It is possible to analyze a set of random events from a deterministic state and also to analyze a deterministic state from a set of random events. Studying random events have applications both ways.

---

random-numbers:~# Unless an algorithm is inherently quantum mechanical, it is only ever pseudorandom at best governed by deterministic algorithms. Hence, one can introduce a seed and reproduce the same set of “random” numbers.

```
x = (a*x+c)%m          # Linear congruential RNG uses three parameters

random()               # The random library uses Mersenne twister algorithm RNG

seed(p)                # Seeding a program re-generates exact random numbers
```

! Pitfall: Never use above codes for cryptography. There exist more sophisticated algorithms and libraries for that.

monte-carlo-integration:~# Useful for higher dimensional integration, Monte Carlo method non-deterministically computes integral by analyzing the fraction of randomly-generated numbers inside the domain.

```
    if y<f(x):         # (Hit-or-miss method) First line indentation
    count += 1         # implies a preceeding loop for randomly
  I = A*count/N        # generating x and y. A refers to the area of
                       # the inscribing box.

    y_sum += f(x)      # (Mean value method) Similar reason for
  I = (l*(y_sum))/N    # indentation.  l denotes interval length.
```

! Pitfall: Perfect for pathological functions but generally an inferior method in terms of efficiency. Be mindful of singularities when generating random numbers as a single run may not detect it immediately.

markov-chain:~# Markov chain Monte Carlo simulation uses Metropolis algorithm governing the acceptance probability of a random move and adding up relevant quantities with each move.

```
if n[i,j]>1 or dn == 1:        # Acceptance condition of a move using the
  if random()<exp(-dE/T):      # Metropolis probability. This is a code snippet
    n[i,j] += dn               # of a quantum system simulation with states
    E += dE                    # n[i,j] and energy E . The move was a discrete
                               # n step and discrete energy change
```

! Pitfall: Make sure that move sets are ergodic. Also, even in a “no-move” step, make sure to still add up the relevant quantity.