Job Interview in the AI-Era: Coding, Systems, Agents

Wei Chen

05/18/2026

PURDUE UNIVERSITY

Syllabus

	Content	Learning Objectives	Instructor
Lecture 1 (May 18)	Foundation: Python: Basics review. - Big O - Python basics (quick recap) - Problems: array & list, array two pointers, array sliding windows - Problems: strings, backtracking, matrix - Problems: linked lists Interview formats, recruitment pipelines, and hiring timelines.	**Technical** - Analyze the time complexity of algorithms - Use Leetcode to practice coding problems. **Communication & Strategy** - Hiring Pipelines - Interviewer (vs) Job Hunter Approach - The inverted Triangle Strategy for Job Hunting	Wei Chen Prateek Chennuri
Lecture 2 (May 20)	Foundation: Data & Scalability. - DSA - Problems: HashMap, stacks, queues - Problems: binary trees, binary search trees - Problems: heaps, graph, combined pattern How to talk to recruiters, professional email etiquette, and phone screens.	**Technical** - Identify better code (time/space) - Understand the coding interview process.	Wei Chen Prateek Chennuri

Syllabus

	Content	Learning Objectives	Instructor
Lecture 3 (May 22)	Foundation: AI agent. Elements of AI agents, usage of popular coding agents, and effective prompt drafting. How to ask for hints or clarification effectively.	**Technical** - Learn basic concepts of an AI agent. - Use the coding agent to assist coding and prepare for interviews. **Communication & Strategy** - Ask for clarification effectively during coding interviews.	Zhaoying Pan
Lecture 4 (May 26)	ML: Data Pipelines: Preprocessing, ETL (Extract, Transform, Load), and handling data noise. Engineering skill: - getting started with an ML repo quickly Problem solving: - principles - toolkit - how to train	**Technical** - Understand the basic idea of a data pipeline **Communication & Strategy** - Understand methodology for getting started on an ML repo - Understand principles of problem solving	Deming Chu

PURDUE UNIVERSITY

Syllabus

	Content	Learning Objectives	Instructor
Lecture 5 (May 27)	ML: Model Lifecycle: Training loops, loss functions, and evaluation metrics (Precision/Recall). How to make an elevator pitch about your resume and adapt it for specific engineering or ML roles.	**Technical** - Understand the general ML pipelining - Understand general ML system design from ideation to deployment **Communication & Strategy** - Tailor resumes for a given target role - Prepare yourself for the role.	Suhas Dara
Lecture 6 (May 29)	AI-assisted coding Panel discussion with industry guests and students with recent interview success.	**Technical** - Familiarize students with Coding interview platforms such as Coderpad. **Communication & Strategy** - Panel Discussion	Gaurav Patel

PURDUE UNIVERSITY

Lecture 1

Foundation: Python

Outline

- Big O
- Python basics (quick recap)
- Coding problems
- Data structure in AI

PURDUE UNIVERSITY

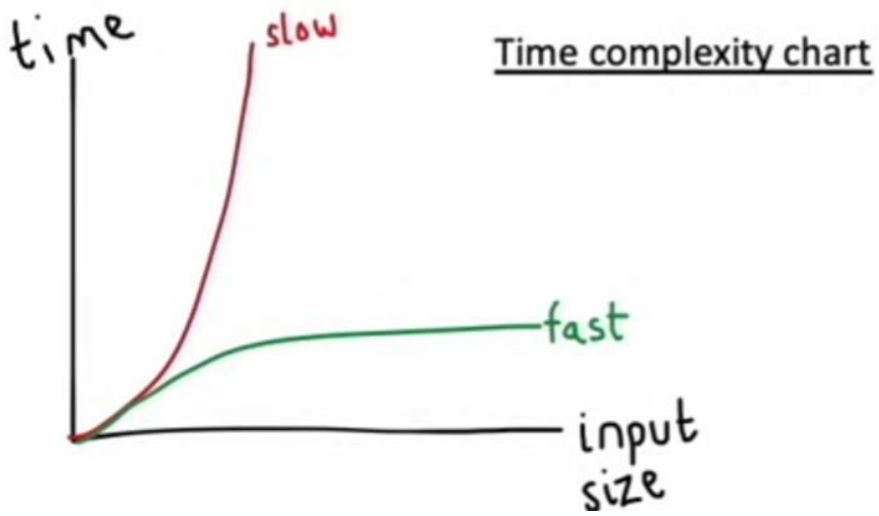
Reference

- https://leetcode.com/discuss/post/5886397/dsa-patterns-you-need-to-know-by-anubhav-x7og/
- https://docs.google.com/spreadsheets/d/1EEYzyD_483B-7CmWxsJB_zycdv4Y5dxnzcoEQtaIfuk/edit?pli=1&gid=329533698#gid=329533698
- https://leetcode.com/problemset/
- https://www.youtube.com/watch?app=desktop&v=IvO88XxNAzs
- https://www.youtube.com/watch?v=vVL6NFzr0Rg

Big (O)

Time and Space Complexity

- **Time Complexity:** describes the amount of time necessary to execute an algorithm
- **Space Complexity:** describes the amount of memory or space utilized by an algorithm/program



PURDUE UNIVERSITY

Big (O)

Time and Space Complexity

- **Why do we need Big O?**
- Big O notation helps us understand how the performance of an algorithm changes as the size of the input grows, providing a simple way to compare and analyze different algorithms' **efficiency**.

Big (O)

Linear Example

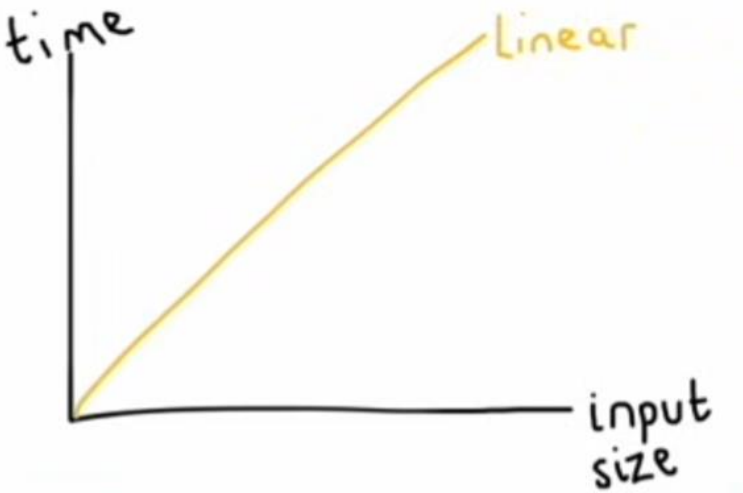
Let's say we are given a problem where we have a list of N numbers (unknown length). We do not know what numbers are in the list.

We are asked to use code to find and return "True" if the number 2 is in list. And if the number is not in the list we return "False".

Our solution could be to go through every position in the list and check if the number at that position is equal to 2.

[3, 10, 2, 7]

```
For number in list:
        if number == 2:
                return True
        else:
                continue
return False
```

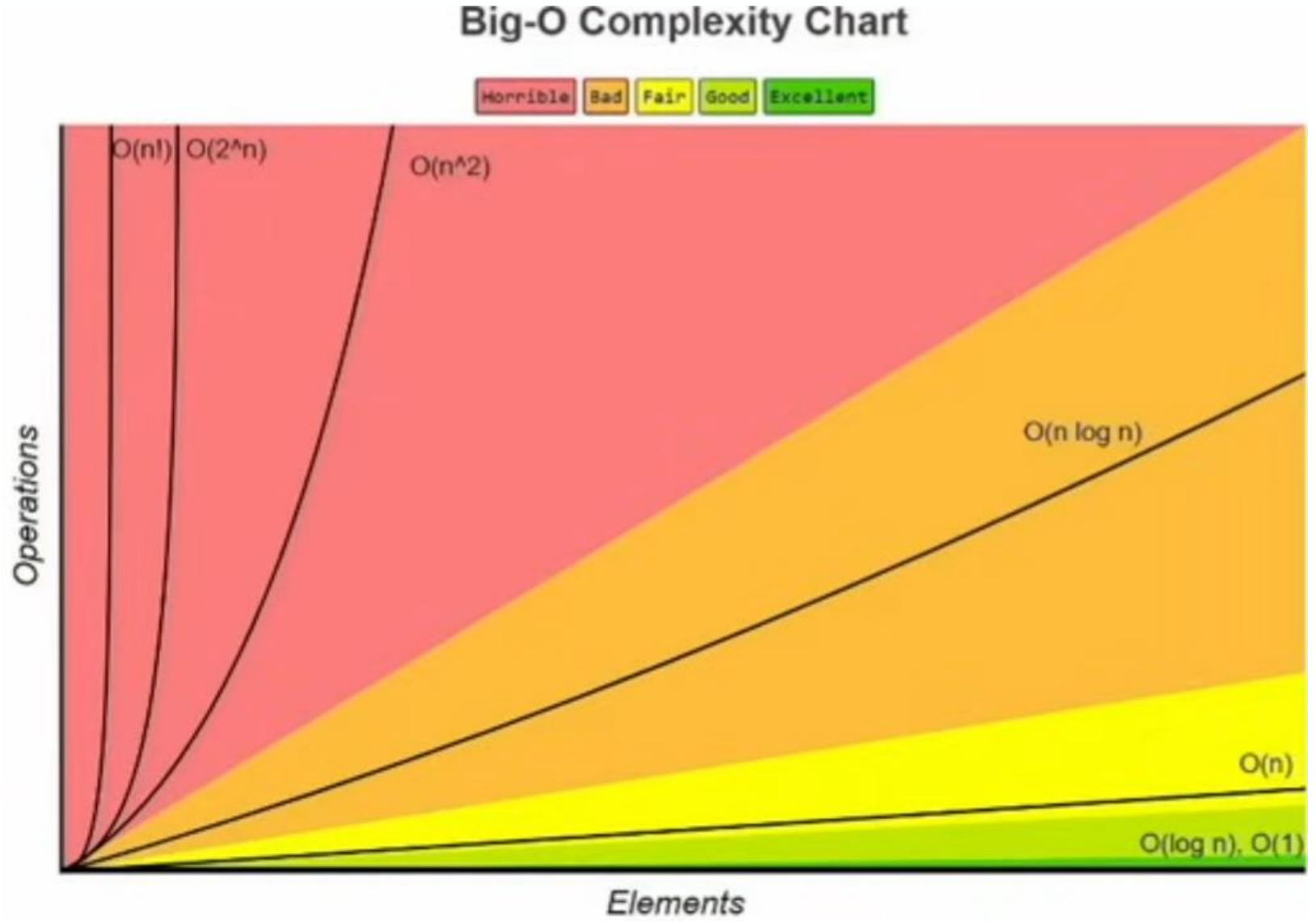


To loop through the entire list of N numbers, and make sure that 2 is or is not in the list, it would take N time. We need to check every number in the list once. Making this solution O(N) – linear time. The longer the list, (the bigger N is), the longer it would take.

PURDUE UNIVERSITY

Big (O)

Reference Chart



PURDUE UNIVERSITY

Python basics

- Variables & data types: dynamically typed
  - Key Types: int, float, bool, str, list, tuple, dict, set
- List: ordered, mutable
  - Append, access, insert, delete
- Dictionaries (HashMaps)
  - Key value store
  - O(1) lookup
- Sets: unordered, unique elements
  - Membership test: O(1)

