CS 5001

LECTURE 9

CLASSES, FUNCTIONS AS OBJECTS

KEITH BAGLEY

FALL 2023

Northeastern University
Khoury College of Computer Sciences

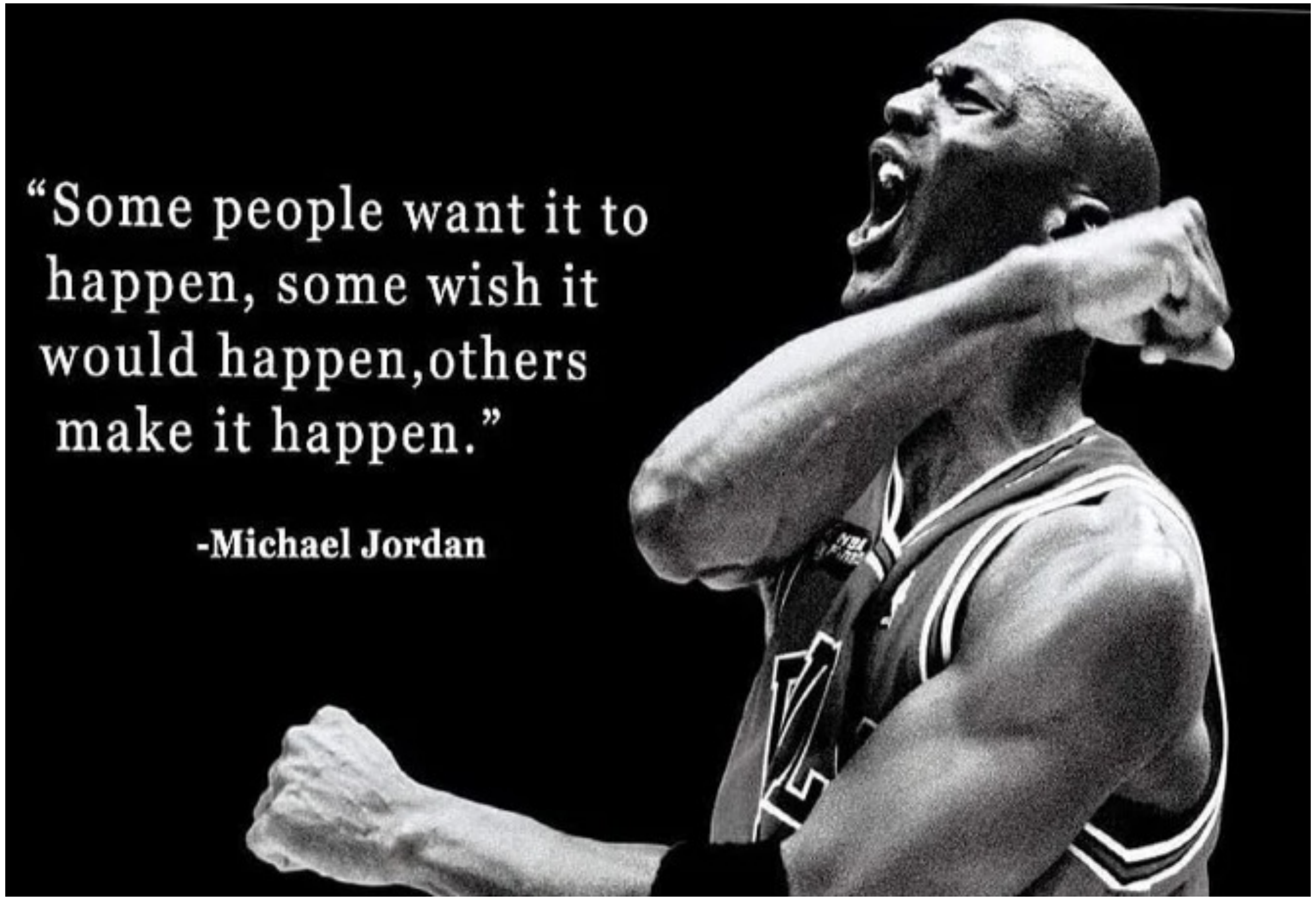
440 Huntington Avenue ▪ 202 West Village H ▪ Boston, MA 02115 ▪ **T** 617.373.2462 ▪ **khoury.northeastern.edu**

AGENDA

- Course Check In
- Review of Classes & Objects
- More Abstract Data Types – Queues, Etc.
- Protocols
- Functions are Objects too (in some languages)
- Lambda & Nested Functions (review)
- Faces of CS
- Launch into Lab
- Q & A

CHECK-IN: HOW ARE YOU DOING?

QUOTE OF THE WEEK


"Some people want it to happen, some wish it would happen,others make it happen."
-Michael Jordan

REVIEW: CLASSES & OBJECTS

- Classes are user-defined types
  - Creating a class is like inventing your own type!

- Classes combine data and functions into one “package”

- Almost everything in Python is some type of object
  - floats and ints and booleans
  - lists, tuples, dictionaries
  - strings

REVIEW: CLASSES & OBJECTS

- Objects are the “thingy”
- Classes are the blueprint for how to make the “thingy”
- A class is like a mold:
  - Design the mold once
    make many models of many colors
  - Objects are **mutable** so you can change their values

REVIEW: MINION EXAMPLE

```
class Minion:
  def __init__(self, name):
      self.name = name
      self.color = "yellow"
  def yell(self, word):
      print(word, "!!!!")
  def get_name(self):
      return self.name
```

Init is a "Constructor". It creates instances of the class
In this example, the name MUST be provided by the user of this class. The color CANNOT be specified

name and color are "attributes" or "instance variables"

We use "getters and setters" to access the attribute values in OOP. Python has "public" accessibility to everything, but we'll follow the standard OO way of well-defined interfaces

REVIEW: ATTRIBUTES

- Besides methods (aka functions) objects keep track of their own data
- This data is called “attributes” or “instance variables”

Open-Minded
I listen to other people and respect their ideas.

