Array Algorithms

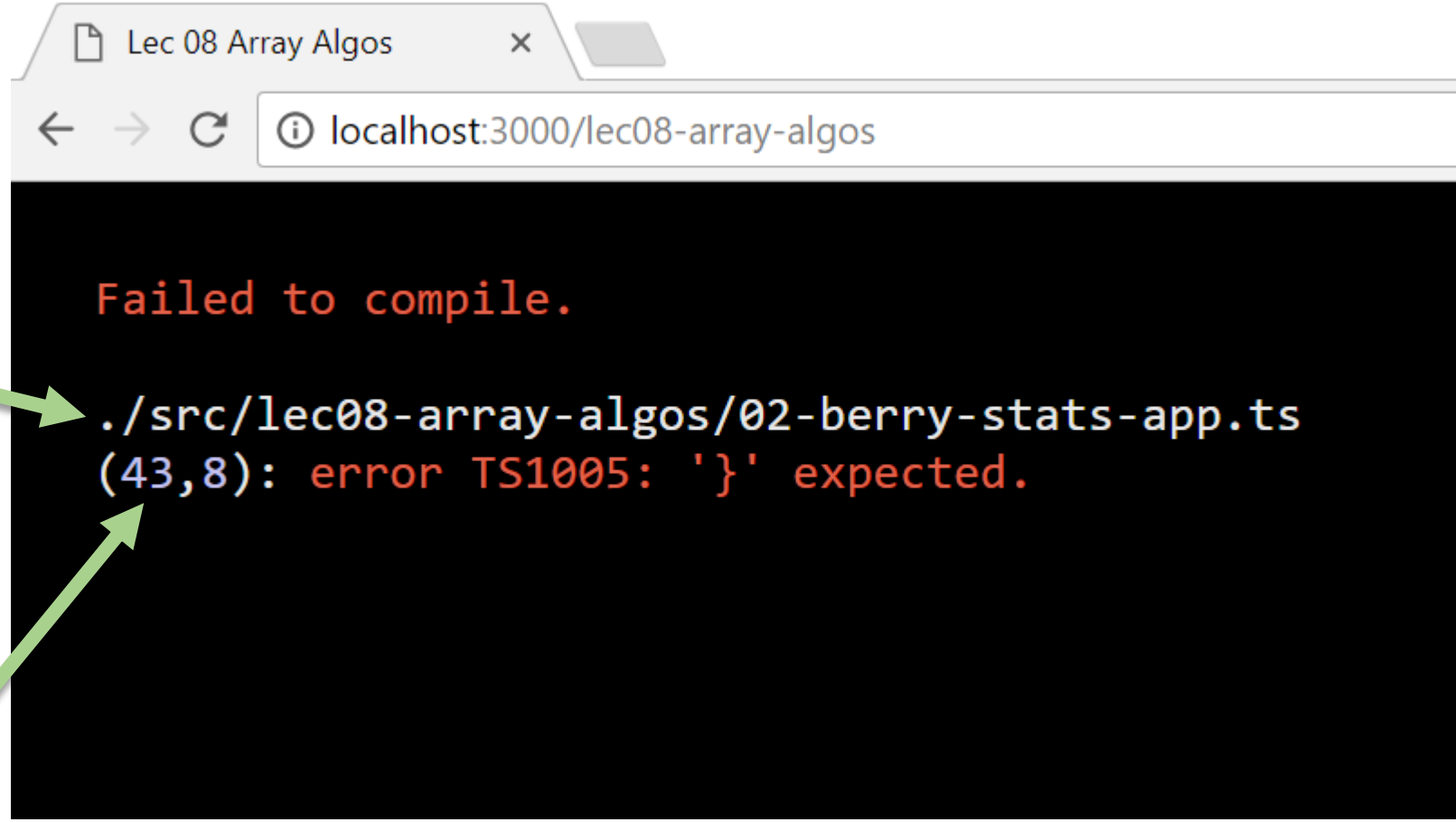
Lecture 08

npm run pull … npm run start

pollev.com/comp110

Fixing the "Black Screen of Death"

- When you see a screen that looks like the black screen to the right:
- The error may be in a file other than the one you are working on. The file with the error is this one.
- The simplest short-term fix in lecture is select all text in the file (Ctrl+A) and then comment it out (Ctrl+/) and save.
- The specific line # of the file that TypeScript *believes* the error is on is this number in parenthesis.



