§7.7-7.8: Pseudo-Random Numbers

17 Oct 2007
CMPT14x
Dr. Sean Ho
Trinity Western University

TRINITY WESTERN UNIVERSITY

Review from §7.0-7.6

- Strings: manipulating text
  - Null-terminated strings
- Application: cryptography (substitution cipher)
  - Creating a library for cryptography
  - Public interface
  - Library-internal helper functions

What's on for today (§7.7, §8.0-8.3)

- §7.7: Application: pseudo-random number generator
  - Accessing global variables
  - Assessing randomness

Application: Random numbers

- A random number (from a uniform distribution) is chosen such that every number within the range is equally likely to be chosen:
  - Uniform distribution on [0..1]
- Making things truly random (high entropy) is very difficult!
  - Hardware random-number generators:
    - Measure radioactive decay of isotopes
    - Brownian motion of particles in a suspension (air)
  - Software pseudo-random number generators

Pseudo-random number generator

- A pseudo-random number generator applies some math operations to the last number generated to get the next number
  - Start with a seed number
  - Hopefully it's "random enough"
  - But really it's completely deterministic:
    - If we start again with the same seed, we'll always get the same sequence of "random" numbers
- e.g., seed=0.10: generates
  - 0.72, 0.23, 0.19, 0.93, 0.54, 0.77, 0.11, ...

DEF: pseudo-random num library

- We only need one public procedure: Random()

```
def random ():
    """Returns a random float between 0 and 1."""

def init_seed (x):
    """Initialize the number generator seed."""
```

- init_seed provides a way for the user to manually set the seed.

IMP: pseudo-random num library

```
"""Pseudo-random number generator.

Sean Ho
CMPT14x example 2006.
"""
from math import exp, log, pi

seed = 0            # persistent across calls to random()
def init_seed (x):
    """Initialize the number generator seed.
    Accessor (set) function for seed."""
    global seed             # access global variable
    seed = x
```

IMP: pseudorandom.py, cont.

```
def random ():
    """Returns a random float between 0 and 1."""
    global seed          # access global variable

    # Try to scramble up seed as much as possible
    seed = seed + pi
    seed = exp (7.0 * log (seed))

    # Only keep the fractional part, in range 0..1
    seed = seed – int (seed)
    return seed
```

Online test of PseudoRandom

- (demo in Python of PseudoRandomTest)
- Library: http://twu.seanho.com/python/pseudorandom.py

- Evaluating "randomness":
  - Graphical evaluations: plot points (x,y) where both coordinates are from Random()
  - Check for dense spots, sparse spots in 1x1 square
  - Python has graphics libraries, but that's beyond the scope of this class

Python's own pseudorandom

- Python has a built-in pseudorandom generator:

```
from random import random
random()
seed()
```

  - Random float in interval [0.0, 1.0)
- Histogram to evaluate randomness
  - Split up interval [0.0, 1.0) into equal-size bins
  - Generate a list of random numbers
  - Count how many numbers fall in each bin

TODO items

- Lab05 due tonight: ch6 # 33 / 35
- HW05 due Fri:
  - ch6 # 25 (hint does not apply in Python)
  - ch6 # 28 (write a Python program to do this)
- 140 Final / 141 midterm next week
  - Wed 24Oct 14:35-15:50 (part 1)
  - Thu 25Oct 13:10-14:15 (part 2)