Principled
I tell the truth and I'm fair. I share and play well with others.

Reflective
I think about my own work. I know when I've done well, and when I could have done better.

Risk-Taker
I have confidence to try new things. I stand up for the things I believe in.

Thinker
I think carefully and show initiative. I make good decisions and I'm a problem solver.

Caring
I care about people, plants, animals and the Earth. I help others and I look after the environment.

Communicator
I can share my ideas with others in many ways, and in more than one language. I listen to others.

Inquirer
I am curious and enjoy learning. I try to find out new things.

Knowledgeable
I know about many things and I remember what I have learnt.

Well-Balanced
I look after my mind and my body I try to stay healthy and happy

Attributes can be used to describe something. Think “characteristics”

<u>**Note**</u>: In other languages, attributes may be designated private or public. Everything in Python is public by default, so it is up to the development teams to maintain good encapsulation & information hiding

REVIEW: DATA STRUCTURES: STACKS

- Stacks are an Abstract Data Type (ADT)
  - **Types** are structures defined by their behavior from a user's viewpoint
- Stacks manage data elements in **L**ast **I**n **F**irst **O**ut (LIFO) order
  - The most recent item added to the stack is the first item removed
  - HOW we choose to do that is up to us!

LET’S USE OUR STACK!

Remember our Binary to Decimal Converter from Lecture 5?

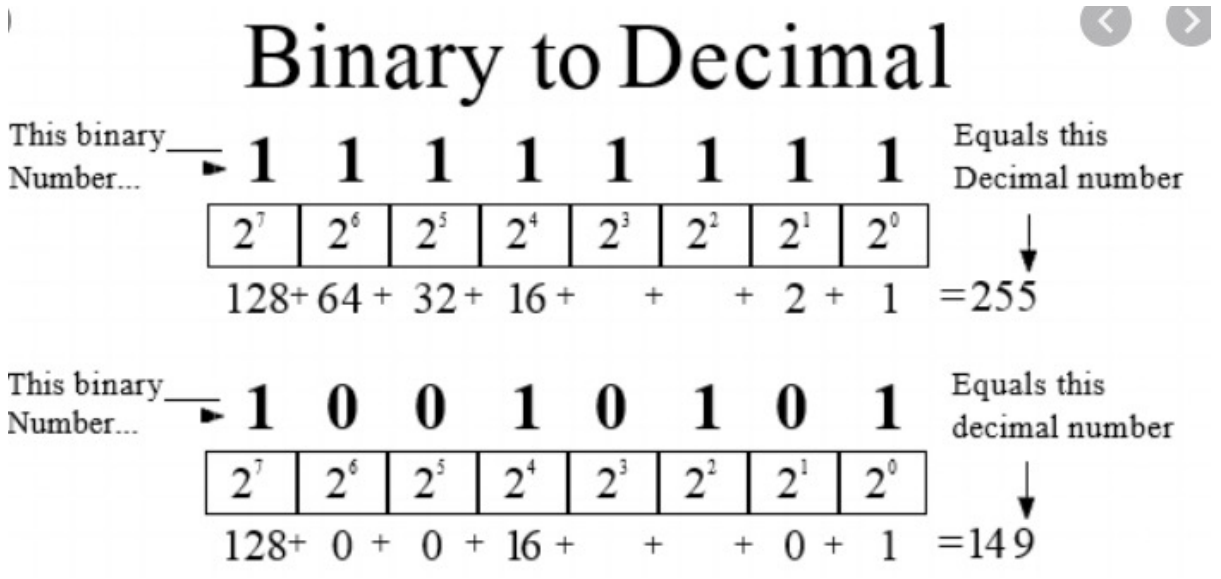
*This week, let’s write a function that **uses a Stack** to do what we did before: Take a binary string of 1’s and 0’s as **input** and **returns** the decimal equivalent*

‘001’ -> 1

‘100’ -> 4

‘101’ -> 5

Etc.



STACK IS LIFO – IS THERE FIFO?

- Yes there is!
- **F**irst **I**n **F**irst **O**ut

- Can you think of an example of FIFO?

QUEUE IS FIFO

- Queues use FIFO ordering
  - Items are removed in the same order that they were added

QUEUES

- Queue operations
  - enqueue() : Put an element at the back of the queue
  - dequeue(): Get the element that is at the front of the queue
  - front(): Look at the element at the front
  - back(): Look at the element at the back/last inserted
  - create(): Create the queue destroy(): Destroy the queue
  - size(): Get the number of elements in the queue

OUR LIST CAN HELP WITH QUEUES TOO!

- Let’s build our Queue in Python now!

```python
"""
    Queue
"""


class Queue:
    """ Queue class - First In First Out """

```

WHAT ABOUT A DEQUE (PRONOUNCED “DECK”)?