Python basics

List

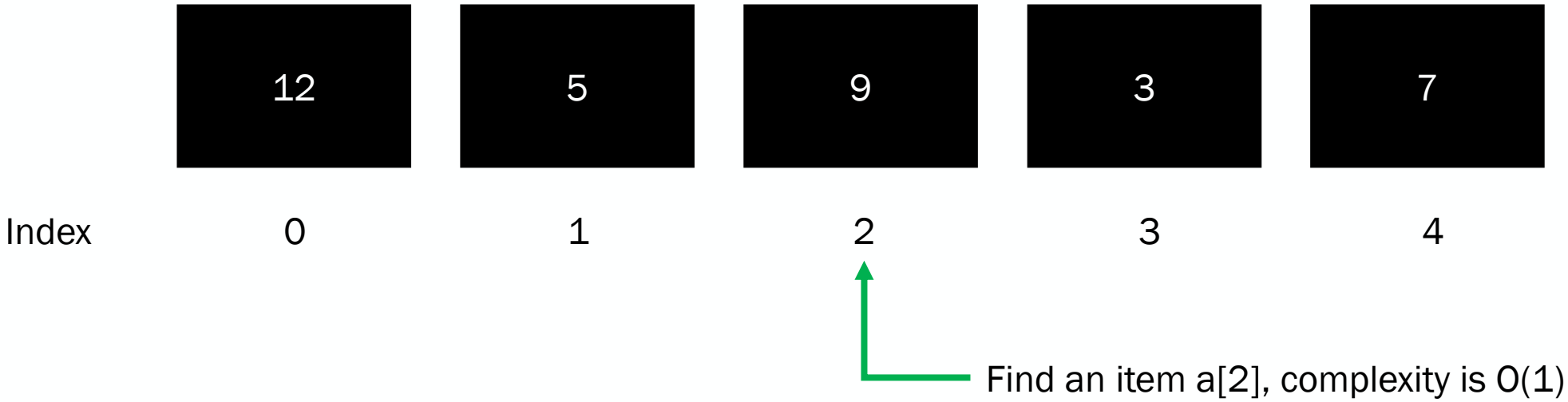
PURDUE UNIVERSITY

Python basics

List

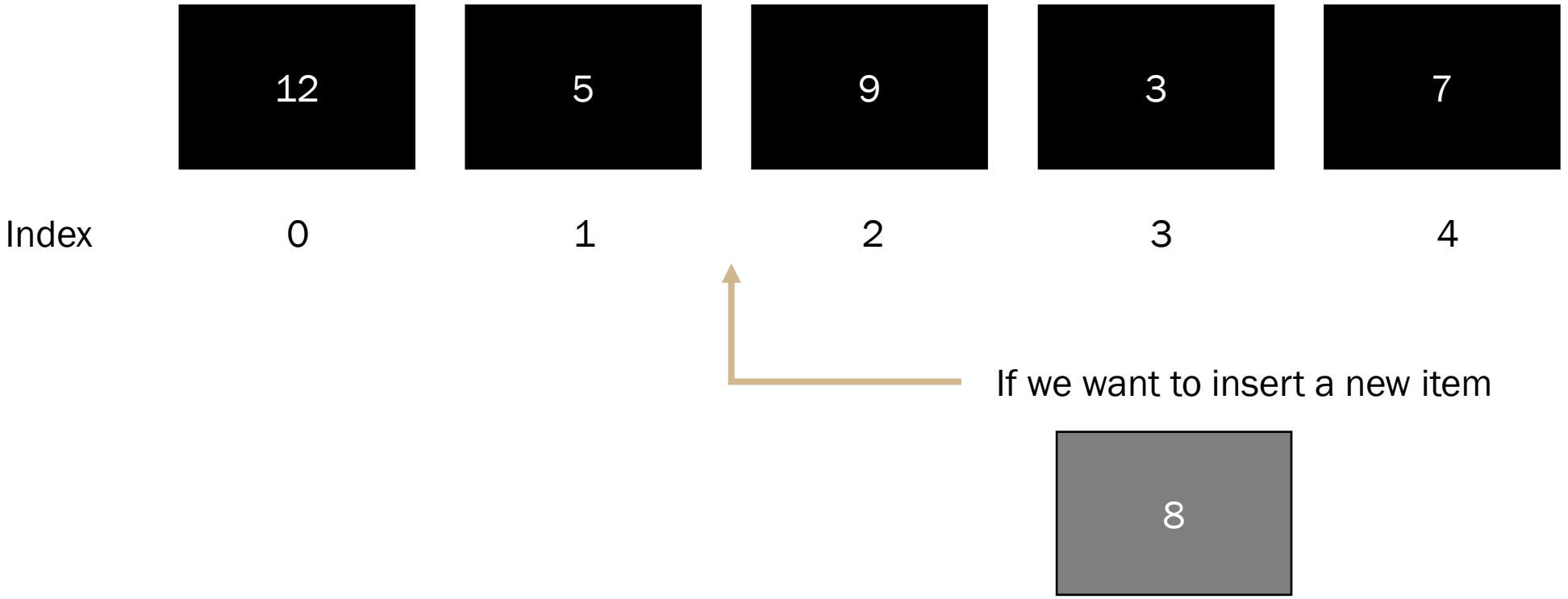
Python basics

List: access



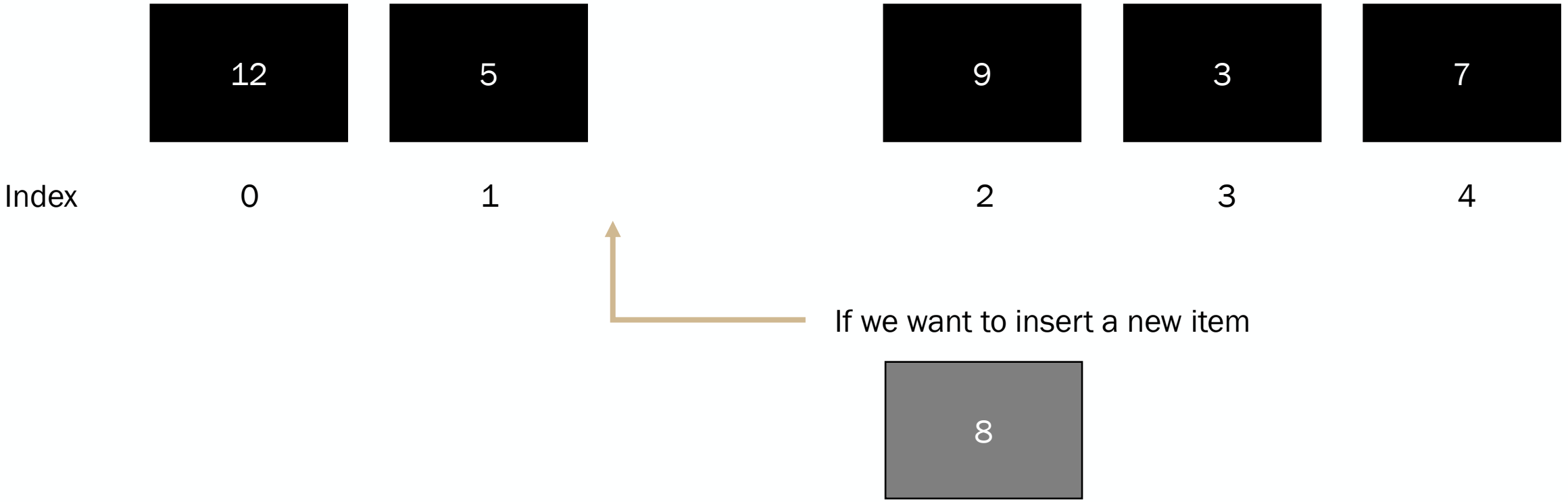
Python basics

List: insert



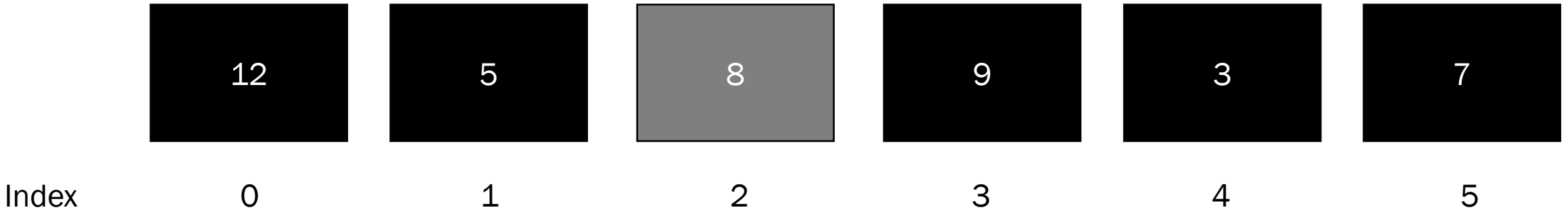
Python basics

List: insert



Python basics

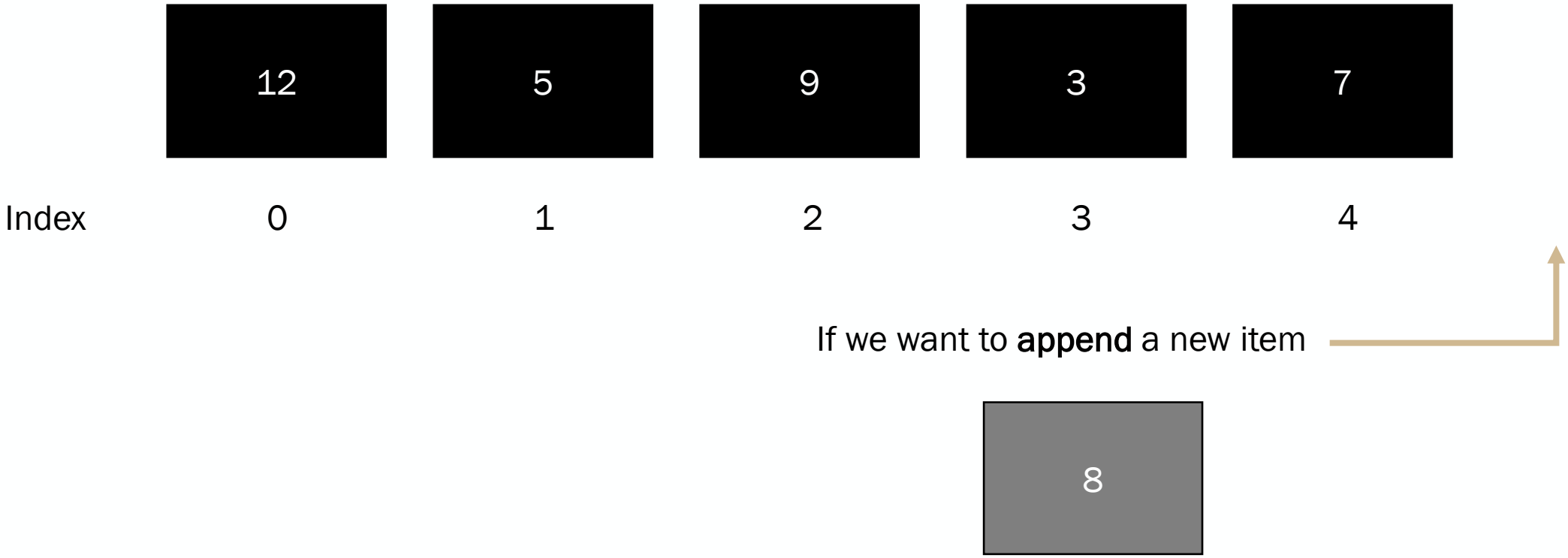
List: insert



The complexity is O(n)

Python basics

List: append



Python basics

List: append

The complexity is O(1)

Python basics

List: syntax

```python
# Create a list
numbers = [1, 2, 3, 4, 5]
names = ["Alice", "Bob", "Charlie"]
mixed = [1, "hello", 3.14, True]

# Access elements
numbers[0]      # first element: 1
numbers[-1]     # last element: 5

# Modify elements
numbers[1] = 20
# numbers is now [1, 20, 3, 4, 5]

# Add elements
numbers.append(6)        # add to the end
numbers.insert(0, 0)     # insert at index 0

# Remove elements
numbers.remove(20)       # remove by value
numbers.pop()            # remove last element
numbers.pop(0)           # remove element at index 0
```

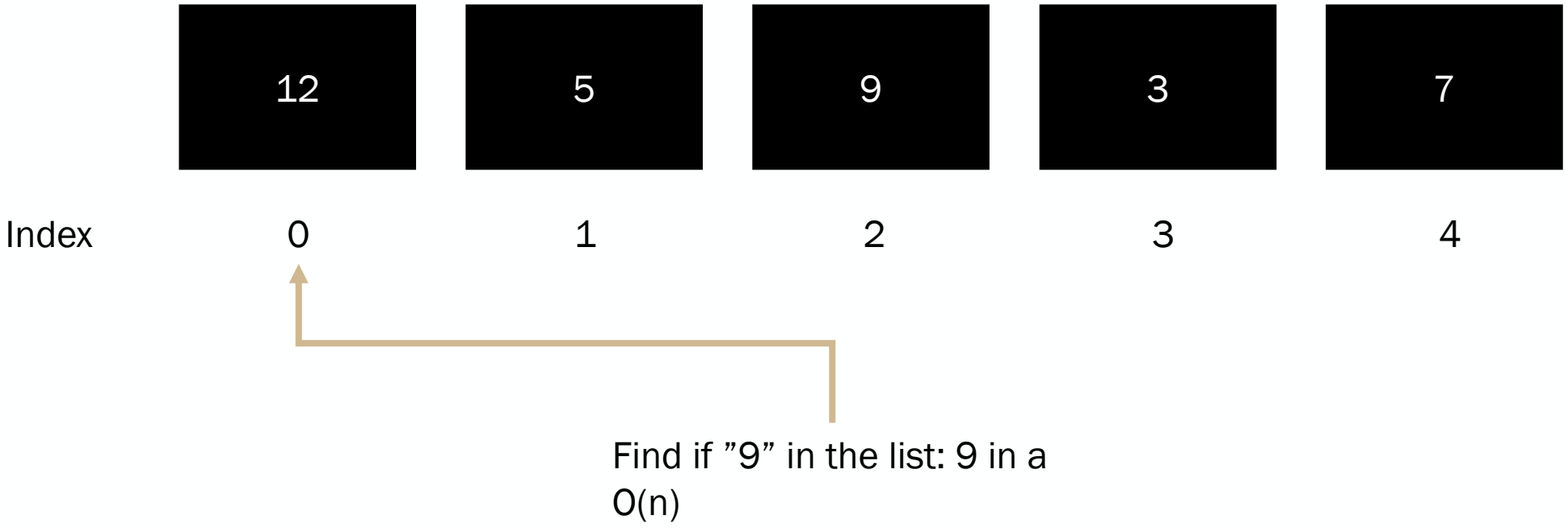
```python
# List length
len(numbers)

# Slicing
numbers[1:4]     # elements from index 1 to 3
numbers[:3]      # first 3 elements
numbers[2:]      # from index 2 to end

# Loop through a list
for n in numbers:
    print(n)

# Check if an item exists
if 3 in numbers:
    print("3 is in the list")
```

```python
numbers.sort()       # sort list
numbers.reverse()    # reverse list
numbers.clear()      # remove all elements
numbers.copy()       # make a copy
numbers.index(3)     # find index of value 3
numbers.count(3)     # count occurrences of 3
```

