

CSE 331: Algorithms & Complexity “A Proof”

Prof. Charlie Anne Carlson (She/Her)

Lecture 1

Monday August 27th, 2025



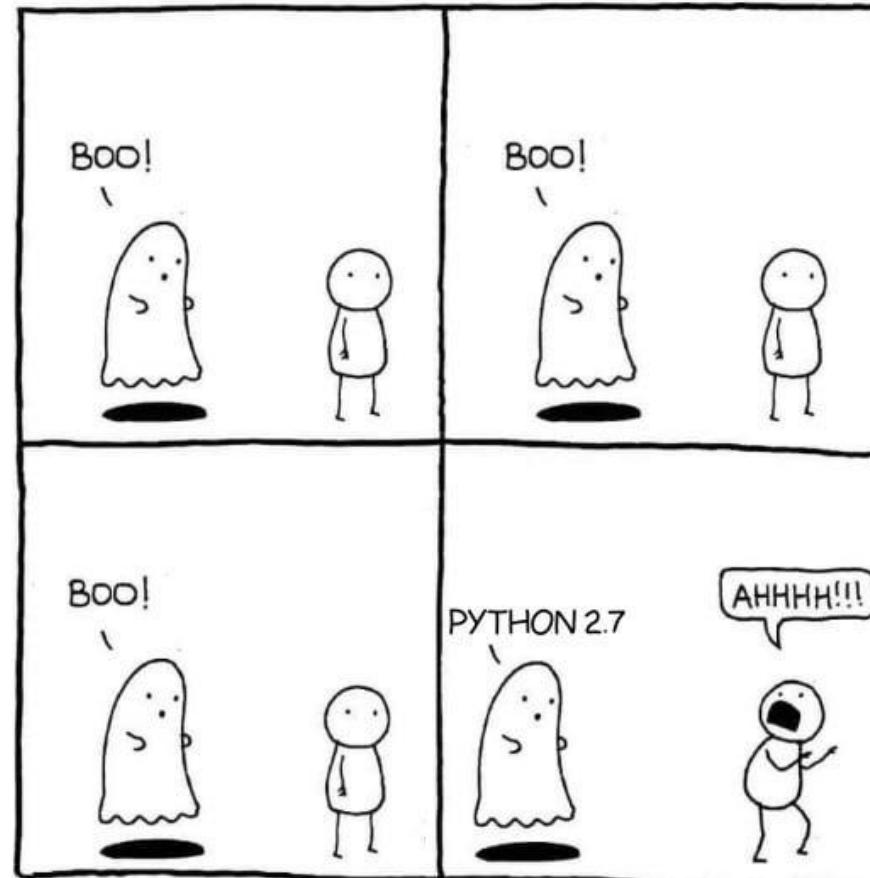
University at Buffalo®

Schedule

1. Course Updates
2. Advice
3. Halting Problem



Course Updates: Syllabus Quiz Fixed!



MrSavagePotato. “Python 2.7.” Reddit, July 25, 2018.

https://www.reddit.com/r/ProgrammerHumor/comments/91vtas/python_27/

Course Updates: Piazza Polls

<p>■ Instr Input on Office Hours</p> <p>Hello All, We would appreciate your input before we decide on the final times for our office hours this semester.</p>	8/25/25	
<p>■ Instr How to do Proofs (and a ...</p> <p>Hello everyone! My name is Tien, and I'm your 331 TA for this Fall 2025 semester. I want to give a heads-up to eve</p>	8/25/25	
<p>■ Instr Course Website & Start ...</p> <p>Hello All, Welcome to CSE 331! I'm looking forward to having you all in my class. I'm sorry to start the semester</p>	8/25/25	
<p>■ Private Search for Teammates!</p>	8/21/25	

[Close the Poll](#) [Download Poll Stats](#) [Clone the Poll](#) [Hide Poll Results](#)

Input on Office Hours

Updated 2 days ago by Charlie Anne Carlson

Hello All,

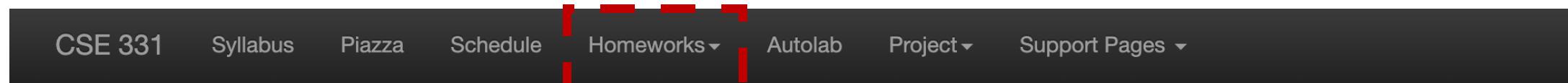
We would appreciate your input before we decide on the final times for our office hours this semester. Please answer the poll with what times you are most likely to attend office hours if they are offered. Please respond before Thursday, August 28th, 2025!

Thank you,

Charlie Anne (and CSE Staff)

Monday 9:00 AM to 10:00 AM

Course Updates: HW0 is out



Homework 0

Due by **11:30pm, Tuesday, September 2, 2025**.

Make sure you follow all the [homework policies](#).

All submissions should be done via [Autolab](#)

The [support page for matrix vector multiplication](#) could be very useful for this homework.

Submitting HW 0 is optional. However, we do encourage you to submit to get familiar with [Autolab](#) and to get some feedback.

