

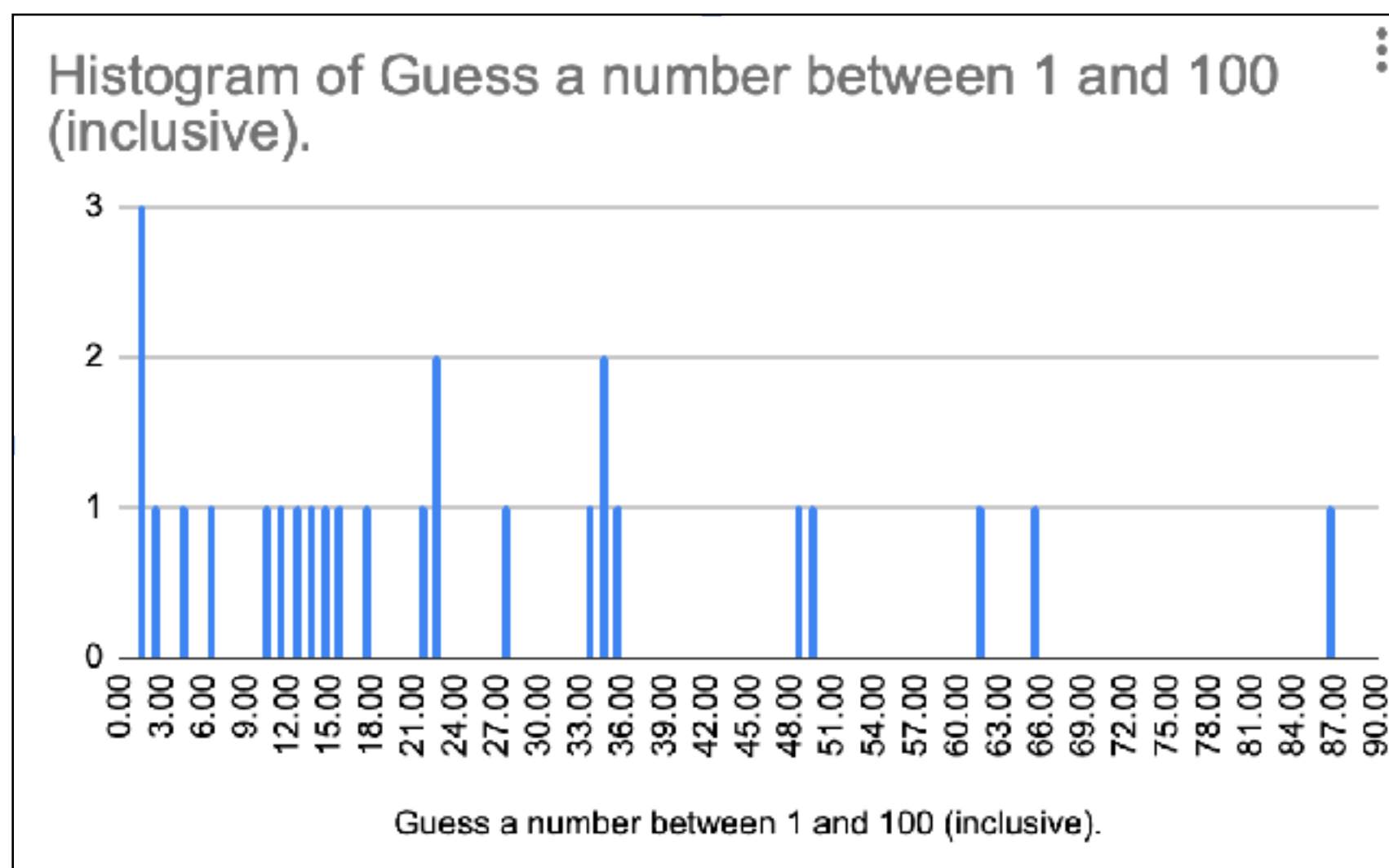
Guess a number between 1 and 100 (inclusive).

*

We will average everyone's guesses before the tutorial, and then compute $\frac{2}{3}$ of the average guess.

The person who guesses closest to $\frac{2}{3}$ of the average guess will win a bag of super-special memo-themed m&ms (we call them... m&mos!!).

Short answer text



Average: 24.8

$\frac{2}{3}$ of avg: 16.5

**Winner: Arthur Le Pargneux
(with a guess of 17)**

Anatomy of a **memo** model

signal start of
memo code
using special
memo syntax

axes of output array
to be computed

scalar free parameters

@memo

def my_model[x: X, y: Y](a, b, c):

 alice: . . .

 bob: . . .

 . . .

return . . .

sequence of **statements**
about agents in the model

expression whose value
to compute for each cell
in returned array
(here, for each x and y)

data types & helper functions
(written in *ordinary* Python, not memo)

X = [1, 2, 3]
Y = range(100)

def f(a):
 return a + 1

the **chooses** statement (like "sample" in WebPPL)

Agent making choice

Domain of choice (list, enum, or array)

↓
bob: **chooses**(a in Actions, wpp=<expression of a>)

↑
Name of choice

↑
"With probability proportional to"

expressions

+, −, *, /
exp(...), **log**(...), **abs**(...)
gaussianpdf(...), ...

other agents'
choices

expectation

↓
E[alice.x + bob.z]

probability

Pr[alice.y >= 0]

the **thinks** statement

Agent doing the thinking

↓
bob: **thinks**[
 alice: chooses(...),
 charlie: chooses(...),
 ...]

↑
What that agent thinks

the **observes** statement (like "condition" in WebPPL)

Agent observing

Choice being observed (square brackets are
a mnemonic for "someone else's choice")

↓
bob: **observes** [alice.x] is y

↑
What the choice is observed to actually be.

**[https://daeh.github.io/
memo-demo/](https://daeh.github.io/memo-demo/)**