CS 1110

Lecture 17: **Card Tricks**

Announcements

Academic Integrity

We have some too-similar handins for A3, and will have to pursue AI cases. Please review the CS1110 web page on the subject, particularly "Specific issues in CS1110."

A4

…is out. Printouts in lecture, version with any later corrections on the web.

No lab

There is no new lab exercise this week.

Slides by D. Gries, L. Lee, S. Marschner, W. White

Quick poker primer

- Basic (straight) version: 5 random cards in your hand
  - 2 of same rank: **pair** (e.g., **3C 3D** 5D 6H 7S)
  - 3 of same rank: **three of a kind** (**5C 5D 5H** 6H 7S)
  - 4 of same rank: **four of a kind** (**6C 6D 6H 6S** 7S)
  - Two separate pairs: **two pair** (**5C 5D 6D 6H** 7S)
  - 3 of one rank, 2 of another: **full house** (**5C 5D 6D 6H 6S**)
  - All cards in rank sequence: **straight** (**5C 6D 7D 8H 9S**)
  - All cards of same suit: **flush** (**5H 7H 8H 10H KH**)
  - All in seq. *and* same suit: **straight flush** (**5D 6D 7D 8D 9D**)
- Less probable hands beat more probable hands
- Some games have > 5 cards; you choose 5 cards

Goal for this lecture

What are the probabilities of all the various poker hands, with varying numbers of cards in the hand?

- Plan: randomly deal out N hands (like maybe a million) and see how many are full houses, straights, etc. The ratio of the number of straights dealt to N is an estimate of the probability of being dealt a straight.
- This requires us to extend our `Hand` class so that it can recognize all eight poker hands. We'll do this, develop testing code along the way, and then answer the question with this **Monte Carlo** experiment.

Stage 1: start implementing

- Stub in methods
- Implement some easy ones
- Python features:
  - multi-way comparison
  - conversion to `bool` in conditionals

multi-way comparisons

`x < y < z` means `x < y and y < z`

`w == x == y == z` means `w == x and x == y and y == z`

One difference: middle operands are **not** evaluated twice.

conversion to Boolean in conditionals

`4.0 + 2` automatically converts 2 to `float`

`if x: ...` automatically converts `x` to `bool`.

`if n:` means `if n != 0:` — for numeric types: zero is false, nonzero true

`if lst:` means `if len(lst) != 0:` — for sequence types: empty is false, nonempty true

Stage 2: start testing

- Form a plan for testing the whole class
- Start by setting up tests for the first methods
- Python features:
  - positional and keyword arguments
  - `assert` statement
- New patterns:
  - `__init__` with multiple disjoint ways to initialize
  - streamlining test cases with lists

positional and keyword arguments

Make arguments optional by giving defaults: `def f(x, y=3, z=4):`

Then the arguments can be matched to parameters two ways:

…by position: `f(1)` (x=1, y=2, z=3) or `f(5, 6)` (x=5, y=6, z=4)

…by keyword: `f(1, z=7)` (x=1, y=2, z=7) or `f(x=5, y=6)` (x=5, y=6, z=4)

the assert statement

`assert` <*Boolean expression*> [, <*string*>]

This statement is for debugging: if the Boolean expression is false, it stops the program, generating an error with a stack trace. If you include the optional string, it is also part of the error message. This statement is useful when the error message from `cunittest2.assert_equal` doesn't contain the information you need.

Stage 3: complete implementation

- Counting suit and rank histograms is useful
- Also simplifies generalization to more cards
- Python features:
  - augmented assignment (+= and friends)
  - `__str__` vs. `__repr__`
- new patterns:
  - computing sums using a for loop
  - computing several sums in parallel
  - computing a maximum using a for loop

augmented assignment

`x += y` is shorthand for `x = x + y`

One difference: expression on the left is **not** evaluated twice.

special methods __str__ and __repr__

We have seen the special method `__str__` that is called to generate a readable string representation of an object. There is also `__repr__` which does something similar.

`__str__` is used by `str()` and `print`. Meant for the user to read.

`__repr__` is used by `repr()`, the interactive prompt, and back-quotes (`` `x` ``). Should be unambigous, for the programmer.

Ideally, `repr()` returns exactly what you would put in your program to create that value—recall how strings work.

Stage 4: the actual experiment

- Now we can finally answer the question!
- Generate N hands, count number of each type
- Python features:
  - Boolean to integer conversion
- new patterns:
  - Monte Carlo estimation

Boolean to integer conversion

Conversions involving `bool` go both ways. When you convert a `bool` to a numeric type, `True` is 1 and `False` is 0.

Example: `x += (y > 3)` increments `x` by 1 if `y` is greater than 3.

Monte Carlo methods

- In CS, "Monte Carlo" is the idea of a program that runs a random experiment many times to estimate some value.
- It can estimate many quantities for which it is very hard or impossible to derive formulas.
- A nice example of trading long computation time for ease of programming!

Casino de Monte Carlo in Monaco.