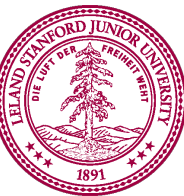
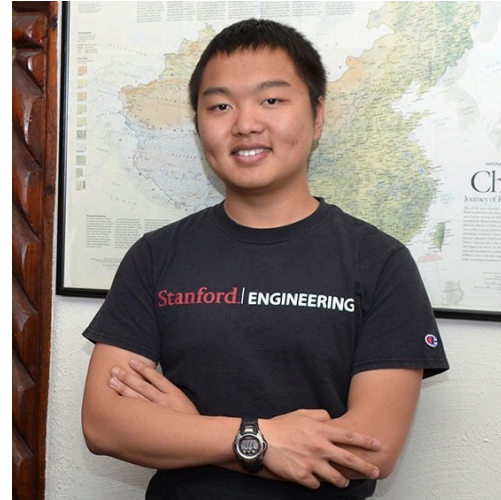


**CS109: Probability for  
Computer Scientists**

# New Teaching Staff Member!!

---



# Office Hours

CS109 Course ▾ Problem Sets ▾ Lecture ▾ Section ▾ Resources ▾ Schedule

- Syllabus
- Honor Code
- Office Hours**

## Office Hours

Office hours start on Monday (June 26th)! Check the calendar for times and locations.

**Where is Huang Basement?**

### CS109 Summer 2023 OH

Today Jun 25 – Jul 1, 2023 Print Week Month Agenda

	Sun 6/25	Mon 6/26	Tue 6/27	Wed 6/28	Thu 6/29	Fri 6/30	Sat 7/1
10am							
11am					10:30 – 12:30p Yunsung's OH Gates 100		
12pm							
1pm				12:45p – 2:45p Kathleen OH Huang Basement			
2pm			2p – 4p Kathleen OH Huang Basement				
3pm		3p – 4:15p Lecture		3p – 4:15p Lecture	3p – 5p Will's OH 240-201	3p – 4:15p Lecture	
4pm							
5pm							

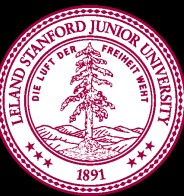


# Section Signups Tomorrow

---



**COMING  
SOON**



# Problem Set #1 is out



CS109 Course > Problem Sets > Lecture > Section

PS1 5. Random Choice

What is the probability that both users will get the same randomly generated password? Provide an answer to three decimal places!

```
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
```

Answer Editor

Numeric Answer: 97 Check Answer

Explanation:

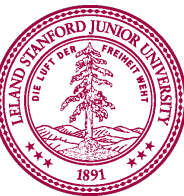
I am so excited! What a good time!