- A deque has the power of Stacks and Queues

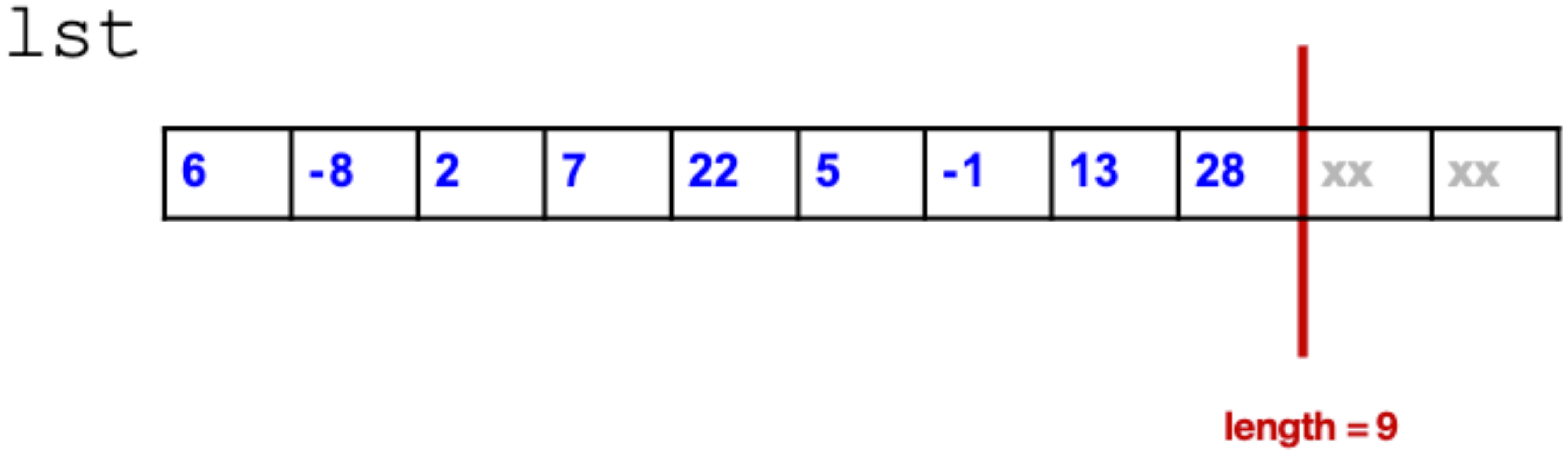
push_front()	Inserts the element at the beginning.
push_back()	Adds element at the end.
pop_front()	Removes the first element from the deque.
pop_back()	Removes the last element from the deque.
front()	Gets the front element from the deque.
back()	Gets the last element from the deque.
empty()	Checks whether the deque is empty or not.
size()	Determines the number of elements in the deque.

THOUGHT QUESTION

- We might consider using a list for implementation again but…
  - Quality of service from an algorithmic complexity level would not be met
    - We'll do algorithmic analysis next week, but remember "superfast access" for Queue and Stack operations?
    - What does push_front() give us if we use a list?

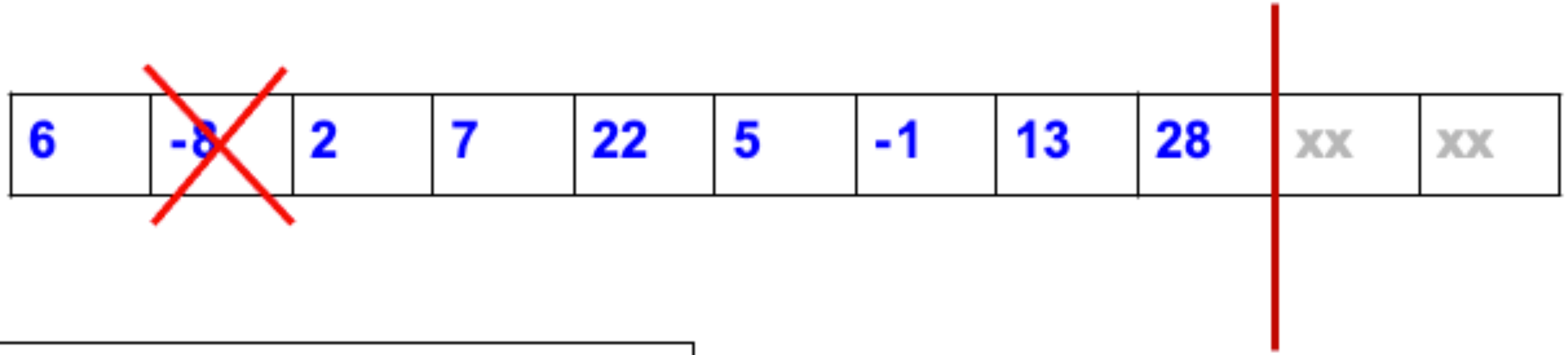
LISTS & THEIR DATA

- Put this example in your back pocket: we'll revisit next week for our algorithmic analysis discussion



LISTS

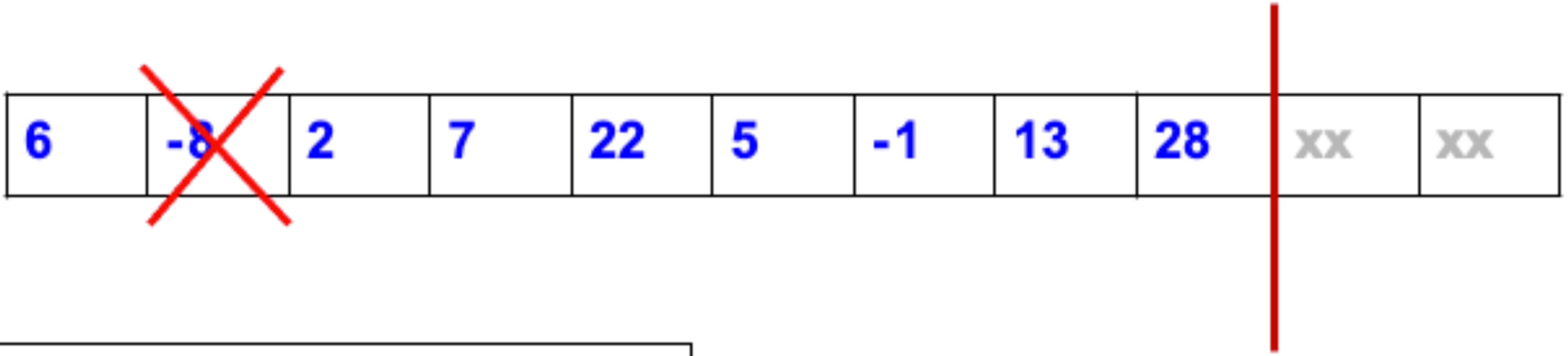
- What looks like ONE step to remove an element:



```
lst.remove(-8)
```

LISTS

- What looks like ONE step – **Is actually 8**
  - We'll "look underneath the hood" to see what's going on

