

2022 Workshop on Gravitational Waves and High-Performance Computing

Geoffrey Lovelace

August 15, 2022 – August 19, 2022

Welcome to the workshop!

- Please make sure you create a **free** account at <https://cocalc.com>
- Workshop supported by the National Science Foundation
- Website with useful materials: <https://geoffrey-lovelace.com/Workshop/2022>



Photos

- We would like to take some photos during the workshop
- The photos would appear on the Cal State Fullerton website, in news stories about the workshop
- If you would prefer to not have your picture taken, please let me know (message me or speak with me during the break) that you'd like to opt out

A commonly held **inaccurate** model of teaching and learning



Joe Reddish, 2001, AAPT, San Diego

Bill Watterson - Calvin and Hobbs

Results from cognitive science and education research

- Learning requires **mental effort**.
- New information must link with what you **already know**.
- **Most people learn best when interacting with others.**

Daily schedule

- Morning: 9:30 AM - 11:00 AM
- Afternoon I: 12:30 PM - 2:00 PM
- Afternoon II: 2:30 PM - 4:00 PM

Tentative schedule

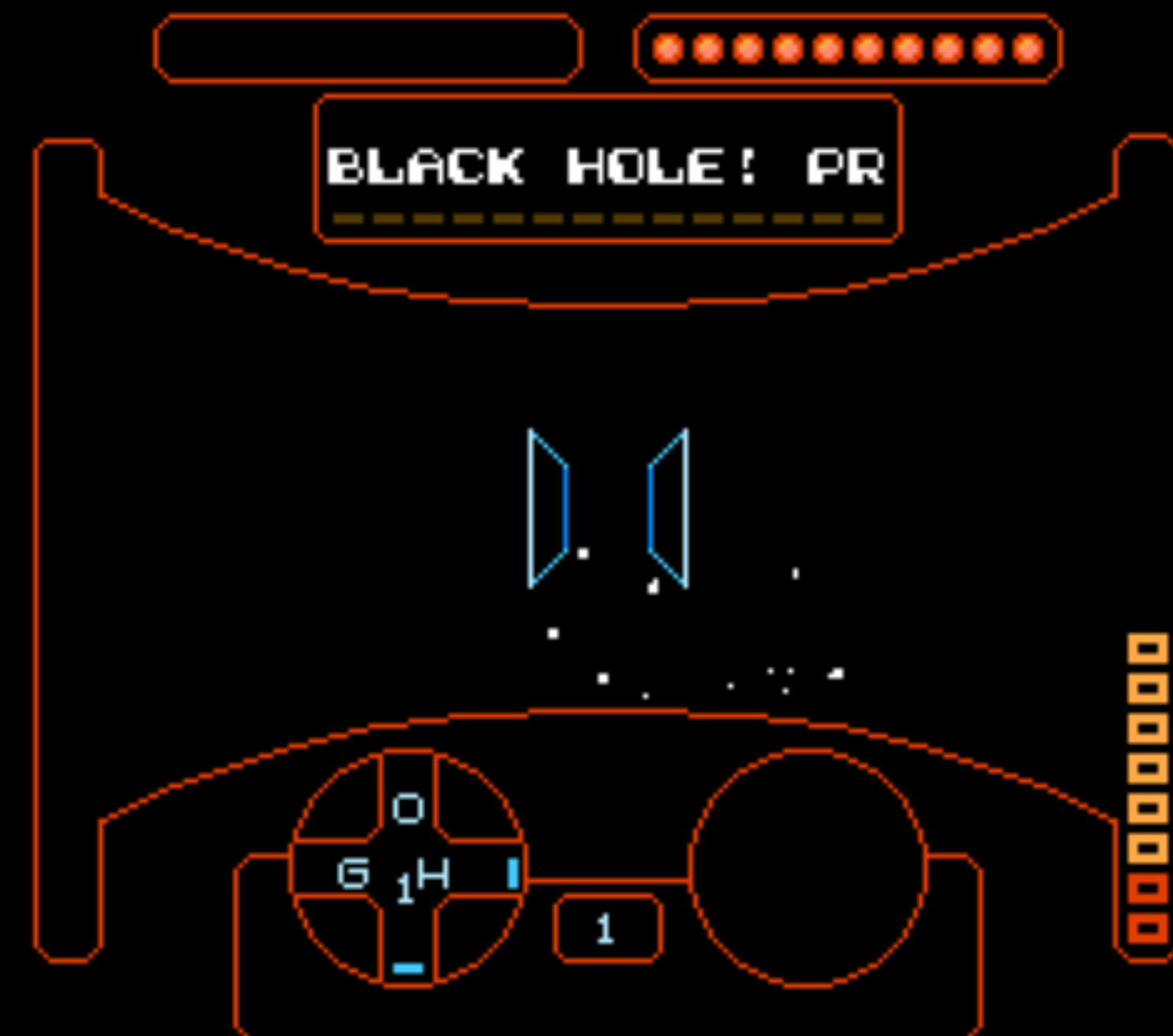
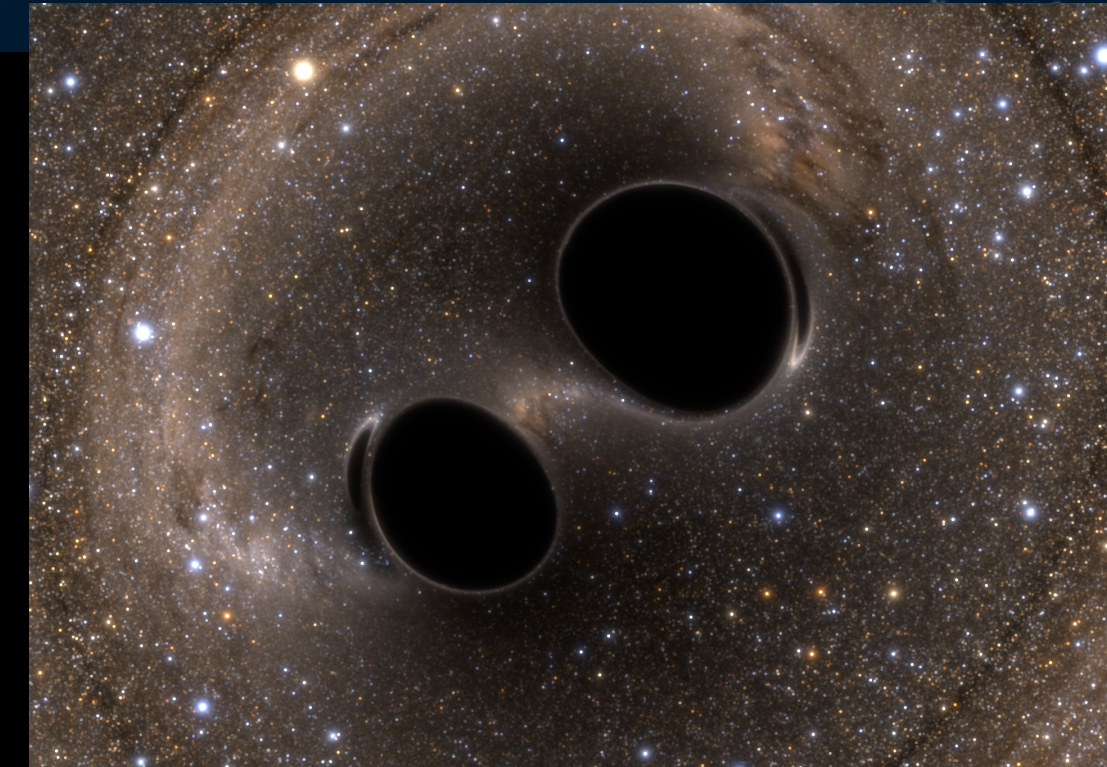
- **Monday:** Powers of 10 & computing, programming with Python
- **Tuesday:** Programming with Python, Unix Command Line, using a supercomputer
- **Wednesday:** Simulating colliding black holes, black holes, gravitational waves
- **Thursday:** Gravitational-wave research, panel discussion, data center tour (if possible)
- **Friday:** visualizing colliding black holes, exit survey

About the pace...

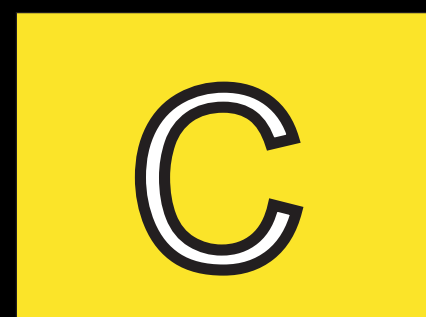
- The pace is intense: you'll be learning a lot
- It's normal to feel confused...that's actually what learning feels like
- There is no such thing as a dumb question!!
- You will get the most out of this experience by participating! It's more like learning a sport or a musical instrument or a language or ...