PURDUE UNIVERSITY

Python basics

List: index

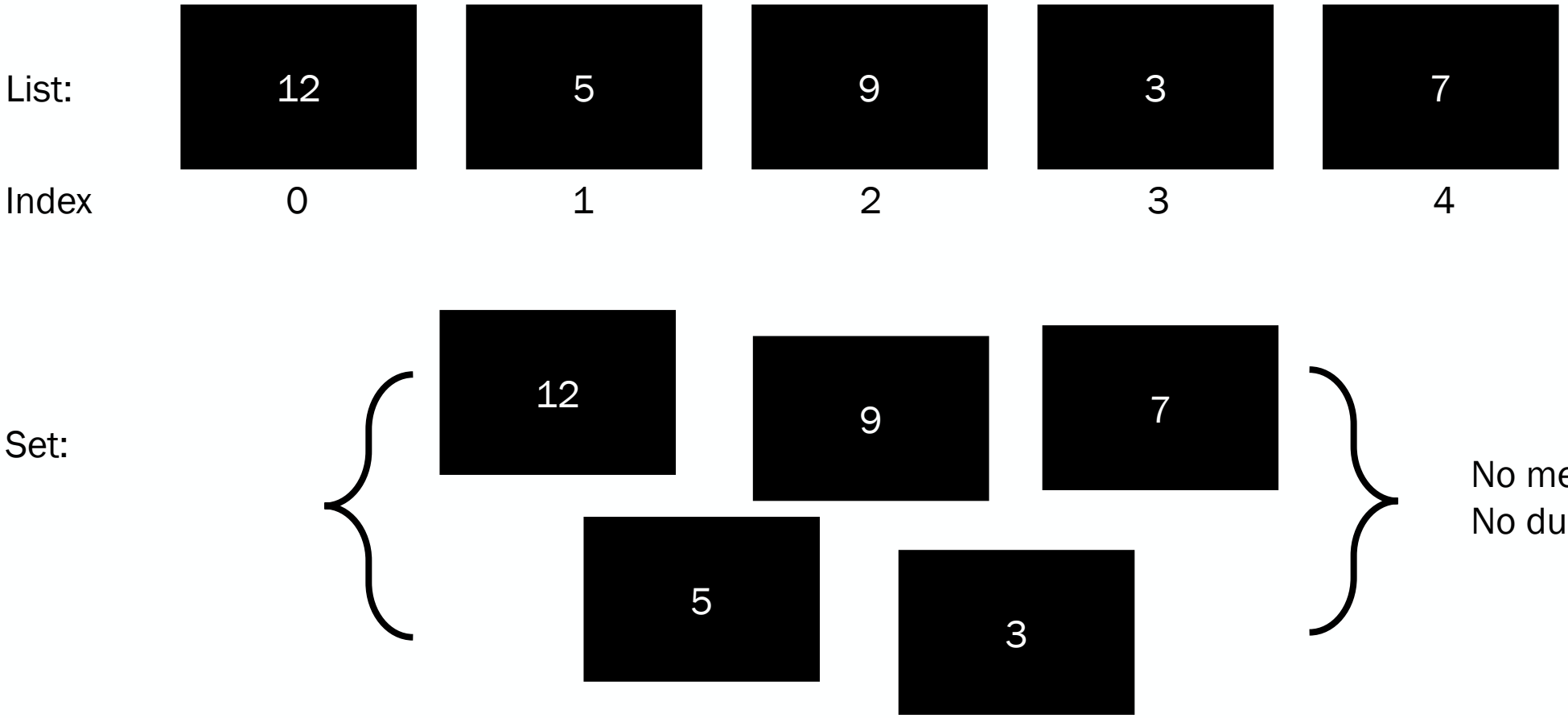


Python basics

Set

Python basics

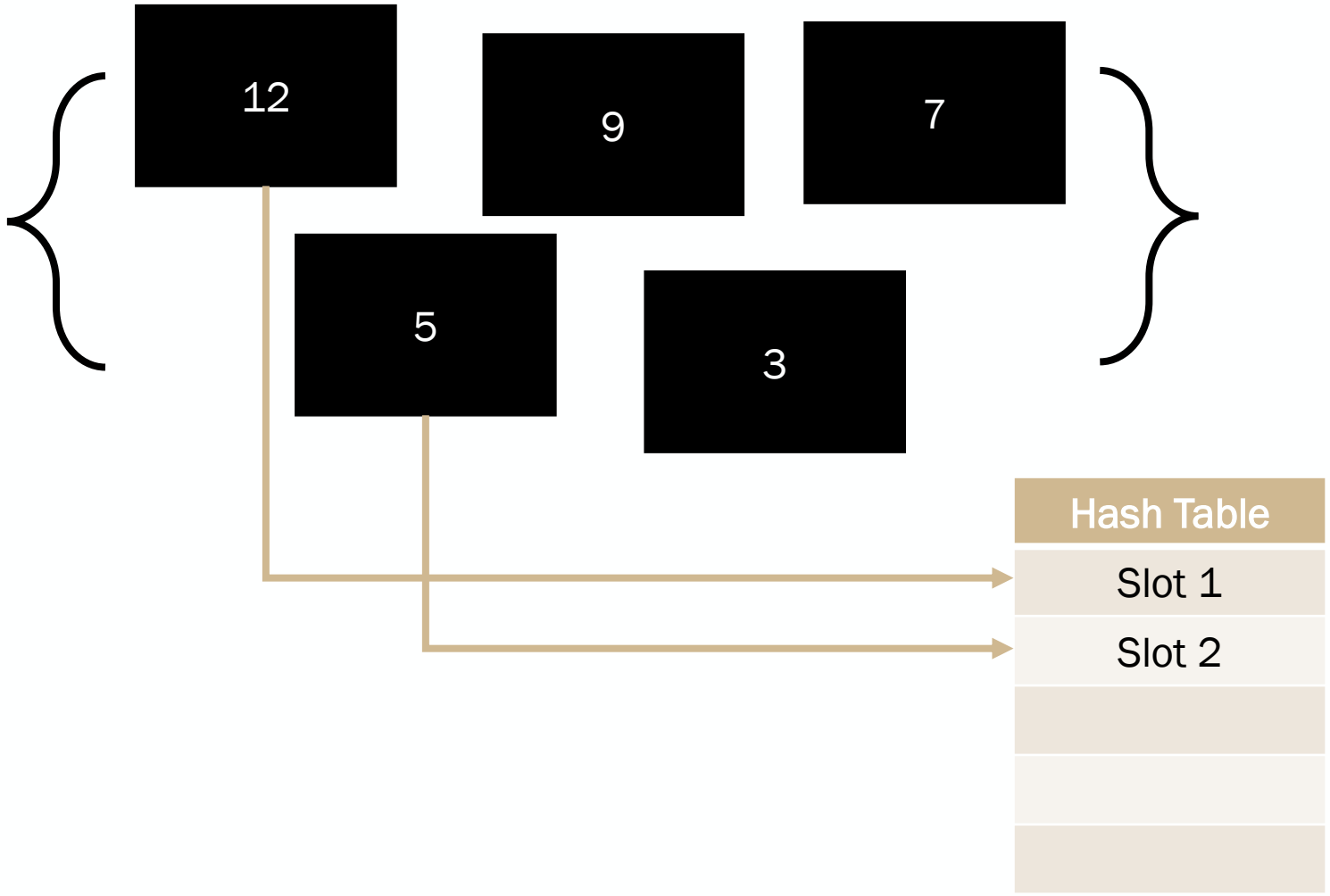
Set



Python basics

Set

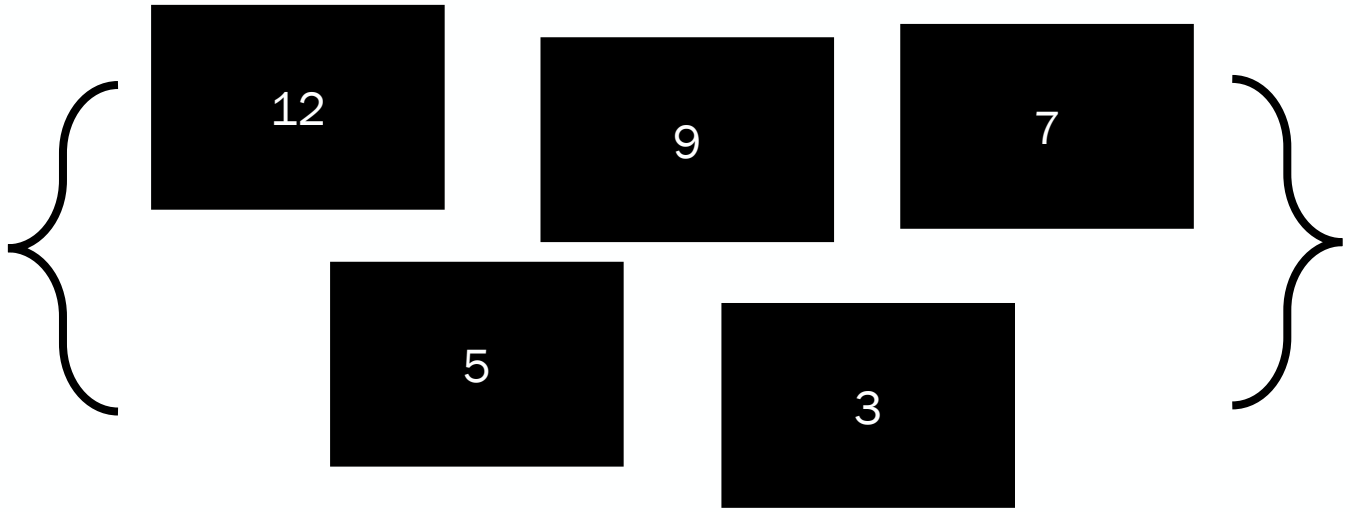
Set:



Python basics

Set: check membership

Set:



9 in s, O(1)

Python basics

Set: syntax

```
# Create a set
nums = {1, 2, 3, 4}
names = {"Alice", "Bob", "Charlie"}

# Empty set
empty = set()      # correct
not_set = {}       # this is an empty dictionary
```

```
nums = {1, 2, 2, 3}
print(nums)   # {1, 2, 3}
```

```
nums.add(5)        # add one element
nums.remove(2)     # remove 2, error if not found
nums.discard(10)   # remove 10 if it exists, no error
nums.pop()         # remove a random element
nums.clear()       # remove all elements
```

```
if 3 in nums:
    print("3 is in the set")
```

```
for x in nums:
    print(x)
```

```
a = {1, 2, 3}
b = {3, 4, 5}

a | b   # union: {1, 2, 3, 4, 5}
a & b   # intersection: {3}
a - b   # difference: {1, 2}
a ^ b   # symmetric difference: {1, 2, 4, 5}
```

```
a.union(b)
a.intersection(b)
a.difference(b)
```

PURDUE UNIVERSITY

Coding Problems

Coding interview guideline

- Coding interview is about both **coding skills** and **communication**
- Process:
  - Read the question, ask clarifying questions
  - Give examples
  - Discuss brute-force solution and improve the solution (if you know the optimal answer, you can skip this)
  - Write clean code, test with edge cases
  - Analyze time and space complexity

Coding Problems

Leetcode

- Try to use LeetCode

Problems: Arrays & Lists

Leetcode

217. Contains Duplicate

Solved

Easy | Topics | Companies

Given an integer array `nums`, return `true` if any value appears **at least twice** in the array, and return `false` if every element is distinct.

Example 1:

> **Input:** `nums = [1,2,3,1]`
>
> **Output:** `true`
>
> **Explanation:**
>
> The element 1 occurs at the indices 0 and 3.

Example 2:

> **Input:** `nums = [1,2,3,4]`
>
> **Output:** `false`
>
> **Explanation:**
>
> All elements are distinct.