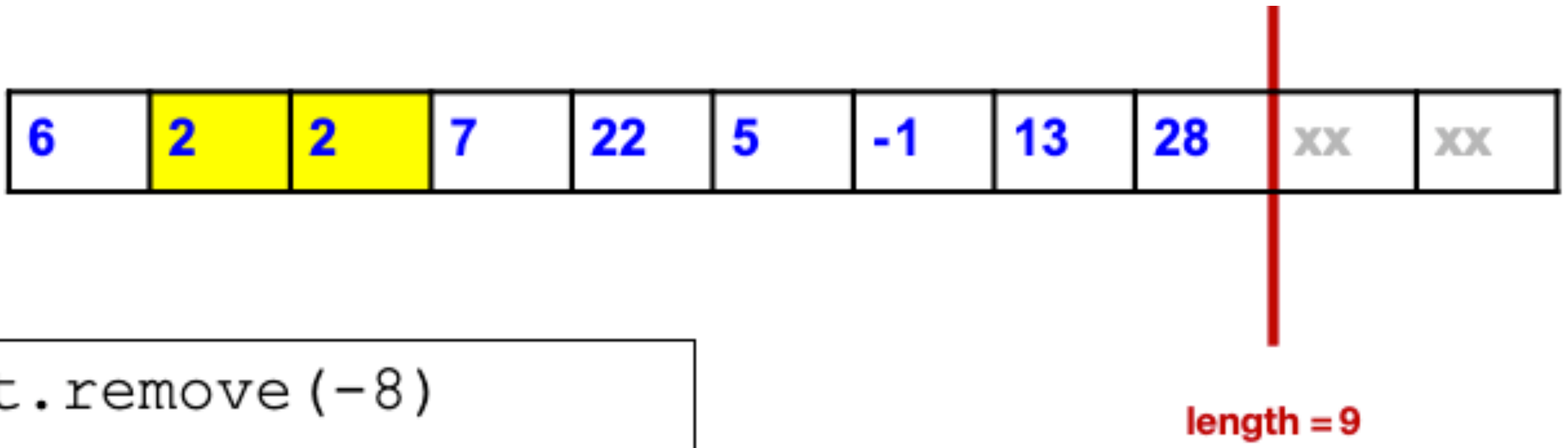


```
lst.remove(-8)
```

length = 9

LISTS

- The “beauty” of Python is that this remains hidden from us…



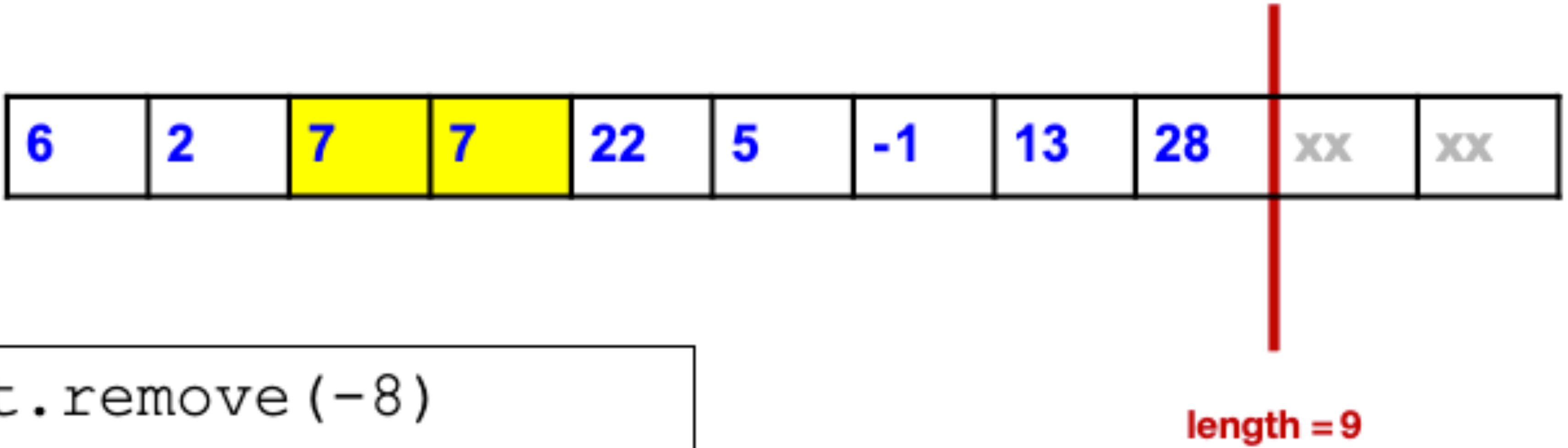
```
lst.remove(-8)
```

Step 1

*Yellow blocks indicate us “shifting” data elements into their new indices since Lists “grow” and “shrink” as needed

LISTS

- Remember that Lists resize as needed. But we CANNOT have gaps in data. Must be contiguous sequences. Hence the shifting here