Use Chrome

- Chrome tends to handle accidental infinite loops better than other web browsers.

- When working on 110 assignments and lectures, we suggest you always use Chrome.

- If you do not want Chrome to be your **default** browser outside of 110, that's fine, just open up Chrome to http://localhost:3000 each time you start and close the browser that opens automatically for you.

Office Hours Tickets

- No matter how small the problem, or how empty office hours is, you must always submit the office hours help request form.

- Why? It ensures two things:

  - First, that the stats for how busy office hours are always accurate.
    It's *super frustrating* to show up believing there's no one being helped when actually there is no one free. Simple questions often take 10 minutes.

  - Second, it shows a track record on your profile of coming in for help and putting in an effort to use the resources you have available.

- TAs are instructed to only help those who have submitted tickets.

Email & Office Hours Best Practices

- **By and large you all have been incredibly awesome to work with in office hours and over e-mail this semester. Keep it up!**

- In a few isolated instances, though, I've read e-mails to TAs or heard reports of office hours interactions that are far beneath Carolina standards. The TAs and I have feelings, too.

- If you get a warning e-mail from me with suggestions on how to be a decent human being, that's the only warning I'll give. Continued issues will be escalated.

Midterm 0 – Next Thursday 9/28

- Special Review Session next Wednesday 9/27 in GSB100 from 5-7pm
- Additional practice problems will be released soon
- Spend quality time understanding everything on the next Problem Set and Worksheet. These are designed to help prepare you for the midterm!

PS2 - Weather Stats

- Analyze the last 30 years of weather data from RDU Airport.
- Autograding will open up by Saturday evening.
- Due Tuesday 9/26 at 11:59pm
  - Submit Sunday for full EC
  - Submit Monday for partial EC
- Pay close attention to what we do in class today. It will help you with the problem set.

WS2 and WS3

- Worksheet 2 due tomorrow by 11:59pm

- TAs will not directly answer questions of "is this right?"
  - Instead:
    - Try asking more pointed questions about topics your uncertain of
    - Try coming up with another example just like it

- Worksheet 3 will go out tonight and be due Monday.
  - Shorter turn around than normal to help be guided preparation for Midterm.

Warm-up #1: What is printed when this code completes?

```
let i: number = 10;
while (i > 0) {
   print(i);
   i = i - 4;
}
```

Answer: 10, 6, 2

```
let i: number = 10;
while (i > 0) {
   print(i);
   i = i - 4;
}
```

Warm-up: Array Question

- Write this down on pen/paper as you work through it!
- What are the elements of array a after the while loop completes?

```
let a: number[] = [];

let i: number = 0;
while (i < 4) {
   if (i <= 0) {
      a[i] = i;
   } else {
      a[i] = a[i - 1] + i;
   }
   i = i + 1;
}

print(a);
```

Warm-up: Array Question

- Answer:

Index	Element
0	0
1	1
2	3
3	6

```
let a: number[] = [];

let i: number = 0;
while (i < 4) {
   if (i <= 0) {
      a[i] = i;
   } else {
      a[i] = a[i - 1] + i;
   }
   i = i + 1;
}

print(a);
```

Hands-on #1: Write a **loop**

- Open **00-increment-operator-app.ts**
- Write a while loop that prints numbers 0 through 3
- Be very careful not to write an infinite loop!
- Done? Check-in on **PollEv.com/comp110**

```
import "introcs";

function main(): void {

    let i: number = 0;
    while (i < 3) {
        print(i);
        i++;
    }

}


main();
```

The Increment Operator: **++**

- Incrementing a variable's value by 1 is so common we have a special operator for doing so.