Insert LaTeX

Check your answer

Auto Submission

Previous Question Next Question

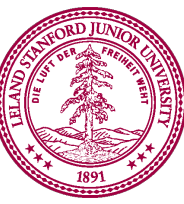


# Write an Agent

The screenshot shows a web browser window with the URL 'cs109'. The page title is 'CS109 Course > Problem Sets > Lecture > Section'. The main content area is titled 'PS1 9. Counting Cards'. On the left, a navigation sidebar shows 11 numbered items, with items 1 through 8 and 10 marked with green checkmarks, and item 9 highlighted. The main text describes counting cards in blackjack and mentions the MIT Blackjack Team and the movie 21. Below the text is a large empty box for the answer. To the right, there is an 'Answer Editor' tab and a 'Solution' tab. The 'Answer Editor' contains a code editor with the following 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
```

Below the code editor are two buttons: 'Run One Game' and 'Test Agent'. At the bottom of the page, there are 'Previous Question' and 'Next Question' buttons.



# Python Review Session



CS109 Course ▾ Problem Sets ▾ Lecture ▾ Section ▾ Resources ▾

Course Reader  
Python Review  
Latex Cheat Sheet

## Python for Probability

We'll hold two Python review sessions throughout the quarter to get you up to speed on what you'll need for the problem sets.

If you want to get more python practice, you can also check out [Python tutorial notebook](#) (make sure you are logged in with your Stanford account)!

- **Session #1:** Thursday June 29th, 4pm Pacific Time. Intro, running programs, Python basics.  
Zoom: [Link](#)
- **Session #2:** Thursday July 6th, 4:30pm Pacific Time. NumPy arrays and random numbers.  
Zoom: [Link](#)

This and Next Thursday

Online and will be recorded!



Course Reader

Python Review

Latex Cheat Sheet

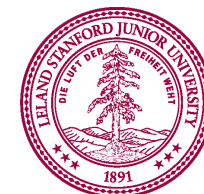
## Schedule

We have a handout to help you get started

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

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

Done



# 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.**

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



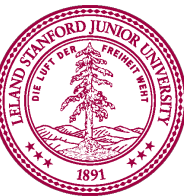


# If you notice a bug?

---

It should be robust, but things can happen.

Let me know: send an email to [cs109@cs.stanford.edu](mailto:cs109@cs.stanford.edu) or email any course staff individually. We need your email and the approximate time you encountered the bug.



**Want to try to hack the  
PsetApp? We will give you  
time after class :-)**



# How to Get the Most Out of Lectures?

---

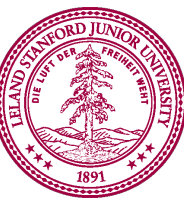
Everyone has different styles of learning!

A lot of the technical details appear in the course reader.

It might help to follow along with the big ideas during class, and refer back to course reader/lectures if you want to fill in details.

**BUT!!**

Find out what works best for **you** 😊 !!



Review

# CS109: From Counting to Machine Learning

---



Counting  
Theory



Core  
Probability



Random  
Variables



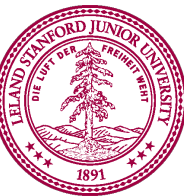
Probabilistic  
Models



Uncertainty  
Theory



Machine  
Learning



# Core Counting

---

## Counting with steps

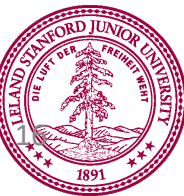
**Definition:** Step Rule of Counting (aka Product Rule of Counting)

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  $m \cdot n$ .

## 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  $|A \text{ or } B| = |A| + |B| - |A \text{ and } B|$ .



# 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

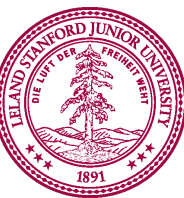
010000  
010001  
**010010**  
010011  
010100  
010101  
**010110**  
010111  
011000  
011001  
**011010**  
011011  
011100  
011101  
**011110**  
011111

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



# Challenge Problem

---

## 1. Strings

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

BOBA, ABOB, OBBA...



End Review

# Permutations I

# Summary of Combinatorics

---

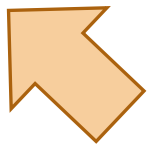
Counting tasks on  $n$  objects

Sort objects  
(permutations)

Choose  $k$  objects  
(combinations)

Put objects in  $r$   
buckets

Distinct  
(distinguishable)



# Orderings of Letters

---

How many letter orderings are possible for the following strings?

1. GATES



# Orderings of Letters

---

gates	getsa	atges	asteg	tegas	egtsa	esgat	sateg
gatse	gesat	atgse	asegt	tegsa	egsat	esgta	saegt
gaets	gesta	ategs	asetg	teags	egsta	esagt	saetg
gaest	gsate	atesg	tgaes	teasg	eagts	esatg	stgae
gaste	gsaet	atsge	tgase	tesga	eagst	estga	stgea
gaset	gstae	atseg	tgeas	tesag	eatgs	estag	stage
gtaes	gstea	aegts	tgesa	tsgae	eatsg	sgate	staeg
gtase	gseat	aegst	tgsae	tsgea	easgt	sgaet	stega
gteas	gseta	aetgs	tgsea	tsage	eastg	sgtae	steag
gtesa	agtes	aetsg	tages	tsaeg	etgas	sgtea	segat
gtsae	agtse	aesgt	tagse	tsega	etgsa	sgeat	segta
gtsea	agets	aestg	taegs	tseag	etags	sgeta	seagt
geats	agest	asgte	taesg	egats	etasg	sagte	seatg
geast	agste	asget	tasge	egast	etsga	saget	setga
getas	agset	astge	taseg	egtas	etsag	satge	setag



# Orderings of letters

---



Step 1:  
Chose first letter  
(5 options)

Step 2:  
Chose 2nd letter  
(4 options)

Step 3:  
Chose 3rd letter  
(3 options)

Step 4:  
Chose 4th letter  
(2 options)

Step 5:  
Chose 5th letter  
(1 option)

# Permutations

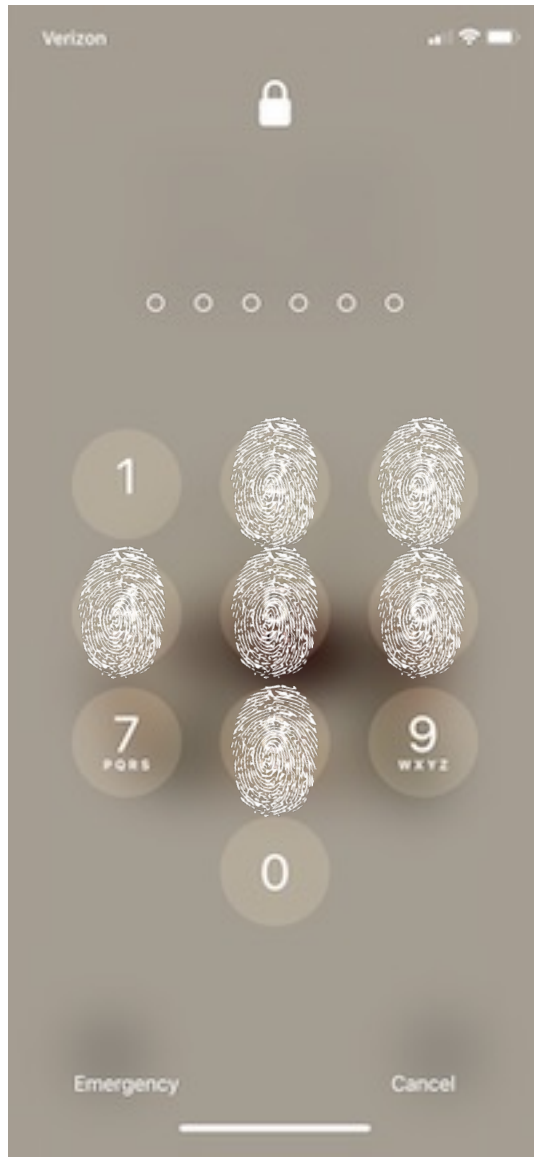
---

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

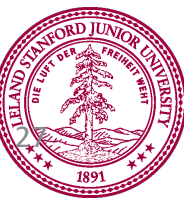
The number of unique orderings (**permutations**) of  $n$  distinct objects is

$$n! = n \times (n - 1) \times (n - 2) \times \cdots \times 2 \times 1.$$

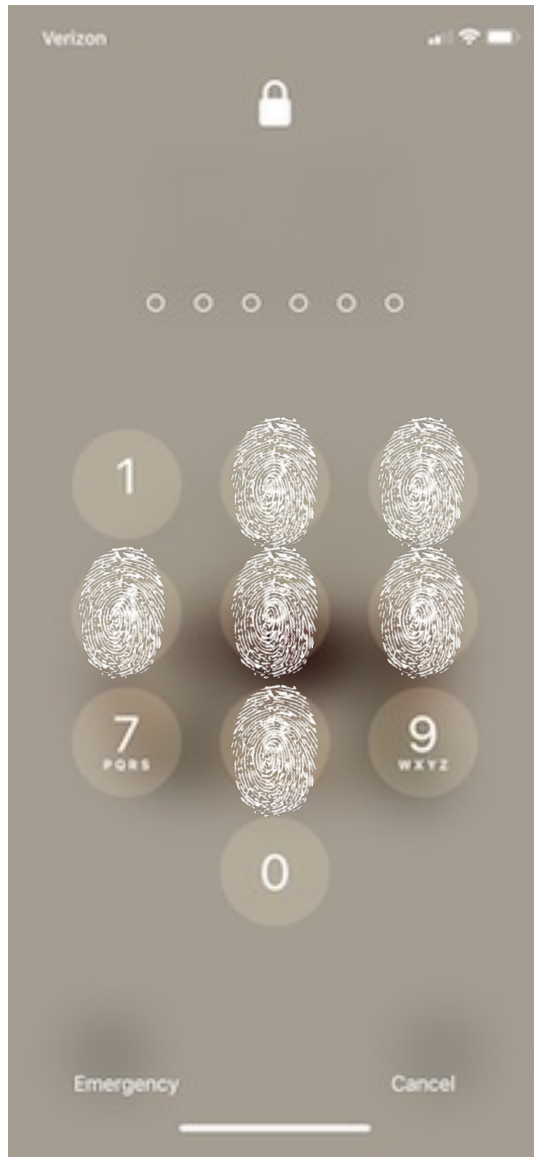
# Unique 6-digit passcodes with **six** smudges



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

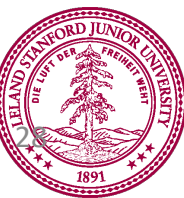


How many unique 6-digit passcodes are possible if a phone password uses each of **six** distinct numbers?

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

How many unique passcodes are possible if a phone password is some ordered subset of any 6 unique digits?

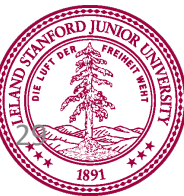
$$\begin{aligned} \text{Total} &= 10 \times 9 \times 8 \times 7 \times 6 \times 5 \\ &= \frac{10!}{4!} = 151200 \text{ passcodes} \end{aligned}$$



# Unique Bit Strings

---

1, 0, 1, 0, 0



# Sort indistinct objects?

---



# Sort $n$ distinct objects



Ayesha



Tim



Irina



Joey



Waddie

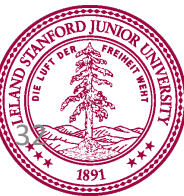
# Sort $n$ distinct objects



## Steps:

1. Choose 1<sup>st</sup> can      5 options
2. Choose 2<sup>nd</sup> can      4 options
- ...
5. Choose 5<sup>th</sup> can      1 option

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



# Permutations II

# Summary of Combinatorics

---

Counting tasks on  $n$  objects

Sort objects  
(permutations)

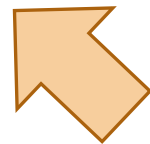
Choose  $k$  objects  
(combinations)

Put objects in  $r$   
buckets

Distinct  
(distinguishable)

Semi-distinct

$n!$



# How many ways can we sort coke cans!

---



Coke



Coke0



Coke



Coke0



Coke0

# Sort $n$ distinct objects



Ayesha



Tim



Irina



Joey



Waddie

# of permutations =

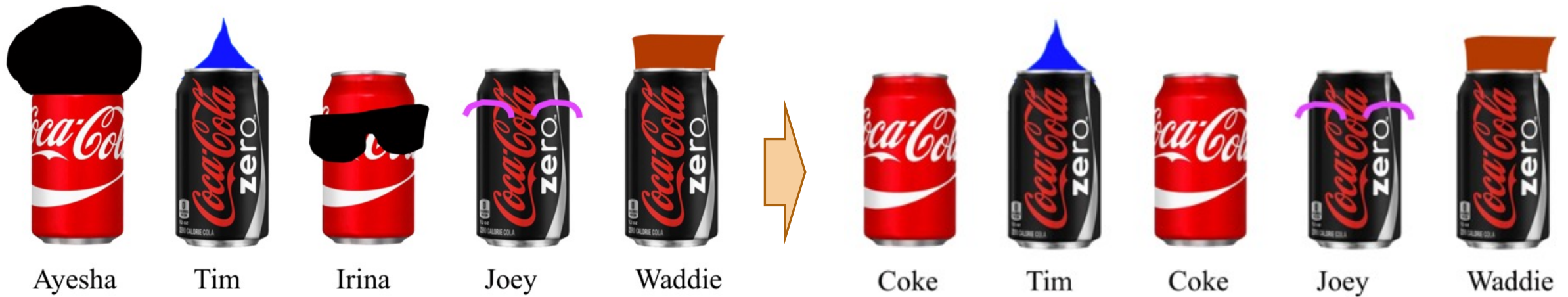
# Sort semi-distinct objects

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, permutations of distinct objects is a two-step process:

$$\begin{array}{ccccc} \text{permutations} & & \text{permutations} & & \text{Permutations} \\ \text{of distinct objects} & = & \text{considering some} & \times & \text{of just the} \\ & & \text{objects are indistinct} & & \text{indistinct objects} \end{array}$$

# Sort semi-distinct objects

---

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

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

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

Permutations of just the indistinct objects



In combinatorics it often helps to overcount permutations and fix for the overcounting

# 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

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

How many permutations?



Coke



Coke0



Coke



Coke0



Coke0

# 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

2. MISSISSIPPI



How many letter orderings are possible for the following strings?

1. BOBA

$$= \frac{4!}{2!} = 12$$

2. MISSISSIPPI

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

# To the Code!

---

baob  
bbao  
obba  
oabb  
boab  
babo  
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)
```