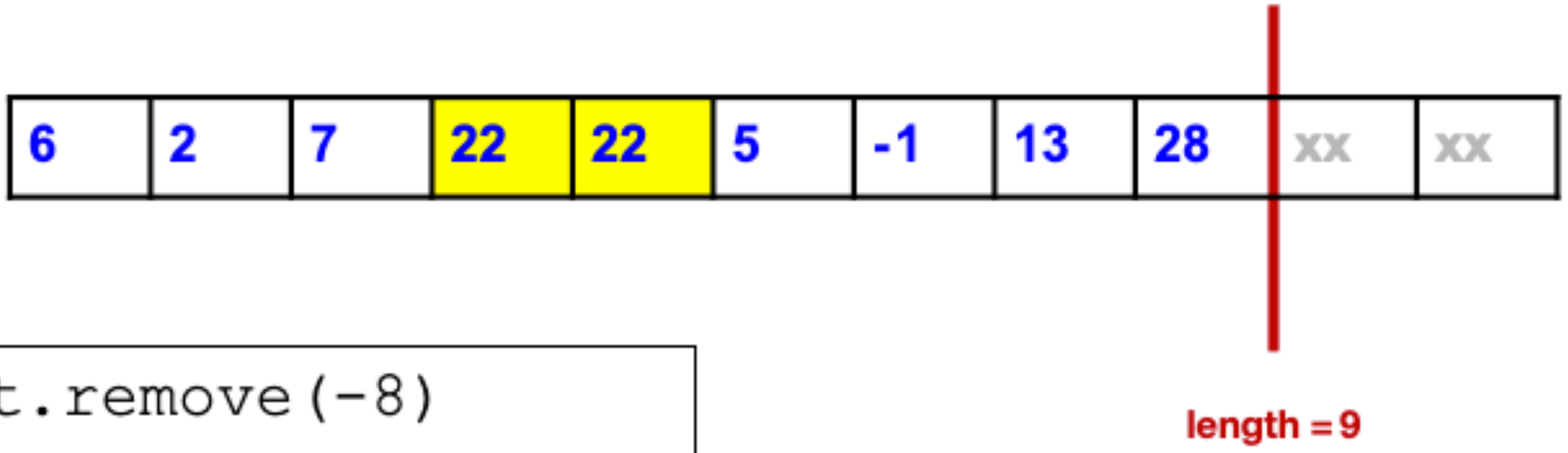


```
lst.remove(-8)
```

```
Step 1
Step 2
```

LISTS

- Python doesn't get "bored" (like you probably are in following this flow) but the work still needs to be done

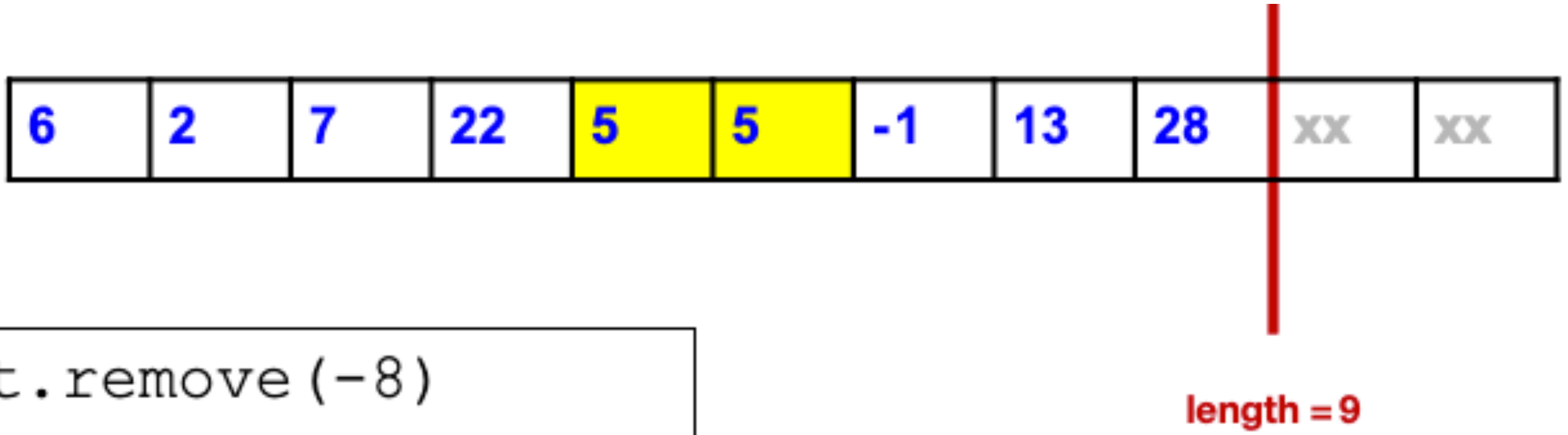


```
lst.remove(-8)
```

```
Step 1
Step 2
Step 3
```

LISTS

- Are we there yet? Nope

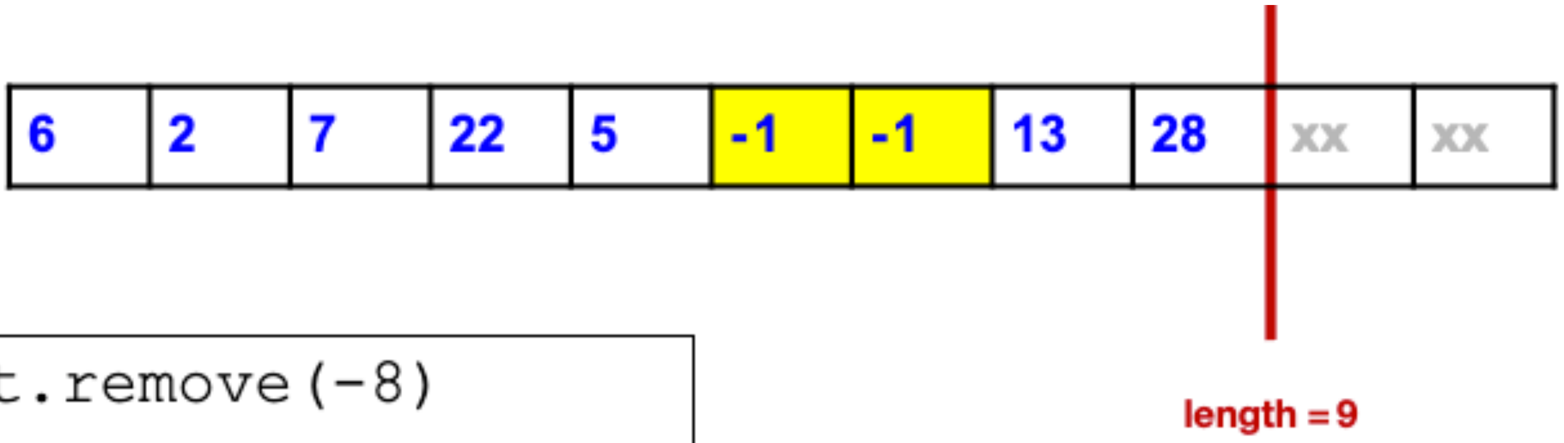


```
lst.remove(-8)
```

```
Step 1
Step 2
Step 3
Step 4
```