<variable>++ **ex: i++;**

- The ++ operator increments a variable's value by 1.
- Using it as a statement, (i.e. **i++;**) it is equivalent to **i = i + 1;**
- Read as "Increment <variable> by 1."

Working with Data

- <u>Today's Goal:</u>
  Use arrays and loops to analyze Joel Berry II's game data from last year.

Today's Data

- Data source: ESPN.com
- The Game Log table to the right was copied and pasted into Excel

Joel Berry II

#2 G | Junior | North Carolina Tar Heels

Hometown Apopka, FL
Height 6-0
Weight 195 lbs

2016-17 Season

PPG	RPG	APG
14.7	3.1	3.6

Go to
Joel Berry II

Player Profile | Game Log | Photos

Joel Berry II Game-by-Game Stats

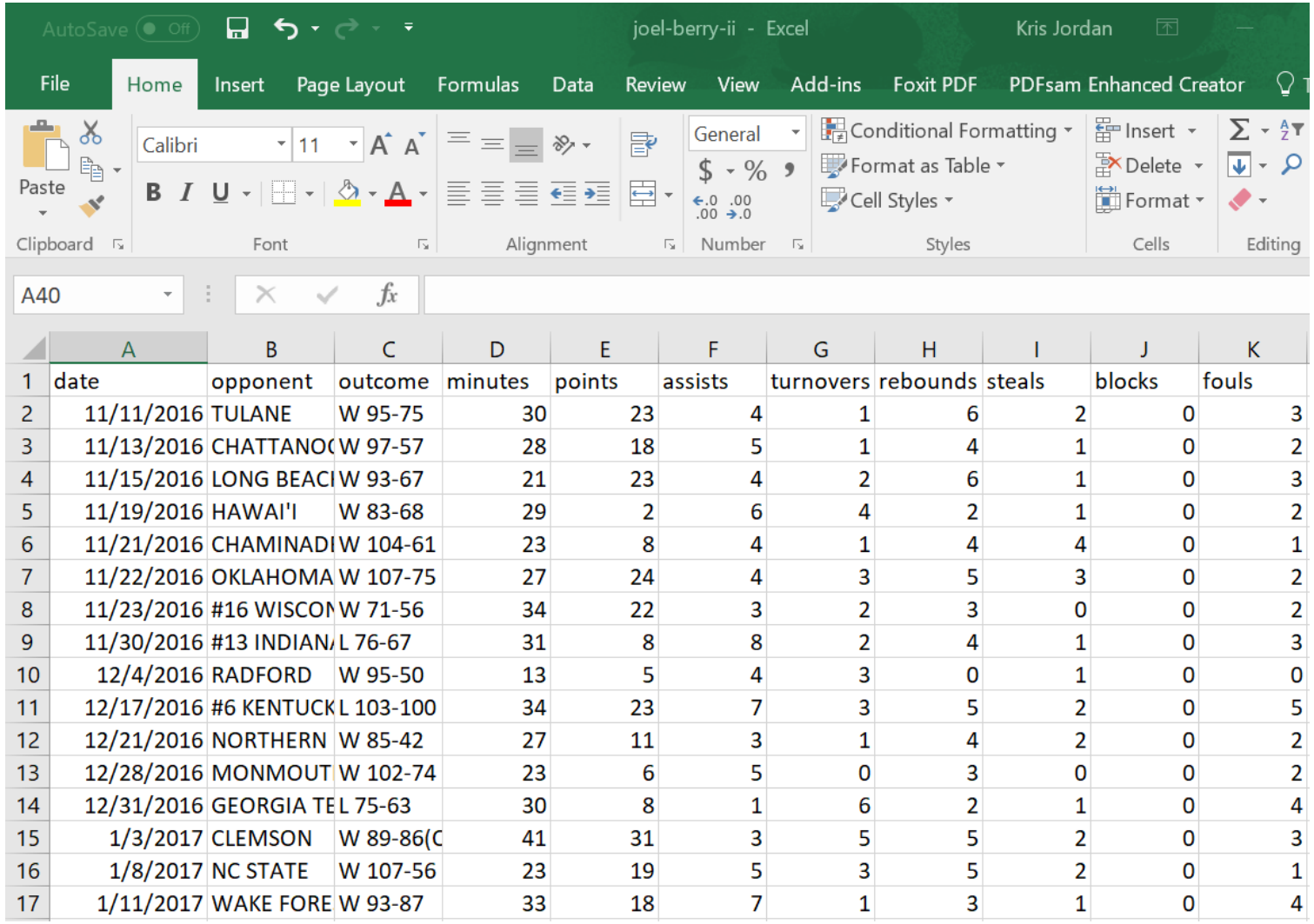
Season: 2016-17

2016-17 GAME LOG

