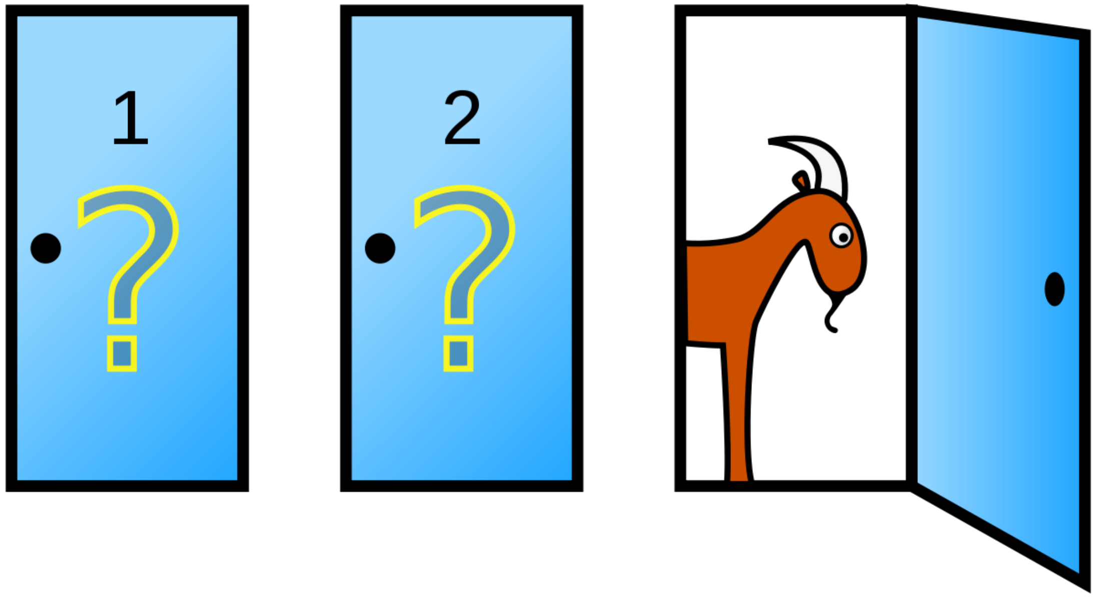
Programming in Python

simple simulations
lecture 8

Department of Cybernetics and Artificial Intelligence
Technical University of Košice
Ing. Ján Magyar, PhD.

Monty Hall Problem

- In a game show there are three doors in front of us, behind one is the main price (a car), behind the others there is a goat. We select one door and another one with a goat behind it is opened. We are given the chance to change our selection.
- Question: Should we swap? Do we increase our chance of victory?



Birthday paradox

- We have *n* students in a room, we want to find out the possibility of (at least) two of them having their birthday on the same day.
- At what value *n* will the probability of shared birthdays be greater than 50%?
- At what value *n* will the probability of shared birthdays be greater than 99%?
- At what value *n* will the probability of shared birthdays be greater than 99.9%?

Lightning paradox

In a city lightning strikes once every 30 days on average, with the same probability for each day. Today lightning struck. On what day is it most likely that it will strike again?

Modelling and simulation

- our goal is to find solutions to problems, which we cannot describe analytically
- often we cannot test each possible output, we simulate only a representative sample
- the result is descriptive, not prescriptive
- the simulation doesn’t give us the result, only describes it

