

CS109: Probability for Computer Scientists

Problem Set #1 is out

newish

The screenshot shows a web-based programming assignment interface. On the left, a sidebar lists "PS1" and "5. Random Choice". Below this, numbered circular buttons from 1 to 11 are shown, with 1, 2, 3, 7, 8, 9, and 10 having green checkmarks. At the bottom of the sidebar are icons for "Previous Question", "Next Question", and navigation arrows.

The main content area displays a question titled "5. Random Choice". The question asks: "What is the probability that both users will get the same randomly generated password? Provide an answer to three decimal places!"

Below the question is a code editor containing Python code:

```
import random

def main():
    user_1_password = generate_password()
    user_2_password = generate_password()

def generate_password():
    part_1 = random.choice([
        'red',
        'funky',
        'smelly'
    ])
    part_2 = random.choice([
        'apple',
        'pear',
        'pineapple'
    ])
    return part_1 + '-' + part_2
```

To the right of the code editor is an "Answer Editor" section. It includes a "Numeric Answer" field set to "97", a "Check Answer" button, and an "Explanation" text area containing the text "I am so excited! What a good time!". A blue arrow points from the text "Check your answer" to the "Check Answer" button.

A blue arrow also points from the text "Insert LaTeX" to the "Explanation" text area.

The word "Auto Submission" is written in blue text on the left side of the screenshot.

Check your answer

Insert LaTeX



Write an Agent newish

The screenshot shows a web-based development environment for a programming assignment. The title bar indicates it's "PSet 1" and the URL is "cs109psets.netlify.app/win22/pset1/countingcards".

PS1 sidebar:

- Home icon
- 1 ✓ (green checkmark)
- 2 ✓ (green checkmark)
- 3 ✓ (green checkmark)
- 4
- 5
- 6
- 7 ✓ (green checkmark)
- 8 ✓ (green checkmark)
- 9 ✓ (green checkmark)
- 10 ✓ (green checkmark)
- 11
- Flag icon
- User icon
- Next Question icon

9. Counting Cards section:

Counting cards refers to when a player keeps track of what cards have already been played during a card-game, in order to have a better estimate of how likely they are to win. Counting cards was successfully used by probability students from MIT to beat casinos worldwide: [MIT Blackjack Team](#) a heist which was popularized by the movie [21](#). The key to counting cards in blackjack is to keep track of the probability of high cards.

In this problem we are going to consider a simpler game called High Card played on a standard 52 card deck. The game works as follows: You decide if you want to play. If you do, the casino deals you a single card. If the card is a high card, (10, Jack, Queen, King or Ace), you win \$20. If it is not, you lose \$20. Another player is playing as well and each game they will play (thus revealing a card). You can play even if you have negative dollars (we assume you will never go bankrupt).

Answer Editor tab is selected. **Solution** tab is available.

Agent Code:

```
1 """
2 counting_agent.py
3 This file defines an agent "counting_agent" which plays the game of
4 High Card. The function gets called each time it is the agents turn.
5 The cards_played list has all cards which have been played so far.
6 """
7
8 def counting_agent(cards_played, actions):
9     return 'play'
10
```

Run One Game and **Test Agent** buttons are present.

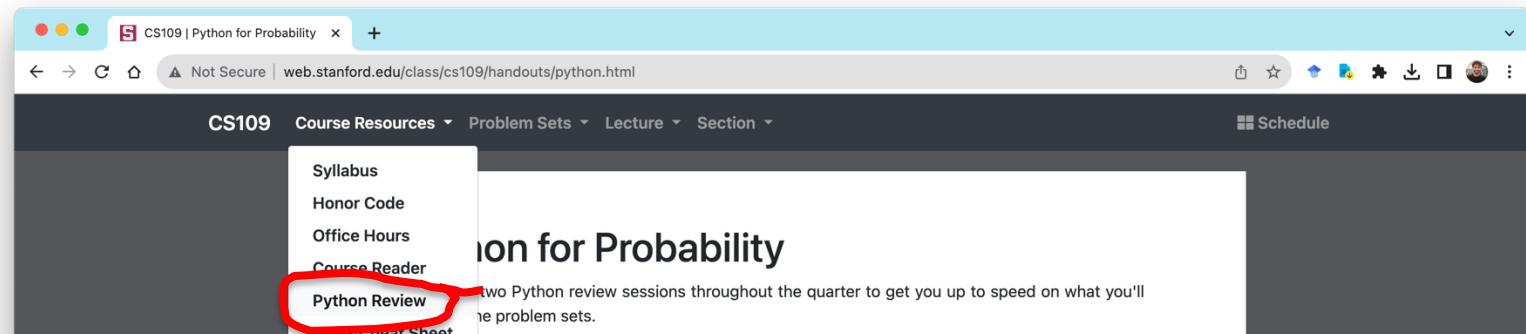
Console area is empty.



Python Review Session

Friday at 4:30-5:30pm PT, recorded

Find links, recordings, and setup here



Learn LaTex

```
1 \begin{aligned}
2 P(E) | \\
3 &= \sum_{i=0}^n e^i \\
4 &= 0.25 \\
5 \end{aligned}
```

$$P(E) = \sum_{i=0}^n e^i \\ = 0.25$$

Done



A screenshot of a web browser window titled "CS109 | Python for Probability". The URL is "Not Secure | web.stanford.edu/class/cs109/handouts/latex/". The page content is a LaTeX guide. On the left, there's a sidebar with links like Syllabus, Honor Code, Office Hours, Course Reader, Python Review, Latex Cheat Sheet (which is currently selected), and Fall 2022 Videos. The main content area has a title "LaTeX Guide" and a sub-section "What is LaTeX?". It explains that LaTeX is a typesetting system used for creating scientific documents and was invented at Stanford. It also mentions that while handwritten homework is still accepted, it's recommended to use LaTeX. Below this, there's a section with "Some Examples" and a tip about right-clicking equations to see TeX commands.



Inline LaTeX

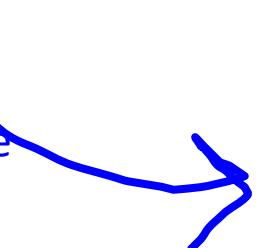
new

Explanation:

≡ Block LaTeX ✓ Inline LaTeX ⚡ Python Image

This is an example of inline latex. Let \$\$Y\$\$

When you type
the closing \$



Explanation:

≡ Block LaTeX ✓ Inline LaTeX ⚡ Python Image

This is an example of inline latex. Let Y

CS109 Course Resources Problem Sets Lecture Section Schedule

Syllabus Honor Code Office Hours Course Reader Python Review Latex Cheat Sheet Fall 2022 Videos Midterm Final

Quick Guide to LaTeX

by Roshini Ravi and Chris Piech

LaTeX is a typesetting system that creates beautiful scientific documents. It is the digital language of science, and it was invented right here at Stanford. You can still submit handwritten homeworks, but we encourage you to use LaTeX.

Some examples that should help you get started! As a helpful tip, you can access the LaTeX source code for any equation in the course reader by right clicking the equation and clicking "Show TeX Commands".



What Makes for a Good Answer

PS1



1 ✓

2 ✓

2l ✓

2c

3

4

5a

5b

6

7

8

9

10

11

12

Etude: Possible Answer Keys (a)

A true-false exam has 15 questions, and precisely three of the answers are false.



Figure: An example of a true-false test to be clear the questions are ordered and that each question can only be answered in one of two ways.

How many different answer keys are possible? In general, you should always show your work for all problems ever submitted for any problem on any problem set so we know how you arrived at your final answer.

Answer Editor

Solution

Answer editing: Off

Explanation:

There exist 15 questions, of which we need to choose 3 to be false and the remaining 12 to be true. The true questions will be indistinguishable from each other, and the false questions will similarly be indistinct. That means we select some subset of size 3 from a set of size 15, and that, by definition, is $\binom{15}{3} = 455$. You can also think of this count as all of the ways to order 15 objects, where each object is in one of two indistinct groups (trues and falses).



If you notice a bug?

It should be robust, but things can happen.

Let me know: send an email to cpiech@stanford.edu or message me on slack. I need your email and the approximate time you encountered the bug.



Honor Code

Always remember: You need to be able to recreate your ability on an exam. And in the real world. This is a foundation course.

Cheating in CS109 is cheating yourself and your friends.

Talk to your friends about the **concepts**, not the solution. Words must be your own.

Practice the **art of teaching**. Three most important things to know:

1. Do not give away the answer
2. Always be respectful
3. Know what you don't know



Improved Cadence

#	Weekday	Date	Topic	Notes
Week 1				
1	Monday	Jan 6	Counting	
2	Wednesday	Jan 8	Combinatorics	
3	Friday	Jan 10	What is Probability?	You are here
Week 2				
4	Monday	Jan 13	Conditional Probability and Bayes	
5	Wednesday	Jan 15	Independence	PSet 1 Due
6	Friday	Jan 17	Random Variables and Binomial	
Week 3				
-	Monday	Jan 20	No Class (MLK Jr Day)	
7	Wednesday	Jan 22	Moments	
8	Friday	Jan 24	Poisson	PSet 2 Due
Week 4				
9	Monday	Jan 27	Continuous Random Variables	
10	Wednesday	Jan 29	Normal Distribution	
11	Friday	Jan 31	Probabilistic Models	PSet 3 Due
Week 5				
12	Monday	Feb 3	Inference	PEP 1
13	Wednesday	Feb 5	Inference II	
14	Friday	Feb 7	General Inference	PSet 4 Due

Done with
counting early

Core probability
early

Random Vars

Chance to Practice
Inference



Late Policy (5 Late Days!)

Pset 1 - Counting for Probability
For Chris Piech

Get Started

Due Date: Wednesday, Jan 15, 10:00 PM Pacific Standard Time (in 8 days).

Grace Period Date: Wednesday, Jan 15, 11:59 PM Pacific Standard Time (in 8 days).