DATE	OPP	RESULT	MIN	FGM-FGA	FG%	3PM-3PA	3P%	FTM-FTA	FT%	REB	AST	BLK	STL	PF	TO	PTS
Fri 11/11	@ TULANE	W 95-75	30	6-12	.500	4-7	.571	7-9	.778	6	4	0	2	3	1	23
Sun 11/13	vs CHATTANOOGA	W 97-57	28	5-8	.625	2-4	.500	6-6	1.000	4	5	0	1	2	1	18
Tue 11/15	vs LONG BEACH ST	W 93-67	21	7-10	.700	2-4	.500	7-7	1.000	6	4	0	1	3	2	23
Sat 11/19	@ HAWAI'I	W 83-68	29	1-9	.111	0-5	.000	0-0	.000	2	6	0	1	2	4	2
Mon 11/21	@ CHAMINADE	W 104-61	23	3-8	.375	2-5	.400	0-0	.000	4	4	0	4	1	1	8
Tue 11/22	vs OKLAHOMA STATE	W 107-75	27	7-10	.700	4-6	.667	6-6	1.000	5	4	0	3	2	3	24
Wed 11/23	vs #16 WISCONSIN	W 71-56	34	9-12	.750	2-3	.667	2-2	1.000	3	3	0	0	2	2	22
Wed 11/30	@ #13 INDIANA	L 76-67	31	3-13	.231	1-6	.167	1-2	.500	4	8	0	1	3	2	8
Sun 12/4	vs RADFORD	W 95-50	13	1-3	.333	1-3	.333	2-2	1.000	0	4	0	1	0	3	5

