Introduction to: Computers & Programming: Random Numbers and Probability in Python

Adam Meyers
New York University
Outline

• The Random Package
• Basics of Probability
• Applications of Randomness and Probability
The Random Package

- `Random.random()` – generates a number between 0 and 1
- `Random.randint(start,end)` – generates an integer between start and end (inclusive)
- `Random.choice(sequence)` – randomly picks an item in sequence
- `Random.seed()` – uses the current time as a seed for pseudo random number generation
  - `Random.seed(a=integer)` – uses a as a seed, so pseudo-random numbers generated are predictable – useful for testing a system
Random Numbers → Random Events

• Die rolls: random.randint(1,6)
  – Die rolls are independent of other die rolls

• Choosing independent events in a list, e.g.,
  – door_states = ['opened', 'locked', 'unlocked', 'destroyed']
  – next_door = door_states.choice()

• Choosing Events that don't repeat
  – next_card = deck.pop(random.randint(0,len(deck)-1))
  – Removes card from deck so it will not come up again
Probability

- Probability(Event_X)
  - Fraction representing how likely Event_X is
  - If there are N equally likely events including Event_X
  - Then Probability(Event_X) = portion of some given set of events is likely to be Event_X

- Example, given a “fair” die with 6 sides,
  - A 6 is expected to be rolled 1/6 of the time
Probability of Totals of 2 Dice

- Fraction of combinations totaling 2, 3, ..., 12 are expected probabilities of these rolls

<table>
<thead>
<tr>
<th>Roll</th>
<th>Combos</th>
<th>Decimal</th>
<th>Roll</th>
<th>Combos</th>
<th>Decimal</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>1/36</td>
<td>.028</td>
<td>8</td>
<td>5/36</td>
<td>.139</td>
</tr>
<tr>
<td>3</td>
<td>2/36</td>
<td>.056</td>
<td>9</td>
<td>4/36</td>
<td>.111</td>
</tr>
<tr>
<td>4</td>
<td>3/36</td>
<td>.083</td>
<td>10</td>
<td>3/36</td>
<td>.083</td>
</tr>
<tr>
<td>5</td>
<td>4/36</td>
<td>.111</td>
<td>11</td>
<td>2/36</td>
<td>.056</td>
</tr>
<tr>
<td>6</td>
<td>5/36</td>
<td>.139</td>
<td>12</td>
<td>1/36</td>
<td>.028</td>
</tr>
<tr>
<td>7</td>
<td>6/36</td>
<td>.167</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
random_trials_with_2_dice

• Program Simulates any number of dice rolls
• Compares frequencies with expected frequencies
• As number_of_trials increases, the results are closer and closer to expected probability
• Try with 100 rolls, 1000 rolls, 10000, rolls, …
• Law of Large Numbers:
  – More trials → Average is Closer to Expected Value
As number of Dice Increase, Distribution of Rolls approaches Gaussian Distribution

• Image from:
Trustworthiness of Scientific Findings Based on Probability

- Trustworthiness of Scientific Findings:
  - Larger differences from expected probability imply that results are not due to chance
  - How large is the dataset?
    - In a small data set, it can be difficult to get reliable result

- Examples:
  - Suppose you think that people who wear hats are tall
    - Evidence: 3 people wear hats and 2 of whom are tall
    - It could be due to chance
    - Evidence is not convincing
  - Suppose you think that 1000 people are related to each other
    - Evidence: Of those 1000 people, 500 have blue eyes
    - There is an 8% chance that a given person has blue eyes
    - This could not be due to chance
    - Evidence is plausible
Technical & Scientific Applications

• Random Sampling
  – Estimate using small sample instead of everything
  – Can be more accurate than using everything

• Machine Learning:
  – Record statistics about labels on items
  – Use statistics to label new items
  – Example: Automatically classifying words with their parts of speech (nouns, verbs, adjectives, etc.)

• Random generation of keys/passwords/etc. for encryption/decryption.
Random Applications in Society

• Choosing winners and losers without objective criteria
  – Lotteries: jobs, housing, draft, …
  – Drawing straws, rock/paper/scissors, odds & evens, flipping coins, …

• Fortune Telling, e.g., Tarot Carts
  – Illusion of Divination is reinforced when randomly chosen selection makes correct prediction
Pointillism Application for Graphics

• Pointillism:
  – https://en.wikipedia.org/wiki/Pointillism

• draw_blurry funcs in random_examples.py
  – Randomly make dots near target locations to give the effect of fuzzy lines.

• Other Unimplemented Details
  – Shading
    • Dots closer together appear darker
    • Dots further apart appear lighter
  – Changing colors of dots
    • Random changes within small range of colors, e.g., choose RGB values with max difference of +/-50
Summary

- (Pseudo) Random numbers handled by the random package
- Distribution of Random Events Determined by Probability
  - More trials → Closer to Expected Value
- Applications: games, fortune telling, computer simulations, graphics, etc.