Advice: Support Materials

The screenshot shows a website header with a dark grey background and white text. The header includes links for CSE 331, Syllabus, Piazza, Schedule, Homeworks (with a dropdown arrow), Autolab, Project (with a dropdown arrow), and Support Pages (with a dropdown arrow). A red dashed box highlights the 'Support Pages' dropdown menu, which is currently open and displays five options: Background Material, Common Mistakes, Algos via Examples, CSE 331 Care Package, and Support Page Home. The main content area below the header features a large title 'CSE 331 Syllabus' in a bold, dark font, followed by the subtitle 'Algorithms and Complexity' in a smaller, italicized font, and the text 'Fall 2025'. At the bottom, a note specifies the 'Time and location: Mondays, Wednesdays and Fridays, 5:00-5:50 PM, KNOX 109.'

CSE 331 Syllabus

Algorithms and Complexity

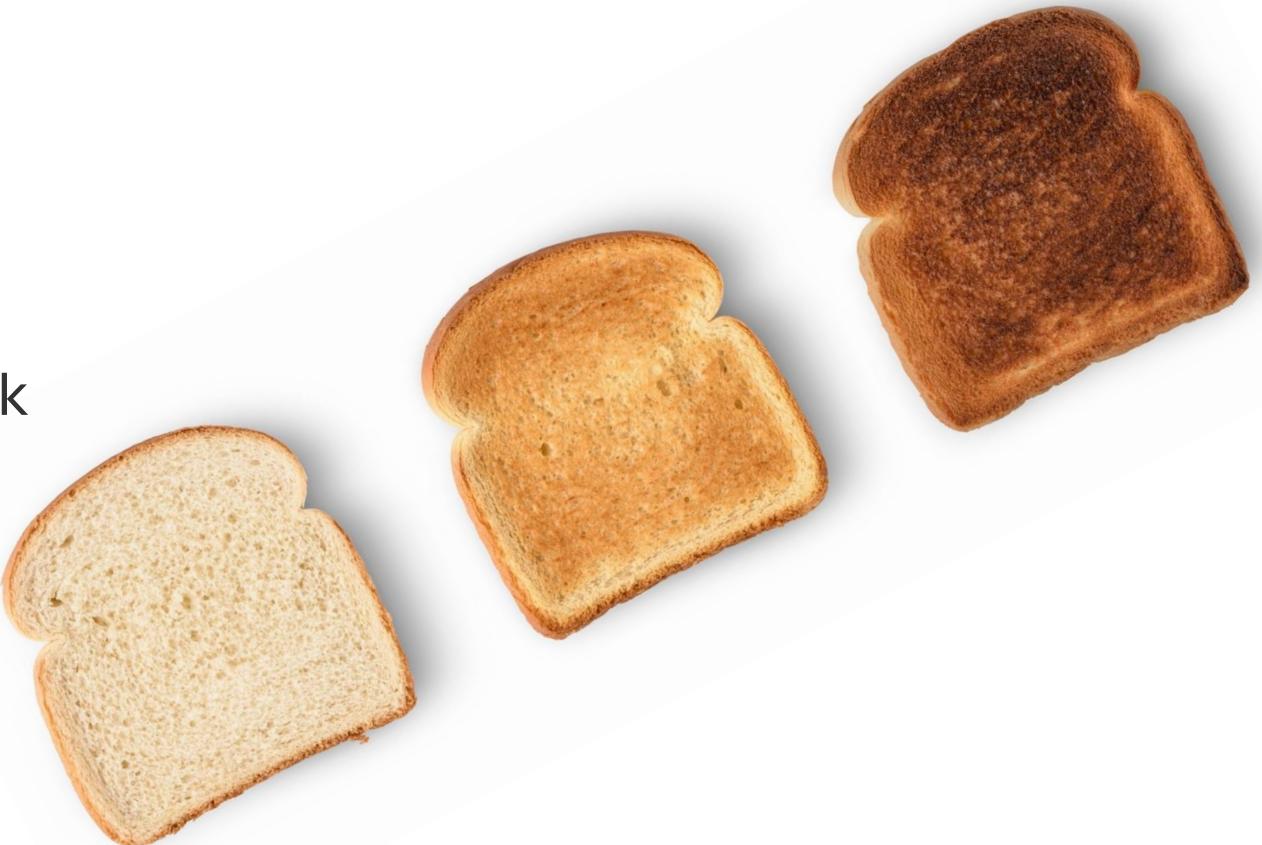
Fall 2025

Time and location: **Mondays, Wednesdays and Fridays, 5:00-5:50 PM, KNOX 109.**

- Background Material
- Common Mistakes
- Algos via Examples
- CSE 331 Care Package
- Support Page Home

Advice: Solving Problems

1. Read problem carefully!
2. Think about problem for a few minutes.
3. Keep notes about ideas.
4. Step away when stuck.
5. Schedule time in advance to work on problems.
6. Go to office hours if stuck.
7. Turn in what you have before the deadline.