Today's Data

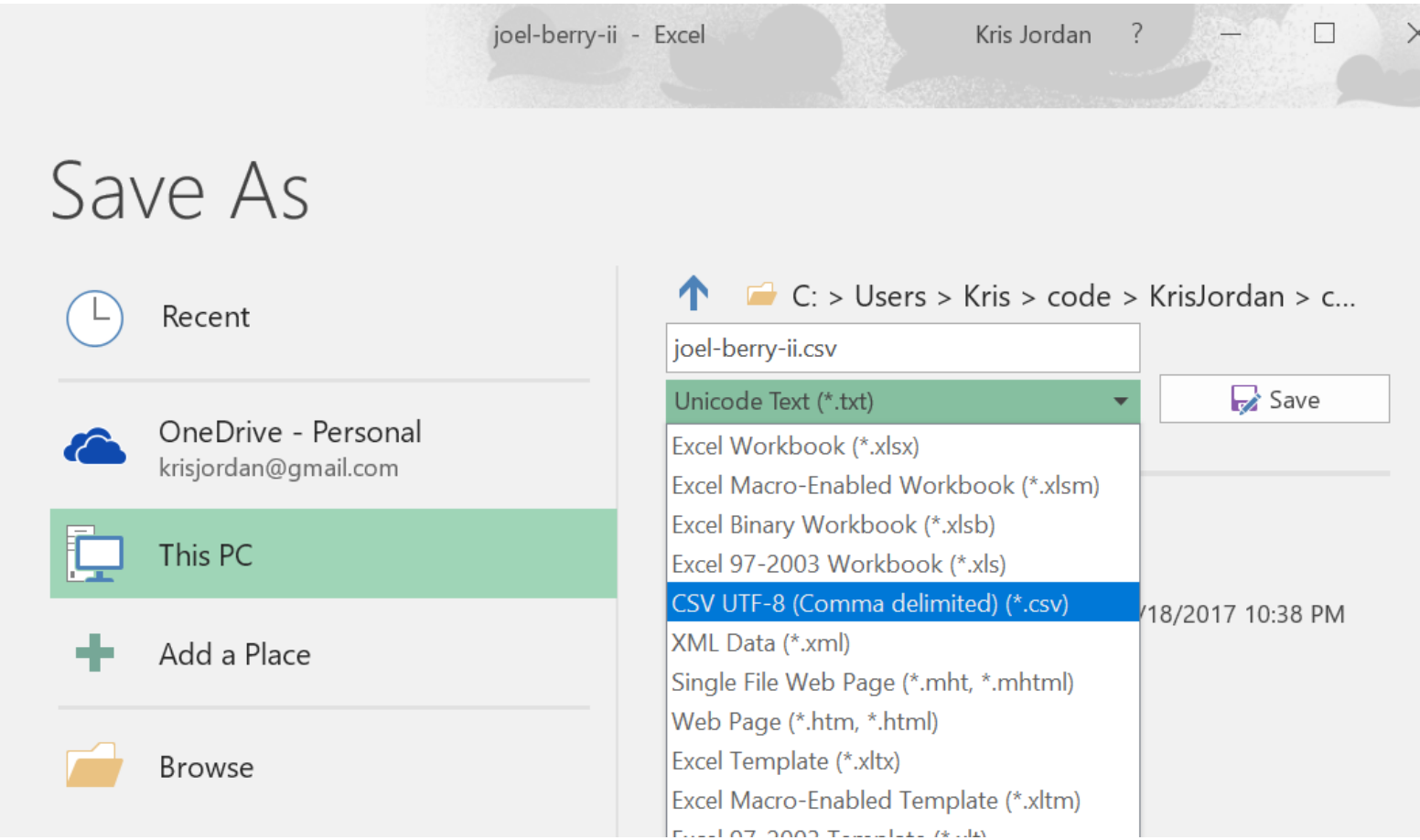
- The table was cleaned up a bit in Excel and formatting removed
- Column header names were changed to match properties we'll use in our code (we'll come back to this soon)



date	opponent	outcome	minutes	points	assists	turnovers	rebounds	steals	blocks	fouls
11/11/2016	TULANE	W 95-75	30	23	4	1	6	2	0	3
11/13/2016	CHATTANOC	W 97-57	28	18	5	1	4	1	0	2
11/15/2016	LONG BEACI	W 93-67	21	23	4	2	6	1	0	3
11/19/2016	HAWAI'I	W 83-68	29	2	6	4	2	1	0	2
11/21/2016	CHAMINADI	W 104-61	23	8	4	1	4	4	0	1
11/22/2016	OKLAHOMA	W 107-75	27	24	4	3	5	3	0	2
11/23/2016	#16 WISCON	W 71-56	34	22	3	2	3	0	0	2
11/30/2016	#13 INDIAN	L 76-67	31	8	8	2	4	1	0	3
12/4/2016	RADFORD	W 95-50	13	5	4	3	0	1	0	0
12/17/2016	#6 KENTUCK	L 103-100	34	23	7	3	5	2	0	5
12/21/2016	NORTHERN	W 85-42	27	11	3	1	4	2	0	2
12/28/2016	MONMOUT	W 102-74	23	6	5	0	3	0	0	2
12/31/2016	GEORGIA TE	L 75-63	30	8	1	6	2	1	0	4
1/3/2017	CLEMSON	W 89-86(C	41	31	3	5	5	2	0	3
1/8/2017	NC STATE	W 107-56	23	19	5	3	5	2	0	1
1/11/2017	WAKE FORE	W 93-87	33	18	7	1	3	1	0	4

Today's Data

- Finally it was saved as a special type of file:
- **CSV UTF-8 (Comma delimited) (*.csv)**
- This is a common data table format that is easy to work with in code.