Example 3:

> **Input:** `nums = [1,1,1,3,3,4,3,2,4,2]`
>
> **Output:** `true`

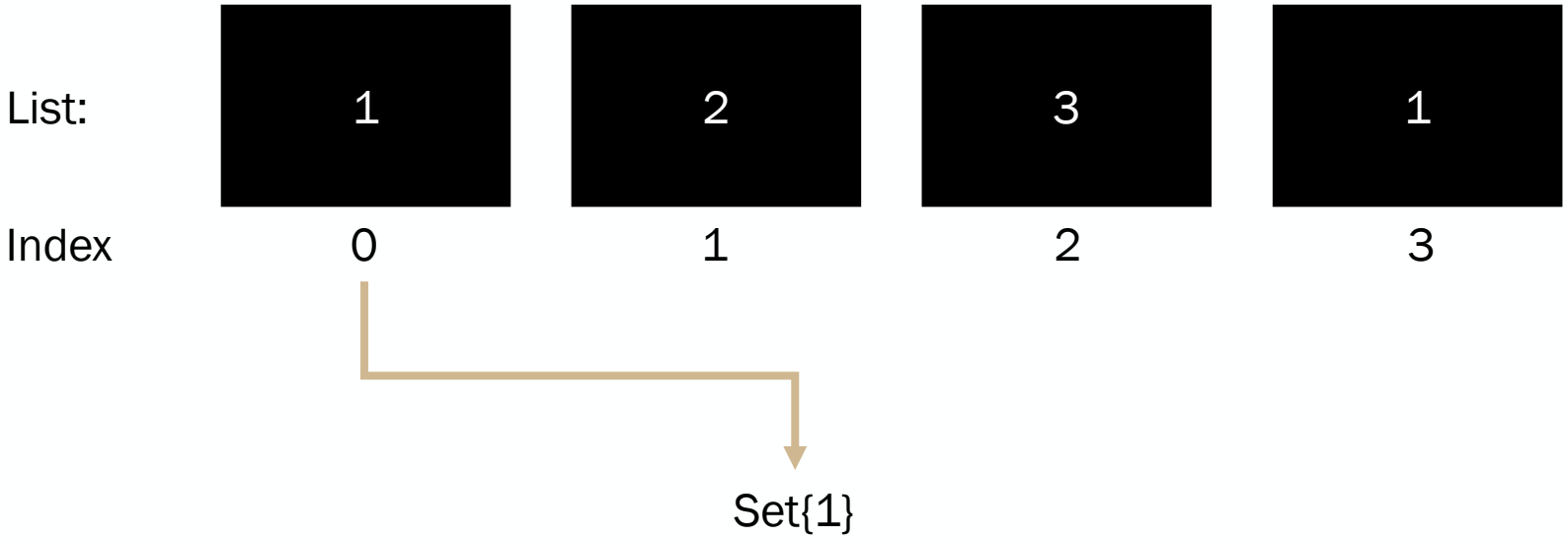
PURDUE UNIVERSITY

Problems: Arrays & Lists

Contains Duplicate

Example 1:

> **Input:** `nums = [1,2,3,1]`
>
> **Output:** `true`
>
> **Explanation:**
>
> The element 1 occurs at the indices 0 and 3.



Problems: Arrays & Lists

Contains Duplicate II

219. Contains Duplicate II

Easy | Topics | Companies

Given an integer array `nums` and an integer `k`, return `true` *if there are two* ***distinct indices*** `i` *and* `j` *in the array such that* `nums[i] == nums[j]` *and* `abs(i - j) <= k`.

Example 1:

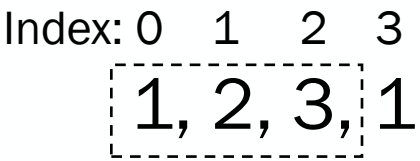
```
Input: nums = [1,2,3,1], k = 3
Output: true
```

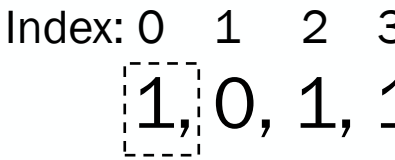
Example 2:

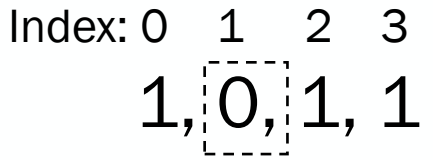
```
Input: nums = [1,0,1,1], k = 1
Output: true
```

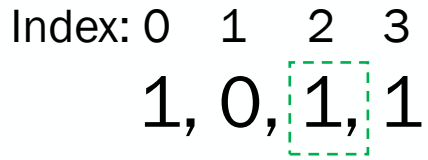
Example 3:

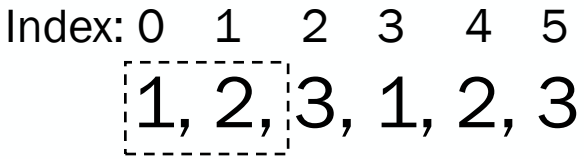
```
Input: nums = [1,2,3,1,2,3], k = 2
Output: false
```

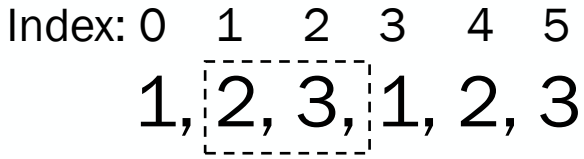


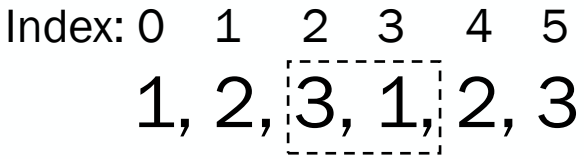


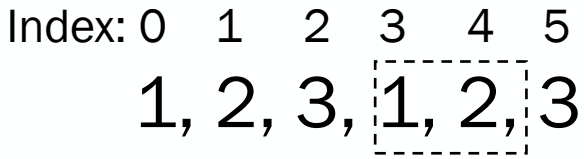








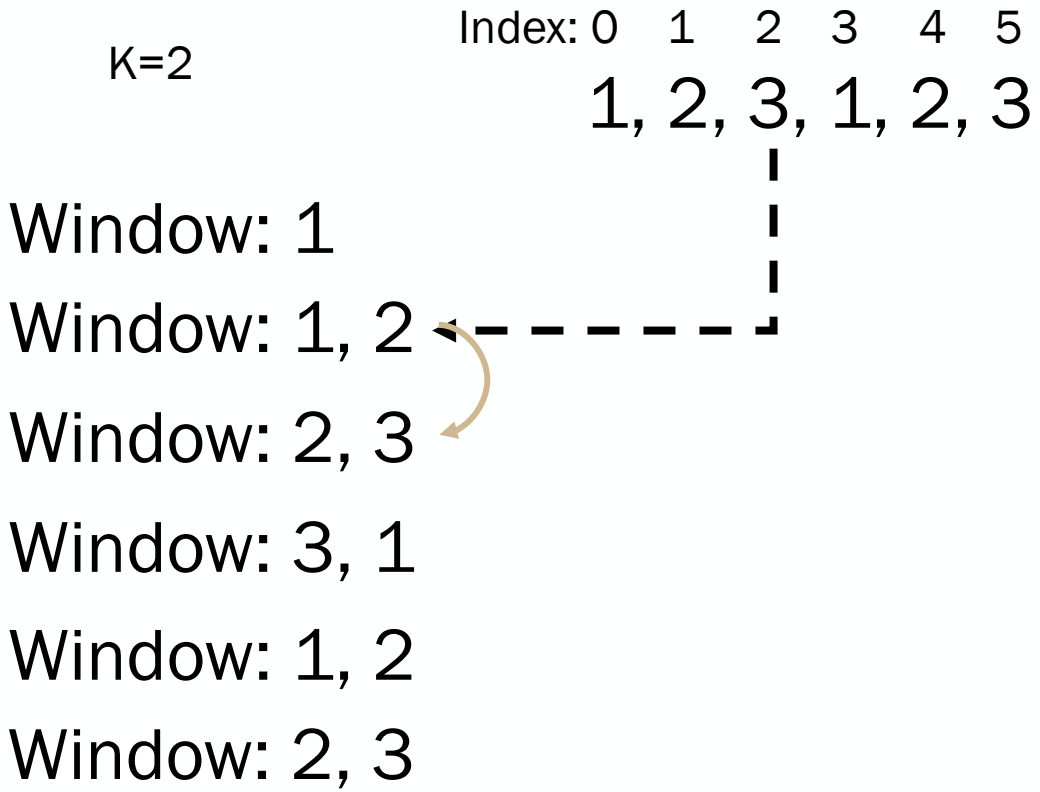




PURDUE UNIVERSITY

Problems: Arrays & Lists

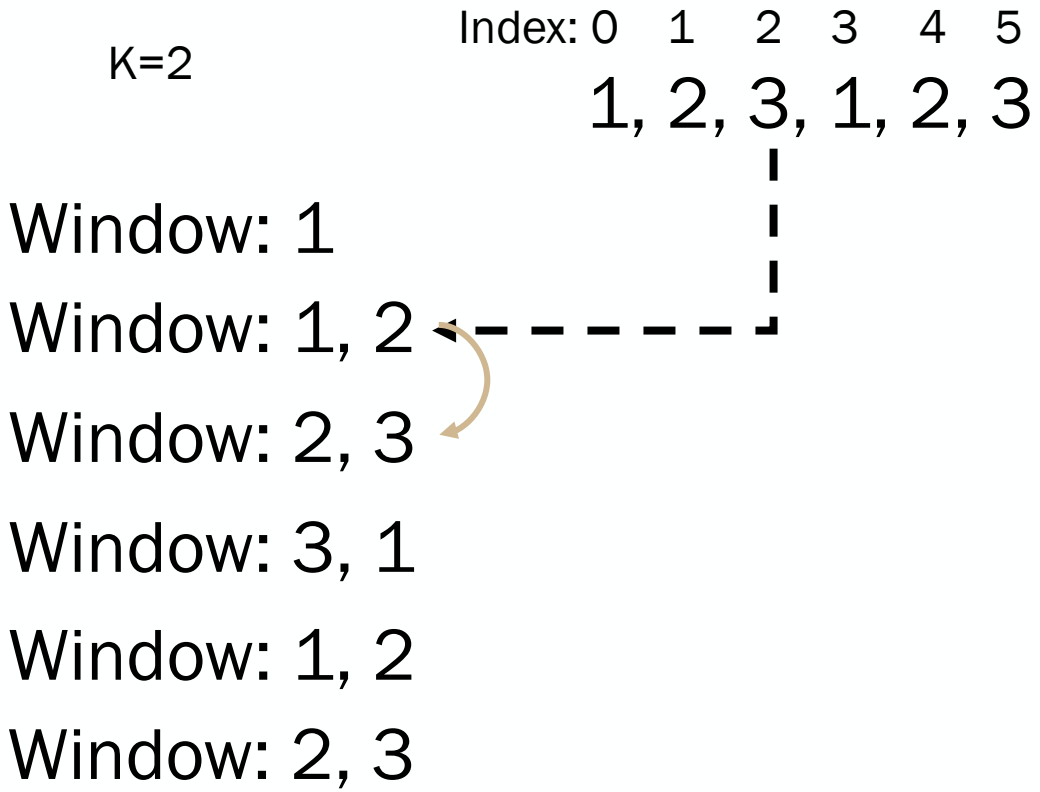
Contains Duplicate II



PURDUE UNIVERSITY

Problems: Arrays & Lists

Contains Duplicate II

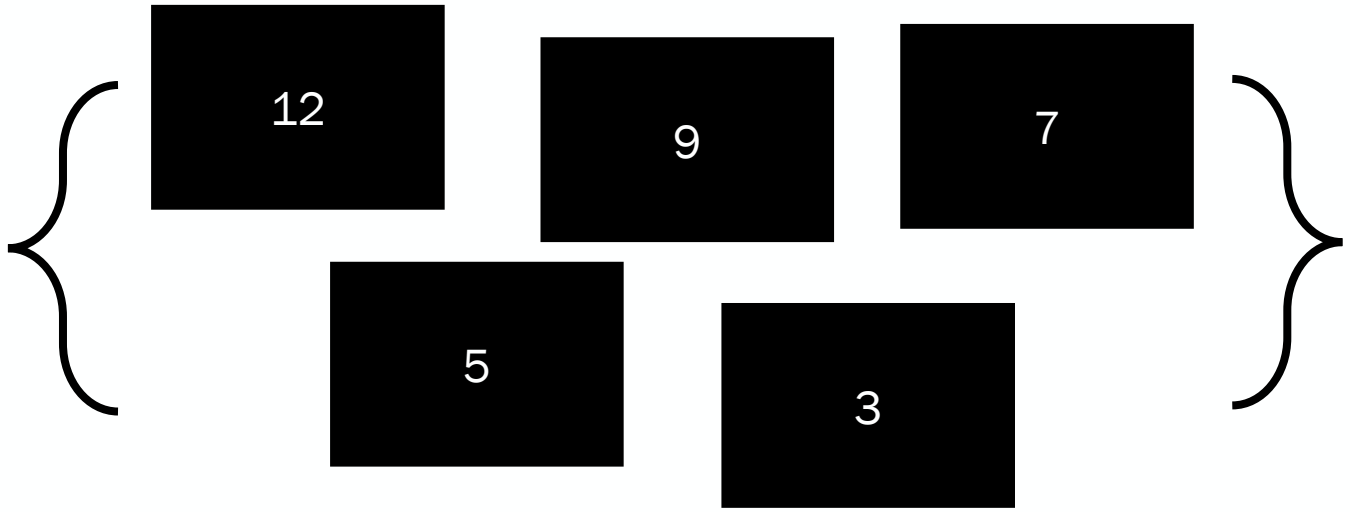


Edge cases:
K=0? List is empty? K is negative?