Solutions Posted: Friday, Jan 17, 11:59 PM Pacific Standard Time (in 10 days).

Late Day Extension
CS109 is a fast paced course and it will be extra work in the next few weeks to catch-up.

Extension: 1 Late Day (26 hours)
New Due Date: Thursday, Jan 16, 11:59 PM Pacific Standard Time (in 9 days)
Reason for request: I had a chance to go watch the sunrise with my friends at the beach and I thought that was pretty important.
Catch-up Plan: I will finish the pset tomorrow and then I will get an early start on the next pset

Extension Request Forms ▾

- Grace period extension
- 26 hour extension (1 Late Day)
- 50 hour extension (2 Late Days)
- Over 50 hour extension

Three types of extensions:

1. Grace period (2 hours)
2. Can take up to 2 late days (50 hours)
3. After the **hard** deadline (> 2 late days)

You give them to yourself. Need to talk to us if:

- A. You need more than 5 late days / qtr
- B. Extension past the hard deadline

But CS109 is a fast class. If you want a long extension I want you to be intentional about how you are going to catch up. Not all late days are created equal (especially before exams).



Section Signups Open Thursday



Class Theme Song Starts Thursday



Chose the top 16 freshest songs.
Song recommendations open
Thursday and close Sunday.

Then we start running a probabilistic
exploration algorithm that will give
you three songs at a time to rate.

We could have our top 16 songs by
the midterm...



**Above
& Beyond**



Review

CS109: From Counting to Machine Learning


Counting
Theory


Core
Probability


Random
Variables


Probabilistic
Models


Uncertainty
Theory


Machine
Learning



Core Counting

Counting with steps

Definition: Step Rule of Counting (aka Product Rule of Counting)

If an experiment has two parts, where the first part can result in one of m outcomes and the second part can result in one of n outcomes regardless of the outcome of the first part, then the total number of outcomes for the experiment is $m \cdot n$.



Scales nicely to many steps

Counting with “or”

Definition: Inclusion Exclusion Counting

If the outcome of an experiment can either be drawn from set A or set B , and sets A and B may potentially overlap (i.e., it is not the case that A and B are mutually exclusive), then the number of outcomes of the experiment is $|A \text{ or } B| = |A| + |B| - |A \text{ and } B|$.



Only scales nicely if “mutually exclusive”



Core Counting

Counting with steps

Definition: Step Rule of Counting (aka Product Rule of Counting)

If an experiment has two parts, where the first part can result in one of m outcomes and the second part can result in one of n outcomes regardless of the outcome of the first part, then the total number of outcomes for the experiment is $m \cdot n$.



Scales nicely to many steps



Fantastic Question

Counting with steps

Definition: Step Rule of Counting with Many Events

If an experiment has k parts, where the first step has n_1 outcomes and the i th step has n_i outcomes (regardless of the result of any earlier steps), then the total number of outcomes of the experiment is:

$$\text{Number of Outcomes} = \prod_{i=1}^k n_i$$

What does this
big pi mean?

↖
Scales nicely to
many steps



Fantastic Question

$$\sum_{i=1}^k n_i$$

```
total = 0  
for i in range(1, k+1):  
    total += n_i
```

$$\prod_{i=1}^k n_i$$

```
total = 1  
for i in range(1, k+1):  
    total *= n_i
```

Sometimes we
are lazy...



$$\prod_i n_i$$



How Many Bit Strings?

Problem: A 6-bit string is sent over a network. The valid set of strings recognized by the receiver must either start with "01" or end with "10". How many such strings are there?

Answer

$$\begin{aligned}N &= |A| + |B| - |A \text{ and } B| \\&= 16 + 16 - 4 \\&= 28\end{aligned}$$

2^4 start with 01

010000
010001
010010
010011
010100
010101
010110
010111
011000
011001
011010
011011
011100
011101
011110
011111

Set *A*

2^4 end with 10

000010
000110
001010
001110
010010
010110
011010
011110
100010
100110
101010
101110
110010
110110
111010
111110

Set *B*



End Review

BOBA

- How many *different* orderings of letters are possible for the string **BOBA**?



Permutations I

Orderings of Letters

How many letter orderings are possible for the following strings?

1. CHRIS

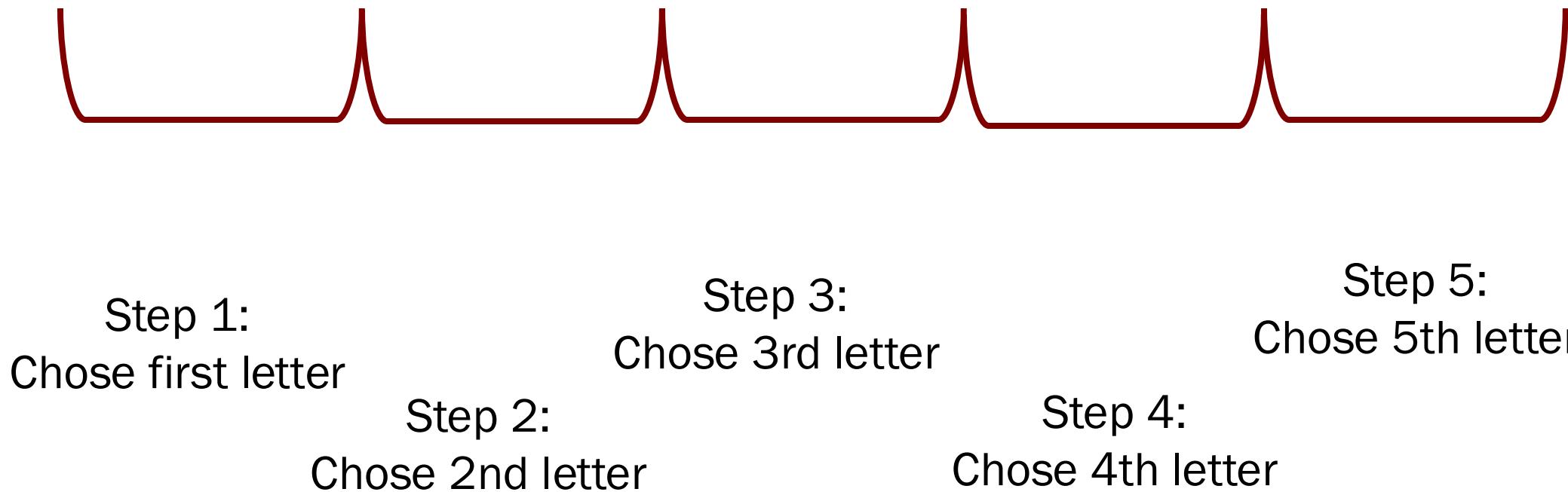
This is Jerry's dog, Doris. She puts her little Doris paw up to her chin when she's thinking.



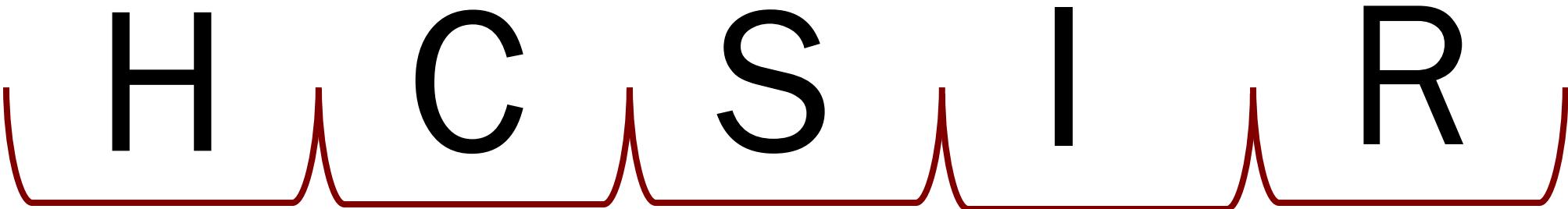
Orderings of Letters

chirs	crish	hicrs	hsirc	irchs	rcish	rschi	shirc
chisr	crshi	hicsr	hsric	ircsh	rcshi	rscih	shrci
chris	crsih	hircs	hsric	irhcs	rcsih	rshci	shric
chrsi	cshir	hirsc	ichrs	irhsc	rhcis	rshic	sichr
chsir	cshri	hiscr	ichsr	irsch	rhcsi	rsich	sicrh
chsri	csihr	hisrc	icrhs	irshc	rhics	rsihc	sihcr
cihrs	csirh	hrcis	icrsh	ischr	rhisc	schir	sihrc
cihsr	csrhi	hrcsi	icshr	iscrh	rhsci	schri	sirch
cirhs	csrih	hrics	icsrh	ishcr	rhsic	scihr	sirhc
cirsh	hcirs	hrisc	ihcrs	ishrc	richs	scirh	srchi
cishr	hcisr	hrsci	ihCSR	isrch	ricsh	scrhi	srcih
cisrh	hcris	hrsic	ihrcs	isrhc	rihcs	scrih	srhci
crhis	hcrsi	hscir	ihrsc	rchis	rihsc	shcir	srhic
crhsı	hcsir	hscrı	ihscr	rchsi	risch	shcri	srich
crihs	hcsri	hsicr	ihsrc	rcihs	rishc	shicr	srihc

Orderings of letters



Orderings of letters



Step 1:
Choose first letter
(5 options)

Step 2:
Choose 2nd letter
(4 options)

Step 3:
Choose 3rd letter
(3 options)

Step 4:
Choose 4th letter
(2 options)

Step 5:
Choose 5th letter
(1 option)

Permutations

A **permutation** is an ordered arrangement of objects.

The number of unique orderings (**permutations**) of n distinct objects is
 $n! = n \times (n - 1) \times (n - 2) \times \cdots \times 2 \times 1$.

Unique 6-digit passcodes with **six** smudges



How many unique 6-digit passcodes are possible if a phone password uses each of **six** distinct numbers?