Advice: Big Three

1. Read Carefully
2. Don't Cheat
3. Work Hard

Questions?

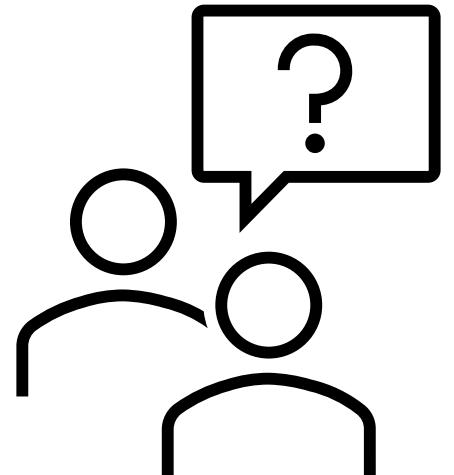


Does it halt?

Q1: What does it mean for a program to “terminate?”

Q2: Can you think of program which does not terminate on any input?

Q3: Can you think of a program that terminates on all but one specific input?



Does it halt?

```
def add (a,b):  
    c = a+b  
    return c
```

```
def badd (a,b):  
    c = a+b  
    while (True):  
        continue  
    return c
```

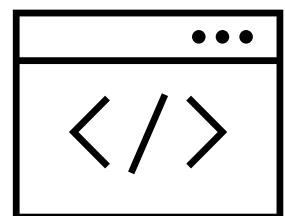
```
def badd2 (a,b):  
    c = a+b  
    if (c == 2):  
        return c  
    while (True):  
        continue
```



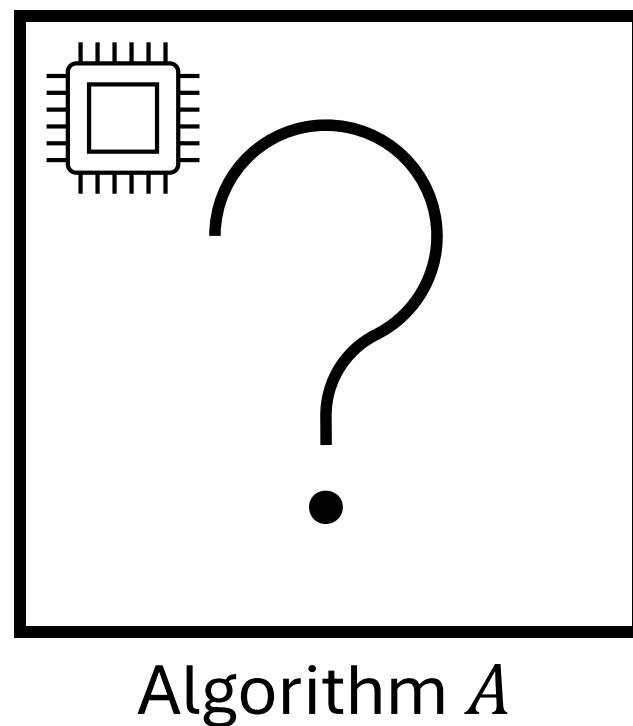
Does it halt?

Input: A Program P .

Output: True if P terminates on all inputs. Otherwise, false.



Input Program P



True if P terminates on all inputs.

False if P does not terminate on at least one input.

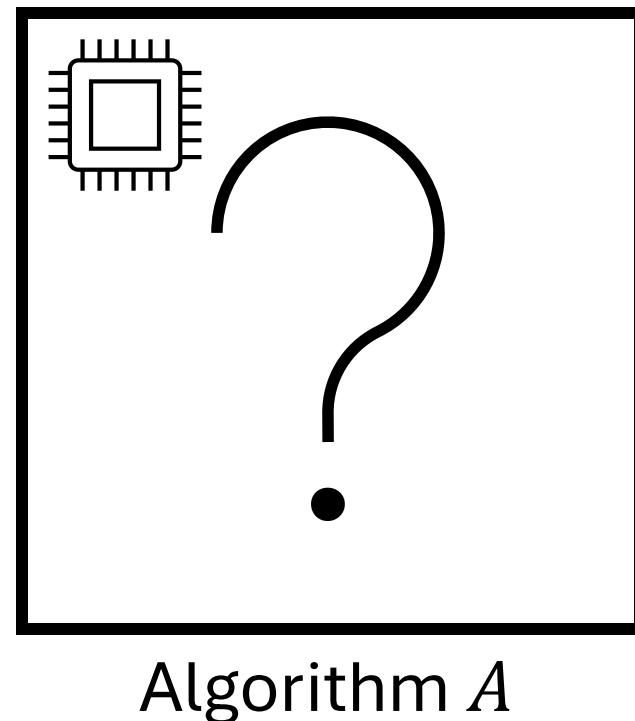
Does it halt?

Input: A Program P .

Output: True if P terminates on all inputs. Otherwise, false.

```
def add (a, b) :  
    c = a+b  
    return c
```

Input Program P



True if P terminates on all inputs (a, b) .