Python basics

Dictionary

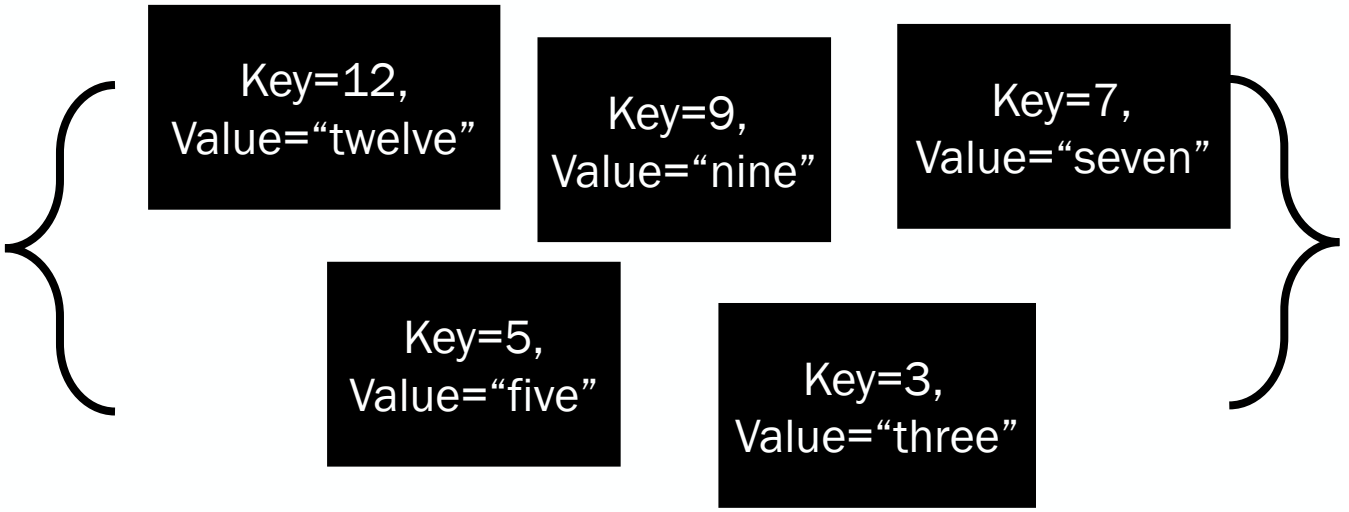
Set:



Python basics

Dictionary

Dictionary:



Python basics

Dictionary: syntax

```python
# Create a dictionary
student = {
    "name": "Alice",
    "age": 20,
    "major": "Computer Science"
}

# Access values by key
print(student["name"])   # Alice
print(student["age"])    # 20

# Safely access values with get()
print(student.get("name"))          # Alice
print(student.get("grade"))         # None
print(student.get("grade", "N/A"))  # N/A

# Add a new key-value pair
student["year"] = "Sophomore"

# Update an existing value
student["age"] = 21
```

```python
# Check if a key exists
if "name" in student:
    print("name exists")

# Remove values
student.pop("major")   # remove by key and return value
del student["age"]     # remove by key

# Loop through keys
for key in student:
    print(key)

# Loop through values
for value in student.values():
    print(value)

# Loop through key-value pairs
for key, value in student.items():
    print(key, value)
```

```python
# Get all keys, values, and items
student.keys()
student.values()
student.items()
```

PURDUE UNIVERSITY

Problems: Arrays & Lists

Two sum

1. Two Sum

Solved

Easy | Topics | Companies | Hint

Given an array of integers `nums` and an integer `target`, return *indices of the two numbers such that they add up to* `target`.

You may assume that each input would have ***exactly* one solution**, and you may not use the *same* element twice.

You can return the answer in any order.

Example 1:

> **Input:** nums = [2,7,11,15], target = 9
> **Output:** [0,1]
> **Explanation:** Because nums[0] + nums[1] == 9, we return [0, 1].

Example 2:

> **Input:** nums = [3,2,4], target = 6
> **Output:** [1,2]

Example 3:

> **Input:** nums = [3,3], target = 6
> **Output:** [0,1]

PURDUE UNIVERSITY

Problems: Arrays & Lists

Two sum

Example 1:

```
Input: nums = [2,7,11,15], target = 9
Output: [0,1]
Explanation: Because nums[0] + nums[1] == 9, we return [0, 1].
```

	2	7	11	15
2				
7				
11				
15				

PURDUE UNIVERSITY

Problems: Arrays & Lists

Two sum

Example 1:

> **Input:** nums = [2,7,11,15], target = 9
> **Output:** [0,1]
> **Explanation:** Because nums[0] + nums[1] == 9, we return [0, 1].

$O(n^2)$

	2	7	11	15
2	4	9	13	17
7	9	14	18	22
11	13	18	22	26
15	17	22	26	30

Problems: Arrays & Lists

Two sum

Example 1:

> **Input:** nums = [2,7,11,15], target = 9
> **Output:** [0,1]
> **Explanation:** Because nums[0] + nums[1] == 9, we return [0, 1].

	2	7	11	15
2	4	9	13	17
7	9	14	18	22
11	13	18	22	26
15	17	22	26	30

Problems: Arrays & Lists

Two sum

Example 1:

> **Input:** nums = [2,7,11,15], target = 9
> **Output:** [0,1]
> **Explanation:** Because nums[0] + nums[1] == 9, we return [0, 1].

	2	7	11	15
2	4	9	13	17
7		14	18	22
11			22	26
15				30

Problems: Arrays & Lists

Two sum

Example 1:

> **Input:** nums = [2,7,11,15], target = 9
> **Output:** [0,1]
> **Explanation:** Because nums[0] + nums[1] == 9, we return [0, 1].

	2	7	11	15
2	4	9	13	17
7		14	18	22
11			22	26
15				30

Problems: Arrays & Lists

Two sum

Example 1:

> **Input:** nums = [2,7,11,15], target = 9
> **Output:** [0,1]
> **Explanation:** Because nums[0] + nums[1] == 9, we return [0, 1].

	2	7	11	15
2		9	13	17
7			18	22
11				26
15				

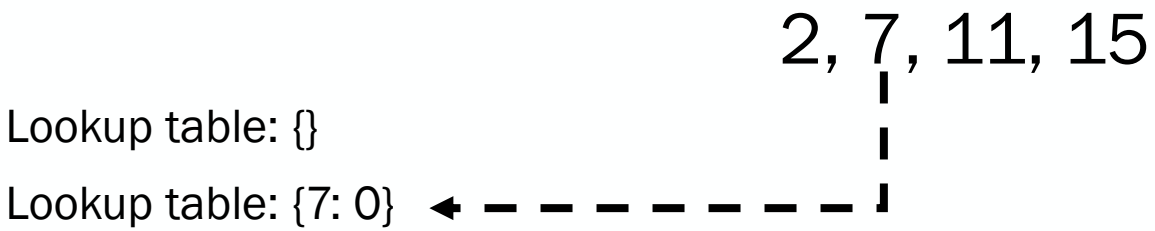
PURDUE UNIVERSITY

Problems: Arrays & Lists

Two sum

Example 1:

> **Input:** nums = [2,7,11,15], target = 9
> **Output:** [0,1]
> **Explanation:** Because nums[0] + nums[1] == 9, we return [0, 1].



Problems: Arrays & Lists

Count Number of Pairs With Absolute Difference K

2006. Count Number of Pairs With Absolute Difference K

Easy | Topics | Companies | Hint

Given an integer array `nums` and an integer `k`, return *the number of pairs* `(i, j)` *where* `i < j` *such that* `|nums[i] - nums[j]| == k`.

The value of `|x|` is defined as:

- `x` if `x >= 0`.
- `-x` if `x < 0`.

Example 1:

Input: nums = [1,2,2,1], k = 1
Output: 4
Explanation: The pairs with an absolute difference of 1 are:
- [**1**,**2**,2,1]
- [**1**,2,**2**,1]
- [1,**2**,2,**1**]
- [1,2,**2**,**1**]

	1, 2, 2, 1
1	1, 1, 0
2	0, 1
2	1
1	

$O(n^2)$

Any better method?

Problems: Arrays & Lists

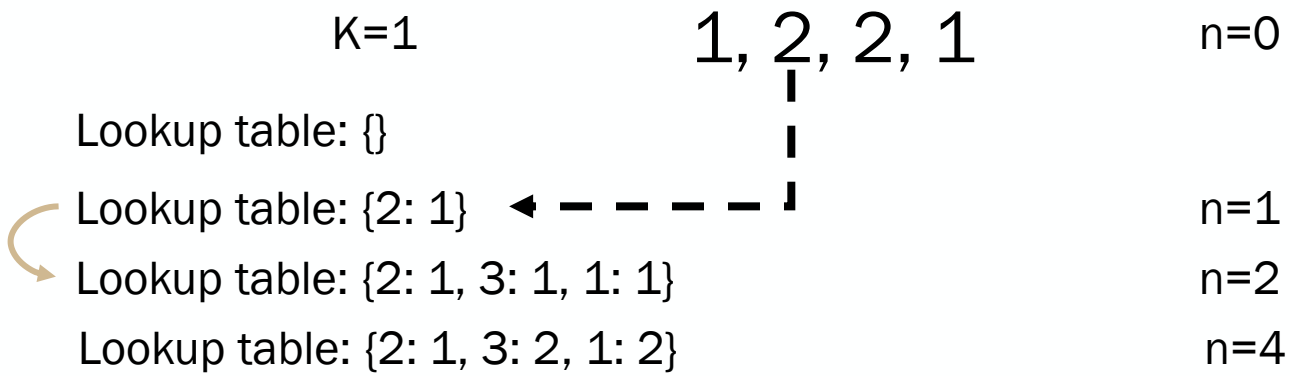
Count Number of Pairs With Absolute Difference K

1, 2, 2, 1

Lookup table: {?}

Problems: Arrays & Lists

Count Number of Pairs With Absolute Difference K



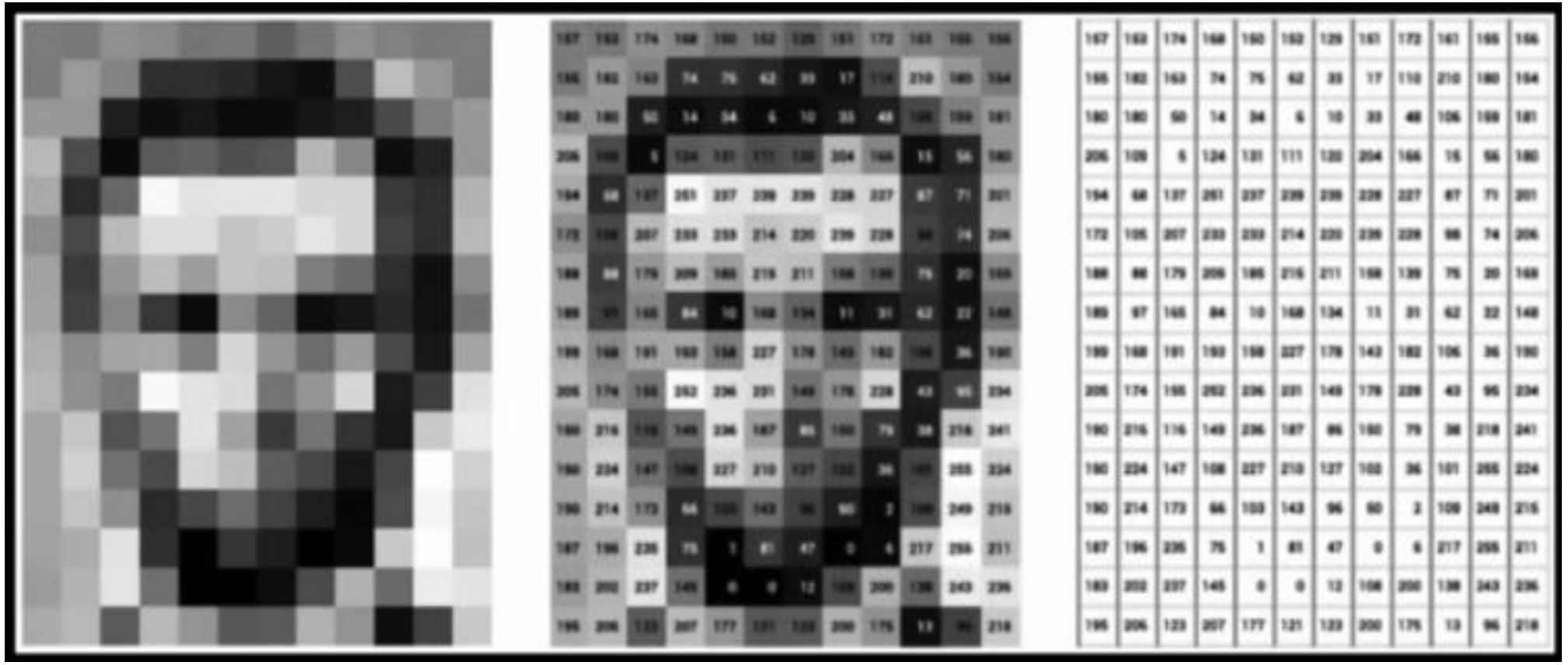
Problems: Arrays & Lists

Leetcode

- Contains Duplicate: https://leetcode.com/problems/contains-duplicate/description/
- Missing number: https://leetcode.com/problems/missing-number/description/
- Find all numbers disappeared in an Array: https://leetcode.com/problems/find-all-numbers-disappeared-in-an-array/description/
- Two sum:
- Move zeroes
- How Many Numbers Are Smaller Than the Current Number
- Minimum Time Visiting All Points
- Spiral Matrix
- Number of Islands