Unique 6-digit passcodes with **six** smudges



How many unique 6-digit passcodes are possible if a phone password uses each of **six** distinct numbers?

$$\text{Total} = 6! = 720 \text{ passcodes}$$

What if you had no smudges, but you knew it was still 6 digits long?

$$\text{Total} = 10^6 = 1 \text{ million passcodes}$$

Permutations II

Summary of Combinatorics

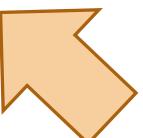
Counting tasks on n objects

Sort objects
(permutations)

Choose k objects
(combinations)

Put objects in r
buckets

Distinct
(distinguishable)



Summary of Combinatorics

Counting tasks on n objects

Sort objects
(permutations)

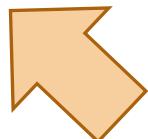
Choose k objects
(combinations)

Put objects in r
buckets

Distinct
(distinguishable)

Some
distinct

$n!$



Unique Bit Strings

1,0,1,0,0



Sort n distinct objects



Sort n distinct objects



Ayesha



Tim



Irina



Joey



Waddie

Sort n distinct objects



Steps:

1. Choose 1st can 5 options
2. Choose 2nd can 4 options
- ...
5. Choose 5th can 1 option

$$\begin{aligned}\text{Total} &= 5 \times 4 \times 3 \times 2 \times 1 \\ &= 120\end{aligned}$$



How many ways can we sort coke cans!



Coke1



Coke0



Coke1



Coke0



Coke0

Sort n distinct objects



Ayesha



Tim



Irina



Joey



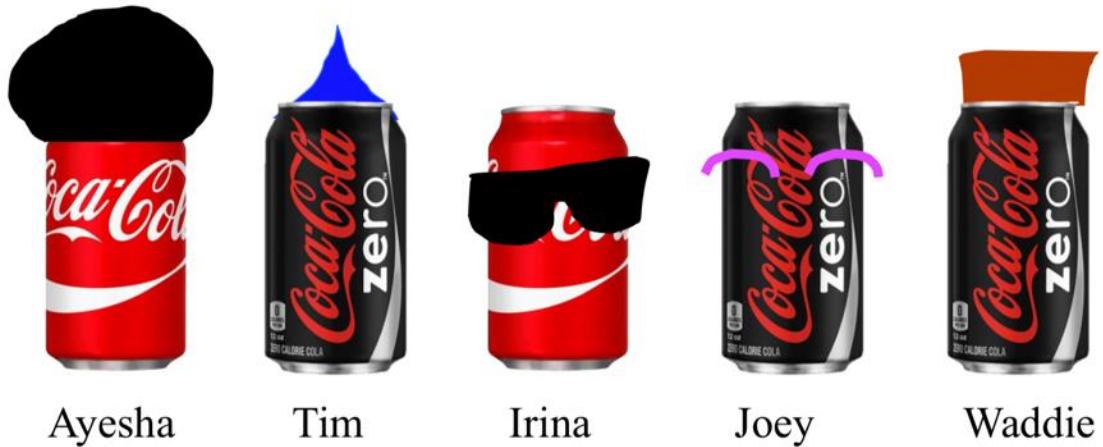
Waddie

of permutations =

Sort semi-distinct objects

Order n
distinct objects $n!$

All distinct



Some indistinct



Sort semi-distinct objects

How do we find the number of permutations considering some objects are indistinct?

By the product rule of counting (aka step rule), permutations of distinct objects is a two-step process:

$$\text{permutations of distinct objects} = \text{permutations considering some objects are indistinct} \times \text{Permutations of just the indistinct objects}$$

Sort semi-distinct objects

How do we find the number of permutations considering some objects are indistinct?

By the product rule of counting (aka step rule), permutations of distinct objects is a two-step process:

permutations
of distinct objects

=

permutations
considering some
objects are indistinct

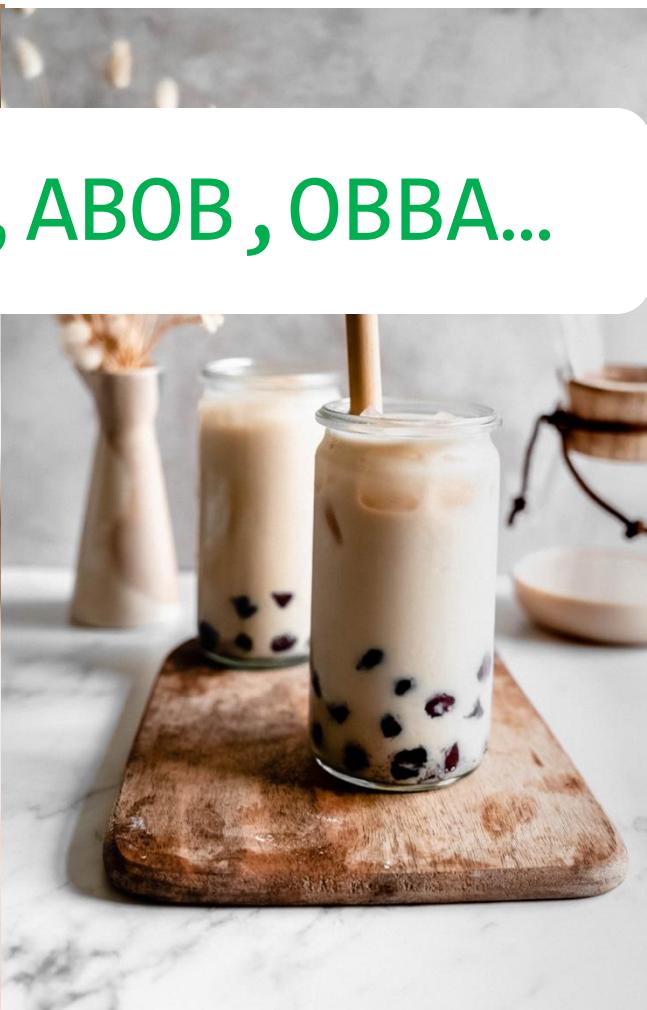
Permutations
of just the
indistinct objects

BOBA

How many *different* orderings of letters are possible for the string **BOBA**?



BOBA, ABOB, OBBA...



Strings

Is this just a regular permutation? $4! = 24$ unique permutations!

boba	bboa
boab	bbao
bboa	boba
bbao	boab
baob	babo
babo	baob
obba	abob
obab	abbo
obba	aobb
obab	aobb
oabb	abbo
oabb	abob



Strings

Is this just a regular permutation? $4! = 24$ unique permutations!

boba	bboa
boab	bbao
bboa	boba
bbao	boab
baob	babo
babo	baob
obba	abob
obab	abbo
obba	aobb
obab	aobb
oabb	abbo
oabb	abob



Strings

Is this just a regular permutation? $4! = 24$ unique permutations!

boba	bboa
boab	bbao
bboa	boba
bbao	boab
baob	babo
babo	baob
obba	abob
obab	abbo
obba	aobb
obab	aobb
oabb	abbo
oabb	abob



Strings

Is this just a regular permutation? $4! = 24$ unique permutations!

boba	bboa
boab	bbao
bboa	boba
bbao	boab
baob	babo
babo	baob
obba	abob
obab	abbo
obba	aobb
obab	aobb
oabb	abbo
oabb	abob



To the Code!

baob
bbao
obba
oabb
boab
bab0
abbo
aobb
boba
abob
bboa
obab

```
import itertools
```

```
def main():
    letters = ['b','o','b','a']
    perms = set(itertools.permutations(letters))
    for perm in perms:
        pretty_perm = "".join(perm)
        print(pretty_perm)
```

```
import math
```

```
def main():
    n = math.factorial(4)
    d = math.factorial(2)
    print(n / d)
```



General approach to counting permutations

When there are n objects such that

n_1 are the same (indistinguishable or **indistinct**), and

n_2 are the same, and

...

n_r are the same,

The number of unique orderings (**permutations**) is

$$\frac{n!}{n_1! n_2! \cdots n_r!}.$$

For each group of indistinct objects,
Divide by the overcounted permutations.

Sort semi-distinct objects

Order n semi-
distinct objects $\frac{n!}{n_1! n_2! \cdots n_r!}$

How many permutations?



Coke



Coke0



Coke



Coke0



Coke0

How many letter orderings are possible for the following strings?

1. BOBA
2. MISSISSIPPI

This is Jerry's dog, Doris. She puts her little Doris paw up to her chin when she's thinking.



Strings

Order n semi-distinct objects $\frac{n!}{n_1! n_2! \cdots n_r!}$

How many letter orderings are possible for the following strings?

1. BOBA $= \frac{4!}{2!} = 12$

2. MISSISSIPPI $= \frac{11!}{1!4!4!2!} = 34,650$

Unique 6-digit passcodes with **five** smudges

Order n semi-distinct objects $\frac{n!}{n_1! n_2! \cdots n_r!}$



How many unique 6-digit passcodes are possible if a phone password uses each of **five** distinct numbers?

Steps:

1. Choose digit to repeat 5 outcomes
2. Create passcode (sort 6 digits:
4 distinct, 2 indistinct)

$$\begin{aligned} \text{Total} &= 5 \times \frac{6!}{2!} \\ &= 1,800 \text{ passcodes} \end{aligned}$$

Summary of Combinatorics

Counting tasks on n objects

Sort objects
(permutations)

Choose k objects
(combinations)

Put objects in r
buckets

Distinct
(distinguishable)

Some
distinct

$$\frac{n!}{n_1! n_2! \cdots n_r!}$$

Combinations I

Summary of Combinatorics

Counting tasks on n objects

Sort objects
(permutations)

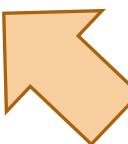
Choose k objects
(combinations)

Put objects in r
buckets

Distinct
(distinguishable)

Some
distinct

Distinct



$$\frac{n!}{n_1! n_2! \cdots n_r!}$$

Combinations with cake

There are $n = 20$ people.