LISTS

- Some languages make you handle this type of resizing manually. Python doesn't

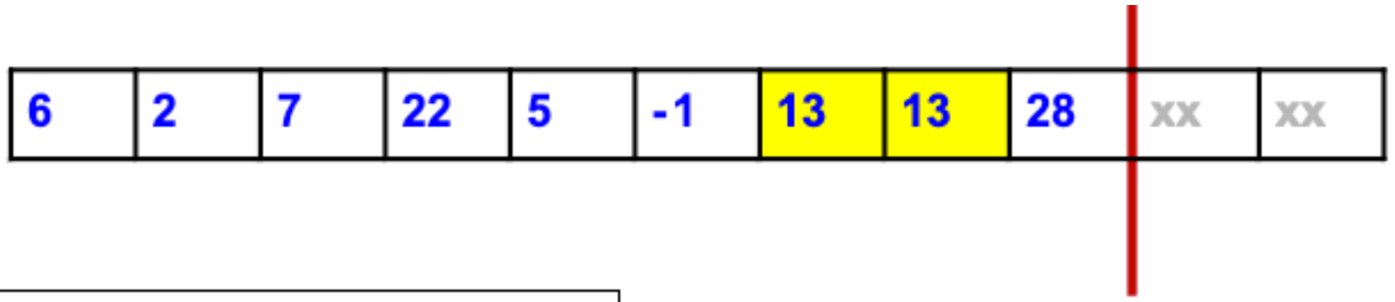


```
lst.remove(-8)
```

```
Step 1
Step 2
Step 3
Step 4
Step 5
```

LISTS

- This is a relatively small list, so imagine more data…

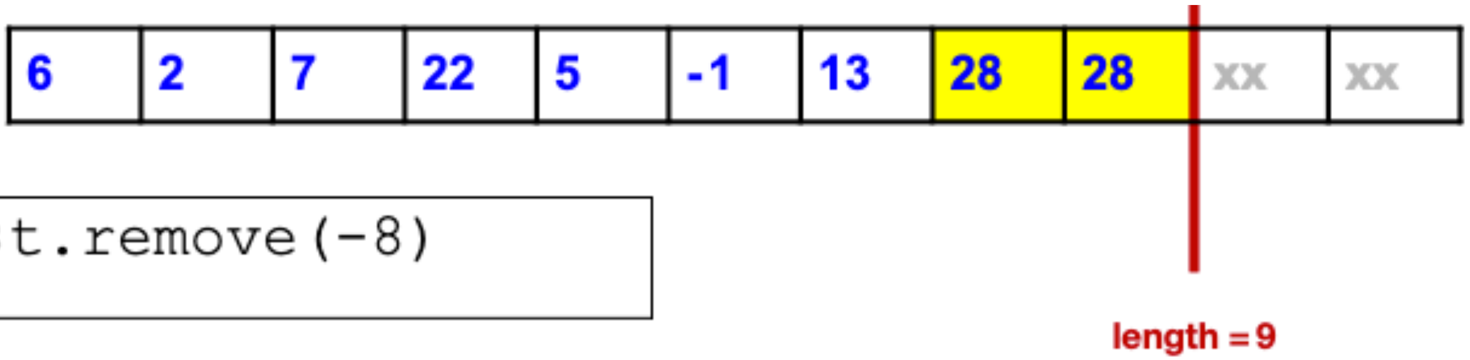


```
lst.remove(-8)
```

```
Step 1
Step 2
Step 3
Step 4
Step 5
Step 6
```

LISTS

- Almost done…

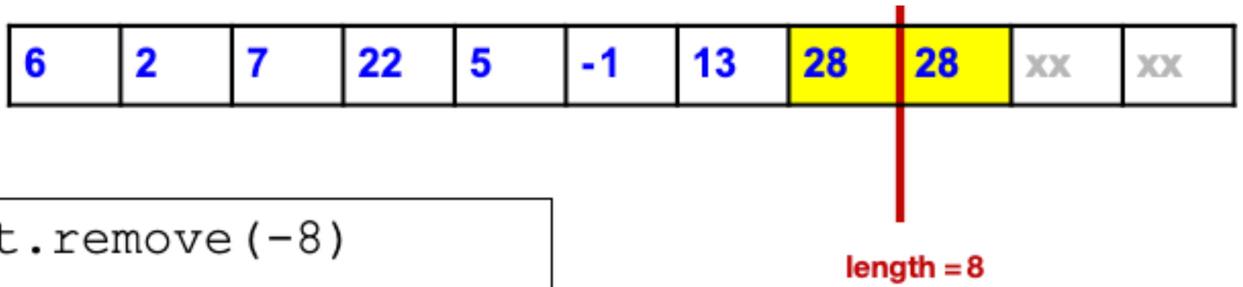


```
lst.remove(-8)
```

Step 1
Step 2
Step 3
Step 4
Step 5
Step 6
Step 7

LISTS

- 8 Steps for a VERY short list. Caveat: *Almost worst case*



```
lst.remove(-8)
```

```
Step 1
Step 2
Step 3
Step 4
Step 5
Step 6
Step 7
Step 8
```

But, what if we were working with “Big Data”?
FB, Instagram, Boeing?

THOUGHT QUESTION - REVISITED

- We need to consider a different approach for a Deque than simply using a List, unless we're explicitly NOT providing O(1) for one of the "push" operations

- We won't write code, but what are your thoughts on how to solve this?

