Statistics

A set of tools for understanding data and making decisions/conclusions/predictions under uncertainty

Randomness

- in the short term, we don't know what will happen (flipping a coin)
- in the long term, we know the **distribution** of possibilities (what outcomes are possible and how often they occur)
  - the crux of randomness

Two definitions of probability

- **proportion** of times an outcome occurs or would occur over infinitely many repetitions of a random action
  - **Frequentist**
  - math is a lot nicer
  - there is a fixed outcome but we don't know it
- a number quantifying our **belief** that an outcome can/will occur
  - **Bayesian**
  - 2000 times more intuitive and math is just as hard
  - random outcome (not fixed)

Both are calculus based

This might be a difference in how the solution is solved (iterative vs recursive)

Probability Model

Consists of two parts:

- sample space: list (set/list of all unique values) of all possibilities
- probability of each outcome

An **event** is an arbitrary set of 0 or more outcomes in a sample space

Axioms of Probability

- axioms : something is so obvious it does not need to be proven
- sample space must be well defined
- For events A and B in the same sample space denoted as “S”:
  - The probability of event A, denoted as P(A) is a number between 0 and 1 (inclusive). [0, 1] notation as well.
    - NOTE: P(A) = 0 means A is “impossible” and P(A) = 1 means A is guaranteed
  - P(S) = 1
    - Some outcome is bound to happen
  - If A and B are disjoint (there are no common outcomes. A is not in B AND B is not in A), then P(A or B) = P(A) + P(B)

Simple rules that follows from the Axioms

- Compliment Rule: Define A^C = A compliment, that is A^C is the event “A does not occur”
  - $P(A^C)$ = 1 - P(A)
- General addition rule: Suppose events A and B have at least one common outcome
  - Define A cap B to be the set of outcomes common to A & B
  - Define A cup B to be the set of outcomes in A, or in B or in both A & B
    - Then (P A cup B) = P(A) + P(B) - P(A cap B)

Example

Random phenomenon: Draw 1 tile from a standard Scrabble bag of 100 tiles

- Sample space 1 (option one):
  - S = the 100 tiles in the bag
  - All tiles are equally likely to be drawn
  - P(draw particular tile) = 1/100 or 0.01 for all
- Sample space 2 (option two):
  - The 27 “letters” (26 letters and 1 blank)
- Let event C = “draw a letter in CAT”
- Let event D = “draw a letter in PET”

P(C) = P(draw a C) + P(draw an A) + P(draw a T) = .02+.09+.06 = 0.17
P(D) = P(draw a P) + P(draw an E) + P(draw a T) = 0.2+0.12+0.06 = 0.2

P(C^C) = P(do not draw any of the letters in CAT) = 1 - P(C) = 1 - .17 = .83
P(C cap D) = P(draw a letter inn **both** CAT & PET) = P(draw a T) = 0.06
P(C cup D) = P(draw a letter in CAT or PET or both words) = P(C) + P(D) - P(C cap D) = 0.17 + 0.2 - 0.06 = 0.31

```
#!/usr/bin/env python3.5

# probability can be calculated by using a hash table in conjunction with a set
# hash tables are used when there are two different letters with the same probability
# using a bare list would result in incorrect calculations of probability
# they would be treated as non unique instances
# in turn allowing for it to filter out needed objects
# this boils down to a set of unique hash tables and summing uo

class hashabledict(dict):
    def __hash__(self):
        return hash(tuple(sorted(self.items())))
value_mapping = {
    "c": 0.02,
    "a": 0.09,
    "t": 0.06,
    "p": 0.02,
    "e": 0.12
}

def get_probability(*args):
    # s has extra new line for code to fit
    s = set((hashabledict({letter: value_mapping[letter]})
        for argument in args for letter in argument))
    return sum([sum(dictionary.values()) for dictionary in s])


print(get_probability("cat", "pet"))
# this some times yields 0.310000000000005 and 0.3099999999999994
# which is essentially the same number
```