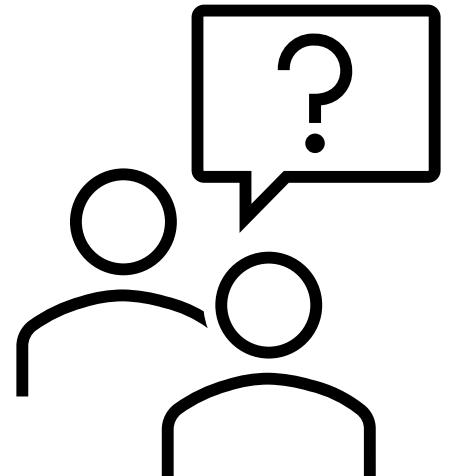
False if P does not terminate on at least one input (a, b) .

Does it halt?

Q4: How would you try to solve this problem?

A4: You would not...

Q5: What could you do instead?

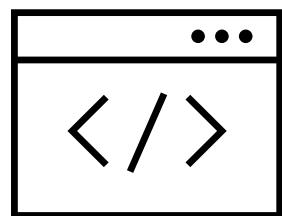


Does it halt 2.0?

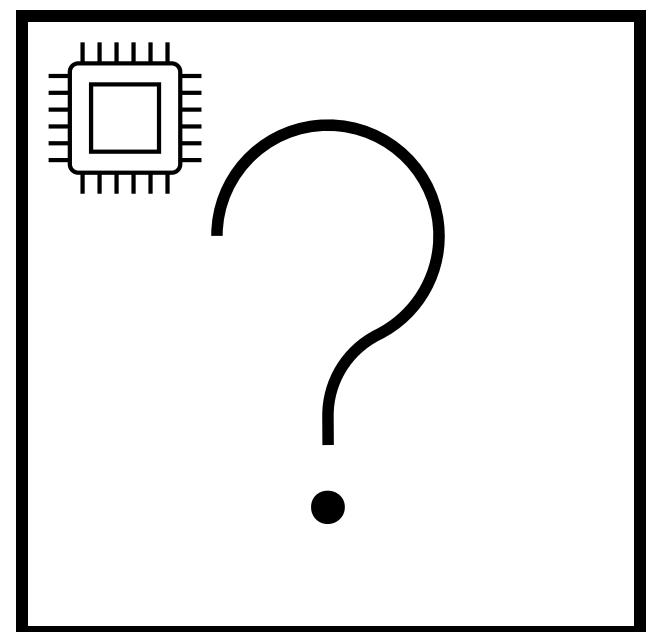
Input: A Program P and Input x .

Output: True if P terminates on x . Otherwise, false.

Program P



Program x



Algorithm A'

True if P terminates on x .

False if P does not terminate on x .

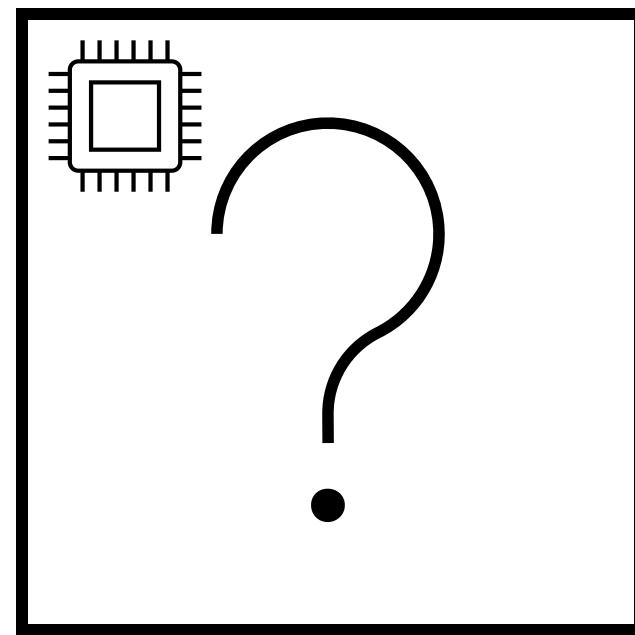
Does it halt 2.0?

Input: A Program P and Input x .

Output: True if P terminates on x . Otherwise, false.

```
def add (a, b) :  
    c = a+b  
    return c
```

Program P and input (a, b)



