Comprehensions (*advanced topic*)

https://visualgo.net/en/list?slide=4

https://www.appcoda.com/ios-concurrency/
Python List as a Queue

**EXAMPLE:**

```python
>>> list = ['c1','c2','c3']
>>> list[0]
'c1'
>>> list = list[1:]
>>> list
['c2', 'c3']
>>> list.append('c4')
>>> list
['c2', 'c3', 'c4']
```

- Method `append()` to add to the queue
- `list[0]` gives us the head of the queue
- `list = list[1:]` deletes the head of the queue from the queue
- Also can use `del list[0]`
Python List as a Queue

A faster implementation according to section 5.1.2

```python
>>> from collections import deque
>>> queue = deque(["Eric", "John", "Michael"])
>>> queue.append("Terry")  # Terry arrives
>>> queue.append("Graham") # Graham arrives
>>> queue.popleft()        # The first to arrive now leaves
'Eric'
>>> queue.popleft()        # The second to arrive now leaves
'John'
>>> queue                   # Remaining queue in order of arrival
deque(["Michael", "Terry", "Graham"])
```
Python List as a Stack

- pop() removes from the same end as append()
Tuples

- Section 5.3: note instead of square brackets, round brackets are used

A tuple consists of a number of values separated by commas, for instance:

```python
>>> t = 12345, 54321, 'hello!'
>>> t[0]
12345
>>> t
(12345, 54321, 'hello!')
>>> # Tuples may be nested:
... u = t, (1, 2, 3, 4, 5)
>>> u
((12345, 54321, 'hello!'), (1, 2, 3, 4, 5))
```
Dictionaries (dict)

- Lists and tuples are indexed by whole numbers (from 0)
- Dictionaries are indexed by a key (usually a string) and return some value associated with the key
- Note: use of curly braces
- Dictionaries are not ordered (like sets) – see next slide
- Methods keys(), values(), items()
- Refer to key + value as an item: encoded as a tuple

```python
>>> tel = {'jack': 4098, 'sape': 4139}
>>> tel['guido'] = 4127
>>> tel
{'sape': 4139, 'guido': 4127, 'jack': 4098}

>>> tel['jack']
4098

>>> del tel['sape']
>>> tel['irv'] = 4127

>>> tel
{'guido': 4127, 'irv': 4127, 'jack': 4098}

>>> list(tel.keys())
['irv', 'guido', 'jack']

>>> sorted(tel.keys())
['guido', 'irv', 'jack']

>>> 'guido' in tel
True

>>> 'jack' not in tel
False
```
Dictionaries (dict)

Example from class...

```python
>>> sorted(d.keys())
[1, 5, 10, 25, 100]
>>> sorted(d.values())
['dime', 'dollar', 'nickel', 'penny', 'quarter']
>>> d.items()
 dict_items([(25, 'quarter'), (10, 'dime'), (100, 'dollar'), (5, 'nickel'), (1, 'penny')])
>>> list(d.items())
[(25, 'quarter'), (10, 'dime'), (100, 'dollar'), (5, 'nickel'), (1, 'penny')]
>>> sorted(d.items())
[(1, 'penny'), (5, 'nickel'), (10, 'dime'), (25, 'quarter'), (100, 'dollar')]
>>> d[1] = 'Penny'
>>> list(d.items())
[(25, 'quarter'), (10, 'dime'), (100, 'dollar'), (5, 'nickel'), (1, 'Penny')]
>>> sorted(d.items())
['Penny', 'dime', 'dollar', 'nickel', 'quarter']
>>> d[1] = 'Penny', 'penny'
>>> d[1]
('Penny', 'penny')
>>> d[100] = ['dollar', '$', 'Dollar']
>>> d
```
Dictionaries (dict)

• Dictionary order preservation depends on the version of Python used ...

New `dict` implementation (Python 3.6)

The `dict` type now uses a “compact” representation based on a proposal by Raymond Hettinger which was first implemented by PyPy. The memory usage of the new `dict()` is between 20% and 25% smaller compared to Python 3.5.

The order-preserving aspect of this new implementation is considered an implementation detail and should not be relied upon (this may change in the future, but it is desired to have this new dict implementation in the language for a few releases before changing the language spec to mandate order-preserving semantics for all current and future Python implementations; this also helps preserve backwards-compatibility with older versions of the language where random iteration order is still in effect, e.g. Python 3.5).
Dictionaries (dict)

• How to print the contents of a dictionary?
• Use a for-loop and method items(): k,v is a tuple

```python
>>> knights = {'gallahad': 'the pure', 'robin': 'the brave'}
>>> for k, v in knights.items():
...     print(k, v)
...     gallahad the pure
robin the brave
```
Dictionaries (dict)

```python
>>> pos
{'likes': ['n', 'v'], 'car': ['n'], 'apple': ['n'], 'smiled': ['v']}
>>> for word, tag in pos.items():
    ...    if 'n' in tag:
    ...        print(word)
    ...
likes
car
apple

>>> pos
{'car': 'n', 'likes': ['n', 'v'], 'apple': 'n', 'smiled': 'v'}
>>> for word, tag in pos.items():
    ...    if tag == 'n' or 'n' in tag:
    ...        print(word)
    ...
    car
    likes
    apple
```

- All values are lists
  Advantage: simplifies the code

- Values are lists or a string
  Advantage: simpler-looking dict
Dictionaries

• Works too!

```python
dict = {'car': 'n', 'likes': ['n', 'v'], 'apple': 'n', 'smiled': 'v'}
for word, tag in dict.items():
    if 'n' in tag:
        print(word)
car
likes
apple
```
Dictionaries (dict)

- function `zip()` pairs up elements from two lists into an iterable

```python
>>> questions = ['name', 'quest', 'favorite color']
>>> answers = ['lancelot', 'the holy grail', 'blue']
>>> for q, a in zip(questions, answers):
...     print('What is your {0}? It is {1}.'.format(q, a))
... What is your name? It is lancelot.
What is your quest? It is the holy grail.
What is your favorite color? It is blue.
```

```python
>>> ll = ['a', 'b', 'c']
>>> l2 = [0, 1, 2]
>>> l3 = ['x', 'y', 'z']
>>> zip(l1, l2, l3)
<zip object at 0x104554dc8>
>>> for v1, v2, v3 in zip(l1, l2, l3):
...     print(v2, v1, v3)
... 0 a x
1 b y
2 c z
```
Dictionaries (dict)

• function zip() doesn't always work quite the same ...

```python
>>> l1 = ['a','b','c']
>>> l2 = [0,1,2]
>>> l3 = ['x','y','z']
>>> zip(l1,l2,l3)
[('a', 0, 'x'), ('b', 1, 'y'), ('c', 2, 'z')]
>>> for v1,v2,v3 in zip(l1,l2,l3):
...     print(v2,v1,v3)
...       0 a x
       1 b y
       2 c z
```
For loops and range()

• `range(start, stop, step)` returns an iterable that can be used with a for-loop
• `range(stop)` start assumed to be 0; step assumed to be 1; range does not include stop

```
>>> list(range(10))
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
>>> list(range(1, 11))
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
>>> list(range(0, 30, 5))
[0, 5, 10, 15, 20, 25]
>>> list(range(0, 10, 3))
[0, 3, 6, 9]
>>> list(range(0, -10, -1))
[0, -1, -2, -3, -4, -5, -6, -7, -8, -9]
>>> list(range(0))
[]
>>> list(range(1, 0))
[]
```
For loops and range()

- Ranges can be indexed like a list
- Can use `in` and `not in`