GW PAC

GRAVITATIONAL WAVE
Physics and Astronomy Center



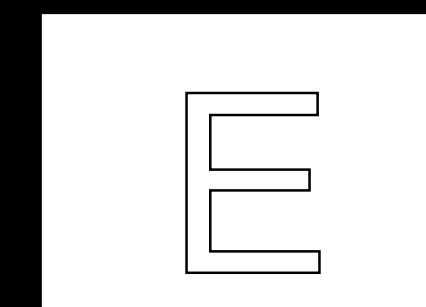
Which brother am I?



Both



Neither



Not sure

Icebreaker

- If you had to gain one superpower, which one would you choose?

A pink square containing the white letter 'A'.

Ability to fly

A green square containing the white letter 'B'.

Power to be invisible

Powers of 10

- How many meters across is Earth?

A	10^6
B	10^7
C	10^8
D	10^9



100 million light years



24
10
meters

Powers of 10

- How many meters across is Earth?

A	10^6
B	10^7
C	10^8
D	10^9

Powers of 10

- How many meters is a light year?

A

10^8

B

10^{12}

C

10^{16}

D

10^{20}

Powers of 10 & computers

Humans

- First entities called “computers” were teams of people
- Divide up the work into operations done in parallel, by hand (perhaps with mechanical aid)
- Redundant calculations to check accuracy
- Since 1700s
- 10^{-1} to 1 FLOPS / human
(decimal operations / second / human)

Image courtesy wikipedia



1949 NACA High Speed
Flight Station “Computer Room”)

Colossus (1942)

- First programmable, digital, electronic computer
- Break codes in World War II Britain
- 5×10^5 FLOPS

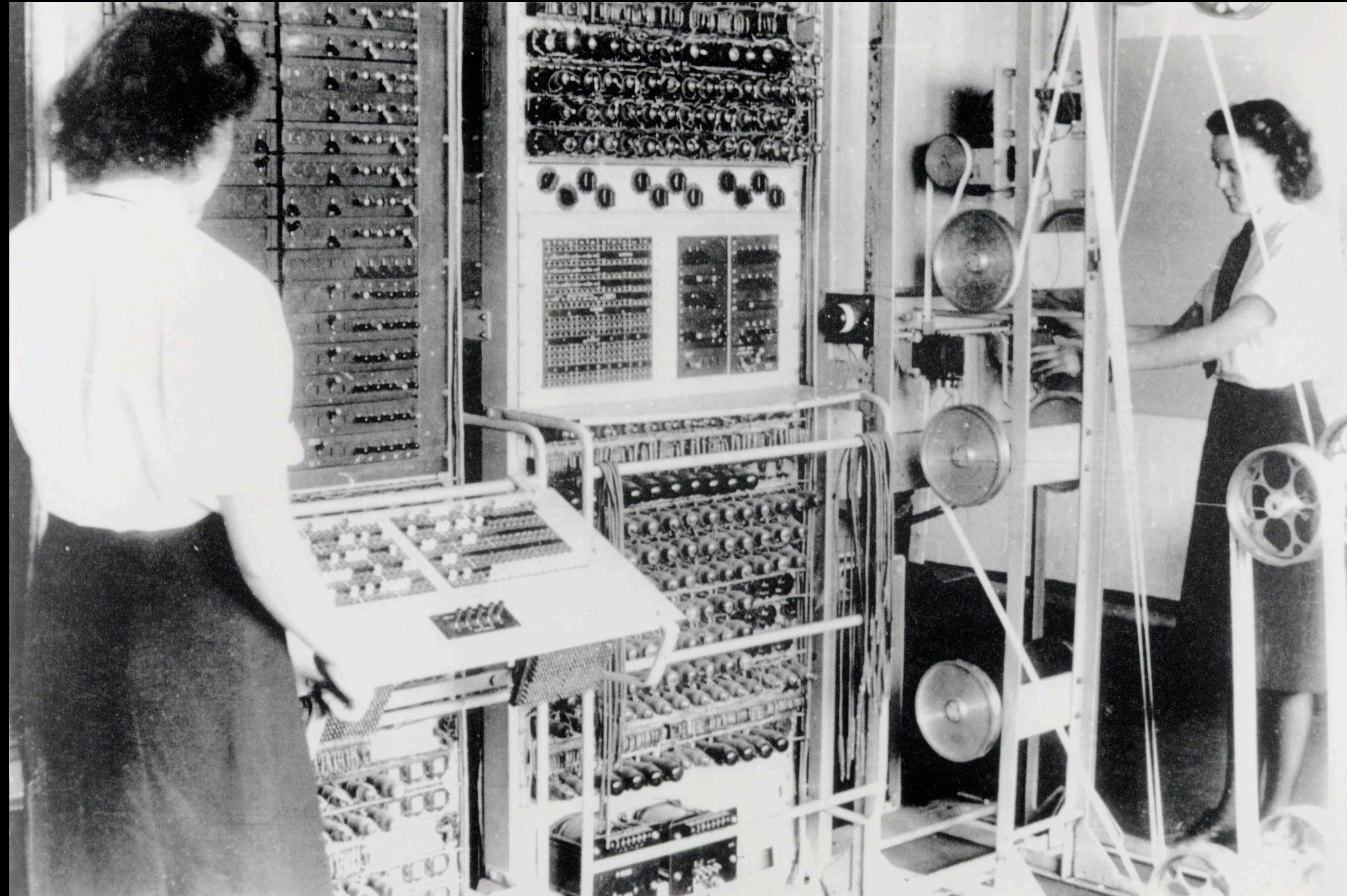


Image courtesy wikipedia

My first Mac (1984)

- First Macintosh
- 1×10^6 FLOPS



My Mac in 2003

- 2 cores
- $1-2 \times 10^9$ FLOPS



Image courtesy Apple

My current Mac

- Apple M1 Max (10 cores)
- 3×10^{11} FLOPS



Image courtesy Apple

My current iPhone

- 6 cores
- 1×10^{11} FLOPS



Image courtesy Apple

Clicker question #1.4

- In 1 second, today's high-end smart phones can perform _____ calculations per second (FLOPS).

A

10^6 (1 million)

B

10^9 (1 billion)

C

10^{11} (100 billion)

D

10^{13} (10 trillion)

Clicker question #1.5

- In 1 second, today's high-end smart phones can perform as many calculations as _____ humans?

A

10^6 (1 million)

B

10^9 (1 billion)

C

10^{11} (100 billion)

D

10^{13} (10 trillion)

For comparison:

Humans alive in 2018: 7.6×10^9

Total humans who ever lived: 10^{11}

Sources: [google.com](https://www.google.com), [pro.org](https://www.pro.org)

Clicker question #1.3

- Today's most powerful computers are _____ times more powerful than *today's high-end personal computers*.

A

10 (ten)

B

10^4 (10 thousand)

C

10^7 (10 million)

D

10^{10} (10 billion)

Ocean supercomputer at Cal State Fullerton

- Supercomputer for Cal State Fullerton Gravitational-Wave Physics and Astronomy Center
- 828 cores
- $\approx 2 \times 10^{12}$ FLOPS



Frontera

- Most powerful computer I have access to
- 470k cores
- 2.35×10^{16} FLOPS



Frontier

- Most powerful computer in the world as of June 2022
- 8.7 million cores
- 1.6×10^{18} FLOPS



High performance computing

- Computing beyond what personal devices can do
- Many cores work together in parallel

FLOPS	Example	Computing Type
10^0	<i>Addition by human with pen & paper</i>	<i>Early</i>
10^5	<i>Room-sized computer in 1940s</i>	
10^6	1980s personal computers (1984)	Personal
10^9	Personal computers around year 2000	
10^{11}	High-end PC/smartphone today	
10^{12}	Small supercomputer today	High-Performance
10^{16}	Most powerful computer I can access	
10^{18}	Most powerful computer in the world	

Clicker question #1.3

- Today's most powerful computers are _____ times more powerful than *today's high-end personal computers*.

A

10 (ten)

B

10^4 (10 thousand)

C

10^7 (10 million)

D

10^{10} (10 billion)

Clicker question #1.6

- In 1 second, the most powerful computer in the world can perform as many calculations as _____ humans?

A

10^8 (100 million)

B

10^{11} (100 billion)

C

10^{14} (100 trillion)

D

10^{18} (1 quintillion)

For comparison:

Humans alive in 2018: 7.6×10^9

Total humans who ever lived: 10^{11}

Sources: [google.com](https://www.google.com), [pro.org](https://www.pro.org)

Clicker question #1.7

- In 1 second, a small supercomputer like Ocean can perform as many calculations as _____ humans?