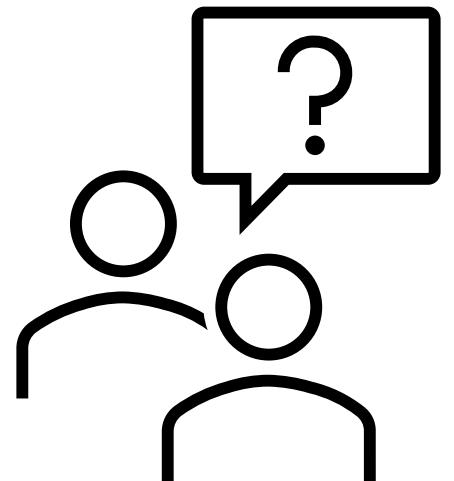
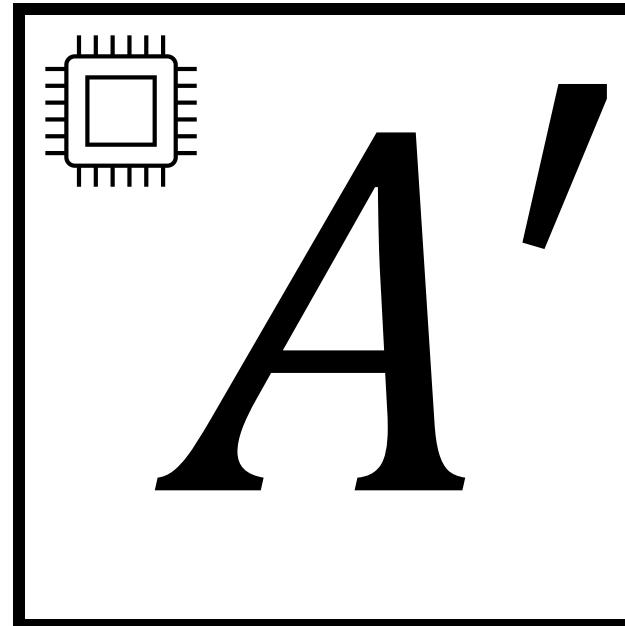
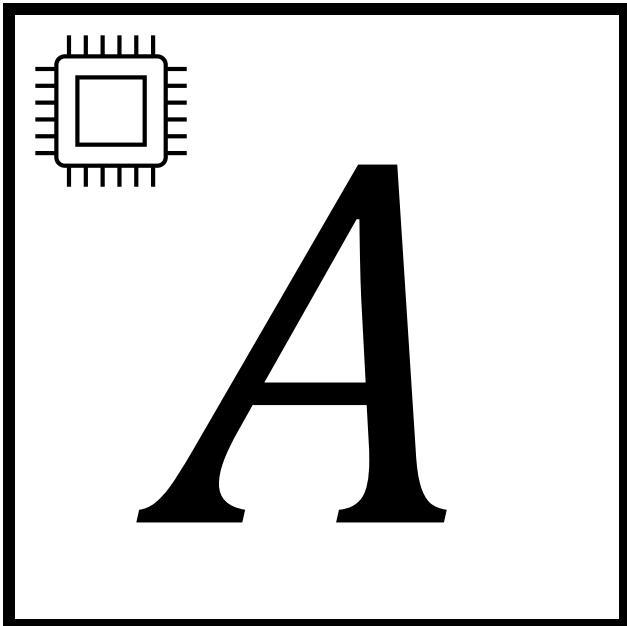
Algorithm A'

True if P terminates on (a, b) .

False if P does not terminate on (a, b) .

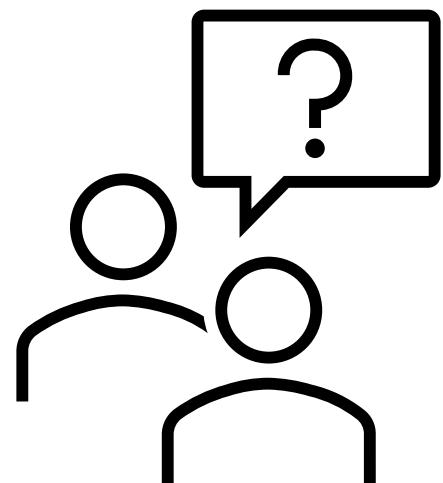
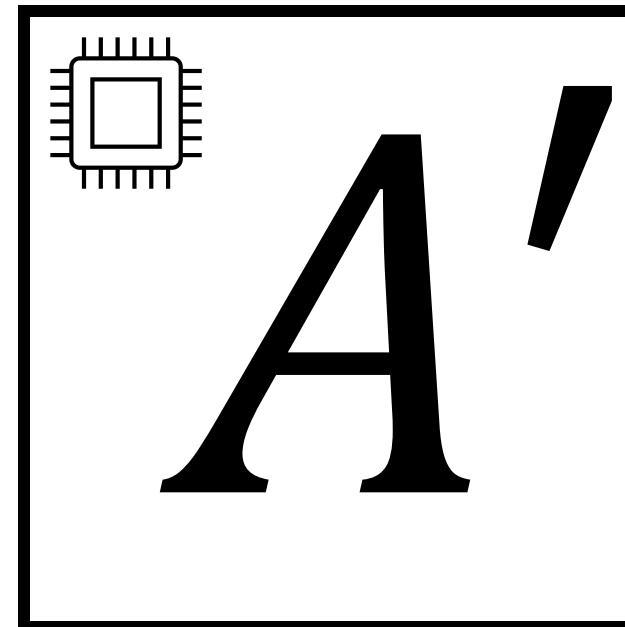
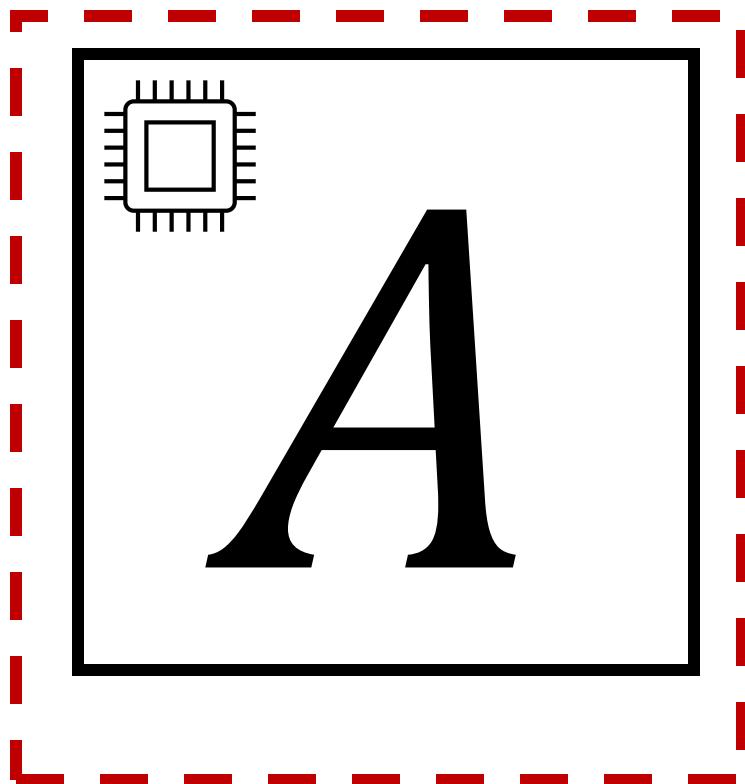
Does it halt?