How many ways can we choose $k = 5$ people to get cake?

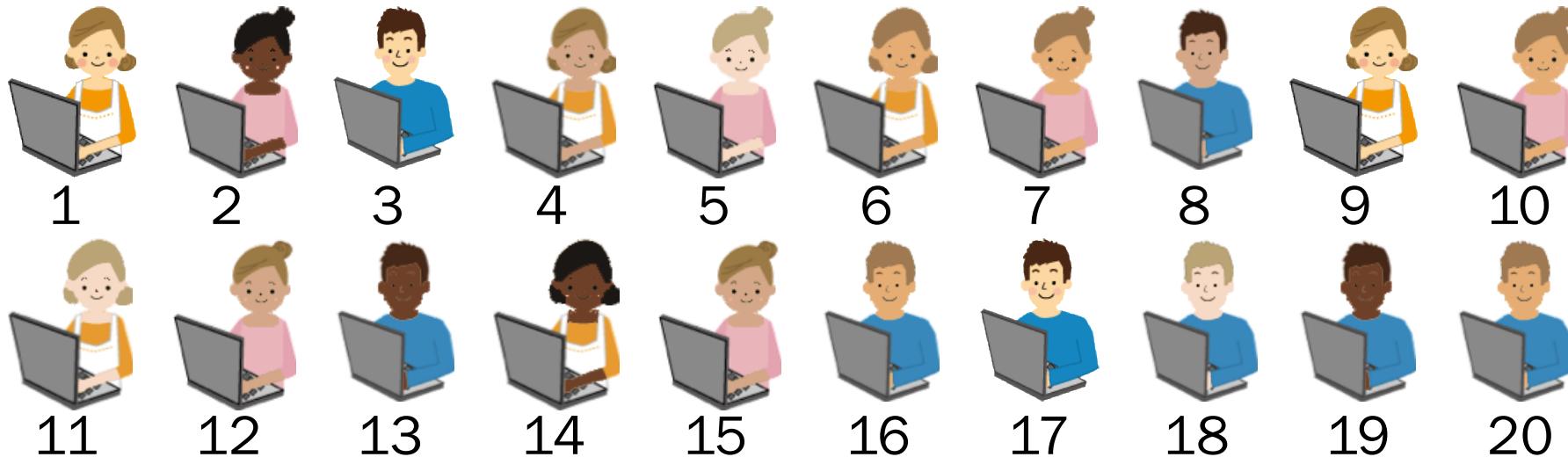


Consider the following
generative process...

Combinations with cake

There are $n = 20$ people.

How many ways can we choose $k = 5$ people to get cake?



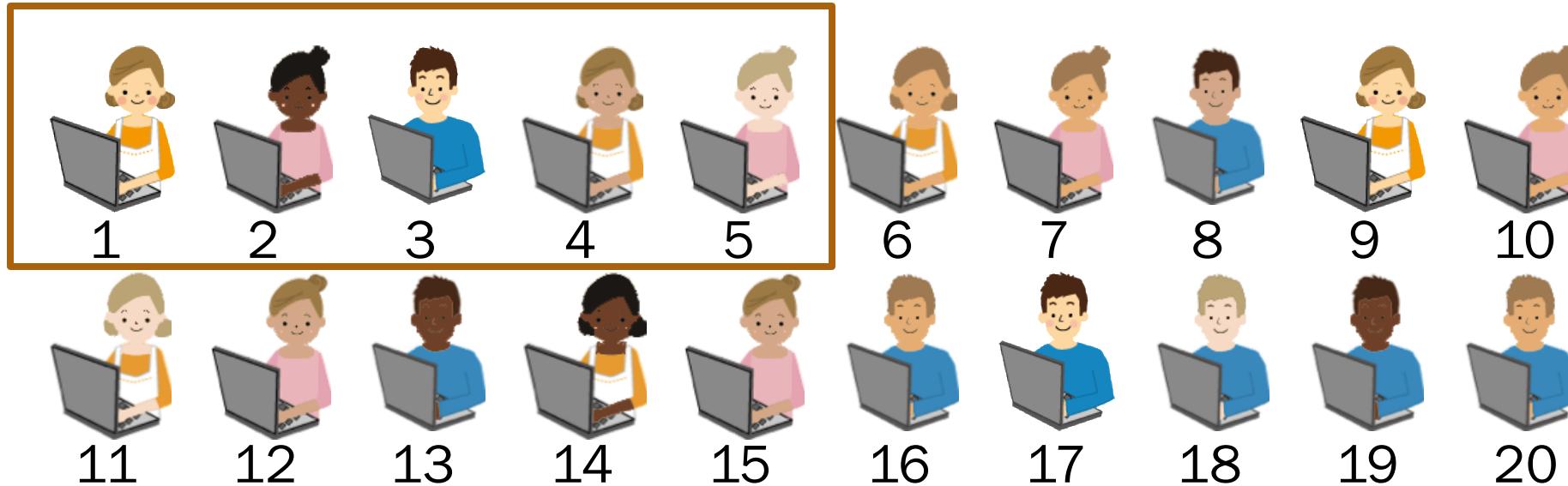
1. n people
get in line

$n!$ ways

Combinations with cake

There are $n = 20$ people.

How many ways can we choose $k = 5$ people to get cake?



1. n people
get in line

$n!$ ways

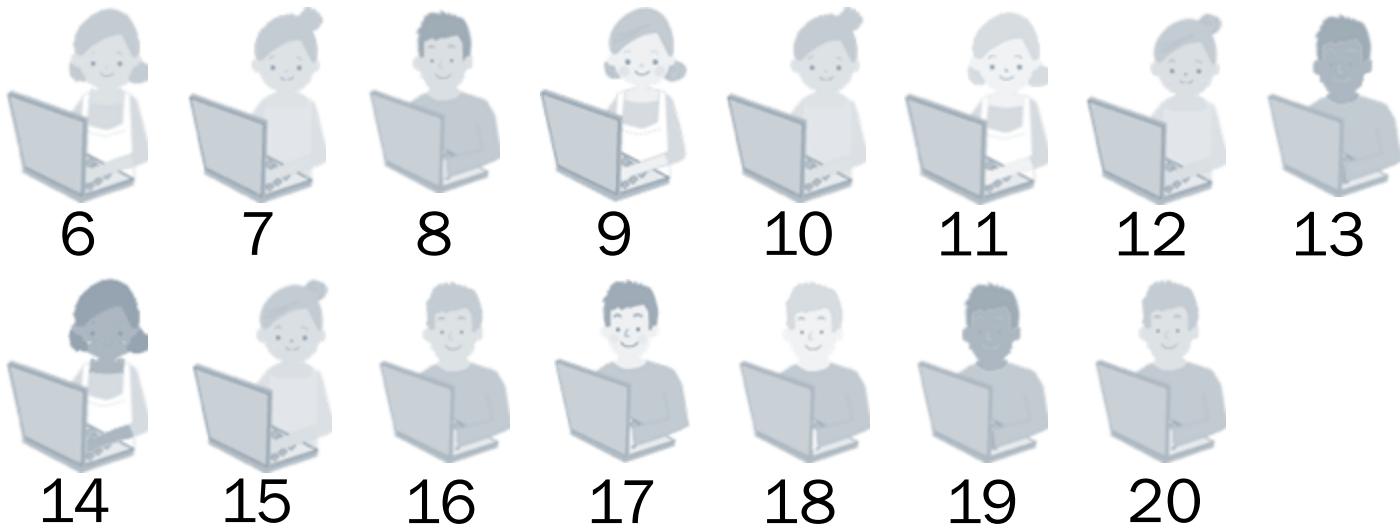
2. Put first k
in cake room

1 way

Combinations with cake

There are $n = 20$ people.

How many ways can we choose $k = 5$ people to get cake?



1. n people
get in line

$n!$ ways

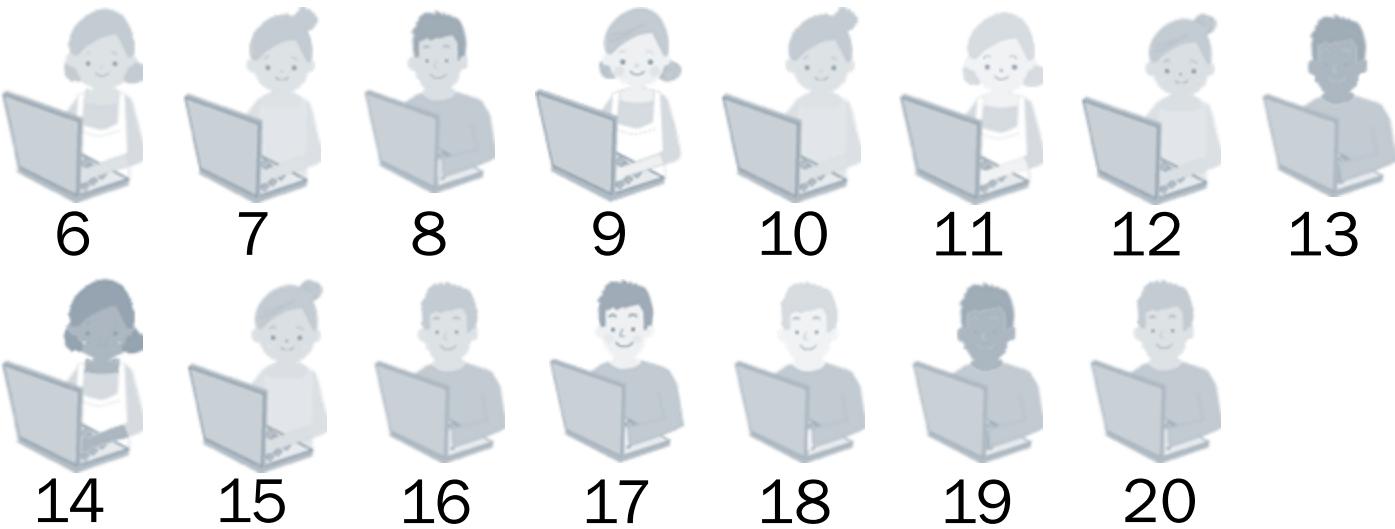
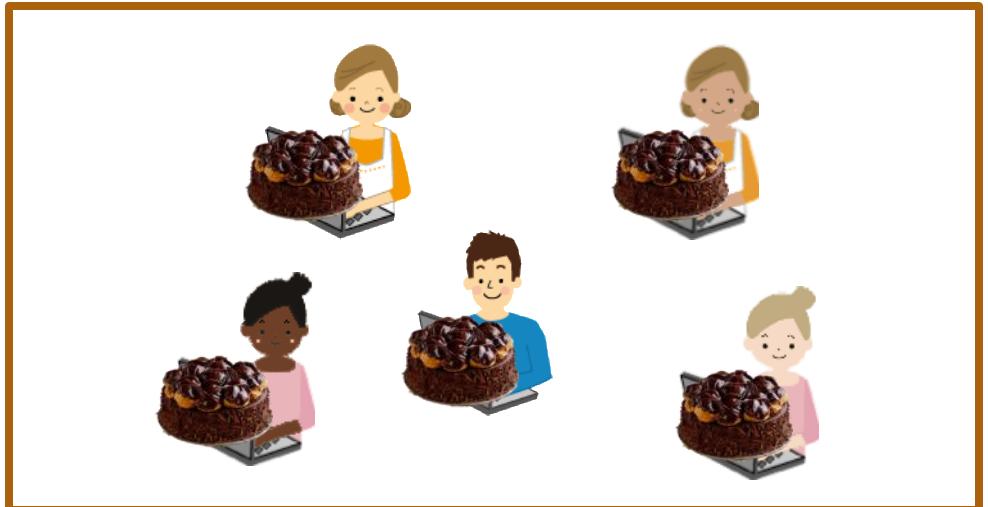
2. Put first k
in cake room

