LING/C SC/PSYC 438/538

Lecture 7

Sandiway Fong

Today's Topics

- A question from last time
- Perl hash and Python dict
- Homework 6

Question from last time

- (len) number of characters vs. number of bytes

```
~$ python3 -c 'print(len("侍"))'
1
~$ python3 -c 'print(len("été"))'
3
~$ python3 -c 'print(len("'l\''été"))'
5
~$
```

```
$ python3 -c 'print(len("a".encode("utf-8")))'
1
$ python3 -c 'print(len("ba".encode("utf-8")))'
2
$ python3 -c 'print(len("侍".encode("utf-8")))'
3
$ python3 -c 'print(len("é".encode("utf-8")))'
2
$
```

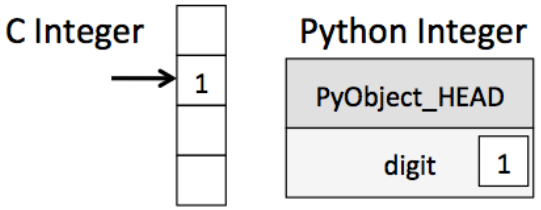
Question from last time

`sys.getsizeof(object[, default])`

Return the size of an object in bytes. The object can be any type of object. All built-in objects will return correct results, but this does not have to hold true for third-party extensions as it is implementation specific.

```
$ python3 -c 'import sys;print(sys.getsizeof("侍"))'
76
$ python3 -c 'import sys;print(sys.getsizeof("a"))'
50
$ python3 -c 'import sys;print(sys.getsizeof("ab"))'
51
$ python3 -c 'import sys;print(sys.getsizeof("a侍"))'
78
```

- Python is inefficient in data storage: why?



Python: strings

- List comprehension:
  - `sentence = ['A', 'big', 'cat', 'in', 'Tucson']`
  - `[x.lower() for x in sentence]`
- Suppose we want to use `.endswith()` in a list comprehension:

> `str.` **`endswith`**(*suffix*[, *start*[, *end*]])
>
> Return `True` if the string ends with the specified *suffix*, otherwise return `False`. *suffix* can also be a tuple of suffixes to look for. With optional *start*, test beginning at that position. With optional *end*, stop comparing at that position.

Reference: https://docs.python.org/3.7/library/stdtypes.html#text-sequence-type-str

Python: strings

```
>>> sentence = ['A', 'big', 'cat', 'in', 'Tucson']
>>> [x.endswith('n') for x in sentence]
[False, False, False, True, True]
>>> [x for x in sentence if x.endswith('n')]
['in', 'Tucson']
>>> [x if x.endswith('n') else 1 for x in sentence]
[1, 1, 1, 'in', 'Tucson']
>>>
```

- conditional list comprehensions

Perl Hashes

perlintro
Perl 5 version 12.1 documentation

Go to top · Download PDF
Show page index · Show recent pages

Search

Home > Overview > perlintro

- **Hashes**

A hash represents a set of key/value pairs:

```
1.  my %fruit_color = ("apple", "red", "banana", "yellow");
```

You can use whitespace and the `=>` operator to lay them out more nicely:

```
1.  my %fruit_color = (
2.      apple  => "red",
3.      banana => "yellow",
4.  );
```

To get at hash elements:

```
1.  $fruit_color{"apple"};          # gives "red"
```

You can get at lists of keys and values with **`keys()`** and **`values()`**.

```
1.  my @fruits = keys %fruit_colors;
2.  my @colors = values %fruit_colors;
```

Hashes have no particular internal order, though you can sort the keys and loop through them.

Just like special scalars and arrays, there are also special hashes. The most well known of these is `%ENV` which contains environment variables. Read all about it (and other special variables) in perlvar.