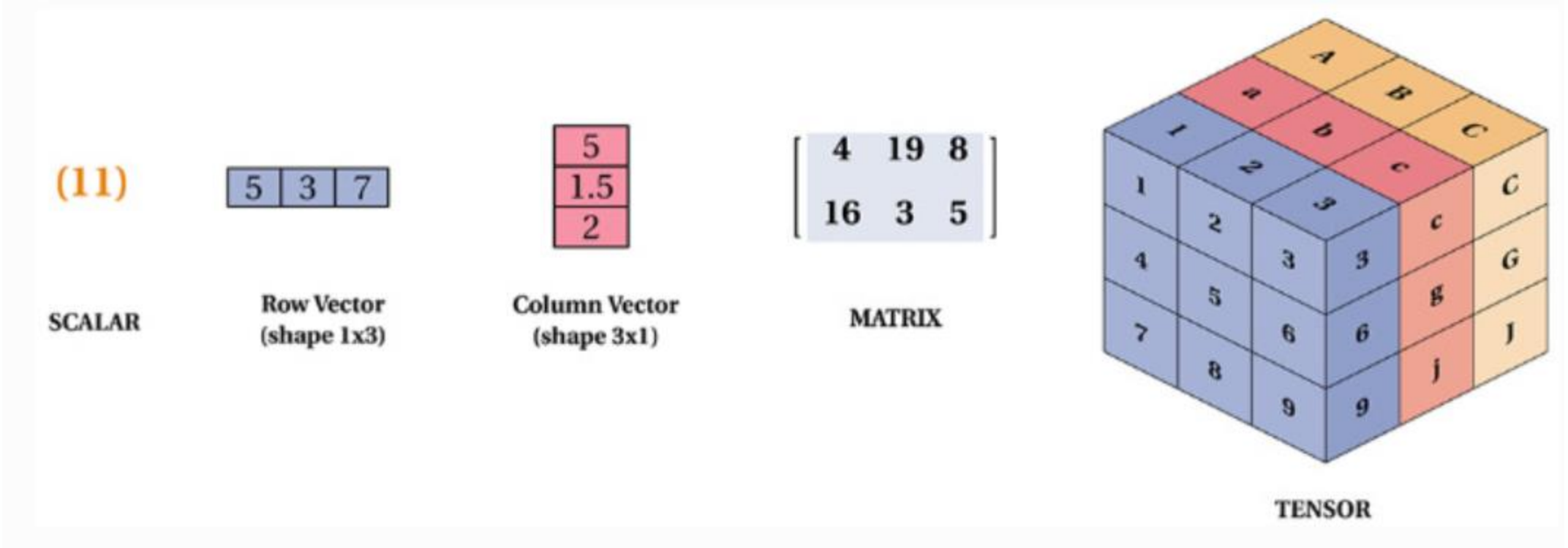
Data Structure for AI

Array & Tensor



Data Structure for AI

Array & Tensor



Data Structure for AI

Array & Tensor, Syntax

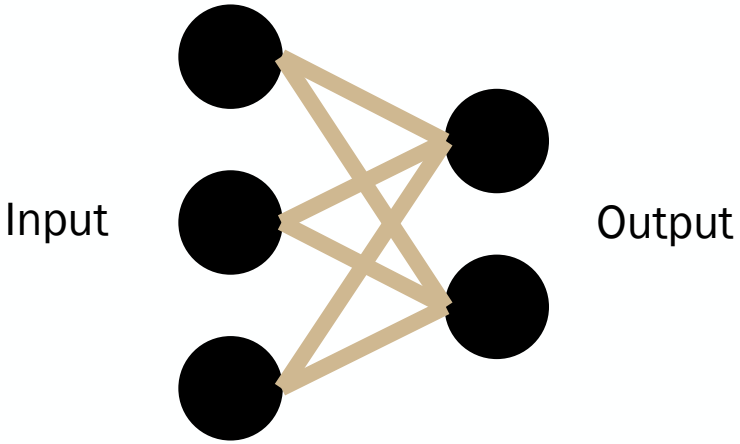
```python
import torch

# Create tensors
x = torch.tensor([1, 2, 3])
y = torch.tensor([[1, 2], [3, 4]])

zeros = torch.zeros(2, 3)
ones = torch.ones(2, 3)
random = torch.randn(2, 3)

print(x)
print(y)


# Check tensor shape and type
x = torch.randn(4, 5)

print(x.shape)      # torch.Size([4, 5])
print(x.dtype)      # data type, e.g. torch.float32
print(x.device)     # cpu or cuda


# Basic math operations
a = torch.tensor([1, 2, 3])
b = torch.tensor([4, 5, 6])

print(a + b)   # tensor([5, 7, 9])
print(a - b)   # tensor([-3, -3, -3])
print(a * b)   # element-wise multiplication
print(a / b)   # element-wise division
print(a ** 2)  # square each element
```

```python
# Matrix multiplication
x = torch.randn(2, 3)
W = torch.randn(3, 4)

y = x @ W

print(y.shape)  # torch.Size([2, 4])


# Indexing and slicing
x = torch.tensor([
    [10, 20, 30],
    [40, 50, 60]
])

print(x[0])       # first row
print(x[1, 2])    # row 1, column 2
print(x[:, 0])    # first column
print(x[:, 1:])   # columns from index 1 to end


# Reshape tensors
x = torch.arange(12)

print(x)
# tensor([0, 1, 2, ..., 11])

y = x.reshape(3, 4)

print(y)
# shape: [3, 4]
```

```python
# Combine tensors
a = torch.randn(2, 3)
b = torch.randn(2, 3)

# concatenate along rows
c = torch.cat([a, b], dim=0)
print(c.shape)  # [4, 3]

# concatenate along columns
d = torch.cat([a, b], dim=1)
print(d.shape)  # [2, 6]


# Reduce tensors
x = torch.tensor([
    [1.0, 2.0, 3.0],
    [4.0, 5.0, 6.0]
])

print(x.sum())       # sum of all values
print(x.mean())      # average of all values
print(x.max())       # largest value

print(x.sum(dim=0))  # sum by column
print(x.sum(dim=1))  # sum by row
```

Data Structure for AI

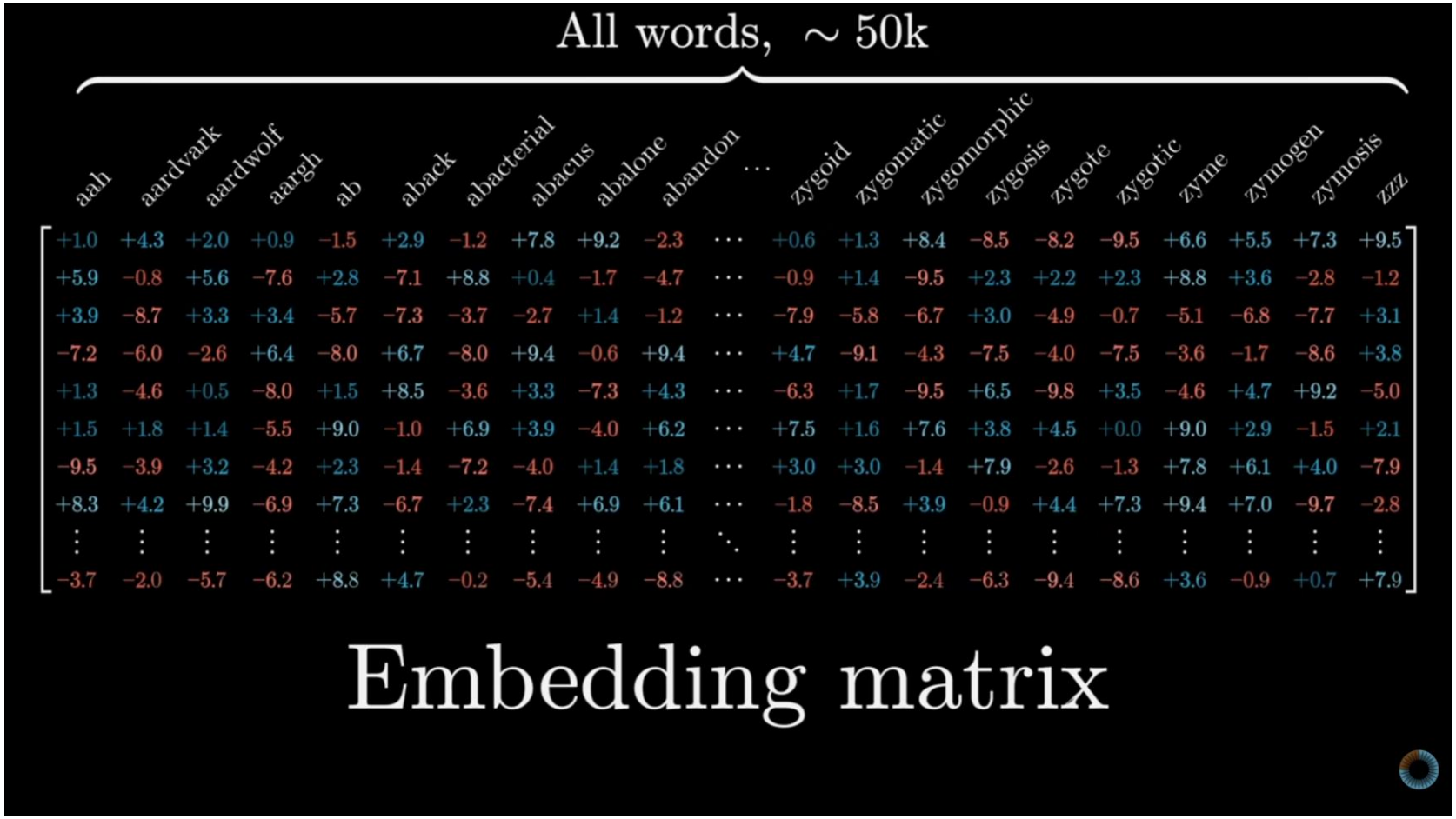
A linear case




```python
import torch

# Input: 4 samples, each sample has 3 features
x = torch.tensor([
    [1.0, 2.0, 3.0],
    [2.0, 1.0, 0.0],
    [0.0, 1.0, 2.0],
    [3.0, 2.0, 1.0]
])

# Weight matrix: map 3 input features to 2 output features
W = torch.tensor([
    [0.2, -0.5],
    [0.3,  0.8],
    [0.5, -0.1]
])

# Bias: one bias for each output feature
b = torch.tensor([0.1, 0.2])

# Linear operation
linear_output = x @ W + b

# ReLU activation
relu_output = torch.relu(linear_output)

print("Linear output:")
print(linear_output)

print("After ReLU:")
print(relu_output)
```