1 way

Combinations with cake

There are $n = 20$ people.

How many ways can we choose $k = 5$ people to get cake?



1. n people get in line

$n!$ ways

2. Put first k in cake room

1 way

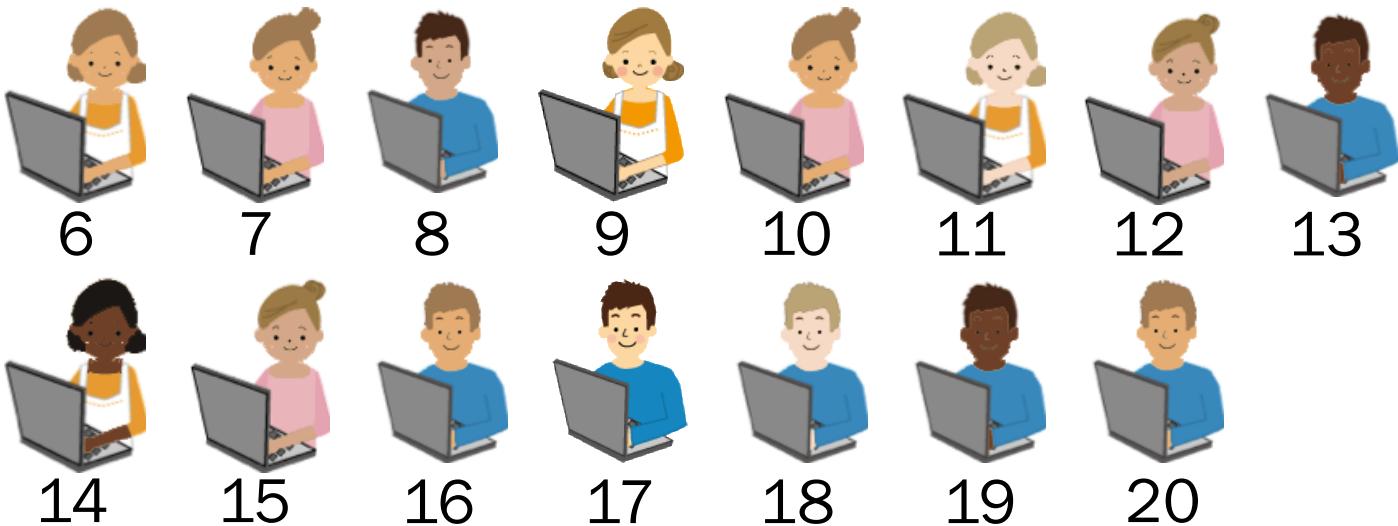
3. Allow cake group to mingle

$k!$ different permutations lead to the same mingle

Combinations with cake

There are $n = 20$ people.

How many ways can we choose $k = 5$ people to get cake?



1. n people get in line

$n!$ ways

2. Put first k in cake room

1 way

3. Allow cake group to mingle

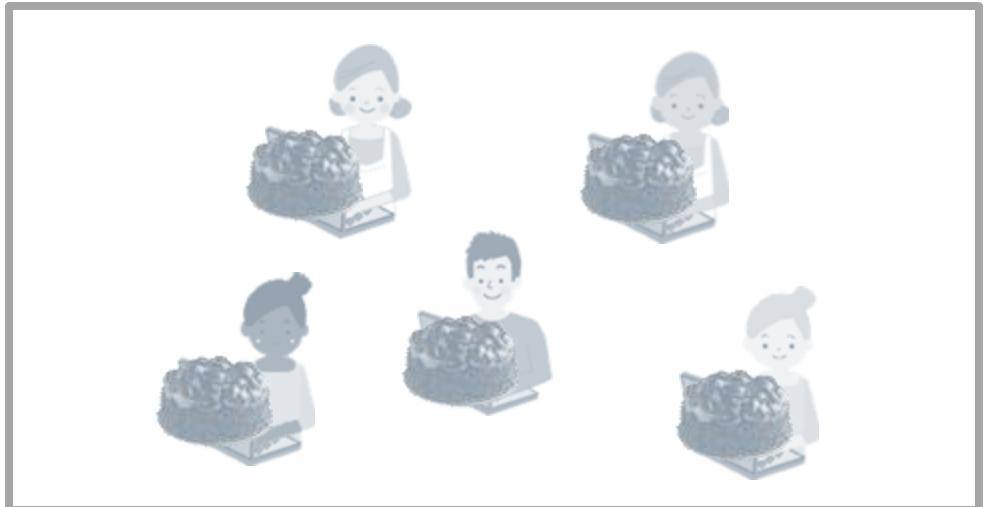
$k!$ different permutations lead to the same mingle

4. Allow non-cake group to mingle

Combinations with cake

There are $n = 20$ people.

How many ways can we choose $k = 5$ people to get cake?



1. n people get in line

$n!$ ways

2. Put first k in cake room

1 way



3. Allow cake group to mingle

$k!$ different permutations lead to the same mingle

4. Allow non-cake group to mingle

$(n - k)!$ different permutations lead to the same mingle

Combinations

A **combination** is an unordered selection of k objects from a set of n **distinct** objects.

The number of ways of making this selection is

$$\frac{n!}{k!(n-k)!} = n! \times 1 \times \frac{1}{k!} \times \frac{1}{(n-k)!}$$

1. Order n distinct objects

2. Take first k as chosen

3. Overcounted: any ordering of chosen group is same choice

4. Overcounted: any ordering of unchosen group is same choice

Combinations

A **combination** is an unordered selection of k objects from a set of n **distinct** objects.

The number of ways of making this selection is

$$\frac{n!}{k!(n-k)!} = n! \times 1 \times \frac{1}{k!} \times \frac{1}{(n-k)!} = \binom{n}{k} \quad \text{Binomial coefficient}$$

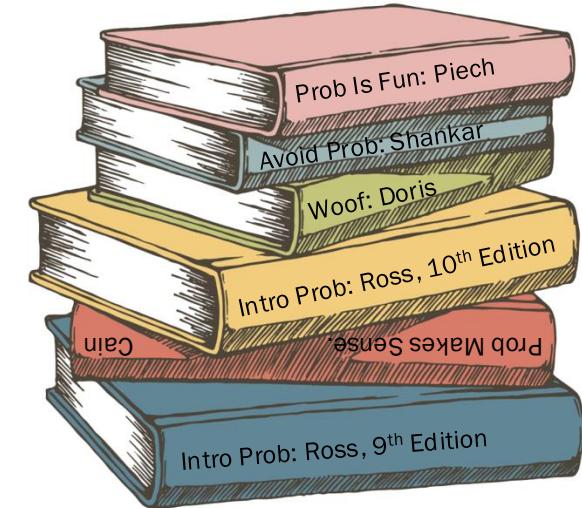
Fun Fact: $\binom{n}{k} = \binom{n}{n-k}$

Probability textbooks

Choose k of
 n distinct objects $\binom{n}{k}$

How many ways are there to choose 3 books from a set of 6 distinct books?

$$\binom{6}{3} = \frac{6!}{3! 3!} = 20 \text{ ways}$$



To the code!