- aka `dict` in Python

Notation:

1. Perl array: @ […]
2. Perl hash: % {…}

Perl Hashes

- Notes on arrays and hashes
  - **arrays** are indexed using integers (0, 1, 2, 3,…) as **keys**
  - **hashes** are like arrays with *user-defined* indexing (*aka* keys)
    aka associative array or hash table or dict (Python)

  - initialization
    (use list notation (or shortcut): ***round brackets and commas***)
    - `@a = ("zero", "one", "two", "three", "four");`
    - `%h = ("zero", 0, "one", 1, "two", 2, "three",  3, "four", 4);`
      (*key/value pairs*)

  - access to individual elements
    (*square brackets vs. curly braces*)
    - `$a[1]` "one"
    - `$h{zero}` 0

no quotes needed for strings
'zero' "zero"

Shortcut:

```
1.    qw(foo bar baz)
```

is semantically equivalent to the list:

```
1.    "foo", "bar", "baz"
```

Perl Hashes

Notes on arrays and hashes

- `@a = ("zero", "one", "two", "three", "four");`
- `%h = ("zero", 0, "one", 1, "two", 2, "three",  3, "four", 4);`

- **output**
  - `print @a` zeroonetwothreefour
  - `print "@a"` zero one two three four
  - `print %h` three3one1zero0two2four4 (note: *different order*)
  - `print "%h"` %h (no interpolation+)

More on Hash tables

- Looping over the **array** of keys:

```
use strict;
my %fruitColor = ("apple", "red", "banana", "yellow", "kiwi", "green");
foreach my $i (keys %fruitColor) {
  print "$i => $fruitColor{$i}\n"
}
```

www.freeimages.com

- Output:

```
$ perl fruits.perl
kiwi => green
apple => red
banana => yellow
$
```

```
%fruitColor = qw(apple red banana yellow kiwi green);
for (keys %fruitColor) {
  print "$_ => $fruitColor{$_}\n"
}
```

  - Notice: order is not guaranteed! (*Implementation dependent*)

More on Hash tables

- Unique key constraint:

```
my %fruitColor = ("apple", "red", "banana", "yellow", "kiwi", "green",
 "apple", "green");
```

- Exists:
  - `exists $fruitColor{apple}`
  - `exists $fruitColor{orange}`

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

```
>>> 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)

```
>>> d = {10:"dime", 100:"dollar", 5:"nickel", 25:"quarter", 1:"penny"}
>>> 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.values())
['Penny', 'dime', 'dollar', 'nickel', 'quarter']
>>> d[1]='Penny', 'penny'
>>> d[1]
('Penny', 'penny')
>>> d[100] = ['dollar', '$', "Dollar"]
>>> d
{25: 'quarter', 10: 'dime', 100: ['dollar', '$', 'Dollar'], 5: 'nickel', 1: ('Penny', 'penny')}
```

Python `dict`

- Dictionary order preservation depends on the version of Python used …

New dict implementation

Python 3.6 onwards

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).

Python `dict`

The `dict()` constructor builds dictionaries directly from sequences of key-value pairs:

```
>>> dict([('sape', 4139), ('guido', 4127), ('jack', 4098)])
{'sape': 4139, 'guido': 4127, 'jack': 4098}
```

In addition, dict comprehensions can be used to create dictionaries from arbitrary key and value expressions:

similar syntax to a list comprehension
[… for *var* in *ITERABLE*]

```
>>> {x: x**2 for x in (2, 4, 6)}
{2: 4, 4: 16, 6: 36}
```

When the keys are simple strings, it is sometimes easier to specify pairs using keyword arguments:

```
>>> dict(sape=4139, guido=4127, jack=4098)
{'sape': 4139, 'guido': 4127, 'jack': 4098}
```

Python `dict`

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

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

Python `dict`

```
>>> 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
```

All values are lists
Advantage: simplifies the code

Trade-off

```
>>> 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
```

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

can be simplified because

```
>>> 'n' in 'n'
True
>>>
```

Python `dict`

- Works too!

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

Less transparent:

- relies on a Python-particular quirk …
- and doesn't always work!