Today's Data

- Here's what the contents of the CSV file look like.

- It is stored in:

data/joel-berry-ii.csv

- Notice it's just plain text!

- Each row gets a line, each column is separated by a comma, hence "Comma Separated Values (CSV)" file.

joel-berry-ii.csv

```
date,opponent,outcome,minutes,points,assists,turnovers,rebounds,steals,blocks,fouls
2016-11-11,TULANE,W 95-75,30,23,4,1,6,2,0,3
2016-11-13,CHATTANOOGA,W 97-57,28,18,5,1,4,1,0,2
2016-11-15,LONG BEACH ST,W 93-67,21,23,4,2,6,1,0,3
2016-11-19,HAWAI'I,W 83-68,29,2,6,4,2,1,0,2
2016-11-21,CHAMINADE,W 104-61,23,8,4,1,4,4,0,1
2016-11-22,OKLAHOMA STATE,W 107-75,27,24,4,3,5,3,0,2
2016-11-23,#16 WISCONSIN,W 71-56,34,22,3,2,3,0,0,2
2016-11-30,#13 INDIANA,L 76-67,31,8,8,2,4,1,0,3
2016-12-04,RADFORD,W 95-50,13,5,4,3,0,1,0,0
2016-12-17,#6 KENTUCKY,L 103-100,34,23,7,3,5,2,0,5
2016-12-21,NORTHERN IOWA,W 85-42,27,11,3,1,4,2,0,2
2016-12-28,MONMOUTH,W 102-74,23,6,5,0,3,0,0,2
2016-12-31,GEORGIA TECH,L 75-63,30,8,1,6,2,1,0,4
2017-01-03,CLEMSON,W 89-86(OT),41,31,3,5,5,2,0,3
2017-01-08,NC STATE,W 107-56,23,19,5,3,5,2,0,1
2017-01-11,WAKE FOREST,W 93-87,33,18,7,1,3,1,0,4
2017-01-14,#9 FLORIDA STATE,W 96-83,35,26,1,2,2,2,0,3
2017-01-16,SYRACUSE,W 85-68,31,10,1,2,1,1,0,2
2017-01-21,BOSTON COLLEGE,W 90-82,35,9,0,2,0,1,1,1
2017-01-26,VIRGINIA TECH,W 91-72,30,15,4,0,3,1,1,2
2017-01-28,MIAMI,L 77-62,30,2,4,2,1,1,0,3
```

Modelling a "Game" with a class

- Each Game has properties associated with it:
  - date
  - opponent
  - points
  - and more...
- These are column names in our data table
- In our program, we'll declare a class to model a single Game's stats with properties for each column in the table we care about.
  - Note: we do not need to use every column but the names of properties much match the column headers in the CSV file.

```
class Game {

    date: string = "";
    opponent: string =
    "";
    points: number = 0;
    fouls: number = 0;

}
```

Reading a CSV into an Array of **`Game`** Objects

- Last week we saw how to process an "array of numbers", i.e. **`number[]`**
- We want to work with our data table as an "array of Games", i.e. **`Game[]`**
- Each row in the data table will have a Game object associated with it. Each column in the data table is a property of the Game object.

index	date	opponent	points	fouls
0	11/11/2016	TULANE	23	3
1	11/13/2016	CHATTANOOGA	18	2
2	11/15/2016	LONG BEACH ST	23	3
3	11/19/2016	HAWAI'I	2	2
4	11/21/2016	CHAMINADE	8	1

`games[2]`

`games[4].points`

How do we prompt the user for a CSV file?