A

10^6 (1 million)

B

10^9 (1 billion)

C

10^{12} (1 trillion)

D

10^{15} (1 quadrilliion)

For comparison:

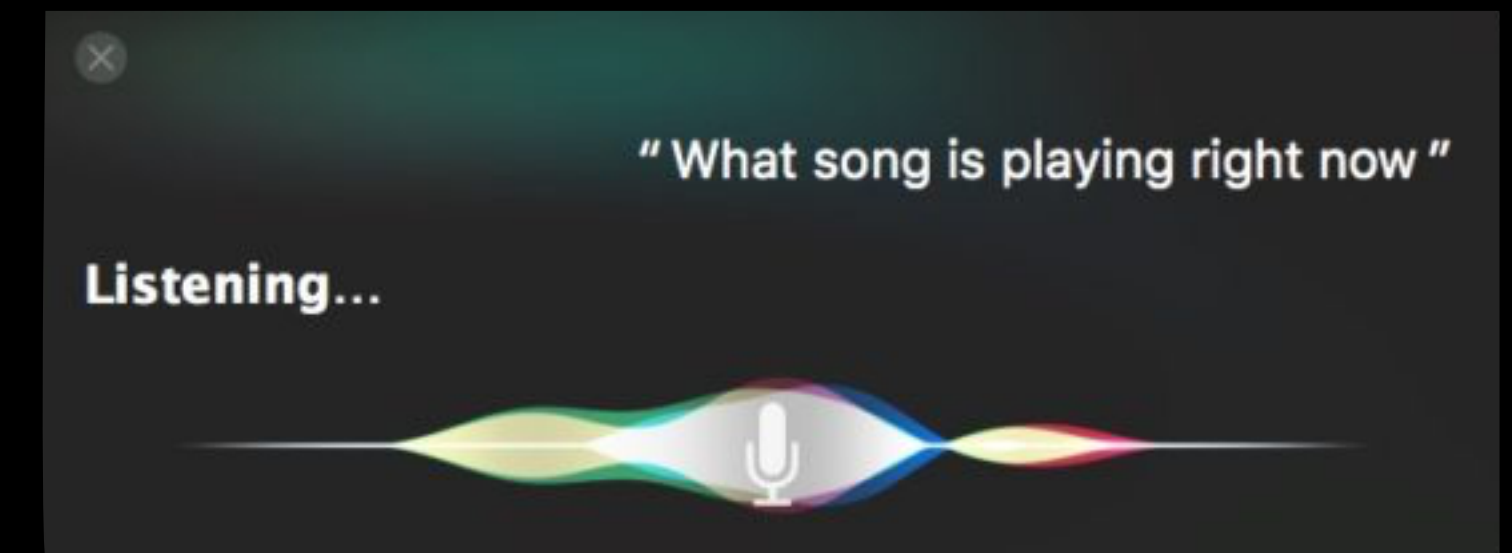
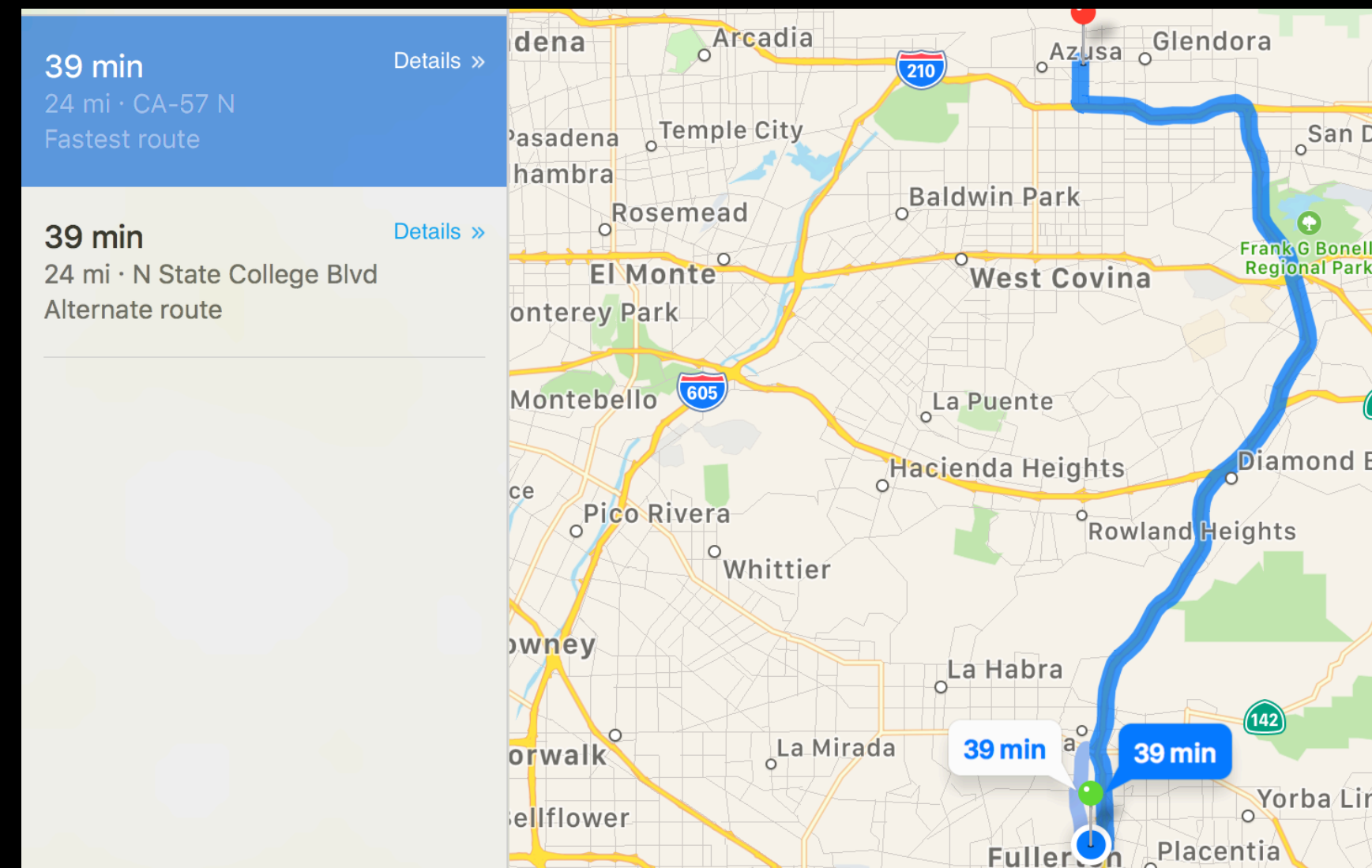
Humans alive in 2018: 7.6×10^9