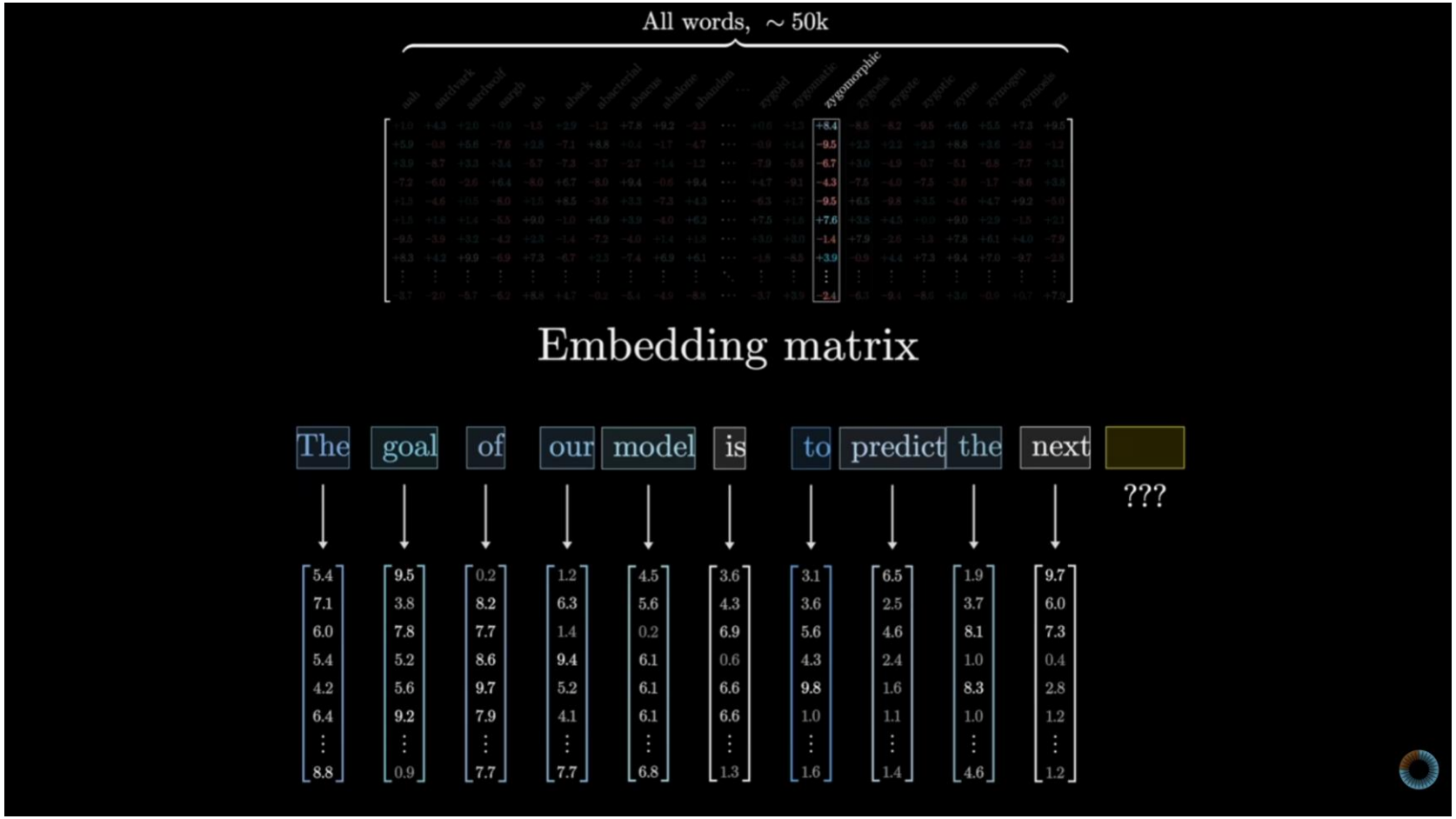
Data Structure for AI

Array & Tensor



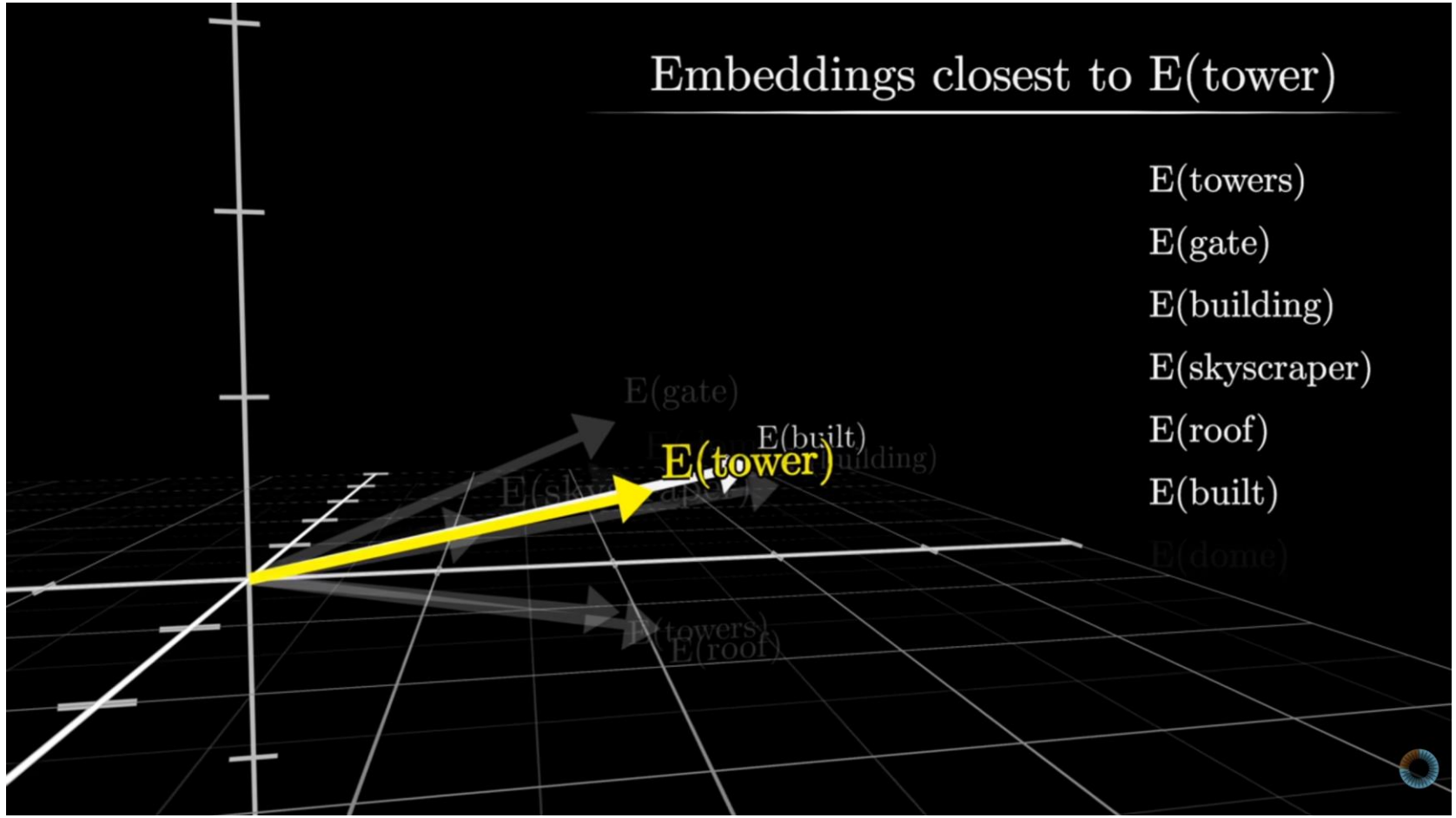
Data Structure for AI

Array & Tensor



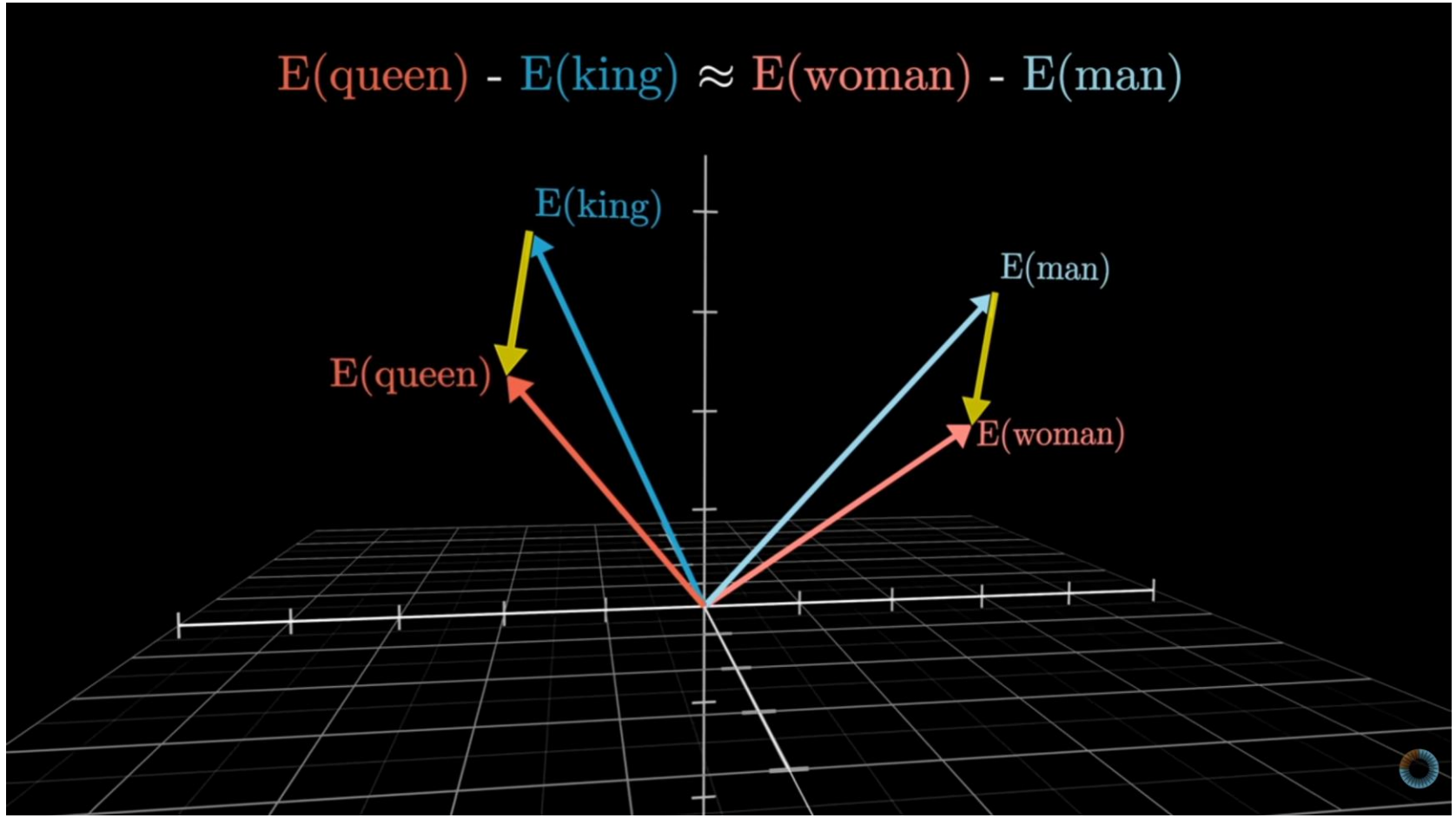
Data Structure for AI

Array & Tensor



Data Structure for AI

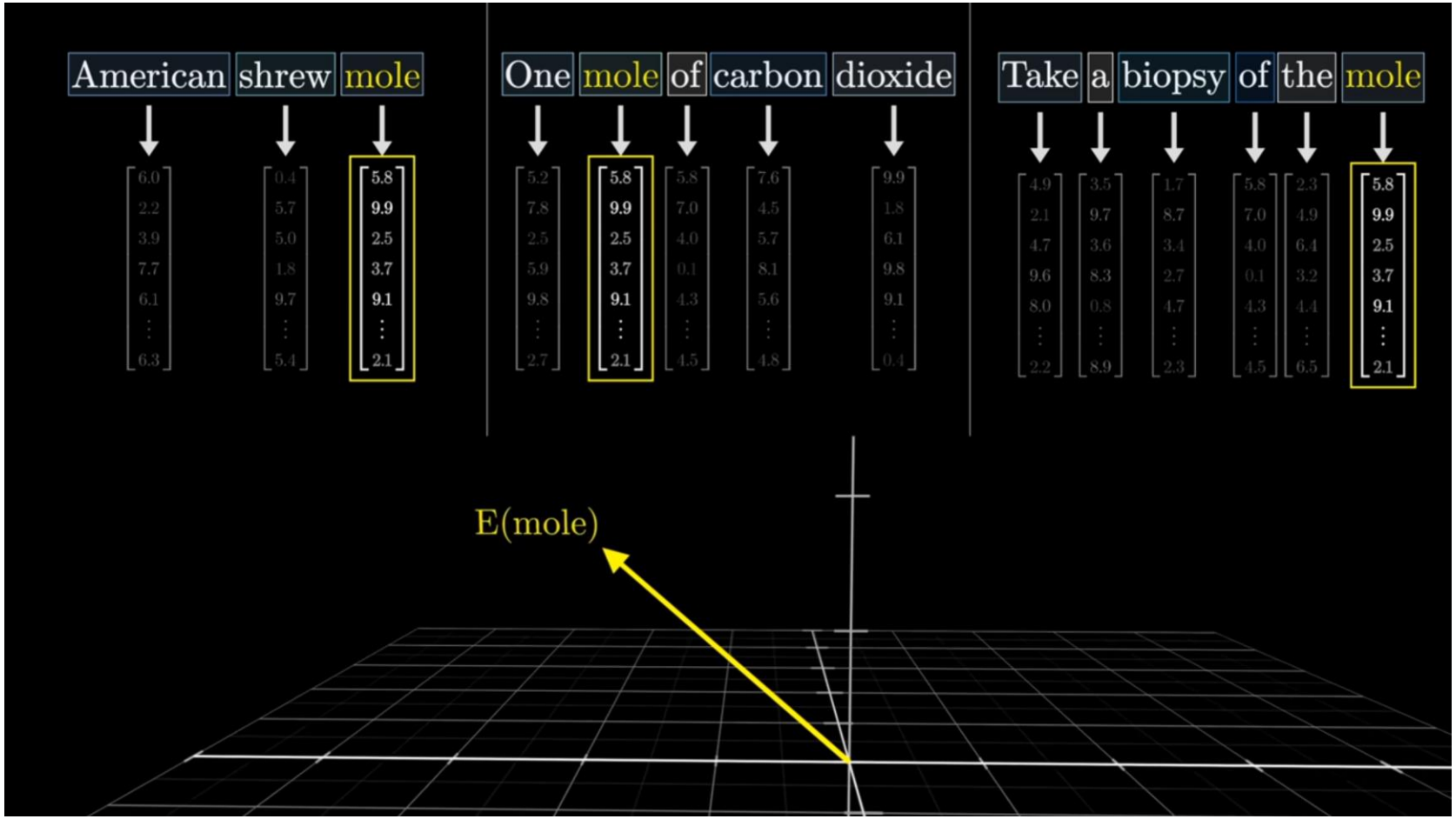
Array & Tensor



PURDUE UNIVERSITY

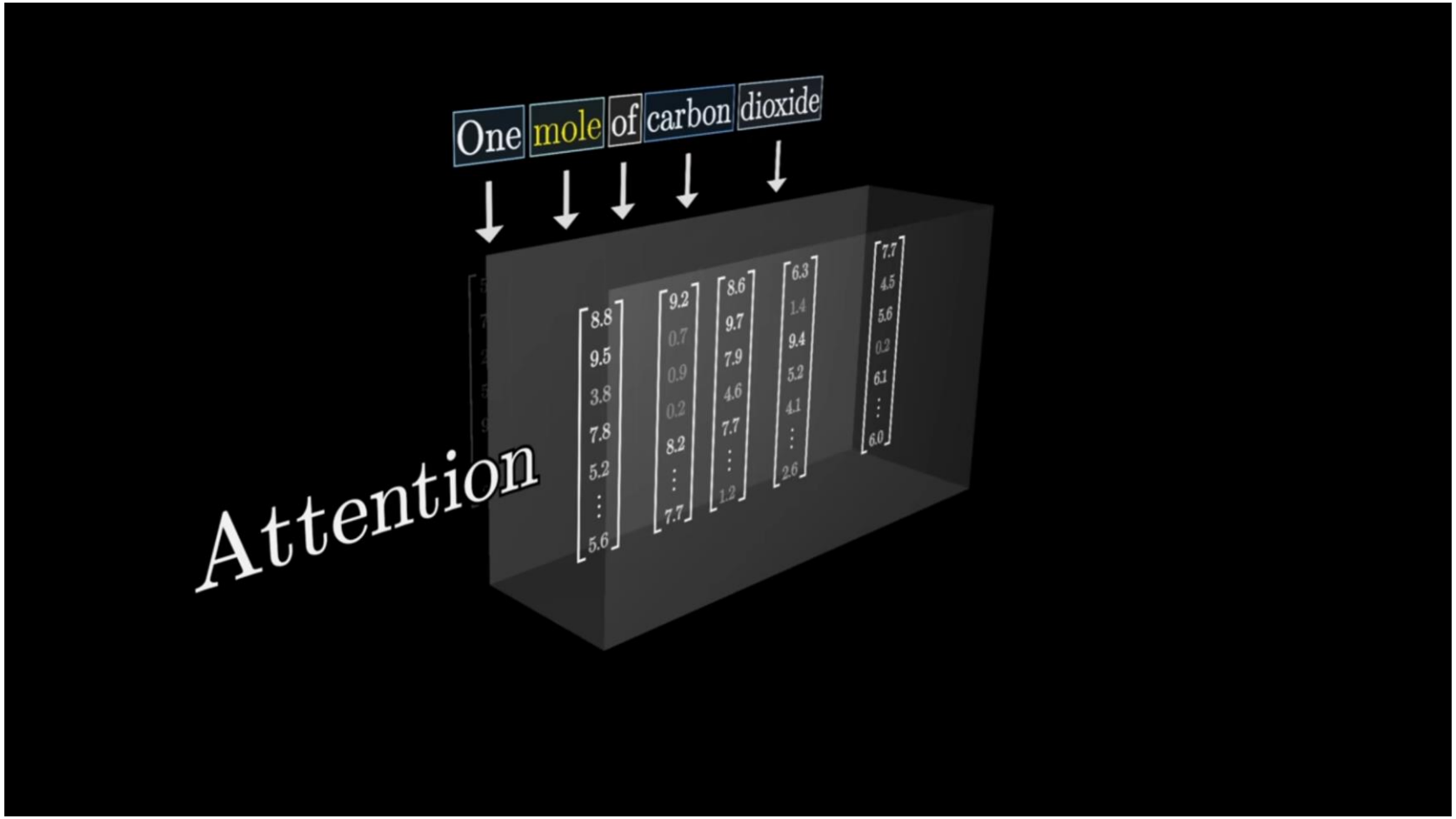
Data Structure for AI

Transformer



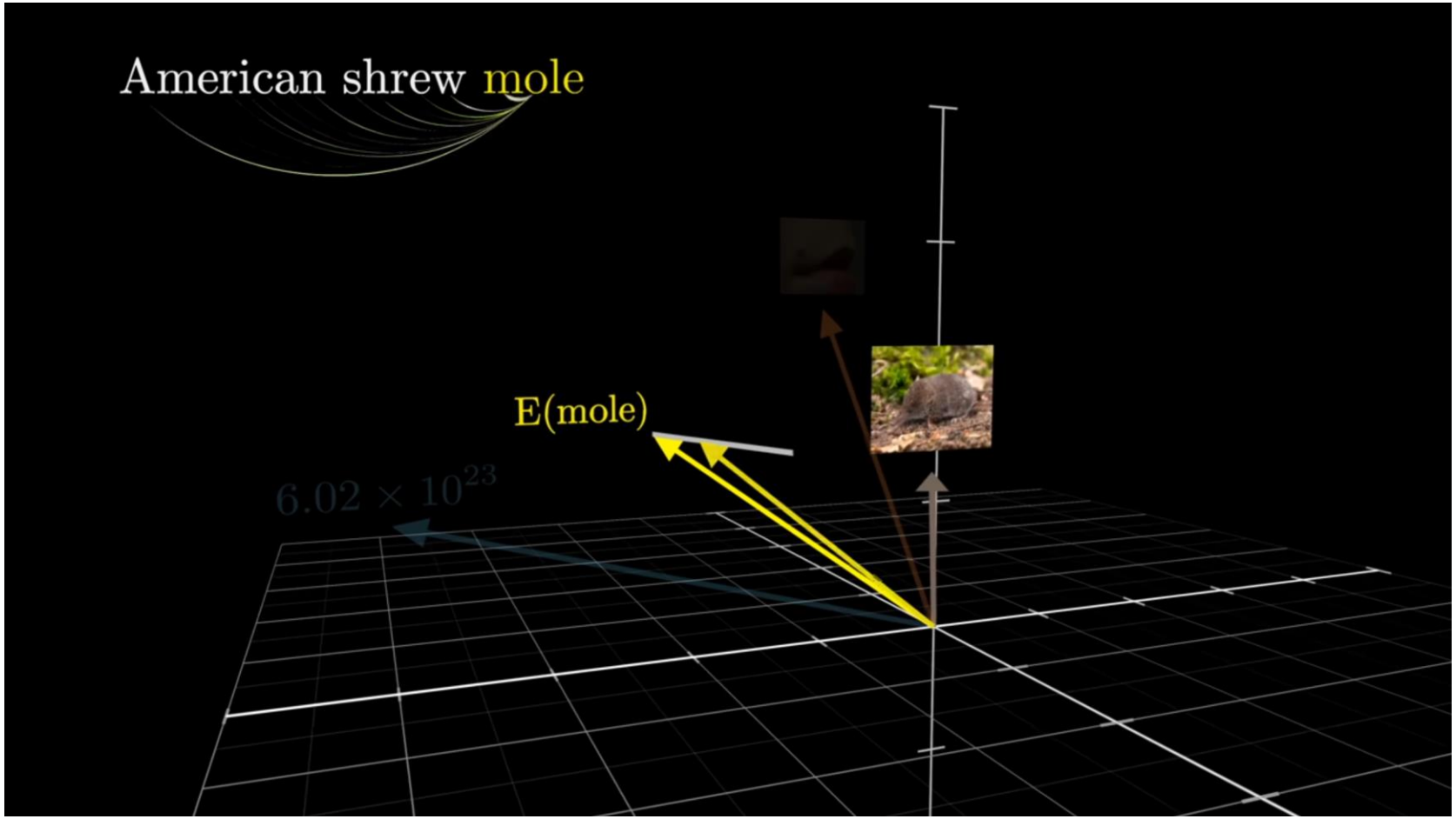
Data Structure for AI

Transformer



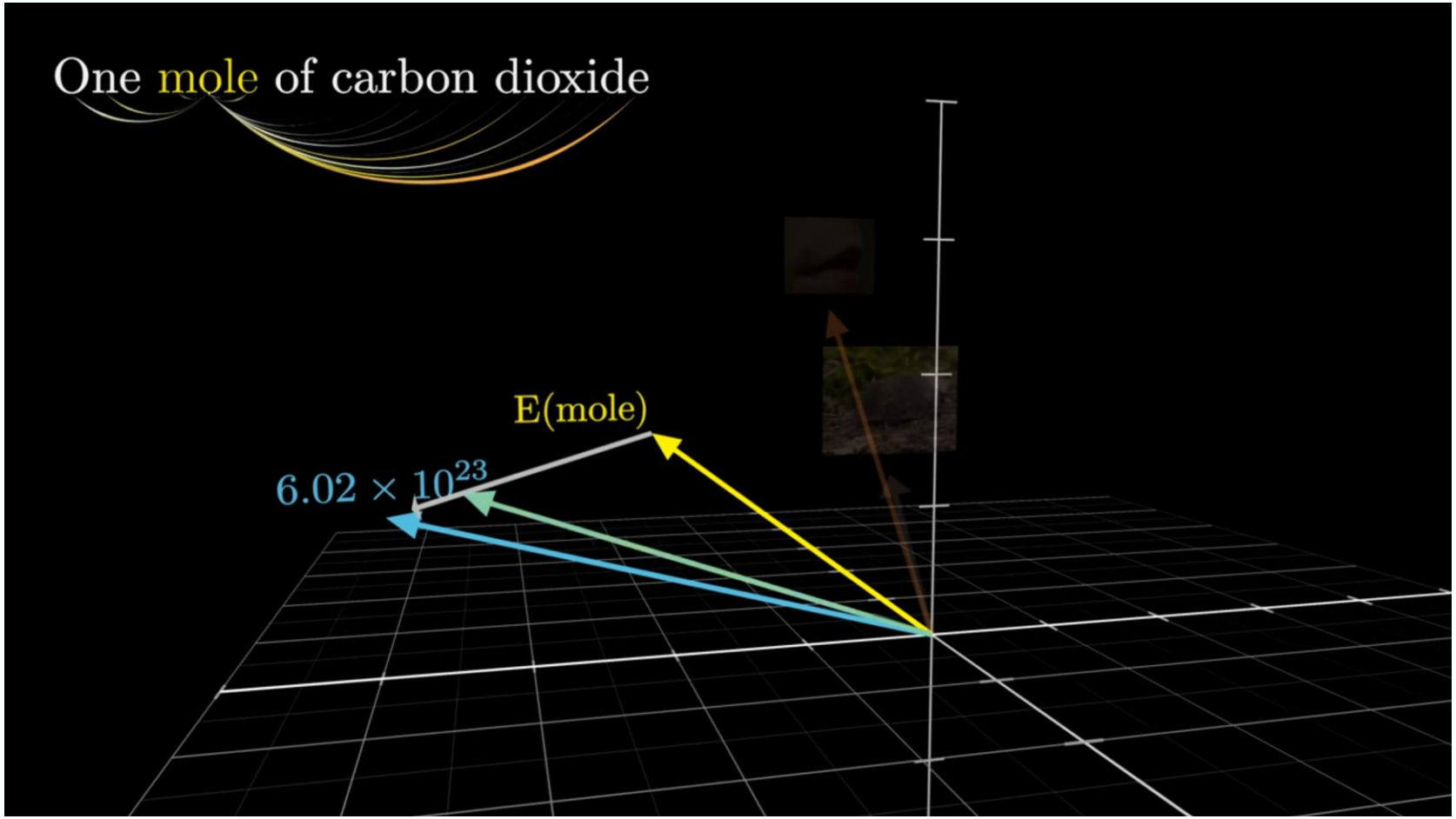
Data Structure for AI

Transformer



Data Structure for AI

Transformer



PURDUE UNIVERSITY

Job Interview in the AI-Era: Coding, Systems, Agents

Additional Materials

Wei Chen

05/18/2026

PURDUE UNIVERSITY®

Python basics

- Tuples: immutable lists
  - Can be used as dictionary keys
- Copy vs. reference
  - Deep copy: Nested structures -> need deep copy
- Functions
  - Lambda Functions
- Strings
- Memory & performance awareness

Problems: Arrays two pointers

- Best time to buy and sell stock
- Squares of a Sorted Array
- 3 sum
- Container with most water
- Longest Mountain in Array

Problems: Arrays sliding window

- Contains Duplicate II
- Minimum Absolute Difference
- Minimum Size Subarray Sum

Problems: Strings

- Valid palindrome
- Longest substring without repeating characters

Problems: Matrix & memory

- Set matrix zeroes

Problems: Backtracking

- Letter Case Permutation
- Subsets
- Combinations
- Permutations

Problems: Linked lists

- Middle of Linked List
- Linked List Cycle
- Reverse Linked List
- Remove Linked List Elements
- Reverse Linked List II
- Palindrome Linked List
- Merge Two Sorted Lists

PURDUE UNIVERSITY