# Summary of Combinatorics

Counting tasks on  $n$  objects

Sort objects  
(permutations)

Choose  $k$  objects  
(combinations)

Put objects in  $r$   
buckets

Distinct  
(distinguishable)

Semi-distinct

$$n!$$

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

# Combinations I

# Summary of Combinatorics

Counting tasks on  $n$  objects

Sort objects  
(permutations)

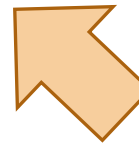
Choose  $k$  objects  
(combinations)

Put objects in  $r$   
buckets

Distinct  
(distinguishable)

Some  
distinct

Distinct



$$n!$$

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

# Combinations with cake

There are  $n = 20$  people.

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

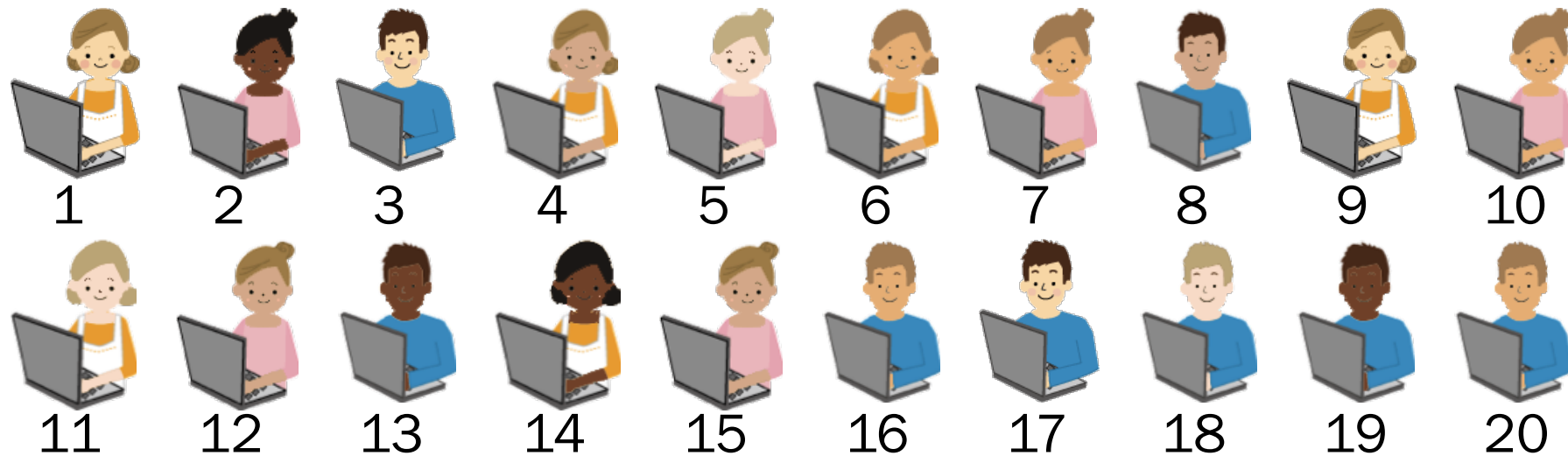


Consider the following  
generative process...

# Combinations with cake

There are  $n = 20$  people.

