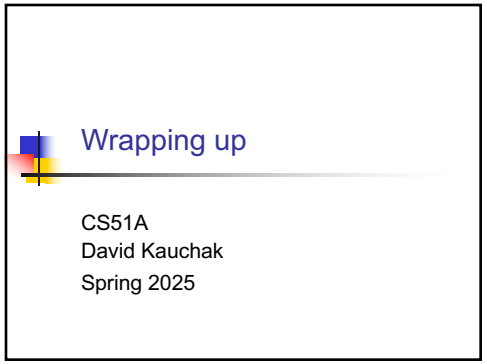
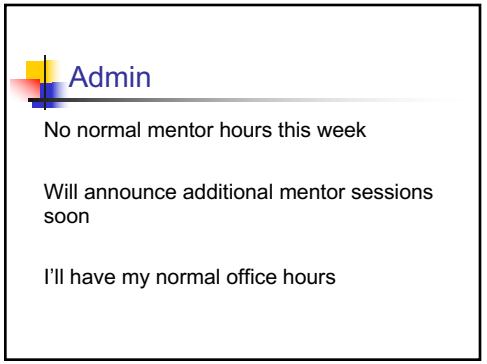
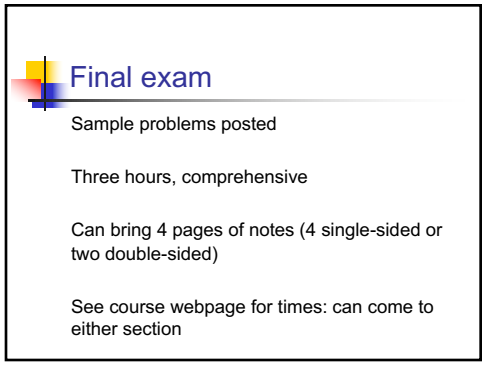

Wrapping up
CS51A
David Kauchak
Spring 2025

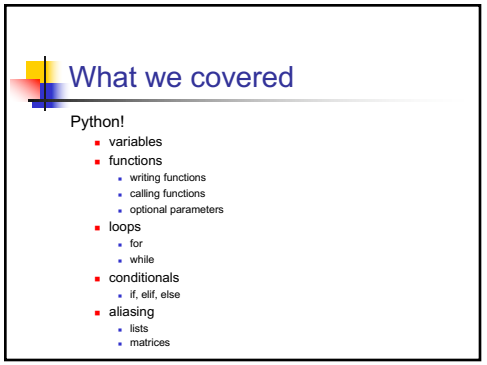
1


Admin
No normal mentor hours this week
Will announce additional mentor sessions soon
I'll have my normal office hours

2


Final exam
Sample problems posted
Three hours, comprehensive
Can bring 4 pages of notes (4 single-sided or two double-sided)
See course webpage for times: can come to either section

3


What we covered
Python!
variables
functions
writing functions
calling functions
optional parameters
loops
for
while
conditionals
if, elif, else
aliasing
lists
matrices

4

What we covered

Python!

- strings
  - string methods
- recursion
- classes
  - self
  - writing your own methods
  - local variables vs. instance variables
- higher order functions
  - passing functions as parameters
  - lambda expression
- file I/O
  - reading/writing from files

5

What we covered

Python!

- built-in modules
  - random
  - turtle
  - deepcopy
- exceptions
  - raising
  - handling (try/except)
- data structures
  - lists
  - tuples
  - dictionaries
  - sets

6

What we covered

Machine learning

Naïve Bayes model

Neural Networks

Search

- algorithms — BFS, DFS (recursive and using stack), best-first
- problem solving
- adversarial search and game playing

Webpages

- html basics
- Reading from webpages

AI ethics

7

Where we started

```
# This program figures out the number of hot dogs
# needed for a BBQ
teran = 1
jasmin = 2
chris = 2 * jasmin
brenda = chris - 1
grace = (brenda+1)//2 + 1 # add 1 to brenda using truncated division to round up

total_hotdogs = teran + jasmin + chris + brenda + grace
print(total_hotdogs)
```

8

Where we ended

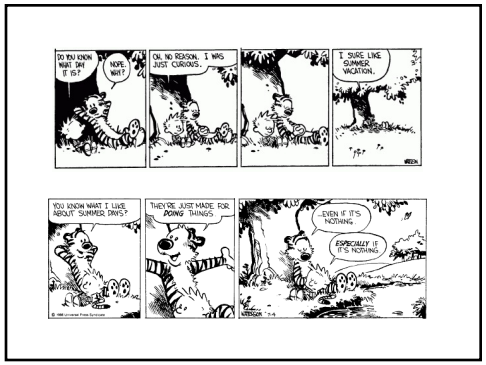
```
class NimGame:
    """ Class to keep track of a nim game. """

    def __init__(self, starting_piles):
        """
        Construct a new game with the list starting_piles
        as the piles
        """
        self.piles = starting_piles[:]

    def get_piles(self):
        """ Returns a copy of the current piles """
        # return a copy to avoid anybody messing with the internal
        # state of the game
        return self.piles[:]

    def make_move(self, pile_number, num_to_remove):
        """
        Move num_to_remove from pile_number.

        Return True if the move was valid, False otherwise.
        """
        if pile_number < 0 or pile_number >= len(self.piles):
            return False
        elif num_to_remove < 0 or num_to_remove > self.piles[pile_number]:
            return False
        else:
            self.piles[pile_number] -= num_to_remove
            return True

    def is_over(self):
        """ Is the game over? """
        return sum(self.piles) == 0

    def __str__(self):
        return str(self.piles)
```

9

Where we ended

```
def play_nim(player1, player2, game_state):
    # player1 will start
    player1_turn = True

    while not game_state.is_over():
        if player1_turn:
            print("Player 1's turn")
            print(game_state)
            (pile_number, num_to_remove) = player1(game_state.get_piles())
            print("Player 1:",end='')
        else:
            print("Player 2's turn")
            print(game_state)
            (pile_number, num_to_remove) = player2(game_state.get_piles())
            print("Player 2:",end='')

        print(str(num_to_remove) + " from pile " + str(pile_number))
        print()

        if not game_state.make_move(pile_number, num_to_remove):
            print("ILLEGAL MOVE!")

        player1_turn = not player1_turn

    # game's over, see who won
    if player1_turn:
        print("Player 2 won!")
    else:
        print("Player 1 won!")
```

10



11