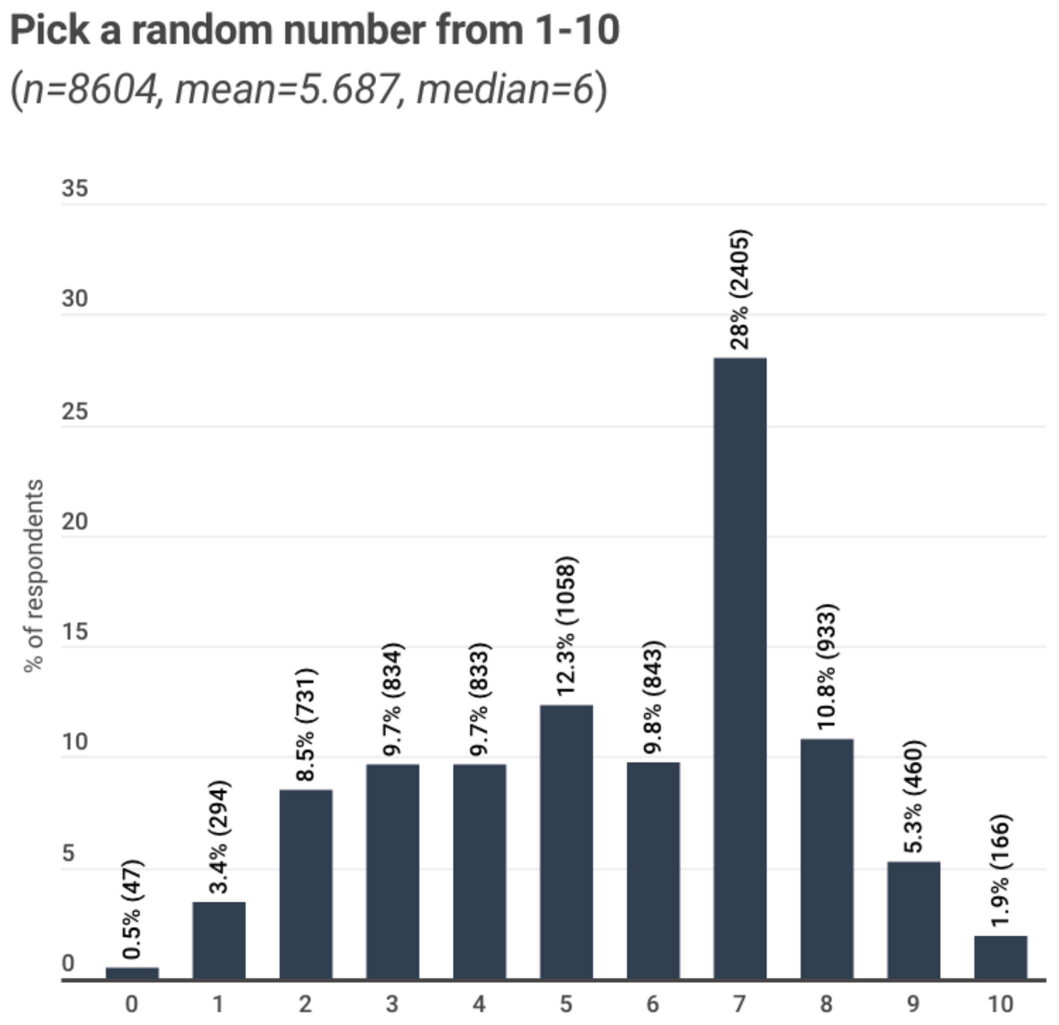
Steps when designing a simulation

1. formulating the question – define the question you want to answer
2. formulating the hypothesis – what do you expect?
3. defining the abstraction and creating computational models
4. (data processing)
5. executing simulations
6. evaluating results
7. evaluating result quality

Computational model

- program representation of the modelled phenomenon or world
- the code itself is not that important
- our goal is to model the real world, we should start with a simple model and keep expanding it iteratively as needed
- we usually use randomness in computation models – stochastic models
- the basic problem with randomness: computers are deterministic machines, how can we generate random numbers?

Can people generate random numbers?



Randomness

- it makes sense to talk about a sequence of random numbers only
- computers generate pseudorandom numbers – they appear to be randomly generated, but in reality we get them using a deterministic algorithm
- the input is usually system time (in milliseconds), or CPU temperature, etc.
- we can generate random numbers – requires input from users (e.g. encoding key)

Randomness in Python

- implemented in `random`
- by default using system time (in `ms`)
- can use OS-specific sources of randomness
- `random.seed(a=None, version=2)`
  - initializes the random value generator
  - `a` – generation parameter
  - `version` – interval of seed values
  - support result recreation

Generating integers

- `random.randint(a, b)`
    - random integer `N`, where `a <= N <= b`

- `random.randrange([start, ] stop[, step])`
    - random element from `range(start, stop, step)`
    - does not generate a `range` object
    - more uniform than `int(random.random() * n)`

Sequence randomness

- `random.choice(seq)`
  - random element from a sequence
- `random.choices(population, weights=None, cum_weights=None, k=1)`
  - selects `k` elements from a population and returns them in a list
  - `weights/cum_weights` – defines the probability of selecting individual elements
- `random.shuffle(x[, random])`
  - shuffles sequence order in a random way
  - if the sequence is immutable, we can use `sample`
- `random.sample(population, k)`
  - selects `k` unique elements of a sequence
  - returns a new list with the values

Generating floating point numbers

- `random.random()`
  - random float from the interval `[0.0, 1.0)`
- `random.uniform(a, b)`
  - random float from the interval `[a, b]` or `[b, a]`
- `random.gauss(mu, sigma)`
  - random number from a normal distribution
  - `mu` – most probable value
  - `sigma` – standard deviation
- `random.expovariate(lambd)`
  - random number from an exponential distribution
  - values are from the interval `[0, ∞)` if `lambd > 0`, `(-∞, 0]` if `lambd < 0`

Simulating probability in Python

- How can we implement a phenomenon with an 80% probability?

```
if random.random() < 0.8:
    # simulate phenomenon
```

Conclusion

- modelling and simulation
- designing a simulation
- computational models
- basic functions in `random`
- simulating probability