Library Ex.: Pseudo-random

19 Oct 2009
CMPT140
Dr. Sean Ho
Trinity Western University

TRINITY WESTERN UNIVERSITY

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 generators

- 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 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 library

```
"""Pseudo-random number generator.
Sean Ho for CMPT140 demo.
"""
from math import exp, log, pi

seed = 0                    # persistent across calls
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

- 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 various graphics libraries, too

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
- See PyTut sect 10.6 for more on random