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



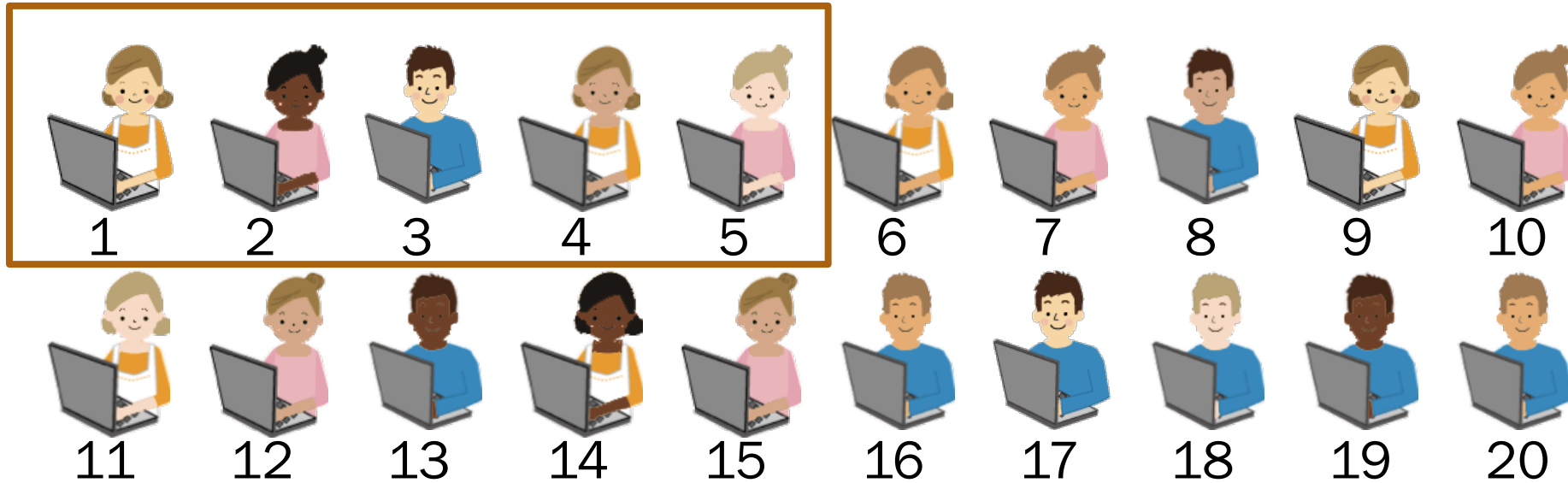
1.  $n$  people  
get in line

$n!$  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

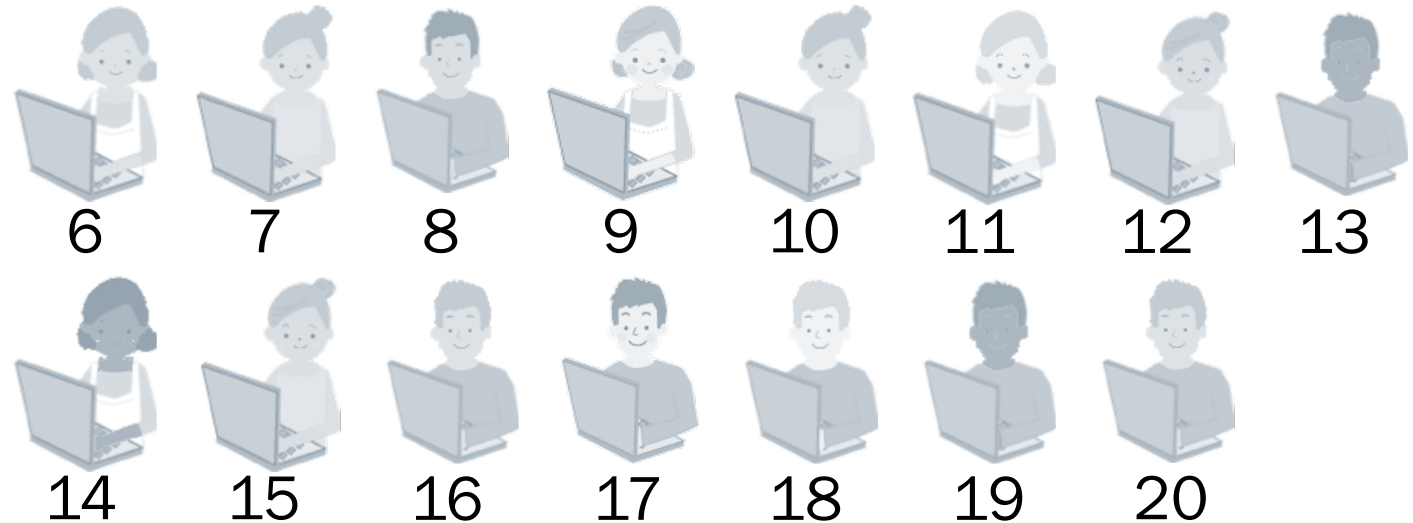
2. Put first  $k$   
in cake room

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

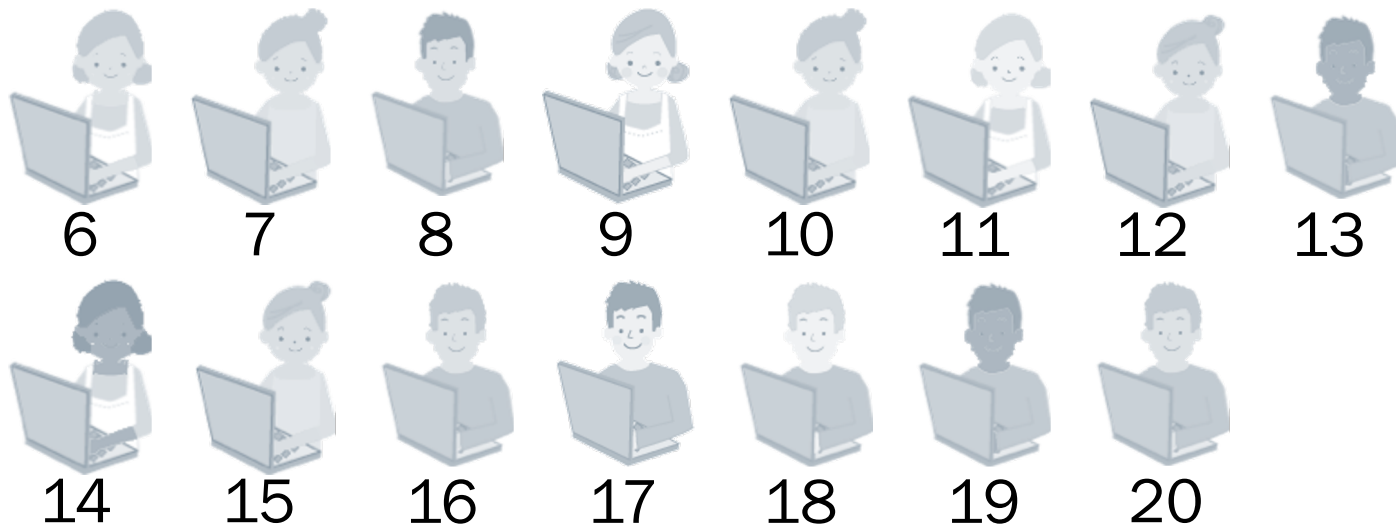
2. Put first  $k$   
in cake room

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

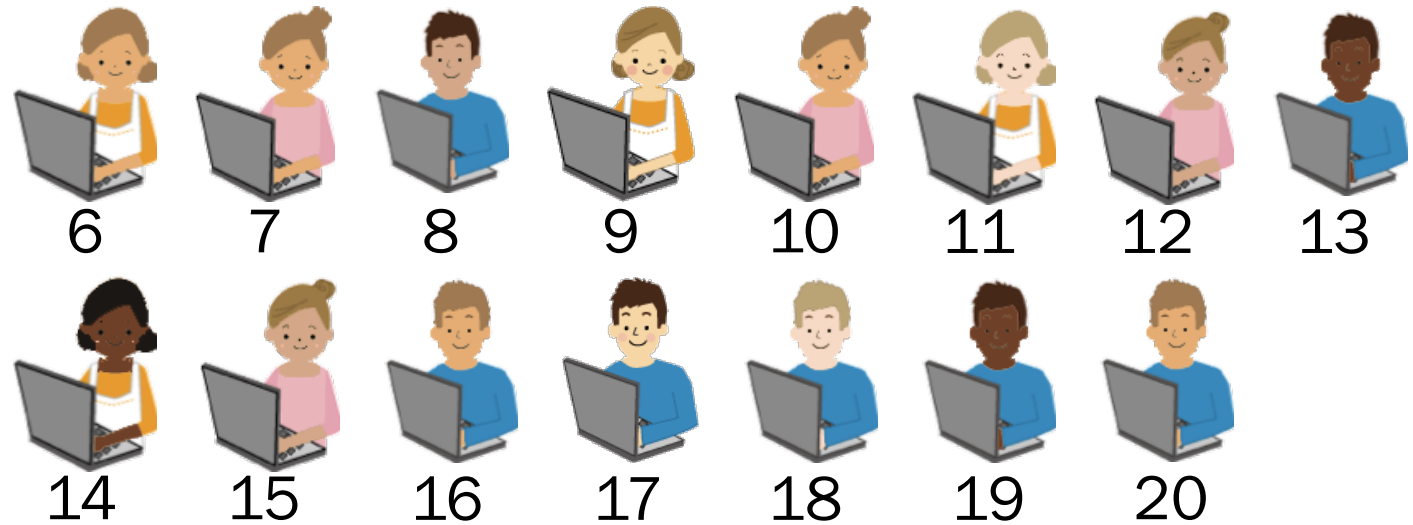
3. **Allow cake group to mingle**

$k!$  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

Kim, Piech + Cain, CS109, Stanford University

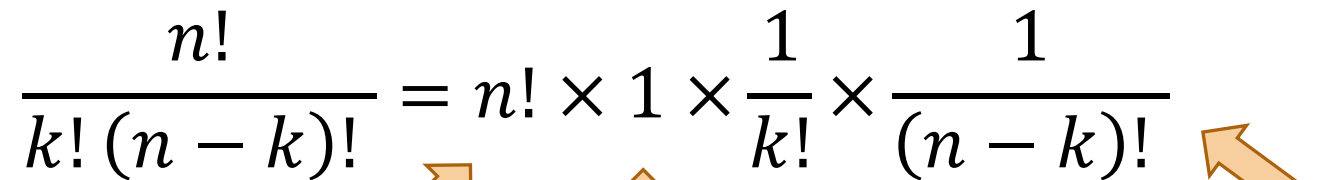
4. Allow non-cake group to mingle

$(n - k)!$  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

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

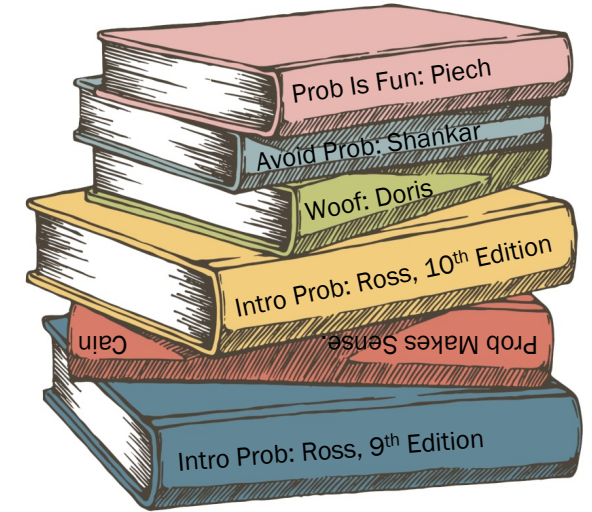
Fun Fact:  $\binom{n}{k} = \binom{n}{n-k}$   $(x+y)^n$