How many unique hands of 5 cards are there in a 52 card deck?



```
def main():
    cards = make_deck()
    all_hands = itertools.combinations(cards, 5)
    for hand in all_hands:
        print(hand)
```

```
def main():
    total = math.comb(52, 5)
    print(total)
```

Buckets and The Divider Method

Summary of Combinatorics

Counting tasks on n objects

Sort objects
(permutations)

Choose k objects
(combinations)

Put objects in r
buckets

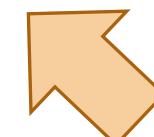
Distinct
(distinguishable)

Some
distinct

Distinct
 $\binom{n}{k}$

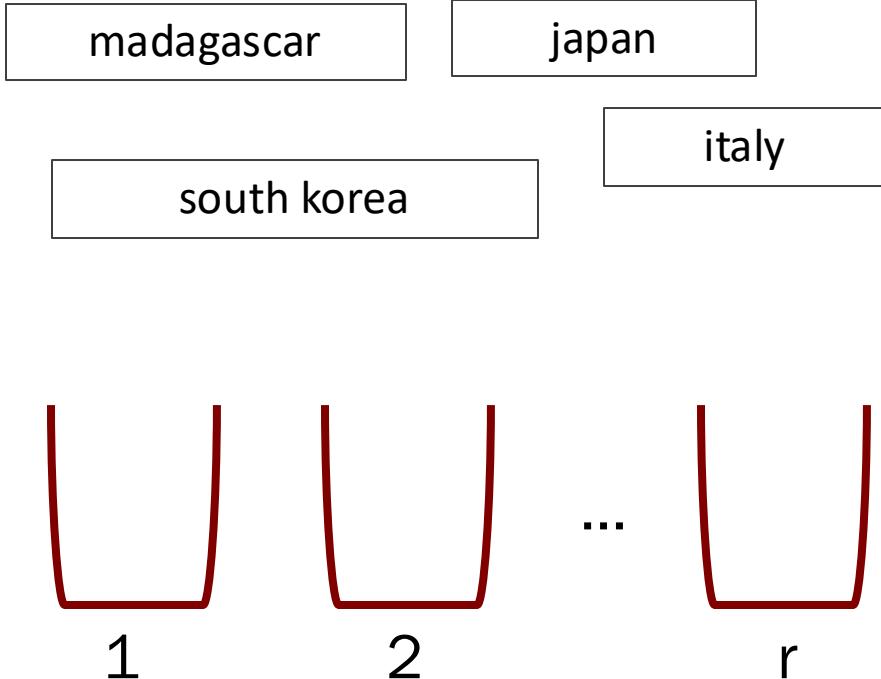
$$\frac{n!}{n_1! n_2! \cdots n_r!}$$

Distinct Indistinct



Balls and urns Hash tables and **distinct** strings

How many ways are there to hash n **distinct** strings to r buckets?



Steps:

1. Bucket 1st string
2. Bucket 2nd string
- ...
- n . Bucket n^{th} string

r^n outcomes

Summary of Combinatorics

Counting tasks on n objects

Sort objects
(permutations)

Choose k objects
(combinations)

Put objects in r
buckets

Distinct
(distinguishable)

Some
distinct

Distinct
 $\binom{n}{k}$

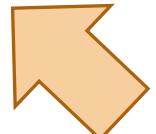
$n!$

$$\frac{n!}{n_1! n_2! \cdots n_r!}$$

r^n

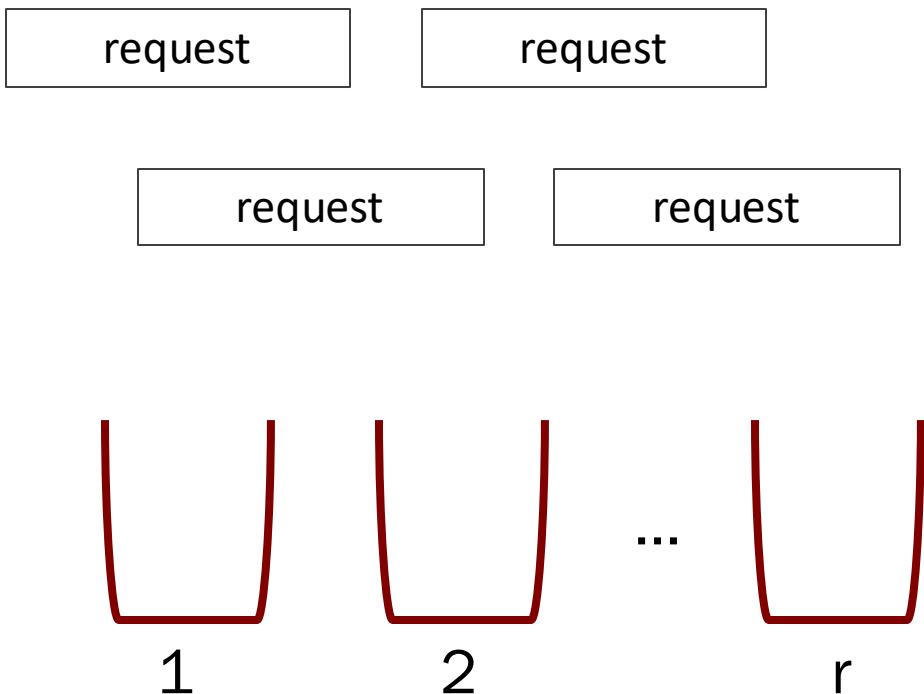
Distinct

Indistinct



Servers and **indistinct** requests

How many ways are there to distribute n **indistinct** web requests to r servers?



Goal

Server 1 has x_1 requests,
Server 2 has x_2 requests,

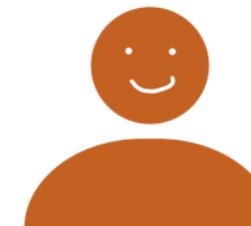
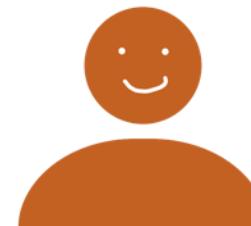
...

Server r has x_r requests

constraint: $\sum_{i=1}^r x_i = n$

Bicycle helmet sales

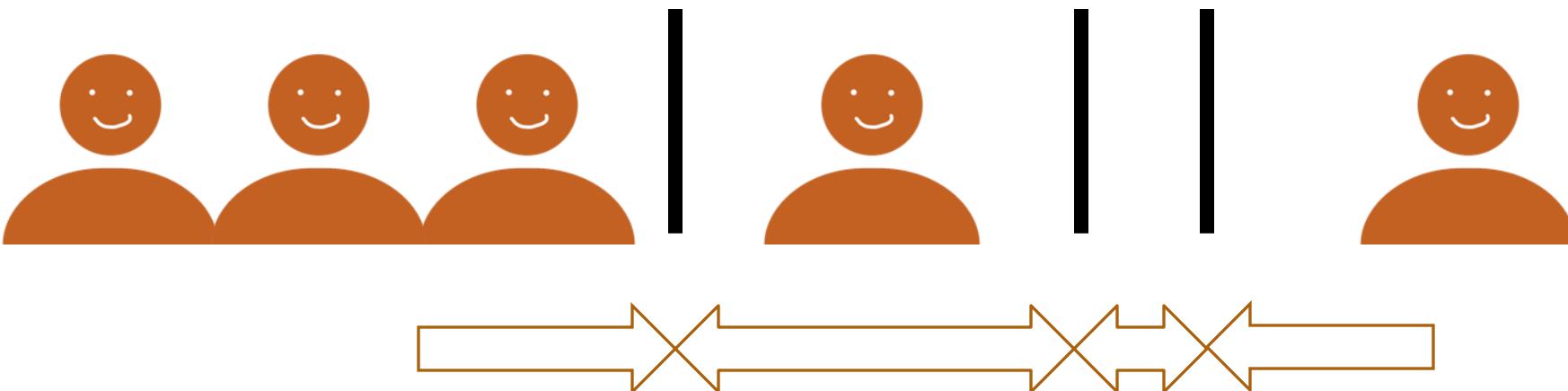
How many ways can we assign $n = 5$ indistinct children to $r = 4$ distinct bicycle helmet styles?



Bicycle helmet sales

1 possible assignment outcome:

Goal Order n indistinct objects and $r - 1$ indistinct dividers.



Consider the
following
generative
process...

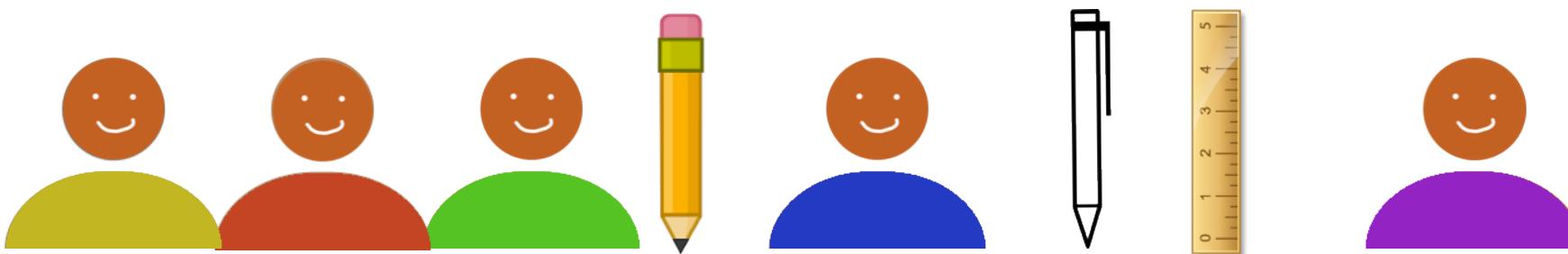


The divider method: A generative proof

How many ways can we assign $n = 5$ indistinct children to $r = 4$ distinct bicycle helmet styles?

Goal Order n indistinct objects and $r - 1$ indistinct dividers.

0. Make objects and dividers distinct

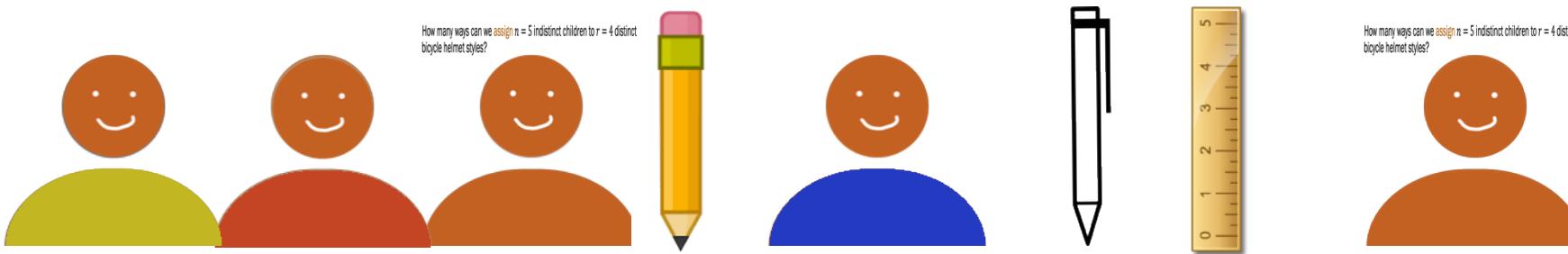


The divider method: A generative proof

How many ways can we assign $n = 5$ indistinct children to $r = 4$ distinct bicycle helmet styles?