- There's a function in the **introcs** library for that!

- Documentation:

```
promptCSV(prompt:string, cname:Class, callback:Function): void
```

- Parameters:

1. **`prompt`** - a string value presented to the user as instructions
2. **`cname`** - the name of the class (i.e. **`Game`**) each row of the CSV corresponds to
3. **`callback`** - the name of the function that will called once the user selects a CSV. The function must declare a single parameter of type **`cname[]`** (i.e. **`Game[]`**).

Follow-along: Calling **promptCSV**

- Open **01-csv-app.ts**

- In the **main** function, call:

```
promptCSV("Select player data CSV", Game, process);
```

- In the **process** function, print:

```
print(games[29].opponent);
print("Points: " + games[29].points);
```

Hands-on #2

- In the process function:

1. Declare a counting variable named **i**
2. Write a **while** loop that repeats while **i** is less than **games.length**
3. Inside the while loop's repeat block:
    1. **Call the `statLine` function** with **games[i]** and print the returned string
    2. **Increment i by 1**

4. Test and confirm that stat lines of all games are printing out.
5. Check-in on PollEv.com/comp110

```
function process(games: Game[]): void {
    print("Processing CSV...");
    let i: number = 0;
    while (i < games.length) {
        print(statLine(games[i]));
        i++;
    }
}
```

Pseudo-algo: counting # of games fouled out

- When a college basketball player accumulates 5 fouls, (s)he fouls out of the game. How can we count the # of games Joel fouled out?

1. Set **count** to 0.
2. Loop element-by-element through an array.
3. If the element's # of fouls is equal to 5, then *increment count*.
4. Reached the end of the array? The count variable now contains the # of games the player fouled out.

<u>Follow-Along:</u> Let's implement a function to calculate the number of games fouled out.

- Open **02-berry-stats-app.ts**

- Notice our **main** function is setup to prompt for a CSV and call the process function already. We also have two functions defined whose implementations are incorrect: gamesFouledOut and totalPoints.

- First, we need to *call* the gamesFouledOut function from the *process* function and print the number it returns.

```
print("Games fouled out: " + gamesFouledOut(games));
```

- Then, we need to implement the algorithm from the previous slide in code inside of the gamesFouledOut function.

Counting the # of games Joel fouled out…

1. Set **count** to 0.
2. Loop element-by-element through an array.
3. If the element's # of fouls is equal to 5, then *increment count*.
4. Reached the end of the array? The count variable now contains the # of games the player fouled out.

```
function gamesFouledOut(games: Game[]): number {
   let count: number = 0;
   let i: number = 0;
   while (i < games.length) {
      if (games[i].fouls === 5) {
         count++;
      }
      i++;
   }
   return count;
}
```

Hands-on #3: Summing Points

- Together: Let's call the **totalPoints** function from process:
  - `totalPoints(games)`

- Your goal: Correctly implement the **`totalPoints`** function with a sum algorithm. It should return the sum of points across all of Joel Berry's games last season.

- Hint #1: You'll need a variable to store the sum as you work element-by-element through the games array.

- Hint #2: You wrote a sum algorithm in lec07 / 03-sum-array-app.ts – how can you modify it to make use of the games array instead?

- **Done? Check-in on PollEv.com/comp110 once you've got sum working.**

- Done early? Try writing an averagePoints function.

```
function totalPoints(games: Game[]): number {
   let sum: number = 0;
   let i: number = 0;
   while (i < games.length) {
      sum = sum + games[i].points;
      i++;
   }
   return sum;
}
```

Problem Set 2: Weather Stats

- The work you do in PS2 will be similar to what we did in class today!
- There are some algorithms you will need to spend some time thinking through in order to find solutions to. ***This will take time!***
- Suggestion: lay out some cards on a table face down and get out a sheet of paper. Try writing down each variable you need and how you would update the variables as you move through your array of cards element-by-element.
- Before you write code, try writing down steps in English.