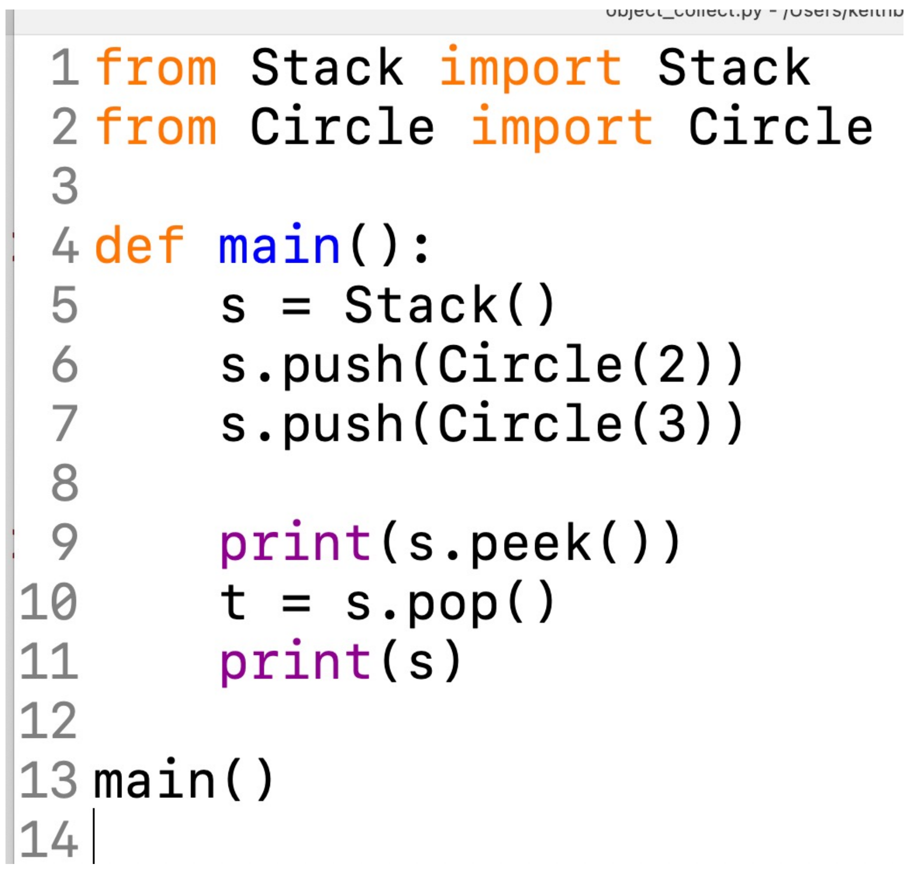
OTHER DATA STRUCTURES

- Trees
  - A variety of tree types here
- Graphs
- We won’t cover these in CS5001, but you’ll be introduced to them in other courses

COLLECTIONS OF OBJECTS

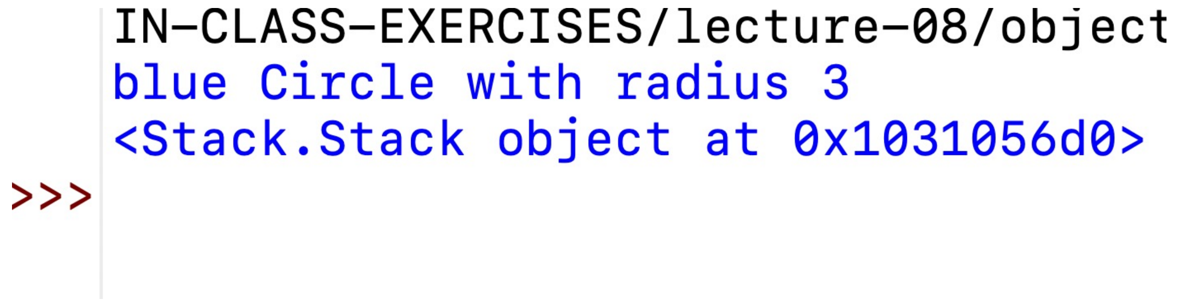
- Thus far, we’ve used objects on a singular level
  - E.g. a single Circle. An Employee, etc.
- We can hold collections of objects as well
  - Plus: collections are objects themselves, so you already know that objects can have collections
- Often, it’s useful to have a collection of objects
  - A list of Circles.
  - A dictionary of SimpleFraction
- What about a Stack of Squares?
  - Sure! Why not? ADTs can be used to collect objects

COLLECTIONS OF OBJECTS



object_collect.py - /Users/keithb

```python
from Stack import Stack
from Circle import Circle

def main():
    s = Stack()
    s.push(Circle(2))
    s.push(Circle(3))

    print(s.peek())
    t = s.pop()
    print(s)

main()
```



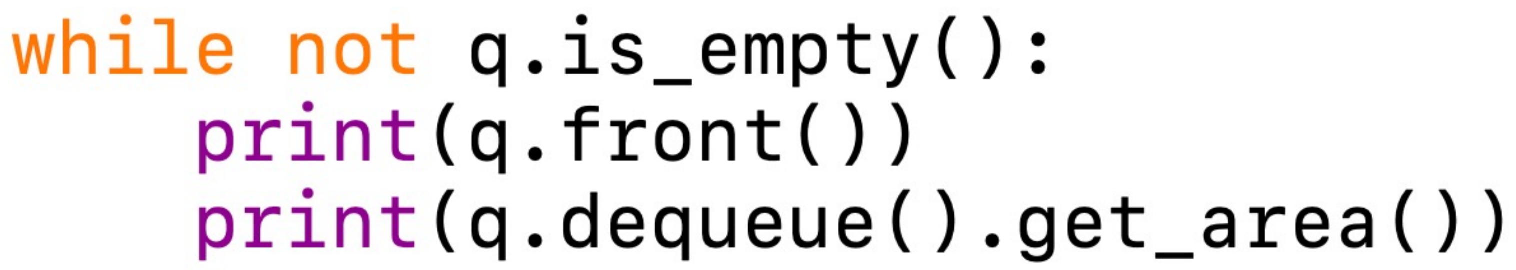
```
IN-CLASS-EXERCISES/lecture-08/object
blue Circle with radius 3
<Stack.Stack object at 0x1031056d0>
>>>
```