Q5: Which is harder to write? Why?



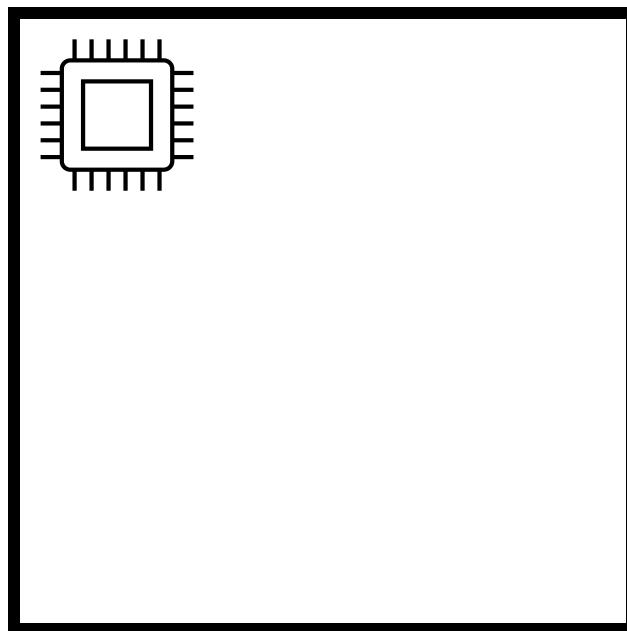
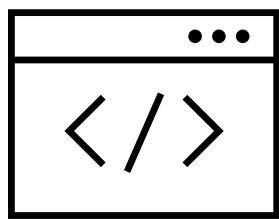
Does it halt?

Q5: Which is harder to write? Why?