Total humans who ever lived: 10^{11}

Sources: [google.com](https://www.google.com), pro.org

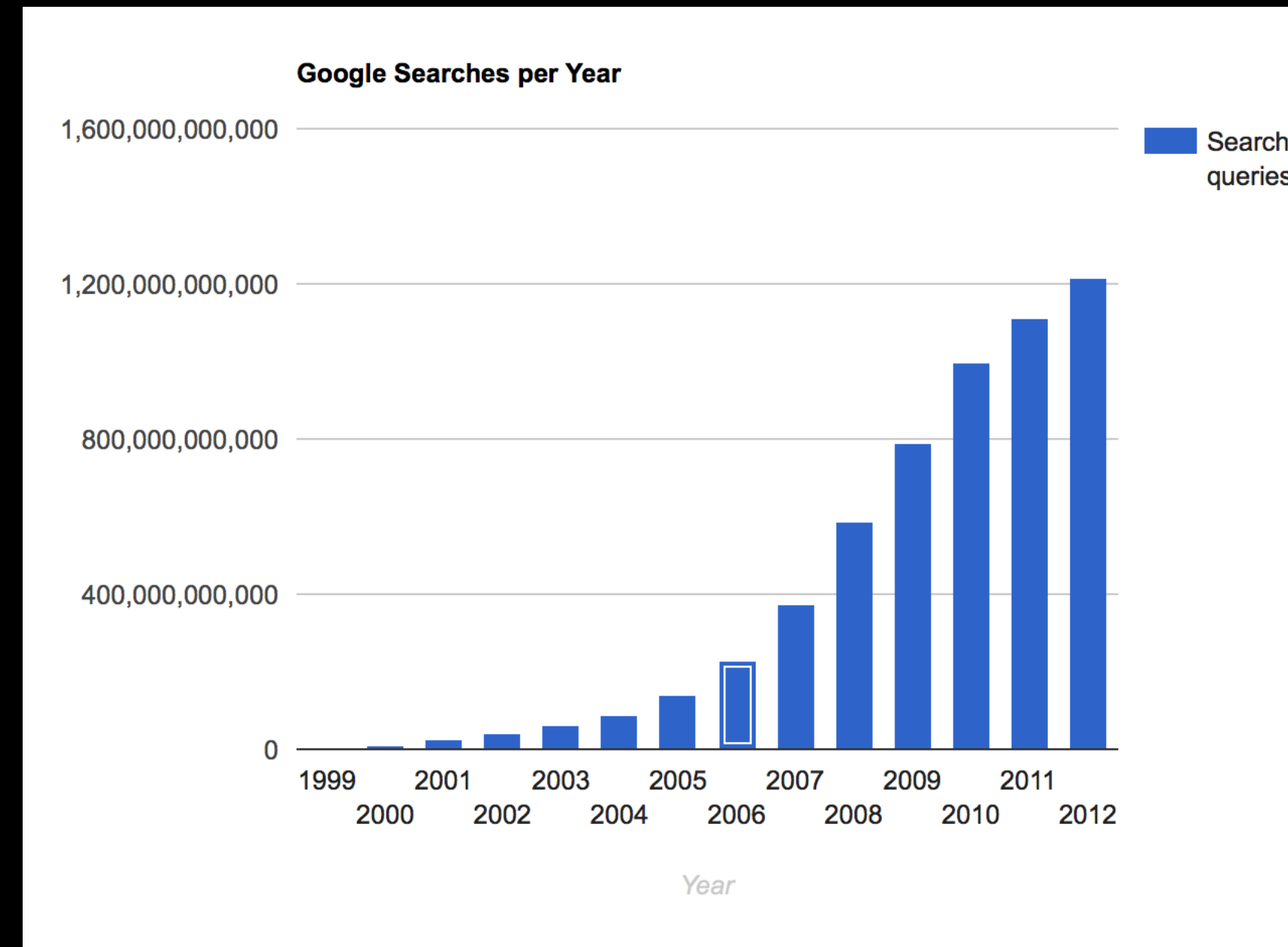
High-performance computing in everyday life

- Cloud computing
- Search the web
- Identify a song
- Get directions
- Voice assistants
- Speech recognition



Example: Google search

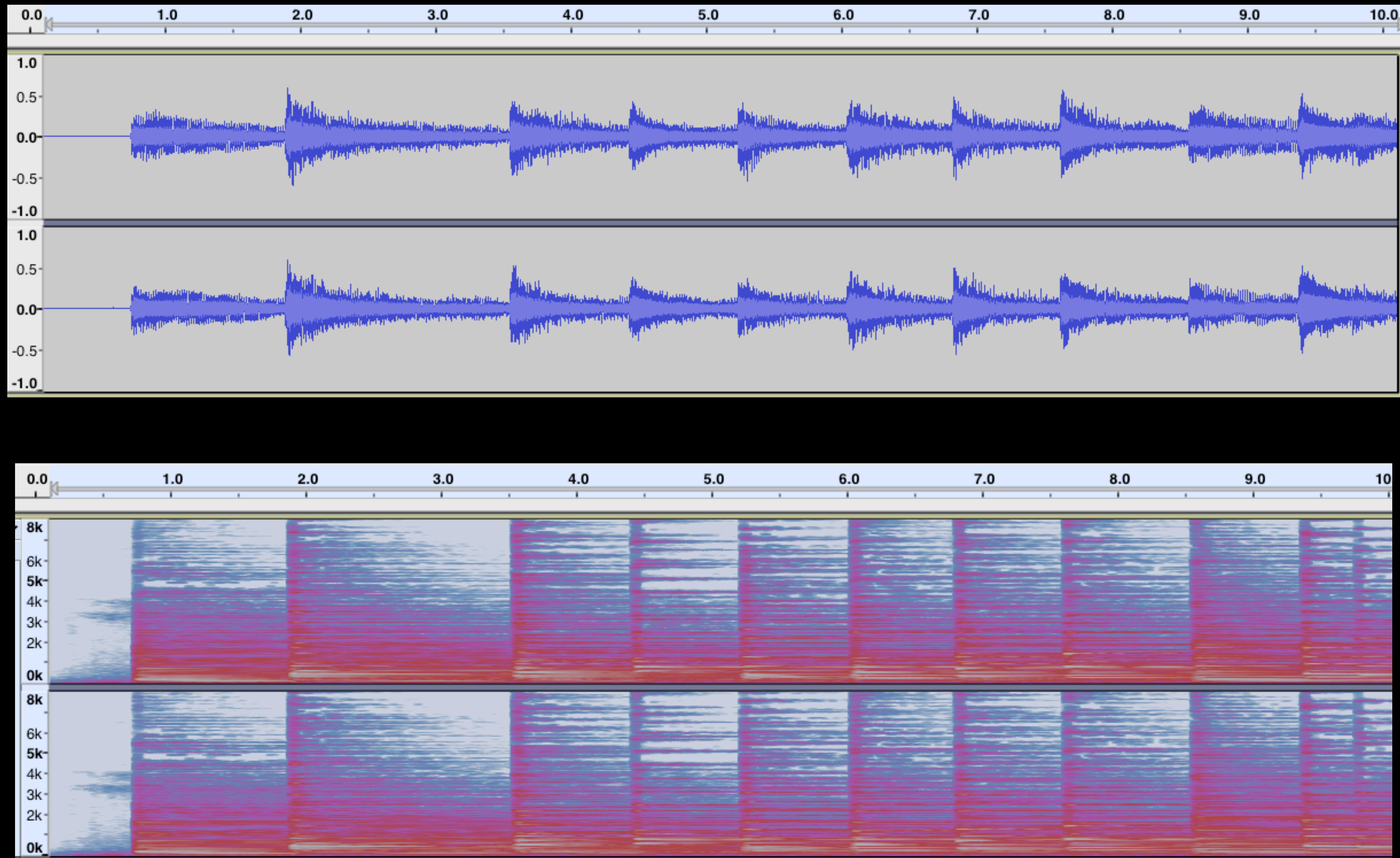
- Search $\sim 10^{13}$ web pages
- 10^3 “servers” per query
- Each query takes about 0.2 seconds
- 4×10^4 queries on average every second of every day
- If each server is “only” 10^9 FLOPS, Google search requires about 10^{16} FLOPS



Images courtesy Google,
internetlivestats.com

Example: Shazam

- 200 queries on average every second of every day
- Convert sound into time-frequency plots, filter to keep only the loudest notes
- Compare to a large library
- Similar to how LIGO searches data for gravitational waves!
- One query is a PC-sized calculation, roughly





Amazon web services data center
Courtesy [amazon.com](https://www.amazon.com)



Image courtesy cnet: Google data center,
Council Bluffs, Iowa
Google: 60,000 searches/second