Goal Order n indistinct objects and $r - 1$ indistinct dividers.

0. Make objects and dividers distinct



1. Order n distinct objects and $r - 1$ distinct dividers

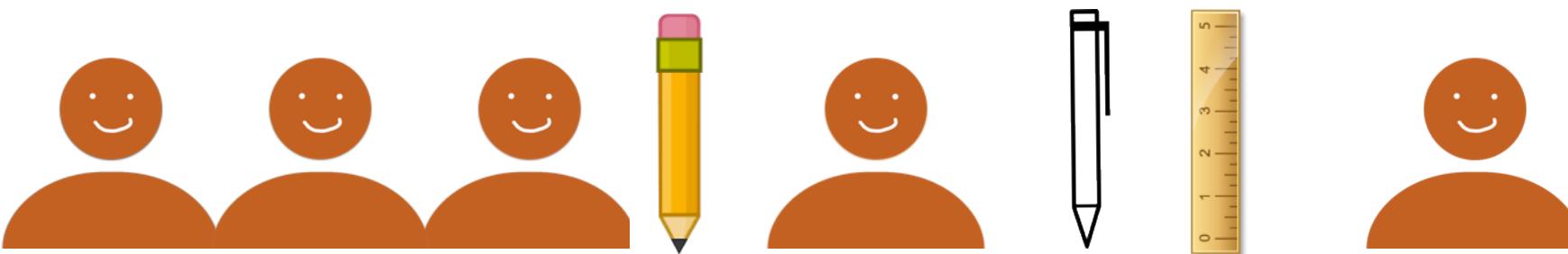
$$(n + r - 1)!$$

The divider method: A generative proof

How many ways can we assign $n = 5$ indistinct children to $r = 4$ distinct bicycle helmet styles?

Goal Order n indistinct objects and $r - 1$ indistinct dividers.

0. Make objects and dividers distinct



1. Order n distinct objects and $r - 1$ distinct dividers

$$(n + r - 1)!$$

2. Make n objects indistinct

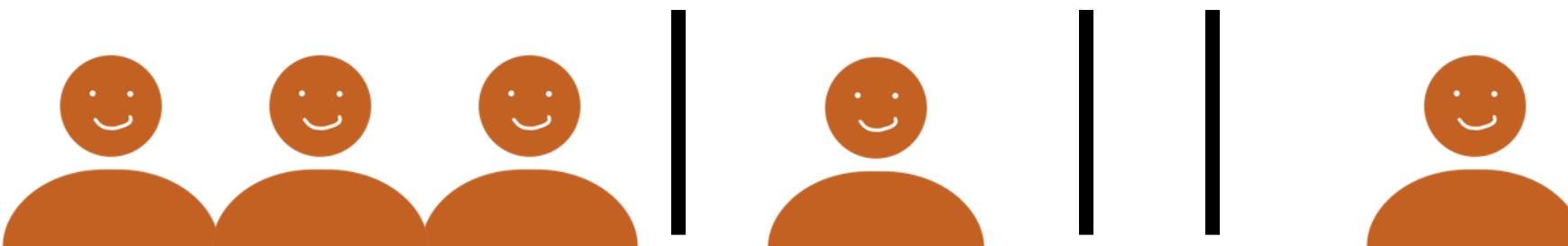
$$\frac{1}{n!}$$

The divider method: A generative proof

How many ways can we assign $n = 5$ indistinct children to $r = 4$ distinct bicycle helmet styles?

Goal Order n indistinct objects and $r - 1$ indistinct dividers.

0. Make objects and dividers distinct



1. Order n distinct objects and $r - 1$ distinct dividers

$$(n + r - 1)!$$

2. Make n objects indistinct

$$\frac{1}{n!}$$

3. Make $r - 1$ dividers indistinct

$$\frac{1}{(r - 1)!}$$

The divider method

The number of ways to distribute n indistinct objects into r buckets is equivalent to the number of ways to permute $n + r - 1$ objects such that n are indistinct objects, and $r - 1$ are indistinct dividers:

$$\text{Total} = (n + r - 1)! \times \frac{1}{n!} \times \frac{1}{(r-1)!}$$

$$= \binom{n + r - 1}{r - 1} \text{ outcomes}$$

Integer solutions to equations

Divider method
(n indistinct objects, r buckets) $\binom{n+r-1}{r-1}$

How many integer solutions (assignments to each x) are there to the following equation:

$$x_1 + x_2 + \cdots + x_6 = 20$$

where for all i , x_i is an integer such that $0 \leq x_i \leq 20$?

Integer solutions to equations

Divider method
(n indistinct objects, r buckets) $\binom{n+r-1}{r-1}$

How many integer solutions (assignments to each x) are there to the following equation:

$$x_1 + x_2 + \cdots + x_6 = 20$$

where for all i , x_i is an integer such that $0 \leq x_i \leq 20$?

Example solution: $2 + 3 + 2 + 5 + 5 + 3$

$$\binom{n+r-1}{r-1} = \binom{20+6-1}{6-1} = \binom{25}{5} = 53130$$

Course Reader for CS109

Search book...

Notation Reference
Random Variable Reference
Python Reference
Calculators

Part 1: Core Probability

Counting
Combinatorics
Definition of Probability
Equally Likely Outcomes
Probability of or
Conditional Probability
Independence
Probability of and
Law of Total Probability
Bayes' Theorem
Log Probabilities
Many Coin Flips
Applications
Enigma Machine
Serendipity
Random Shuffles
Bacteria Evolution

Combinatorics

Counting problems can be approached from the basic building blocks described in the first section: [Counting](#). However some counting problems are so ubiquitous in the world of probability that it is worth knowing a few higher level counting abstractions. When solving problems, if you can find the analogy from these canonical examples you can build off of the corresponding combinatorics formulas:

1. [Permutations of Distinct Objects](#)
2. [Permutations with Indistinct Objects](#)
3. [Combinations with Distinct Objects](#)
4. [Bucketing with Distinct Objects](#)
5. [Bucketing with Indistinct Objects](#)
6. [Bucketing into Fixed Sized Containers](#)

While these are by no means the only common counting paradigms, it is a helpful set.

Permutations of Distinct Objects

Definition: Permutation Rule

A permutation is an ordered arrangement of n distinct objects. Those n objects can be permuted in $n \cdot (n-1) \cdot (n-2) \cdots 2 \cdot 1 = n!$ ways.

This changes slightly if you are permuting a subset of distinct objects, or if some of your objects are indistinct. We will handle those cases shortly! Note that unique is a synonym for distinct.

Combinatorics

chrispiech.github.io/probabilityForComputerScientists/en/part1/combinatorics/#bucketing_indistinct

Course Reader for CS109

Search book...

Notation Reference
Random Variable Reference
Python Reference
Calculators

Part 1: Core Probability

- Counting
- Combinatorics**
- Definition of Probability
- Equally Likely Outcomes
- Probability of or
- Conditional Probability
- Independence
- Probability of and
- Law of Total Probability
- Bayes' Theorem
- Log Probabilities
- Many Coin Flips
- Applications
 - Enigma Machine
 - Serendipity
 - Random Shuffles
 - Bacteria Evolution

Part 2: Random Variables

Bucketing with Indistinct Objects

While the previous example allowed us to put n distinguishable objects into r distinct groups, the more interesting problem is to work with n indistinguishable objects.

Divider Method:

Suppose you want to place n indistinguishable items into r containers. The divider method works by imagining that you are going to solve this problem by sorting two types of objects, your n original elements and $(r - 1)$ dividers. Thus, you are permuting $n + r - 1$ objects, n of which are the same (your elements) and $r - 1$ of which are the same (the dividers). Thus the total number of outcomes is:

$$\frac{(n + r - 1)!}{n!(r - 1)!} = \binom{n + r - 1}{n} = \binom{n + r - 1}{r - 1}$$

The divider method can be derived via the "Stars and Bars" method. This is a creative construction where we consider permutations of indistinguishable items, represented by stars *, and dividers between our containers, represented by bars |. Any distinct permutation of these stars and bars represents a unique assignments of our items to containers.

Imagine we want to separate 5 indistinguishable objects into 3 containers. We can think of the problem as finding the number of ways to order 5 stars and 2 bars *****| |. Any permutation of these symbols represents a unique assignment. Here are a few examples:

|*| represents 2 items in the first bucket, 1 item in the second and 2 items in the third.

****| |* represents 4 items in the first bucket, 0 item in the second and 1 items in the third.

| |***** represents 0 items in the first bucket, 0 item in the second and 5 items in the third.

Why are there only 2 dividers when there are 3 buckets? This is an example of a fence-post problem". With 2 dividers you have created three containers. We already have a method for counting permutations with some indistinct items. For the example above where we have seven

Combinatorics

chrispiech.github.io/probabilityForComputerScientists/en/part1/combinatorics/#bucketing_indistinct

Course Reader for CS109

Search book...

Notation Reference
Random Variable Reference
Python Reference
Calculators

Part 1: Core Probability

Counting
Combinatorics
Definition of Probability
Equally Likely Outcomes
Probability of or
Conditional Probability
Independence
Probability of and
Law of Total Probability
Bayes' Theorem
Log Probabilities
Many Coin Flips
Applications
Enigma Machine
Serendipity
Random Shuffles
Bacteria Evolution

Part 2: Random Variables

Part A: Say you are a startup incubator and you have \$10 million to invest in 4 companies (in \$1 million increments). How many ways can you allocate this money?