Universal Turing Machine

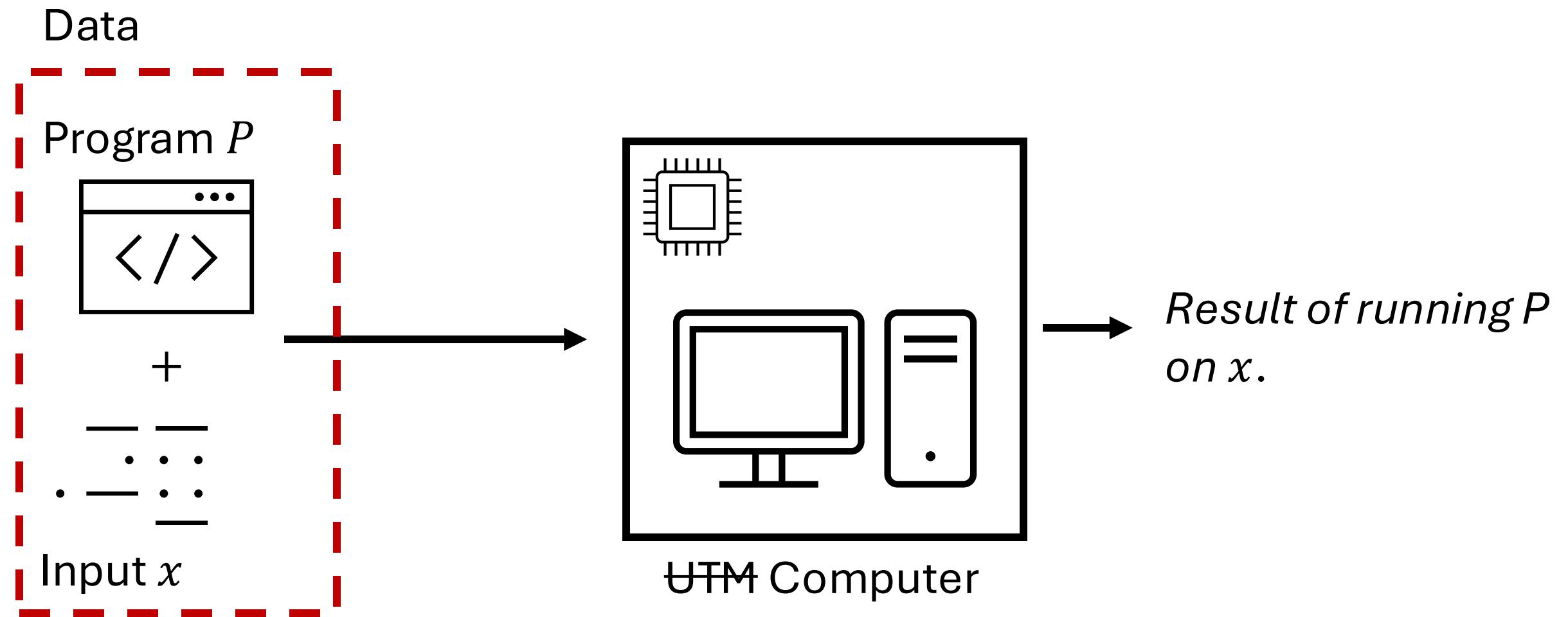
Program P



Result of running P on x .

Input x

Universal Turing Machine



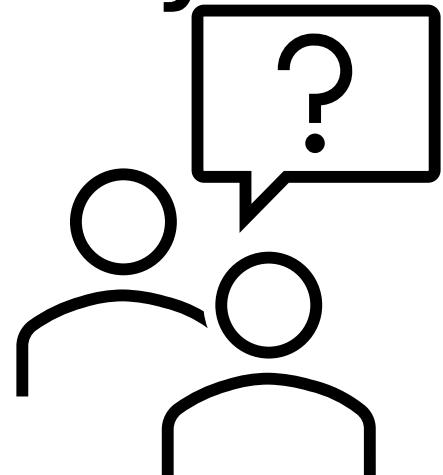
A Wild Theorem

Theorem: There exist no algorithm/program that will always halt and will always correctly decide Halting Problem 2.0.

A Wild Theorem

Q6: How would you prove this?

Theorem: There exist no algorithm/program that will always halt and will always correctly decide Halting Problem 2.0.



Proof by Contradiction

1. Assume negation of proposition.
2. Derive a contradiction from such an assumption.
3. Conclude that the original assumption is wrong and thus... proposition is true.



Theorem Proof

Suppose there exists an algo/program $\text{MAGIC}(P, I)$ that decides if program P terminates on input I .

```
def magic( P, I ):  
    # This is a magic box so there is no real code here!  
    ''' Return True if P halts on I and False otherwise'''
```

Theorem Proof

That means $\text{MAGIC}(P, I)$ always terminates and returns the correct answer.

```
def magic( P, I ):  
    # This is a magic box so there is no real code here!  
    ''' Return True if P halts on I and False otherwise'''
```

Theorem Proof

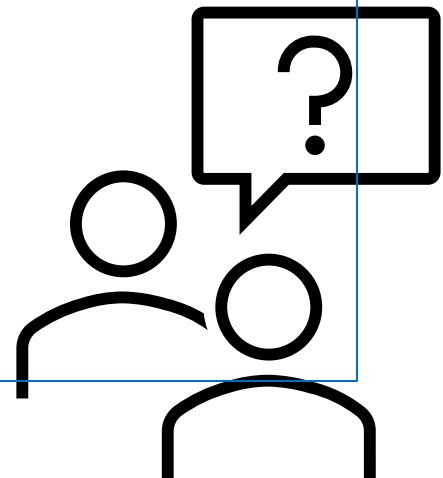
Define $\text{Contradiction}(P)$:

```
def contradiction ( P ): # This function takes a program as an input
    #Run magic on (P,P)
    if magic(P,P): # Use an UTM to make this call
        while True:
            continue # Do nothing
    return # Just terminate if magic(P,P) returns False
```

That's not right...