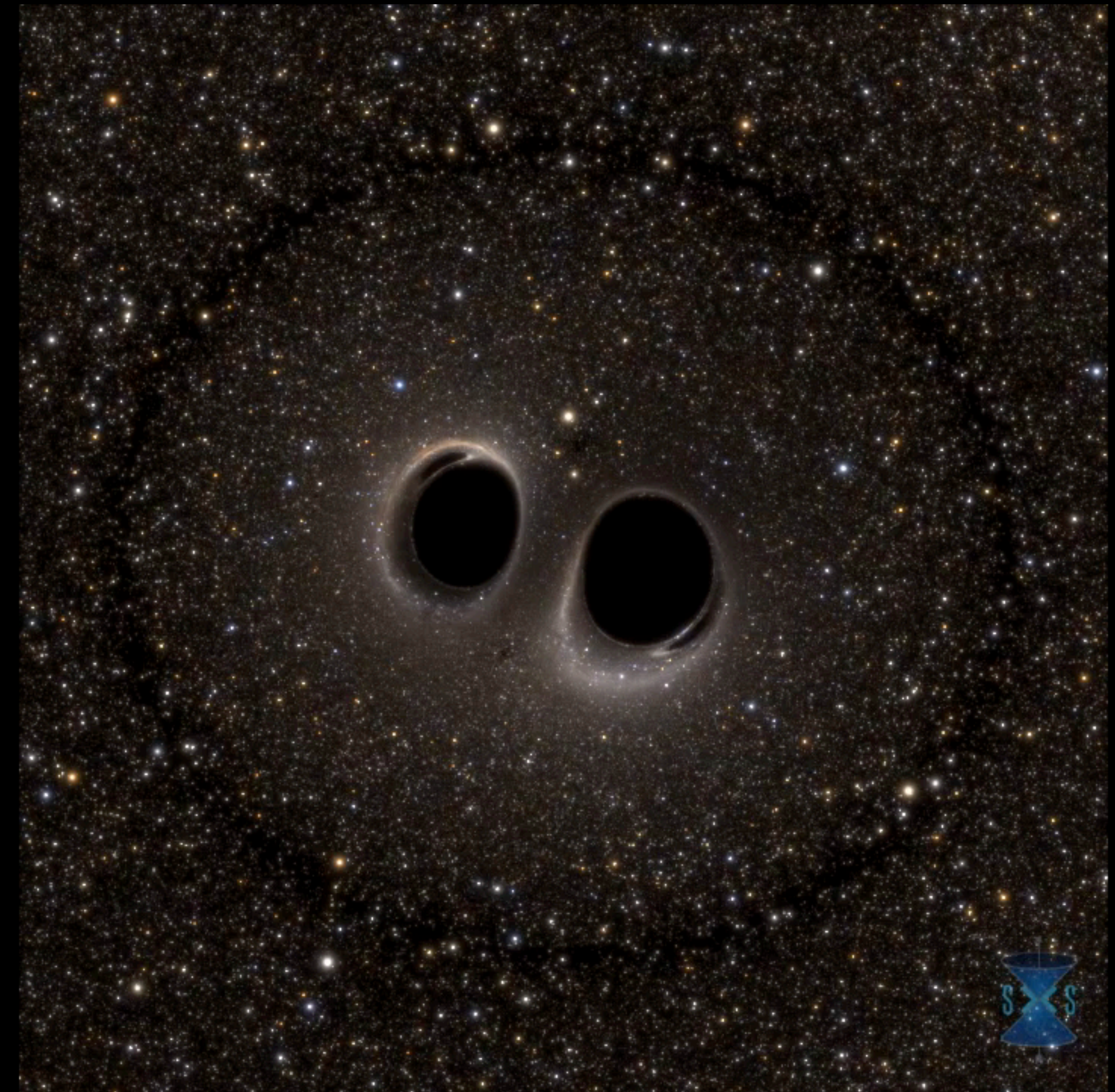


Microsoft Azure data center
(courtesy [sensorslab.co](https://www.sensorslab.co))

Provide many 10^{15} FLOPS
of performance to customers

High-performance computing for science

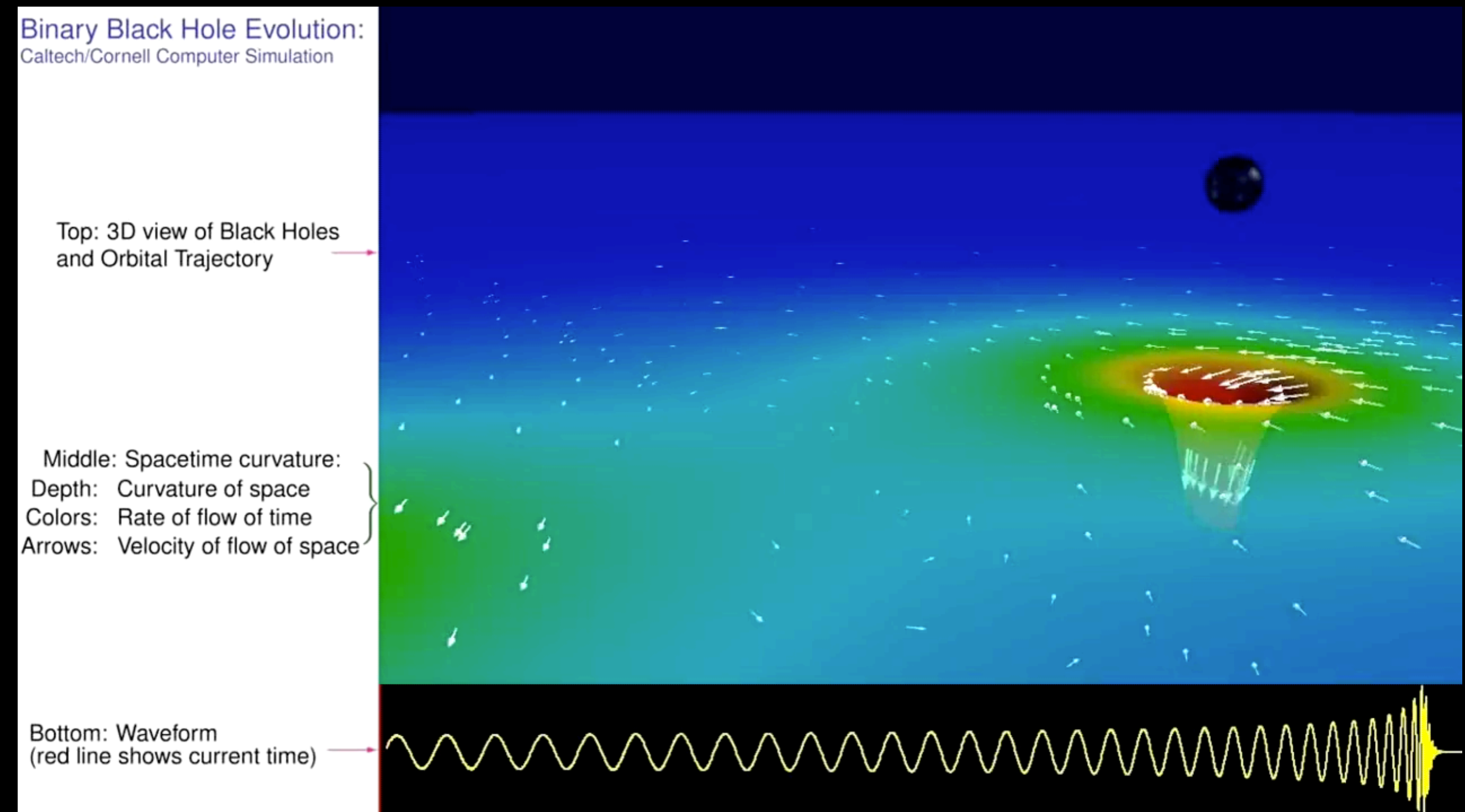
- Solve otherwise unsolvable problems
- Insight into scientific data & results
 - Experimental measurements
 - Results of calculations
 - Complicated pencil & paper results



Movie & calculation by undergraduate
Haroon Khan, Nick Demos,
Simulating eXtreme Spacetimes collaboration

Example: Simulating colliding black holes

- Head-on collision example
 - About 1.4×10^{17} floating-point operations
 - 3 days (48 (slower) cores on ocean)
 - Several weeks on a typical laptop
- Inspiral, merger, ringdown example
 - About 7.8×10^{17} floating-point operations
 - 17 days (48 cores before merger, 36 cores after, on orca)
 - Months on a typical laptop



Programming with Python

Programming is like magic

- Say the right cryptic words and something cool happens
- Mess up a word and the spell fizzles



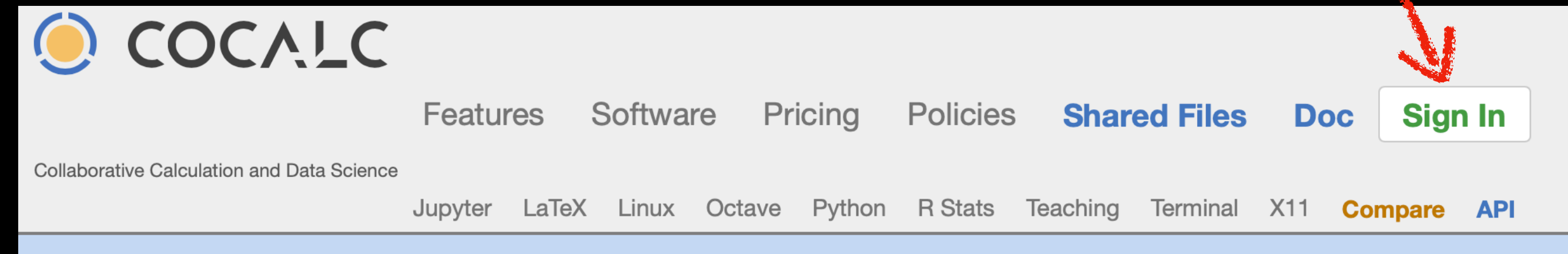
Cocalc

- <https://cocalc.com>
- Limited paid service
- This course: ~\$20/month paid plan (I paid, don't worry!)

