Cryptographic Random Numbers

random numbers
- a sequence of uniformly distributed numbers over a defined set
- predicting future values based on what has been observed in the past is impossible

in computers?
- generating, impossible
- random number generators don't produce a random sequence because of their deterministic nature
- if something is predictable, it cannot be random

types of random numbers
- pseudo-random sequences
  - a sequence of random numbers that looks random
  - and passes all the statistical tests of randomness
  - but these numbers are not safe for cryptographic purposes
- cryptographically secure pseudo-random sequences
  - pseudo-random and unpredictable
  - given the algorithm, hardware, all previous bits in the stream, etc
    - it is still infeasible to predict the next outcome
  - these numbers are considered cryptographically secure
- real random sequences
  - cryptographically secure pseudo-random and cannot be reproduced
  - computers cannot produce a real random sequence of numbers
  - usually, some external data (e.g., radioactive decay information, mouse movement, etc) is used
    - e.g., lava lamps at Cloudflare
      - https://blog.cloudflare.com/randomness-101-lavarand-in-production/

random numbers are heavily used in cryptography
- especially for key generation
- the security of most cryptosystems depends on the quality of the random number sequence it uses

seed
- an integer that acts as the starting point to generate a series of random numbers
- a unique seed produces a unique sequence of random numbers
  - that is repeatable!
  - therefore, not secure for cryptographic purposes
  - how/why is this useful?

philosophical question: does anything truly random even exist?

Python examples