**Q7: What if we run
Contradiction (Contradiction) on the UTM?**

```
def contradiction ( P ): # This function takes a program as an input  
  
#Run magic on (P,P)  
if magic(P,P): # Use an UTM to make this call  
    while True:  
        continue # Do nothing  
  
return # Just terminate if magic(P,P) returns False
```



Case I: P terminates on P

Q8: What happens if Contradiction terminates on Contradiction?

```
def contradiction ( P ): # This function takes a program as an input

#Run magic on (P,P)
if magic(P,P): # Use an UTM to make this call
    while True:
        continue # Do nothing

return # Just terminate if magic(P,P) returns False
```

Case I: P terminates on P

Q8: What happens if Contradiction terminates on Contradiction?

```
def contradiction ( P ): # This function takes a program as an input  
  
#Run magic on (P,P)  
if magic(P,P) True: # Use an UTM to make this call  
    while True:  
        continue # Do nothing | Does not terminate!  
    return # Just terminate if magic(P,P) returns False
```



Case II: P does not terminates on P

Q9: What happens if Contradiction does not terminate on Contradiction?

```
def contradiction ( P ): # This function takes a program as an input

#Run magic on (P,P)
if magic(P,P): # Use an UTM to make this call
    while True:
        continue # Do nothing

return # Just terminate if magic(P,P) returns False
```

Case II: P does not terminates on P

Q9: What happens if Contradiction does not terminate on Contradiction?

```
def contradiction ( P ): # This function takes a program as an input  
  
#Run magic on (P,P)  
if magic(P,P) False : # Use an UTM to make this call  
    while True:  
        continue # Do nothing  
  
    Does terminate!  
    return # Just terminate if magic(P,P) returns False
```

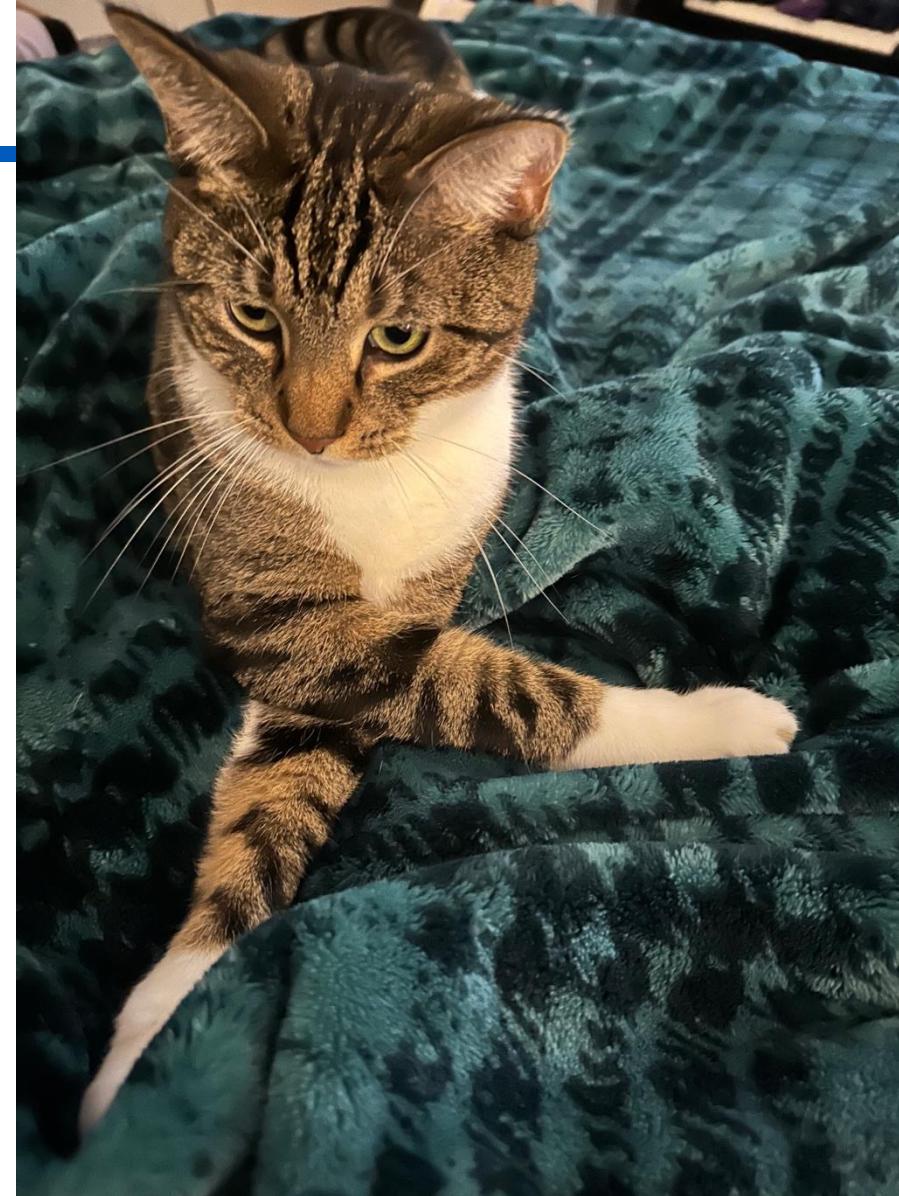


Proof Conclusion

1. Assume MAGIC exists.
2. Use MAGIC to construct Contradiction.
3. “Law of excluded middle”
 1. If Contradiction (Contradiction) terminates then does not...
 2. If Contradiction (Contradiction) does not terminate then does...
4. Thus, something we said must be wrong.

Q10: What could be wrong?

Questions?



Celebrate!



Proof by Induction

Proof by induction,
idk, I don't understand
math.