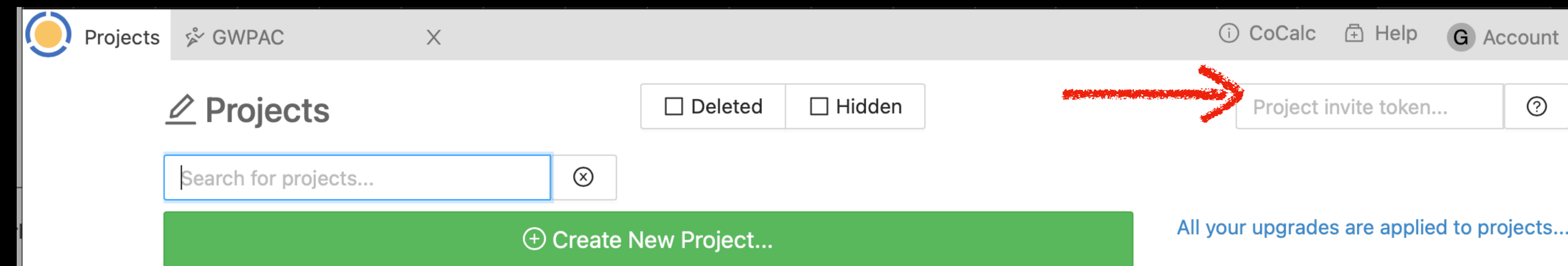


Image courtesy amazon, Tech Vision

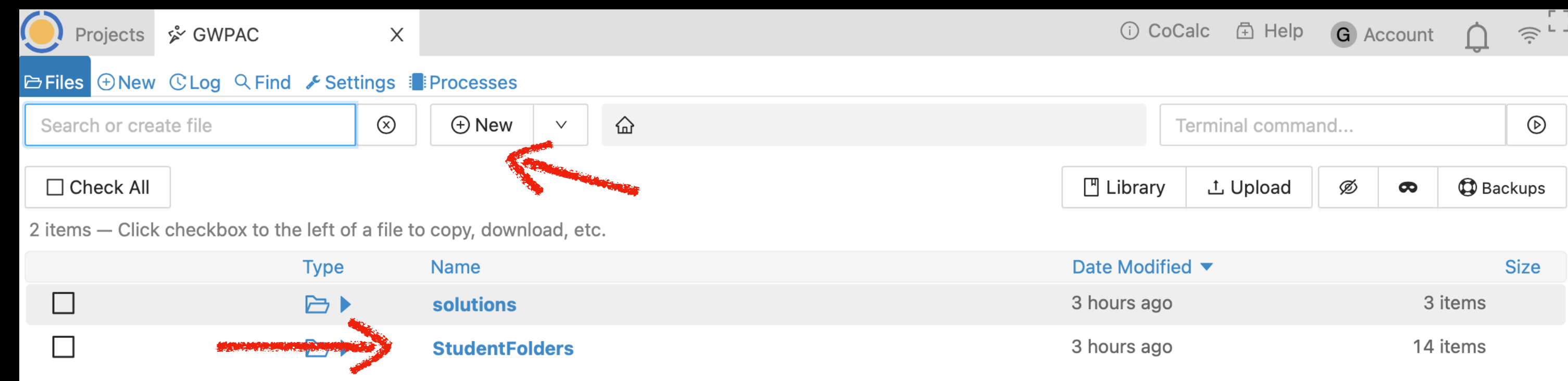
- Open <https://cocalc.com> and sign in



- See zoom chat for the "token": enter it in the "token" box and press enter



- Click "Day1.ipynb"



- Scroll to your name, and click in the box saying "# Insert code here" labeled below your name

- Enter this code: VBBNN1tpwckqE5mw

Output

- Your program needs to tell you the result
- Tradition since 1974: first program prints "Hello world"
- Python (language commonly used in scientific computing) makes this easy

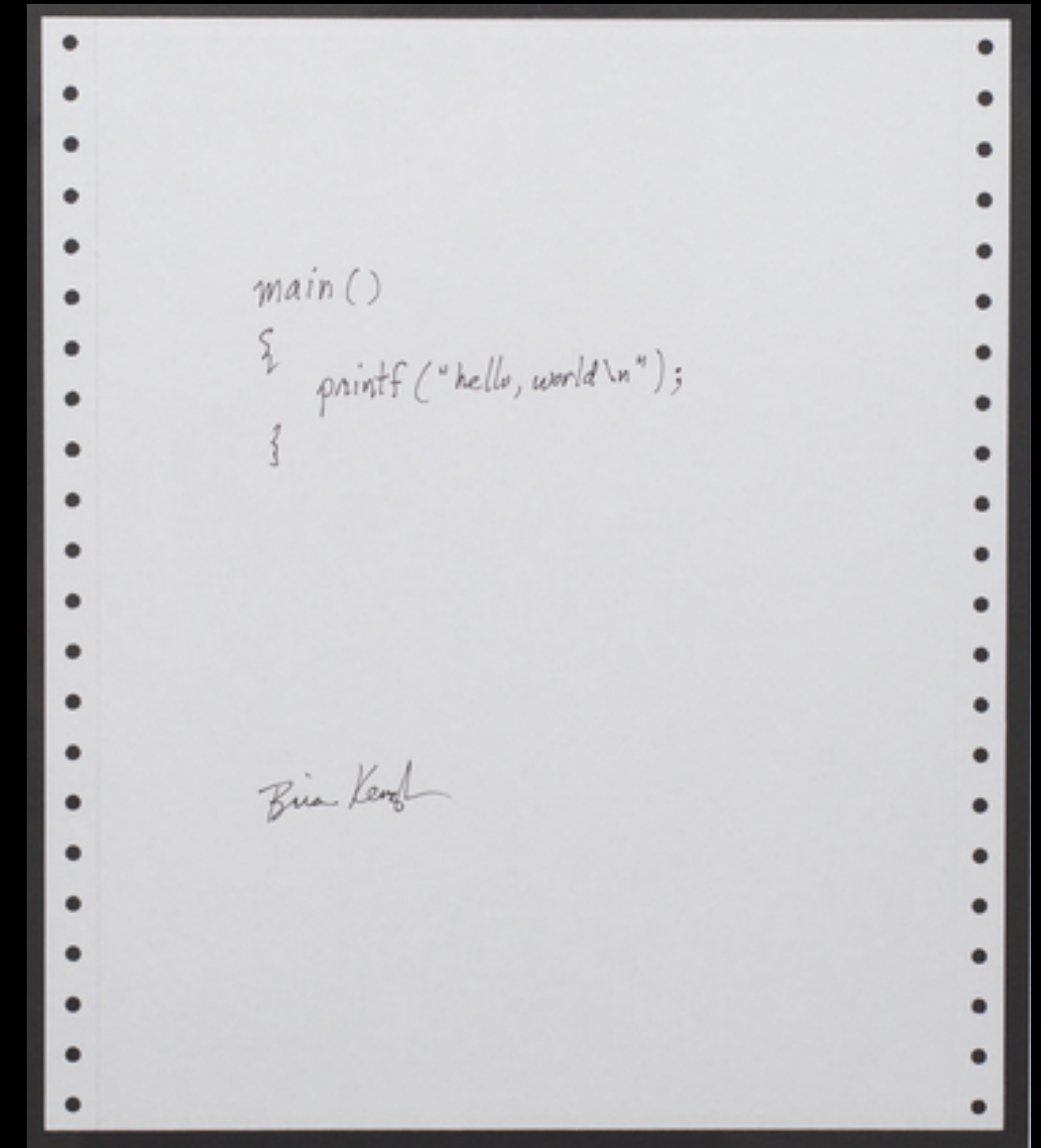
Try:

```
print("Hello, world!")
```

- Print basically anything

Try:

```
print(4*4+4-4)
```



Brian Kernighan
(early UNIX developer), 1978

Libraries

- Don't reinvent the wheel when you want to hit the road
 - (But OK if you want to learn how to make wheels)
- Python has *many* libraries for numerical computing & everything else
- By "Libraries", I mean any pre-written code that you can use in your programs

Try in tutor:

```
import math  
print(math.pi)
```


Math

Try in tutor (only type the left hand side of the ==):

- Arithmetic operations built in `(4 + 4) * 4 / 4 - 4`
- Exponents with `**` `4 ** 4 == 256`
- Scientific notation `4e4 == 40000`
- The rest in the math library `math.sin(4)`
`math.sqrt(4)`

Expressions

- Value = piece of data of a particular type

4.444

"Hello world"

- Type = kind of data

float

string

- Operator = combine values to get a new value

+ - * /

+

- Behavior depends on type

- Expression = group of values and operators

4.0 * 3.0 - 2.0

"Hello" + " world"

- Python evaluates expressions, like a calculator

Clicker question #2.0

- What does Python get when it evaluates this expression?