```
>>> 'abc' in 'abc'
True
>>> 1 in 1
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: argument of type 'int' is not iterable
```

Python `dict`

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

```
>>> 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.
```

```
>>> l1 = ['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
>>>
```

Python `dict`

- function zip() doesn't always work quite the same …

```
Python 2.7.10 (default, Jul 15 2017, 17:16:57)
[GCC 4.2.1 Compatible Apple LLVM 9.0.0 (clang-900.0.31)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> 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')
>>>
```

```
>>> l1 = ['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
>>>
```

Homework 6

Disemvoweling, **disemvowelling** (see doubled *L*), or **disemvowelment** of a piece of alphabetic text is rewriting it with all the vowel letters removed.[1][*full citation needed*] This original sentence:

> The quick brown fox jumps over the lazy dog

would, after being disemvowelled, look like this:

> Th qck brwn fx jmps vr th lzy dg

Disemvoweling is a common feature of SMS language[1] as disemvoweling requires little cognitive effort[*citation needed*] to read, so it is often used where space is costly. The word *disemvoweling* is a portmanteau combining *vowel* and *disembowel*.[1]

The word was used with precisely this meaning in the 1939 novel *Finnegans Wake*

https://en.wikipedia.org/wiki/Disemvoweling

Background

—No more than Richman's periwhelker.

—Nnn ttt wrd?

—Dmn ttt thg.

—A gael galled by scheme of scorn? Nock?

—Sangnifying nothing. Mock!

—*Fortitudo eius rhodammum tenuit?*

—Five maim! Or something very similar.

—I should like to euphonise that. It sounds an isochronism. Secret speech Hazelton and obviously disemvowelled. But it is good laylaw too. We may take those wellmeant kicks for free granted, though *ultra vires*, void and, in fact, unnecessarily so. Happily you were not quite so successful in the process verbal whereby you would sublimate your blepharospasmockical suppressions, it seems?

—What was that? First I heard about it.

Finnegan's Wake by James Joyce
Book 3: Episode 3

disemvowelled

- wordplay: *periwhelker* = periwinkle
- how about?
  - Nnn ttt wrd? (*Not a word*)
  - Dmn ttt thg. (*Damn that thing*)

clemsonhgic.wpengine.com/

Background

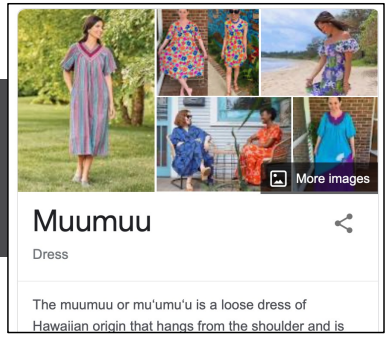
Where Have All the Vowels Gone?

Consider the muumuu.

By John Williams • Dec. 29, 2018

- https://www.nytimes.com/2018/12/29/style/vowels-no-more.html
  - tech companies like **Tumblr** and **Flickr** arrived on the scene, dropping e's both for distinctiveness and because the altered names made it easier to trademark, claim domain names on the internet and conduct other practical business.

After all, there are words that can hardly do without them: muumuu, audio and oboe, just to queue up a few. One cannot text someone "b" and expect them to know one is referring to an oboe.



Homework 6

- Read https://en.wikipedia.org/wiki/Disemvoweling
- Part 1:
  - write a Perl program to remove vowels *a, e, i, o, u* from words typed into the command line. (Don't worry about *y*.)
- Hint: use `split` from previous lecture
- Possible template for your program (*see slide ahead*)
- **Example:**

```
Desktop$ perl dv.perl quick brown fox
qck brwn fx
```

Homework 6

```
1my @vowels = qw(a e i o u A E I O U);¶
2my  %vowel;¶
3foreach $v (@vowels) {¶
4   $vowel{$v} = undef¶
5}¶
6foreach $word (@ARGV) {¶
7¶
8¶
9¶
10¶
11¶
12¶
13}¶
14print "\n"¶
```

- Simple Recipe:

  (*but not resulting in the shortest program*)
  - Take each word, e.g. *rabbit*
    - Split it into characters, e.g. *r a b b i t*
    - for each character, print it if it's not a vowel
  - at the end of a word, print a space
- at the end of all the words, print a newline
- Could make use of:
  - `exists $vowel{$char}`
  - https://perldoc.perl.org/functions/exists.html

Homework 6

- Part 2:
  - Modify your program for part 1 to not delete leading vowels
  - Example:
    - `If a sentence is unreadable`
    - `f sntnc s nrdbl`
    - `If a sntnc is unrdbl`

Homework 6

- Submit code and example runs in one PDF file
  - *submit partial code if you cannot complete it*
- Due next Sunday night: reviewed next Monday