Class 1: Static Checking

6.102 — Software Construction
Spring 2024

6.102 does not have lectures

6.102 does not have ~~lectures~~


Class time will be spent doing **active learning**


Laptops are required for 6.102

power

- there are not many power outlets: charge your laptop before class
- if someone needs power, remember: sharing is caring
- cords must be on the floor, not a tripping hazard

wireless network

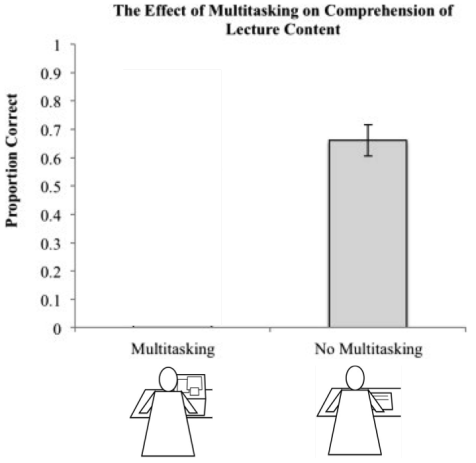
- please turn off Wifi on all your other devices (except your laptop)

Laptops are evil

Sana, Weston, Cepeda. “Laptop multitasking hinders classroom learning for both users and nearby peers.” *Computers & Education*, March 2013.

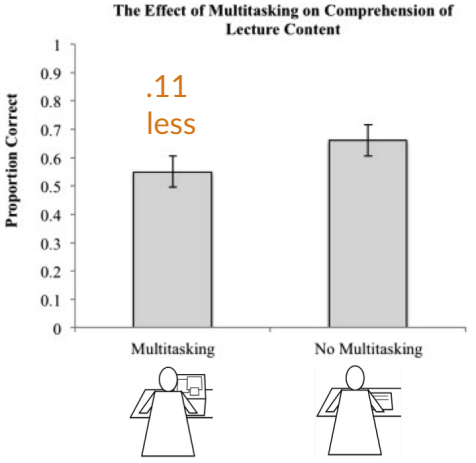
Laptops are evil

Sana, Weston, Cepeda. “Laptop multitasking hinders classroom learning for both users and nearby peers.” *Computers & Education*, March 2013.



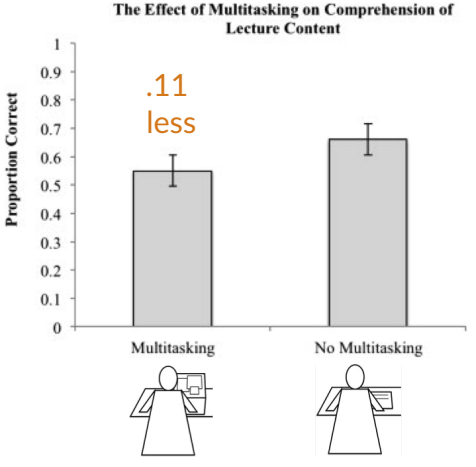
Laptops are evil

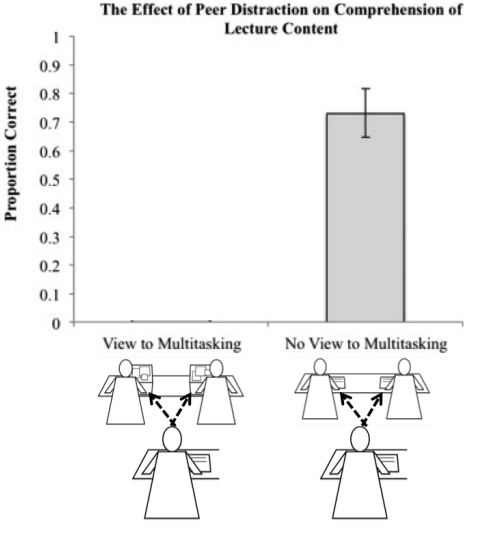
Sana, Weston, Cepeda. “Laptop multitasking hinders classroom learning for both users and nearby peers.” *Computers & Education*, March 2013.



Laptops are evil

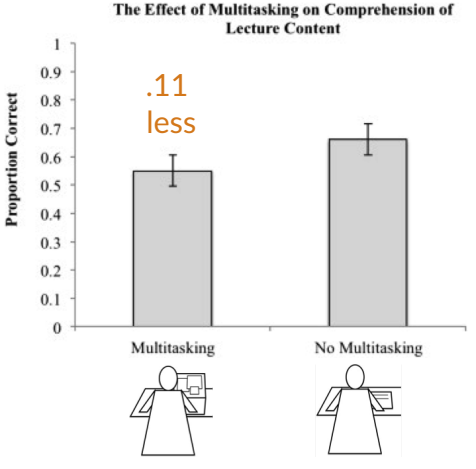
Sana, Weston, Cepeda. "Laptop multitasking hinders classroom learning for both users and nearby peers." *Computers & Education*, March 2013.