`4.0 * 3.0 - 2.0`

A

4.0

B

10.0

C

Some other number

D

An error

Try out some expressions

```
4.0 * 3.0 - 2.0
```

```
"Hello" + " world"
```


Try out some expressions

```
print(4.0 * 3.0 - 2.0)
```

```
print("Hello" + " world")
```

```
#make up your own
```

Some types we will need

- Float
- Int
- String
- Boolean

Type: float

- **Values:** real numbers (“numbers with decimal points”)

- Examples `4.1234` `4.0` `4.4e2` `-5.2e-3`

- If you don't include a decimal point, it is an integer!

- **Operators:** `+` `-` `*` `/` `**`

Try in tutor:

```
print(22.0 / 7.0)
```

```
print(8.0**2.0)
```

```
print(type(4))  
print(type(4.0))
```

```
print(-3.0e-3 * 10.0)
```

```
print(1.0/3.0)
```

Type: int

- **Values:** integers (whole numbers, positive, negative, zero)

- Examples `-4` `742352046` `7` `-33`

- Don't use commas when typing an int or float

- **Operators:** `+` `-` `*` `**` `/` `//` `%`

Try in tutor:

```
print(2**8)
```

```
print(4 * 3 - 2)
```

```
print(7 / 3) #float in Python3,  
            #int in Python2 (avoid!)
```

```
print(7 // 3) # quotient  
print(7 % 3)  # remainder
```


Clicker question #2.1

- In Python 3, what is the value of this expression?

```
10 // 3 + 1
```

A

4

B

4.333333333333333

C

Some other number

D

An error

Type: boolean

- **Values:** true or false
 - Examples `True` `False`
- **Operators:** `and` `or` `not`
 - `a and b` is true if both are true, false otherwise
 - `a or b` is true if a is true, b is true, or both are true
is false if both a and b are false
 - `not a` is true if a is false, false if a is true

= and ==

- = stores results in a named object ("variable")

```
myNumber = 4  
print(myNumber * myNumber)
```

```
print(myNumber * myNumber == 16)  
True
```

- == tests whether two objects are equal

```
print(2 + 2 == 5)  
False
```

Try some of these

- = stores results in a named object ("variable")
- == tests whether two objects are equal

```
print(2 + 2 == 4 and 3 + 3 == 6)
```

```
print(2 + 2 == 4 and 3 + 3 == 7)
```

```
print(2 + 2 == 4 or 3 + 3 == 7)
```

```
print(not 3 + 3 == 7)
```

```
a = True  
b = True  
c = False  
d = False
```

```
# Pick a few of these
```

```
print(a)
```

```
print(not c)
```

```
print(not a)
```

```
print(a or b)
```

```
print(a or c)
```

```
print(c or d)
```

```
print(a and b)
```

```
print(a and c)
```

```
print(c and d)
```


Converting types

Try in tutor:

```
q = 4  
print("The number is "+q)
```

```
q = 4  
print("The number is "+str(q))
```

```
print(type(4))  
print(type(str(4)))  
print(type(float(4)))
```

Clicker question #2.2

- What does this line print?

```
import math  
print("The value of pi is "+math.pi)
```

A

The value of pi is 3.141592653589793

B

The value of pi is math.pi

C

Something else but not an error

D

An error

Clicker question #2.2

- What does this line print?

```
import math  
print("The value of pi is "+str(math.pi))
```

A

The value of pi is 3.141592653589793

B

The value of pi is math.pi

C

Something else but not an error

D

An error

Comments

- Comments explain what you're doing
- Use comments to explain your code
- Use names that help explain, even without comments

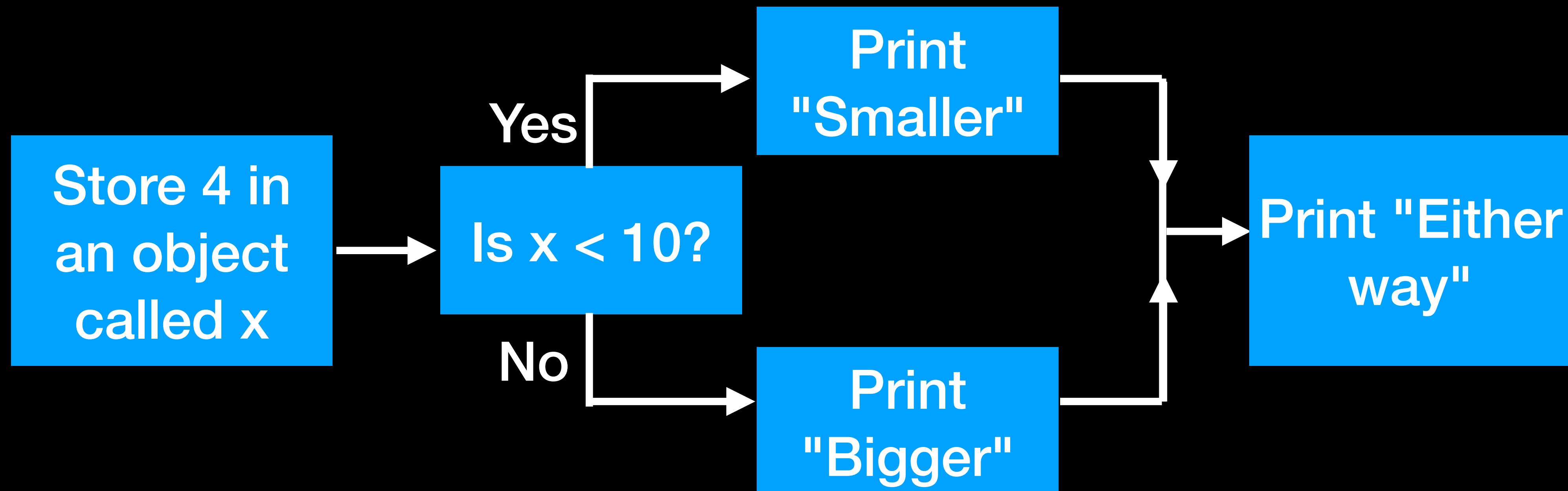
```
# Say hello to someone by name  
personName = "Geoffrey"  
print("Hello " + personName)
```


If/else

- If does the first indented thing if the stuff in () is True
- Otherwise it does the indented stuff under "else"

```
x = 4
if (x < 10):
    print("Smaller")
else:
    print("Bigger")
print("Either way.")
```

Try in tutor!



Clicker question #2.2b

- What does this program print?

```
x = 4
if x==10 or x==11:
    print('yes')
else:
    print('no')
```

A

Yes

B

No

C

The code gives an error

Clicker question #2.2

- What does this program print?

```
x = 4
if x==10 or 11:
    print('yes')
else:
    print('no')
```

A

Yes

B

No

C

The code gives an error

Clicker question #2.2

- What does this program print?

```
x = 4
if x==10 or 11:
    print('yes')
else:
    print('no')
```

```
x = 4
if false or true:
    print('yes')
else:
    print('no')
```

A

Yes

B

No

C

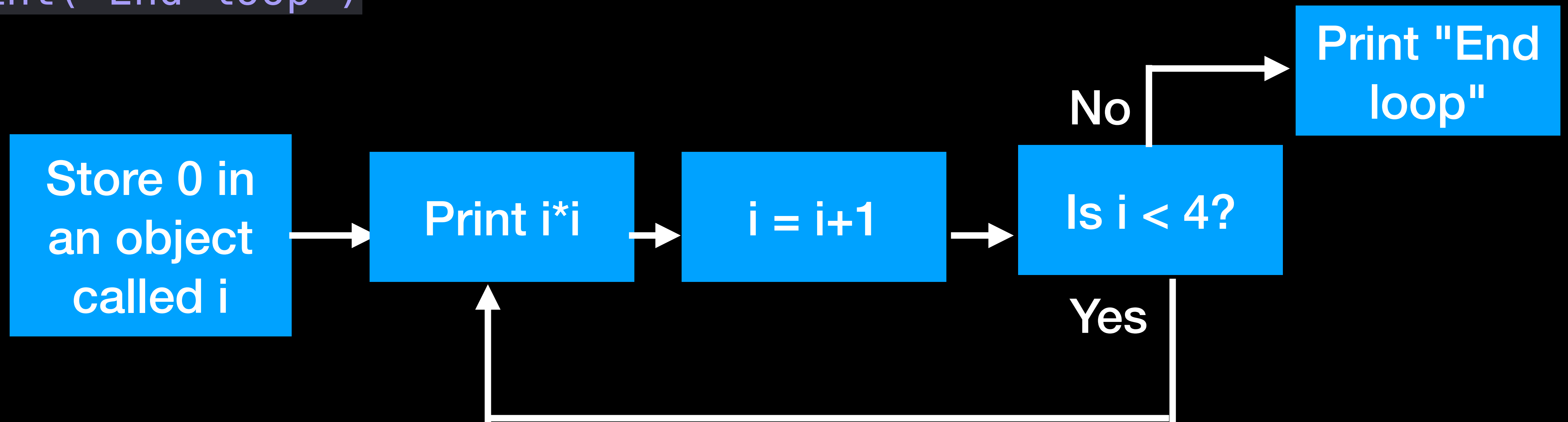
The code gives an error

Try in tutor!

```
i = 0
while i < 4:
    print(i*i)
    i = i + 1
print("End loop")
```