# 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?



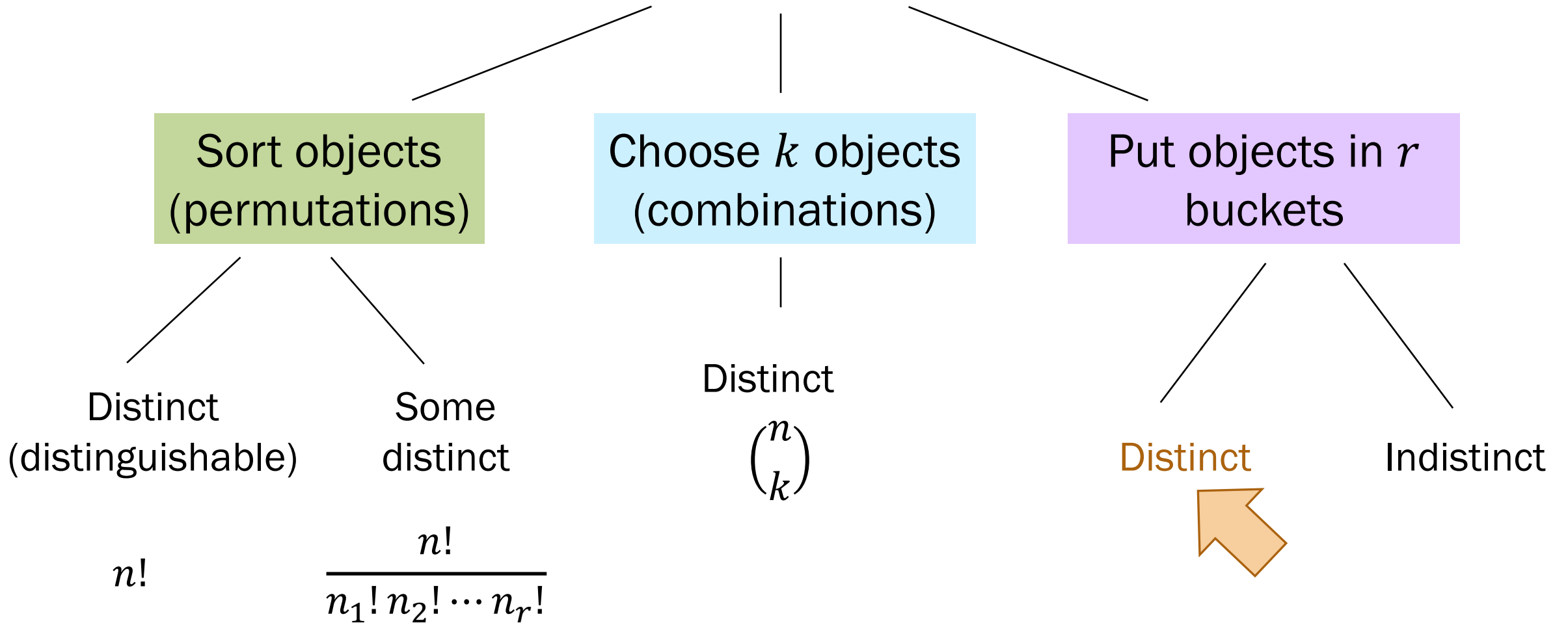
```
def main():  
    cards = make_deck()  
    all_hands = itertools.combinations(cards, 5)  
    for hand in all_hands:  
        print(hand)
```

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

# Buckets and The Divider Method

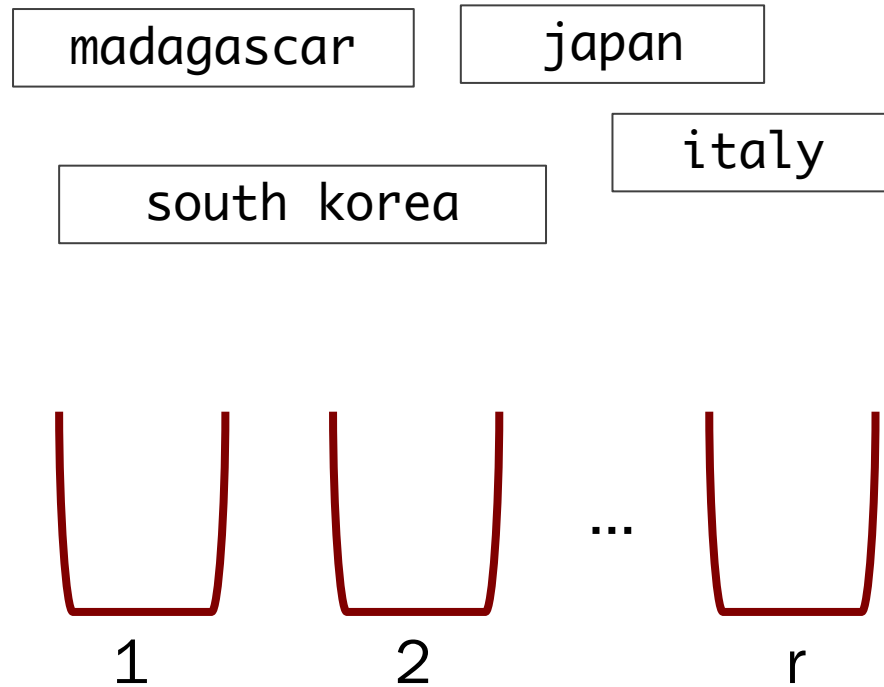
# Summary of Combinatorics

Counting tasks on  $n$  objects



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

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

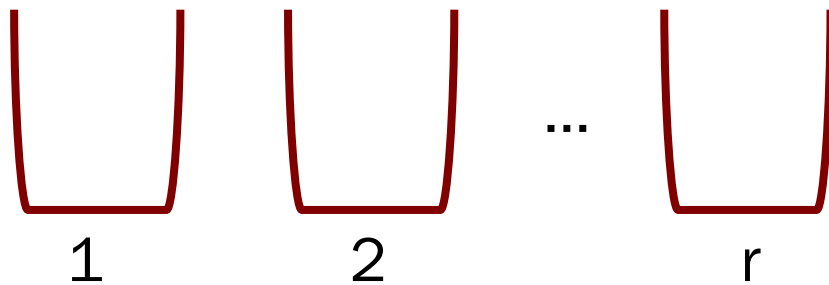
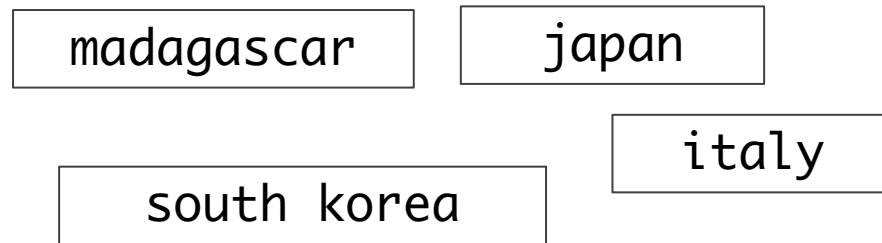


# ~~Balls and urns~~ Hash tables and **distinct** strings

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

Steps:

1. Bucket 1<sup>st</sup> string
2. Bucket 2<sup>nd</sup> string
- ...
- $n$ . Bucket  $n^{\text{th}}$  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

Distinct

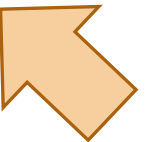
Indistinct

$$n!$$

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

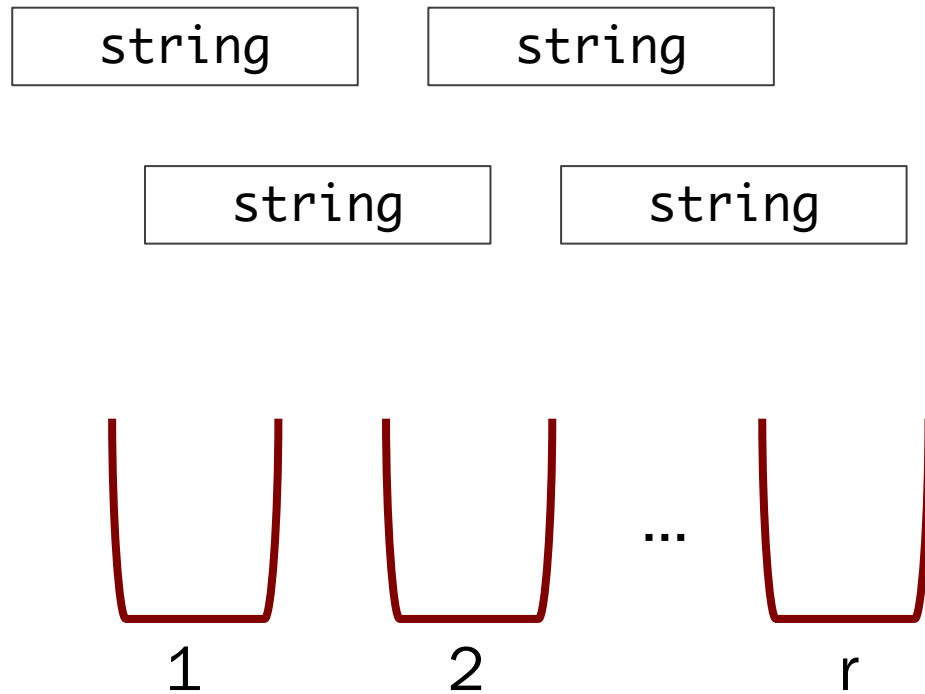
$$\binom{n}{k}$$

$$r^n$$



# Servers and **indistinct** requests

How many ways are there to distribute  $n$  **indistinct** strings to  $r$  buckets?