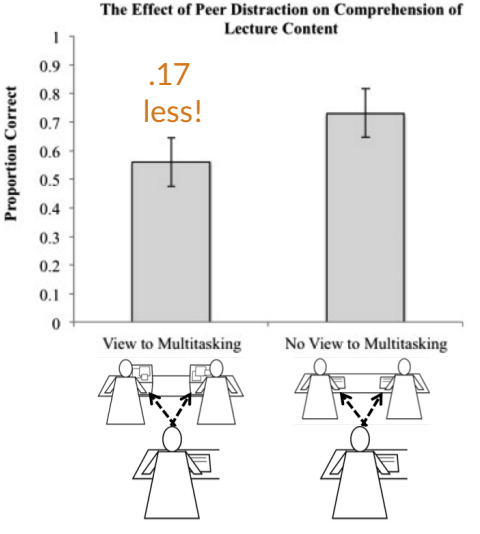




Laptops are evil

Sana, Weston, Cepeda. "Laptop multitasking hinders classroom learning for both users and nearby peers." *Computers & Education*, March 2013.





Laptops are evil

Expect to ~~close~~ lower your laptop frequently

Please practice now

What is 6.102 about?

What is 6.102 about?

Nanoquiz

Every class will start with a 3-minute “are you prepared?” quiz

- This quiz is just for you and your own brain
  - closed-book, closed-notes
  - nothing else on your screen:
    - no 6.102 website or readings
    - no VS Code, no TypeScript compiler, no programming tools
    - no web search, no discussion with other people
- ~~Close~~ Lower your laptop screen when you’re done

 yellkey.com/**impact**

```
data = [ 2, 4, 6 ]
total = 0
average = 0
n = 0
for value in data:
    n += 1
    total += value
    average = total / n
    print(average)
```

What does this code print?

A. `1 1 1`
B. `1 2 3`
C. `2 3 4`
D. `2 4 6`
E. error before the program starts
F. error while the program is running

```
data = [ "2", "4", "6" ]
total = 0
average = 0
n = 0
for value in data:
    n += 1
    total += value
    average = total / n
    print(average)
```

What does this code print?

A. `0 0 0`
B. `2 3 4`
C. `"2" "3" "4"`
D. `2 12 82`
E. error before the program starts
F. error while the program is running

Side-by-side programming: cumulative averages

1. Pair up with someone to work with

Side-by-side programming: cumulative averages

1. Pair up with someone to work with
2. Start collaborating on the code

yellkey.com/**positive**

   - if you did the Getting Started, you already have `ic01-static-checking`, so you don't need to run `git clone` or `npm install` again

Side-by-side programming: cumulative averages

1. Pair up with someone to work with
2. Start collaborating on the code

yellkey.com/**positive**

   - if you did the Getting Started, you already have `ic01-static-checking`, so you don't need to run `git clone` or `npm install` again

3. Open the Python file `cumulative-avg.py` and make the code into a **function**

Side-by-side programming: cumulative averages

1. Pair up with someone to work with
2. Start collaborating on the code

yellkey.com/**positive**

   - if you did the Getting Started, you already have `ic01-static-checking`, so you don't need to run `git clone` or `npm install` again

3. Open the Python file `cumulative-avg.py` and make the code into a **function**
4. Make sure it's a **useful** function

Side-by-side programming: cumulative averages

1. Pair up with someone to work with
2. Start collaborating on the code

yellkey.com/**positive**

   - if you did the Getting Started, you already have `ic01-static-checking`, so you don't need to run `git clone` or `npm install` again

3. Open the Python file `cumulative-avg.py` and make the code into a **function**
4. Make sure it's a **useful** function

Handing in work with Constellation vs. Git

- In class: if you see your partner's typing, then the staff has it too ✓
- Problem sets: you need to `git push` and check Didit

What would be a good way to support cumulative averages with a **decay factor** `gamma`?

$$\texttt{average}_i = (\texttt{data}_i + \gamma \cdot \texttt{data}_{i-1} + \gamma^2 \cdot \texttt{data}_{i-2} + \ldots + \gamma^i \cdot \texttt{data}_0) / i$$

A. Declare `gamma` as a global variable:

```
gamma = 1.0
def cumulative_average(list_of_numbers):
    ...
```

B. Add `gamma` as a parameter:

```
def cumulative_average(list_of_numbers, gamma):
    ...
```

C. Define a new function:

```
def decaying_cumulative_average(list_of_numbers, gamma):
    ...
```

The real question: what can go wrong with each of these choices?

Today

- Python ⇒ TypeScript
- Static checking
- The big three goals of 6.102

Your TODOs:

- Signup form
  - if you already filled it out, don't do it again
  - otherwise, **fill it out right now**, or you will not have access to the course

- Course website at `mit.edu/6.102`
- Readings 1 & 2 due tomorrow 10pm
- PS0 is already out
  - part I: Git and TypeScript exercises
  - part II: TypeScript coding
  - PS0 alpha submission due Mon 10 pm