Class 27: Taming of the Plagarist

cs1120 Fall 2009
David Evans

Menu

- Python Dictionaries
- History of Object-Oriented Programming

PS6-related talk **tomorrow**:
Thursday, October 29 at **2:00 p.m**, Scholars' Lab in the Alderman Library

"Disruptive Construction of Game Worlds"
Shane Liesegang (UVa 2004 CogSci major/CS minor)
Bethesda Softworks

For extra credit on PS6: mention something in your answer to Question 8 that you learned from this talk.

Python Dictionary

Dictionary abstraction provides a lookup table. Each entry in a dictionary is a

<*key*, *value*>

pair. The *key* must be an immutable object. The *value* can be anything.

dictionary[*key*] evaluates to the *value* associated with *key*. Running time is approximately constant!

Dictionary Example

```
>>> d = {}
>>> d['UVa'] = 1818
>>> d['UVa'] = 1819
>>> d['Cambridge'] = 1209
>>> d['UVa']
1819
>>> d['Oxford']

Traceback (most recent call last):
  File "<pyshell#93>", line 1, in <module>
    d['Oxford']
KeyError: 'Oxford'
```

Create a new, empty dictionary

Add an entry: key 'UVa', value 1818

Update the value: key 'UVa', value 1819

Histogramming

Define a procedure histogram that takes a text string as its input, and returns a dictionary that maps each word in the input text to the number of occurences in the text.

Useful string method: split()
outputs a list of the words in the string

```
>>> 'here we go'.split()
['here', 'we', 'go']
```

```
def histogram(text):
    d = {}
    words = text.split()
    for w in words:
        if w in d:
            d[w] = d[w] + 1
        else:
            d[w] = 1
    return d
```

```
>>> d = histogram(declaration)
>>> show_dictionary(d)
of: 79
the: 76
to: 64
and: 55
our: 25
their: 20
has: 20
for: 20
in: 18
He: 18
a: 15
these: 13
...
```

Showing the Dictionary

```
def show_dictionary(d):
    keys = d.keys()
    okeys = sorted(keys, lambda k1, k2: d[k2] - d[k1])
    for k in okeys:
        print str(k) + ": " + str(d[k])
```

Author Fingerprinting (aka Plagarism Detection)

"The program identifies phrases of three words or more in an author's known work and searches for them in unattributed plays. In tests where authors are known to be different, there are up to 20 matches because some phrases are in common usage. When *Edward III* was tested against Shakespeare's works published before 1596 there were 200 matches."

The Times, 12 October 2009

```
def histogram(text):
    d = {}
    words = text.split()
    for w in words:
        if w in d:
            d[w] = d[w] + 1
        else:
            d[w] = 1
    return d
```

```
def phrase_collector(text, plen):
    d = {}
    words = text.split()
    words = map(lambda s: s.lower(), words)
    for windex in range(0, len(words) - plen):
        phrase = tuple(words[windex:windex+plen])
        if phrase in d:
            d[phrase] = d[phrase] + 1
        else:
            d[phrase]= 1
    return d
```

Dictionary keys must be immutable: convert the (mutable) list to an immutable tuple.

```
def common_phrases(d1, d2):
    keys = d1.keys()
    common = {}
    for k in keys:
        if k in d2:
            common[k] = (d1[k], d2[k])
    return common
```

```
def get_my_homepage():
    return urlopen('http://www.cs.virginia.edu/evans/index.html').read()
```

```
>>> ptj = phrase_collector(declaration, 3)
>>> pde = phrase_collector(get_my_homepage(), 3)
>>> c = common_phrases(ptj, pde)
>>> len(c)
0
```

```
>>> pde = phrase_collector(get_my_homepage(), 2)
>>> ptj = phrase_collector(declaration, 2)
>>> c = common_phrases(ptj, pde)
>>> show_phrases(c)
('principles', 'and'): (1, 1)
('has', 'kept'): (1, 1)
('has', 'been'): (1, 1)
('and', 'our'): (1, 1)
('design', 'to'): (1, 1)
('not', 'be'): (1, 1)
('of', 'all'): (1, 1)
('they', 'have'): (1, 1)
('by', 'the'): (1, 1)
('protection', 'of'): (1, 1)
('with', 'a'): (1, 1)
('as', 'we'): (1, 1)
('is', 'the'): (2, 1)
('them', 'to'): (1, 2)
('to', 'a'): (1, 2)
('the', 'state'): (1, 2)
('people', 'to'): (1, 2)
...
('of', 'the'): (4, 12)
```

Possible Project Idea

Make a website that allows visitors to compare text samples for common phrases.

Reminder: if you want to do a "super ambitious" web application project instead of PS7, you need to convince me by Monday (November 2). You should have a team, idea for a project, and justification explaining why it is "super ambitious".