## Goal

Bucket 1 has  $x_1$  string,  
Bucket 2 has  $x_2$  string,

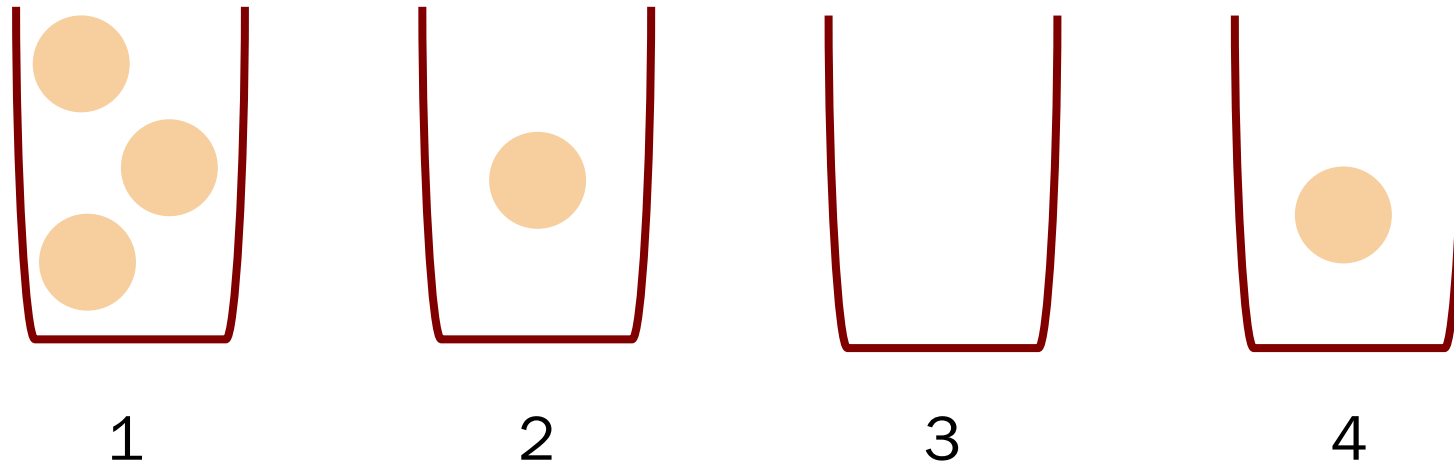
...

Bucket  $r$  has  $x_r$  string

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

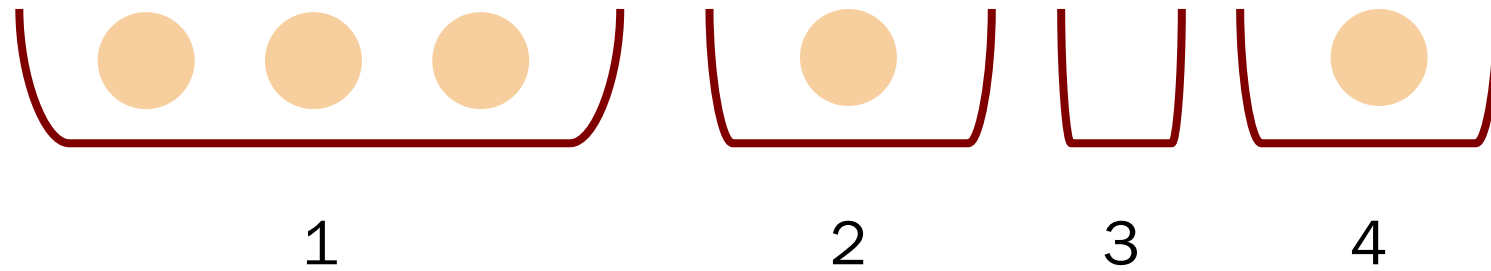
# Servers and **indistinct** requests

How many ways are there to distribute  $n$  **indistinct** strings to  $r$  buckets?



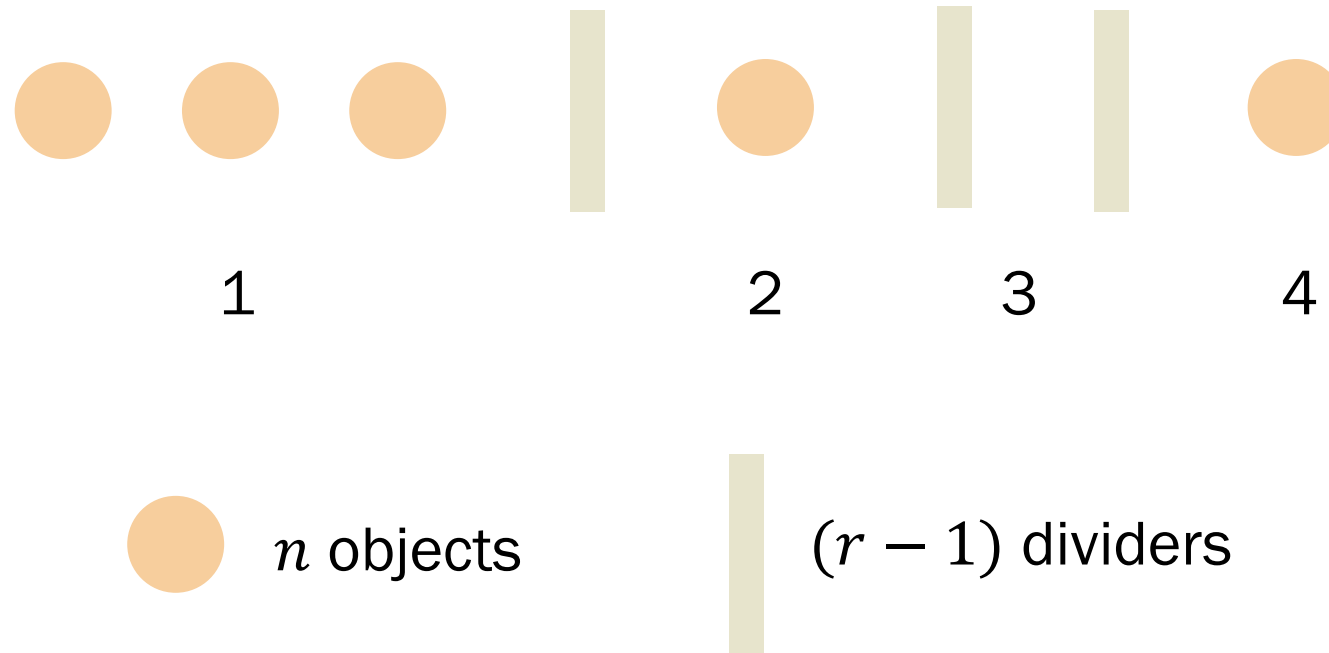
# Servers and **indistinct** requests

How many ways are there to distribute  $n$  **indistinct** strings to  $r$  buckets?



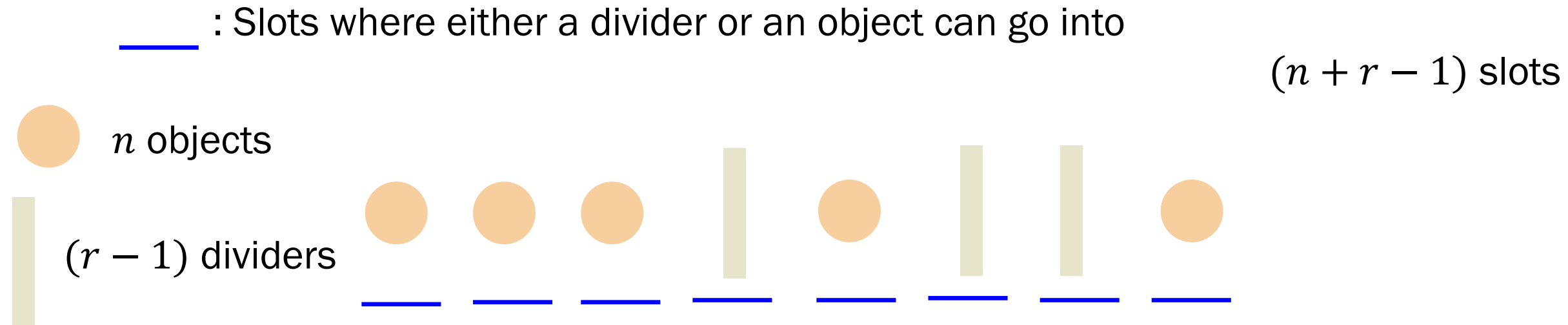
# Servers and **indistinct** requests

How many ways are there to distribute  $n$  **indistinct** strings to  $r$  buckets?