Loops

0
1
4
9



Loops

```
for i in [0,1,2,3]:  
    print(i*i)
```

0
1
4
9

```
i = 0  
while i < 4:  
    print(i*i)  
    i = i + 1
```

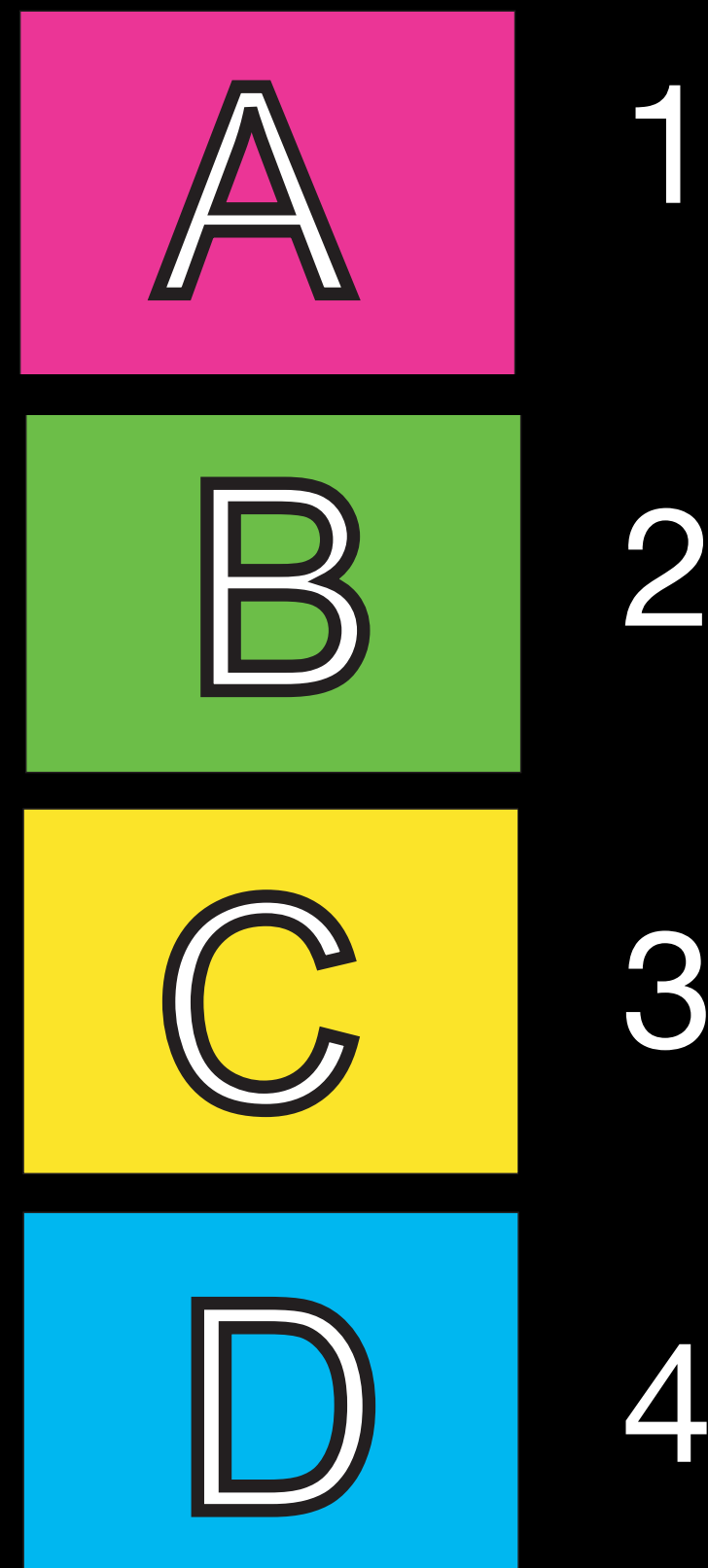
0
1
4
9

So far, our programs just run & stop...
How do programs with a user interface work?

Clicker question #2.3

- What does this program print?

```
j = 1
while j < 3:
    j = j + 1
print(j)
```



Clicker question #2.4

- What does this program print?

```
product = 1
j = 1
while j < 3:
    product = product * j
    j = j + 1
print(product)
```

A

1

B

2

C

6

D

24

Clicker question #2.4b

- What does this program print?

```
product = 1
j = 1
while j < 4:
    product = product * j
    j = j + 1
print(product)
```

A

1

B

2

C

6

D

24

Clicker question #2.4c

- What value of x makes the program print 24?

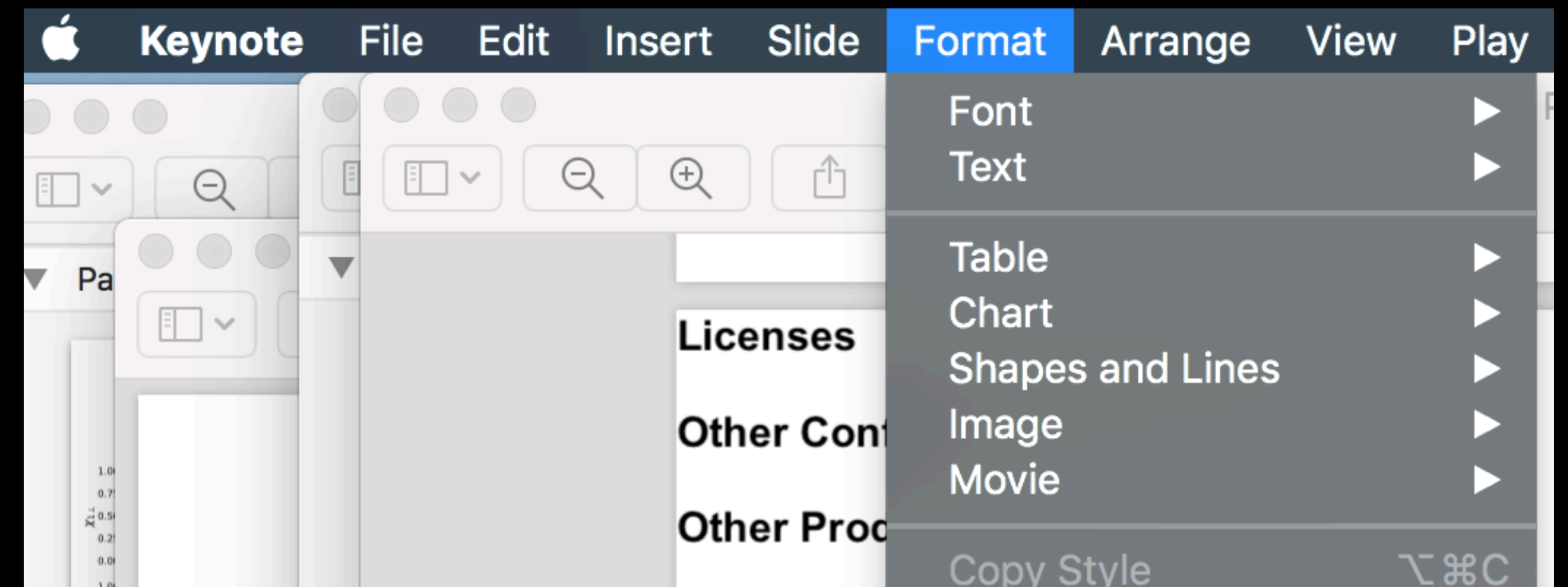
```
product = 1
j = 1
while j < x:
    product = product * j
    j = j + 1
print(product)
```

A	3
B	4
C	5
D	6

Loops

- Real life:
event loop
- Event = key press,
mouse/trackpad
click,
...

```
while message != quit:  
    message = get_next_message()  
    process_message(message)
```



My first program

- Basic, 1987
- Python equivalent

```
10 PRINT "GEOFFREY"  
20 GOTO 10
```

```
done = False  
while not done:  
    print("Geoffrey")
```

```
while true:  
    print("Geoffrey")
```