History of Object-Oriented Programming

Object-oriented programming is an exceptionally bad idea which could only have originated in California. Edsger Dijkstra

I don't know how many of you have ever met Dijkstra, but you probably know that arrogance in computer science is measured in nano-Dijkstras. Alan Kay

The people who are the worst at programming are the people who refuse to accept the fact that their brains aren't equal to the task. Their egos keep them from being great programmers. The more you learn to compensate for your small brain, the better a programmer you'll be. The more humble you are, the faster you'll improve.

Edsger Dijkstra, 1972 Turing Award

http://www.cs.utexas.edu/users/EWD/ewd03xx/EWD340.PDF

Computing in World War II

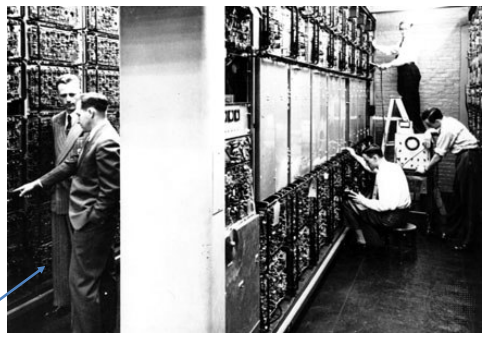
Cryptanalysis (Lorenz: Collossus at Bletchley Park, Enigma: Bombes at Bletchley, NCR in US)

Ballistics Tables, calculations for Hydrogen bomb (ENIAC at U. Pennsylvania)

Batch processing: submit a program and its data, wait your turn, get a result

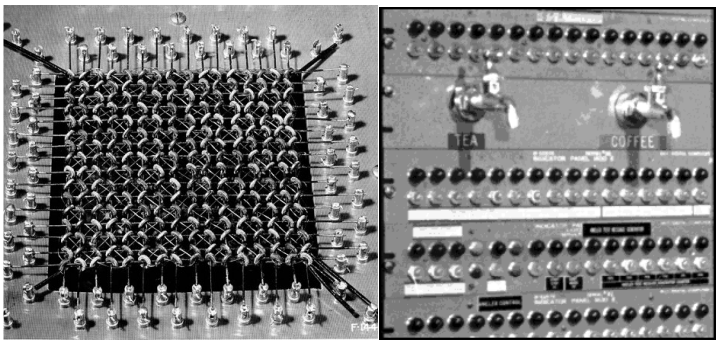
Building a flight simulator required a different type of computing: interactive computing

Pre-History: MIT's Project Whirlwind (1947-1960s)



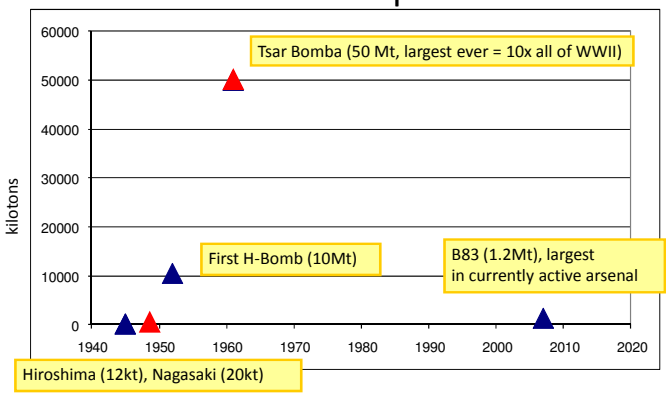
Jay Forrester

Whirlwind Innovations



Magnetic Core Memory
(first version used vacuum tubes)

Short or Endless Golden Age of Nuclear Weapons?



Semi-Automatic Ground Environment (SAGE)

MIT/IBM, 1950-1982

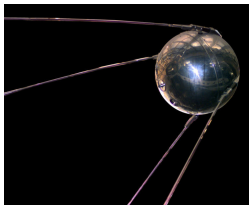
Coordinate radar stations in real-time to track incoming bombers

Total cost: ~$55B

(more than Manhattan Project)

R-7 Semyorka

First intercontinental ballistic missile
First successful test: August 21, 1957



Sputnik: launched by R-7, October 4, 1957

What does all this have to do with object-oriented programming?

(To be continued Friday...)

Charge

- **PS6 due Friday**
- Friday: Trick-or-Treat Protocols, Interpreters

PS6-related talk **tomorrow**:
Thursday, October 29 at **2:00 p.m**, Scholars' Lab in the Alderman Library

"Disruptive Construction of Game Worlds"
Shane Liesegang (UVa 2004 CogSci major/CS minor)
Bethesda Softworks

For extra credit on PS6: mention something in your answer to Question 8 that you learned from this talk.