Solution: This is just like putting 10 balls into 4 urns. Using the Divider Method we get:

$$\text{Total ways} = \binom{10 + 4 - 1}{10} = \binom{13}{10} = 286$$

This problem is analogous to solving the integer equation $x_1 + x_2 + x_3 + x_4 = 10$, where x_i represents the investment in company i such that $x_i \geq 0$ for all $i = 1, 2, 3, 4$.

Part B: What if you know you want to invest at least \$3 million in Company 1?

Solution: There is one way to give \$3 million to Company 1. The number of ways of investing the remaining money is the same as putting 7 balls into 4 urns.

$$\text{Total Ways} = \binom{7 + 4 - 1}{7} = \binom{10}{7} = 120$$

This problem is analogous to solving the integer equation $x_1 + x_2 + x_3 + x_4 = 10$, where $x_1 \geq 3$ and $x_2, x_3, x_4 \geq 0$. To translate this problem into the integer solution equation that we can solve via the divider method, we need to adjust the bounds on x_1 such that the problem becomes $x_1 + x_2 + x_3 + x_4 = 7$, where x_i is defined as in Part A.

Part C: What if you don't have to invest all \$10 M? (The economy is tight, say, and you might want to save your money.)

Solution: Imagine that you have an extra company: yourself. Now you are investing \$10 million in 5 companies. Thus, the answer is the same as putting 10 balls into 5 urns.

$$\text{Total} = \binom{10 + 5 - 1}{10} = \binom{14}{10} = 1001$$

This problem is analogous to solving the integer equation $x_1 + x_2 + x_3 + x_4 + x_5 = 10$, such that $x_i \geq 0$ for all $i = 1, 2, 3, 4, 5$.

Summary of Combinatorics

Counting tasks on n objects

Sort objects
(permutations)

Choose k objects
(combinations)

Put objects in r
buckets

Distinct
(distinguishable)

Some
distinct

Distinct

$$n!$$

$$\frac{n!}{n_1! n_2! \cdots n_r!}$$

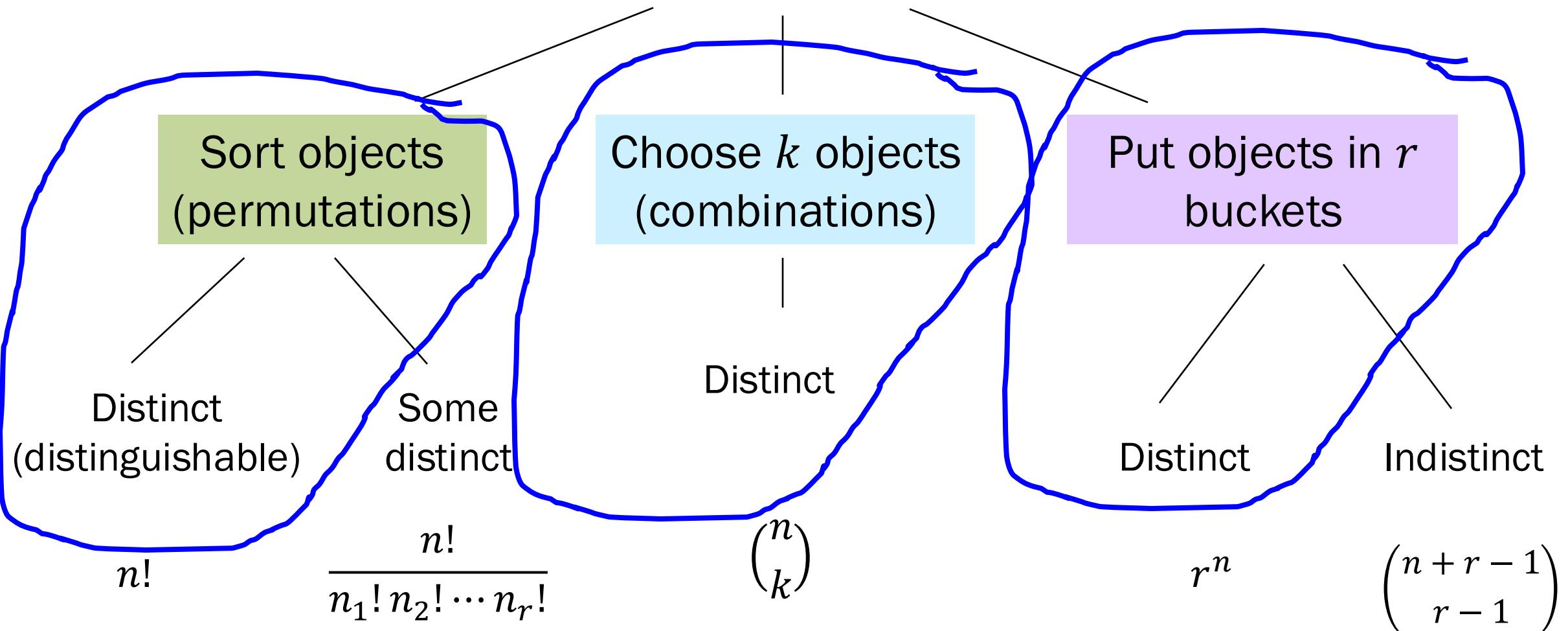
$${n \choose k}$$

$$r^n$$

$${n+r-1 \choose r-1}$$

Summary of Combinatorics

Counting tasks on n objects



Ready for the first 6 problems (and the rest on Friday)

CS109PsetApp – Overview Pset 1 – Counting for Probabilistic Robotics

Robot Paths

Imagine you have a robot (🤖) that lives on an 7×8 grid (it has 7 rows and 8 columns). The robot starts in cell (1, 1) and can take steps either to the right or down (no left or up steps). How many distinct paths can the robot take to the destination (🚩) in cell (7, 8)?

7 rows

8 columns

Answer Editor Solution

Numeric Answer: Enter your answer Check Answer

Explanation:

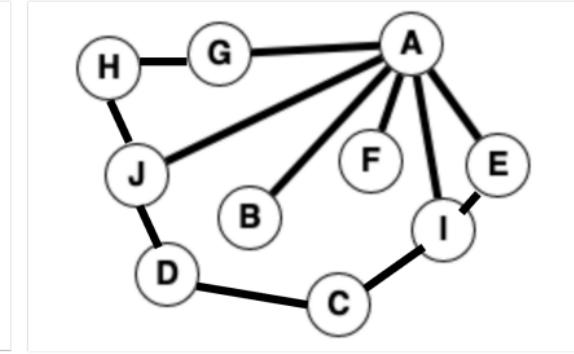
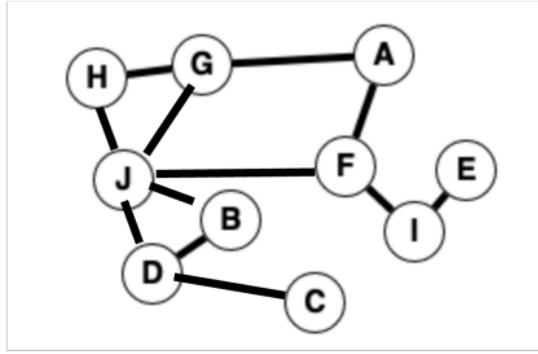
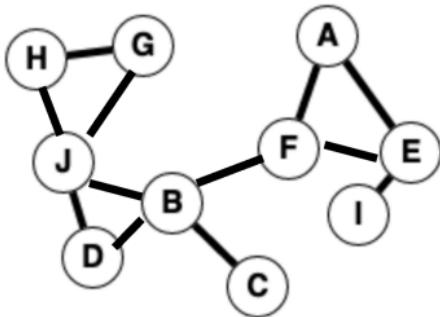
Block LaTeX Inline LaTeX Python Image

Previous Question Next Question

Formalizing a Word Problem is Hard

3 Random Edges [25 points]

Here are three different networks each with 10 nodes and 12 random edges:



- a. (4 points) Consider a network with 10 nodes. Count the number of possible locations for undirected edges. An undirected edge from node A to node B is not distinct from an edge from node B to node A. You can assume that an edge does not connect a node to itself.

Want to go deep?

The screenshot shows a web browser window with the title bar "Enigma Machine". The URL in the address bar is "chrispiech.github.io/probabilityForComputerScientists/en/examples/enigma/". The main content area displays a dark-themed page with a large title "Enigma Machine". To the left, a sidebar titled "Course Reader for CS109" contains a search bar and several links: "Notation Reference", "Random Variable Reference", "Calculators", "Part 1: Core Probability" (which is expanded to show "Counting", "Combinatorics", "Definition of Probability", "Equally Likely Outcomes", "Probability of or", "Conditional Probability", "Independence", "Probability of and", "Law of Total Probability", "Bayes' Theorem", "Log Probabilities", "Many Coin Flips", "Worked Examples", "Enigma Machine" which is highlighted in grey, "Serendipity", and "Bacteria Evolution"). Below this, another section titled "Part 2: Random Variables" includes a link to "Random Variables". The main text on the page discusses the Enigma machine's role in World War II code-breaking, mentioning the difficulty of finding unique configurations and the use of a large machine to search for them. A black and white photograph of a woman operating a large, complex mechanical device (the Bombe) is shown, with a caption below it.

Enigma Machine

One of the very first computers was built to break the Nazi “enigma” codes in WW2. It was a hard problem because the “enigma” machine, used to make secret codes, had so many unique configurations. Every day the Nazis would choose a new configuration and if the Allies could figure out the daily configuration, they could read all enemy messages. One solution was to try all configurations until one produced legible German. This begs the question: How many configurations are there?

The WW2 machine built to search different enigma configurations.

The enigma machine has three rotors. Each rotor can be set to one of 26 different positions. How