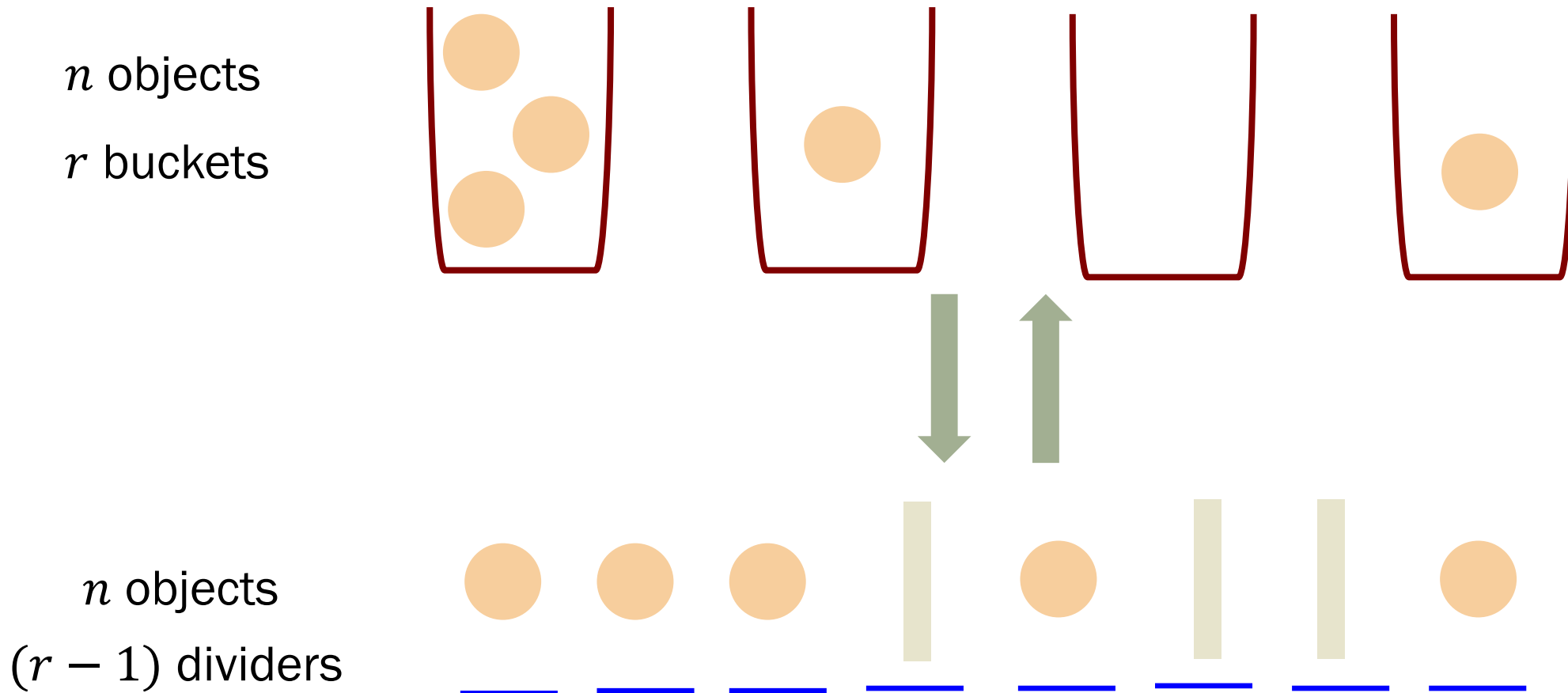
# Servers and **indistinct** requests

How many ways are there to distribute  $n$  **indistinct** strings to  $r$  buckets?



# Servers and **indistinct** requests

How many ways are there to distribute  $n$  **indistinct** strings to  $r$  buckets?

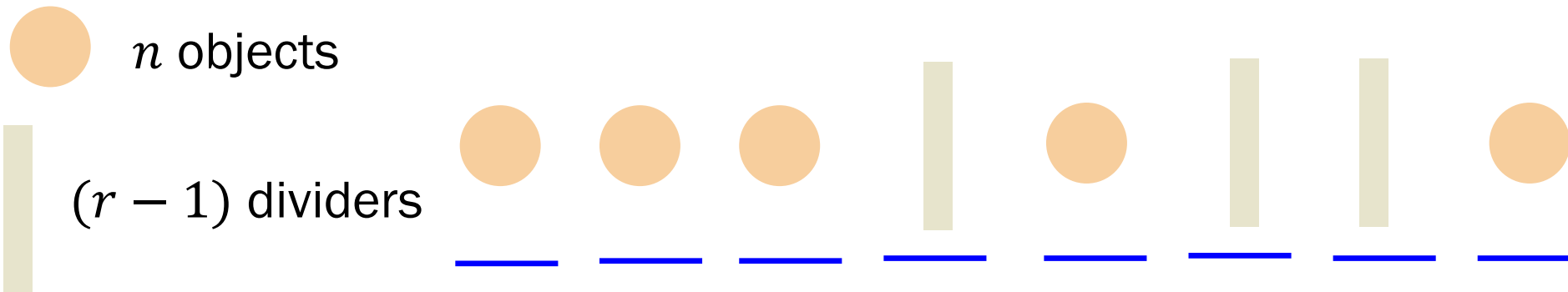


# Servers and **indistinct** requests

How many ways are there to distribute  $n$  **indistinct** strings to  $r$  buckets?

       : Slots where either a divider or an object can go into

$(n + r - 1)$  slots



Answer

1. (Combination) Choose  $(r - 1)$  divider slots out of  $(n + r - 1)$  distinct slots
2. (Permutation) Arrange  $(r - 1)$  indistinct dividers and  $n$  indistinct objects

# The divider method

---

The number of ways to distribute  $n$  indistinct objects into  $r$  buckets is equivalent to the number of ways of:

1. Choosing  $(r-1)$  slots among  $(n+r-1)$  distinct slots to place the divider
2. Arranging  $(r-1)$  indistinct dividers and  $n$  indistinct objects.

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

# Integer solutions to equations

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

How many integer solutions are there to the following equation:

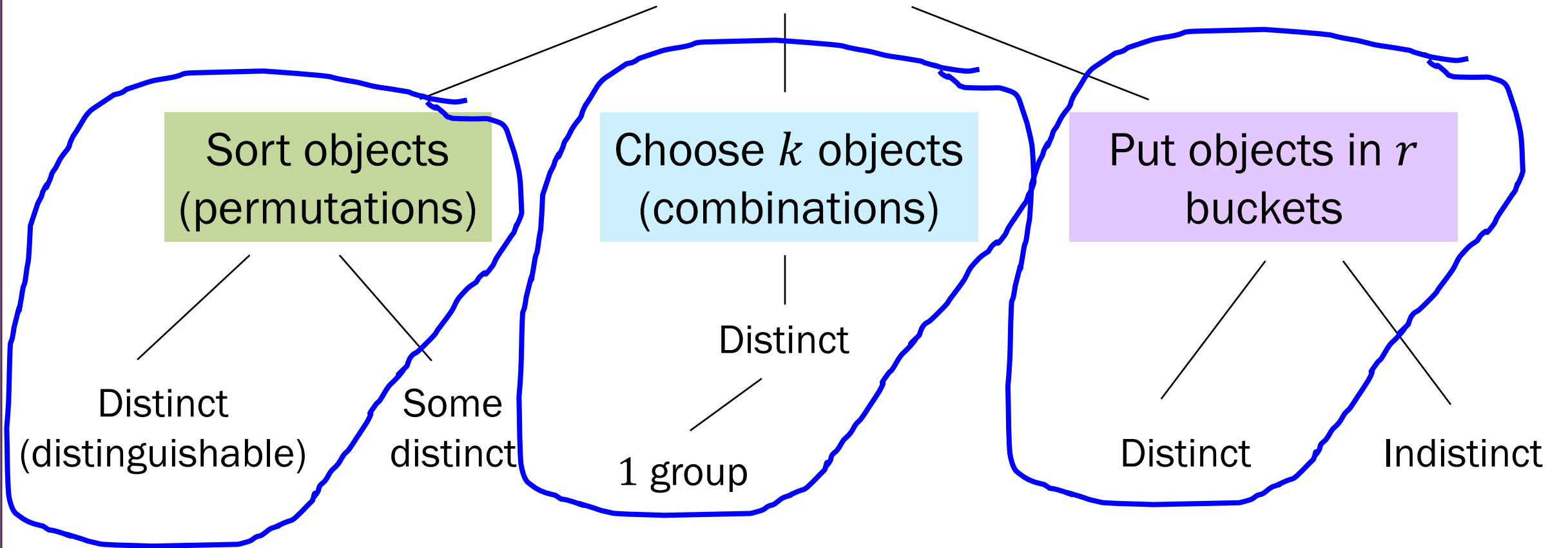
$$x_1 + x_2 + \cdots + x_r = n,$$

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

Positive integer equations can be solved with the divider method.

# Summary of Combinatorics

Counting tasks on  $n$  objects



# To the Code!

---

baob  
bbao  
obba  
oabb  
boab  
babo  
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)
```

# 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)
```

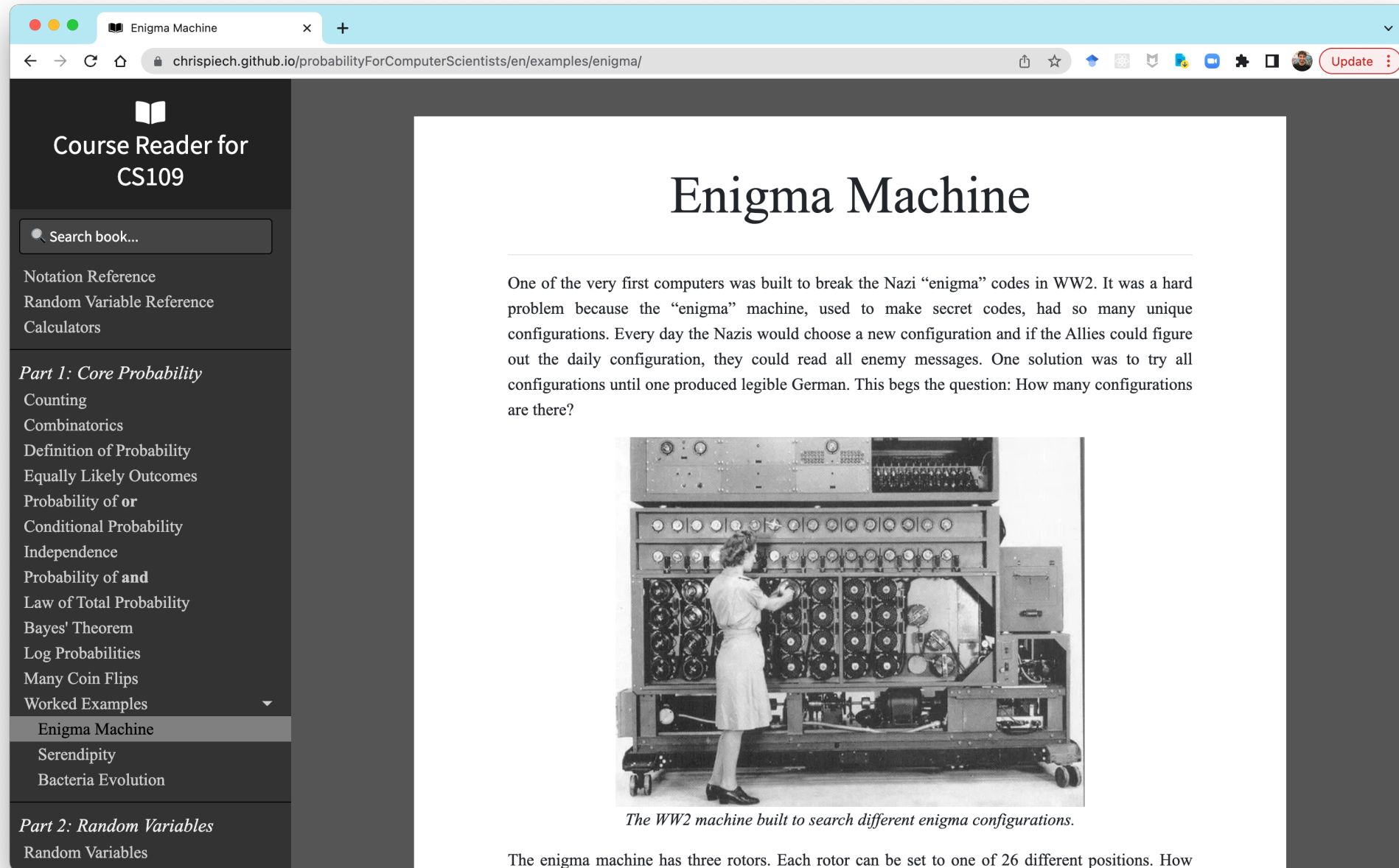
```
def main():  
    total = math.comb(52, 5)  
    print(total)
```

# Ready for the first 4 problems (and the rest on Friday)

Now, more than ever, get started early

The screenshot shows a web browser window with the URL `cs109psets.netlify.app/win22/pset1/robot_paths`. The page title is "PS1 4. Robot Paths". On the left, a vertical navigation bar contains numbered buttons from 1 to 11. Buttons 1 through 3 are green with checkmarks, and button 4 is white with a black border and a black number, all enclosed in a purple hand-drawn circle. Buttons 6 through 11 are grey. The main content area contains the text: "Imagine you have a robot (🤖) that lives on an 7 x 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)?" Below the text is a 7x8 grid. A robot icon is in the top-left cell (row 1, column 1), and a red triangle icon is in the bottom-right cell (row 7, column 8). The grid is labeled "7 rows" on the left and "8 columns" at the bottom. To the right of the grid is an "Answer Editor" section with a "Numeric Answer:" field containing the text "Enter your answer" and a "Check Answer" button. Below that is an "Explanation:" section with a rich text editor toolbar containing icons for Block LaTeX, Image, Bold, Italic, and Underline. At the bottom of the page are "Previous Question" and "Next Question" buttons.

# Want to go deep?



The screenshot shows a web browser window with the address bar displaying `chrispiech.github.io/probabilityForComputerScientists/en/examples/enigma/`. The page content includes a sidebar for a course reader and a main article titled "Enigma Machine".

**Course Reader for CS109**

Search book...

Notation Reference  
Random Variable Reference  
Calculators

*Part 1: Core Probability*

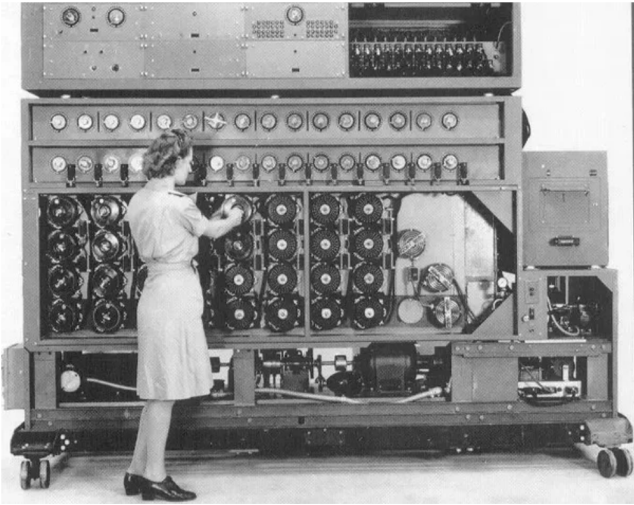
- Counting
- Combinatorics
- Definition of Probability
- Equally Likely Outcomes
- Probability of **or**
- Conditional Probability
- Independence
- Probability of **and**
- Law of Total Probability
- Bayes' Theorem
- Log Probabilities
- Many Coin Flips
- Worked Examples
- Enigma Machine**
- Serendipity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

*Part 2: Random Variables*

- Random Variables

## 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