PROTOCOLS (INTERFACES IN SOME LANGUAGES)

- A **Protocol** is a related set of methods that defines the explicit communication to which that object can respond.
  - Also known as “interfaces” in language like Java and C#
- Methods in a protocol are generally related, and give some sense of significant behavior for a class
  - E.g. Employees can calculate their pay() and accept_raise()
- Python is dynamically typed, so the concept of protocols allows for a high degree of flexibility, regardless of the class

PROTOCOLS & DYNAMIC TYPING

- An example with Python
  - How does it KNOW which KIND of object?
  - Runtime POLYMORPHISM (dynamic binding) and (with Python) dynamic typing

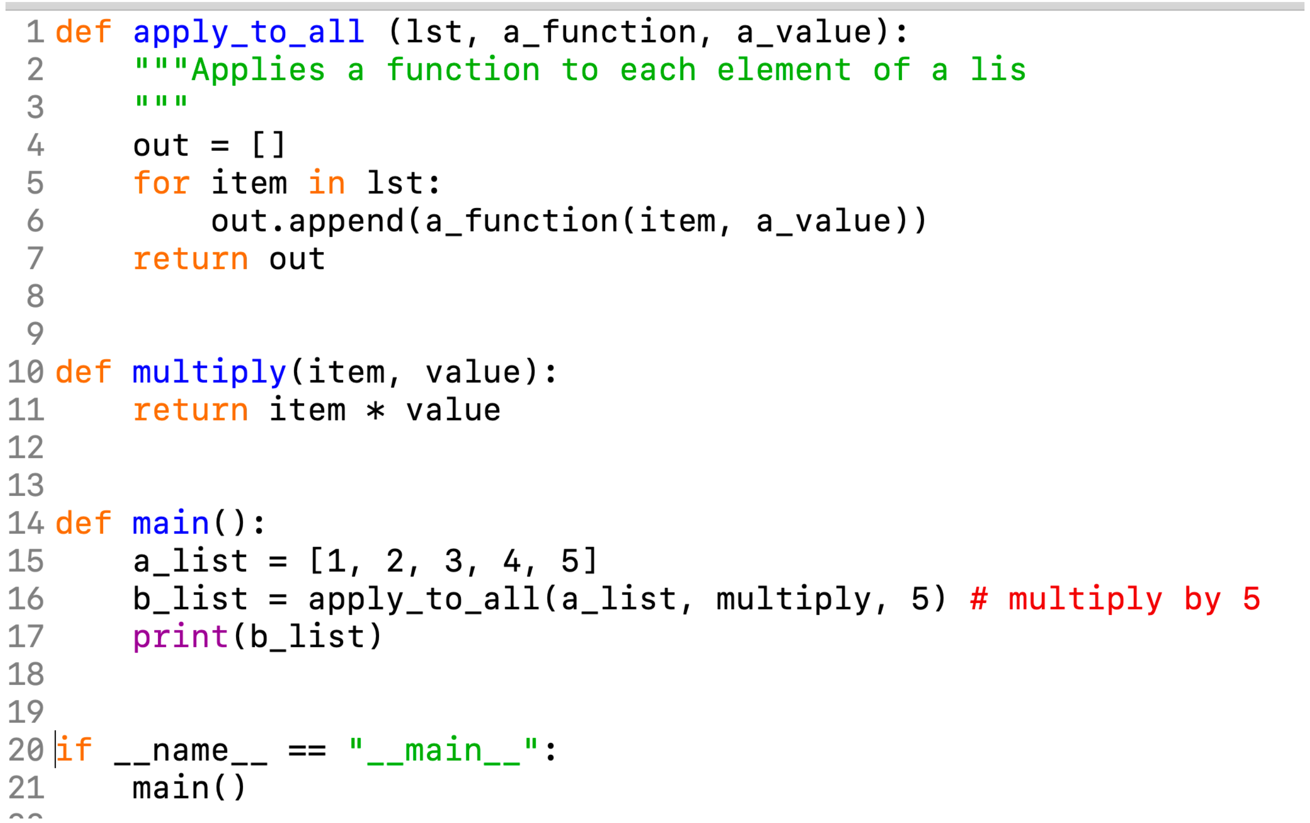


```
while not q.is_empty():
    print(q.front())
    print(q.dequeue().get_area())
```

Let’s write the code for this with Squares and Circles!

FUNCTIONS ARE OBJECTS TOO

- Functions are objects too, and can be passed to other functions



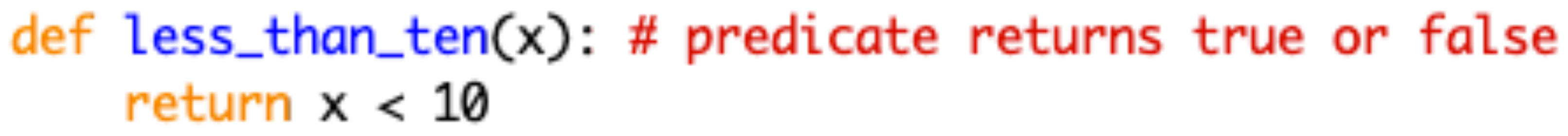
```python
def apply_to_all (lst, a_function, a_value):
    """Applies a function to each element of a lis
    """
    out = []
    for item in lst:
        out.append(a_function(item, a_value))
    return out


def multiply(item, value):
    return item * value


def main():
    a_list = [1, 2, 3, 4, 5]
    b_list = apply_to_all(a_list, multiply, 5) # multiply by 5
    print(b_list)


if __name__ == "__main__":
    main()
```

Function that takes function as parameter

Function we'll pass

QUICK NOTES: PREDICATES & LAMBDA

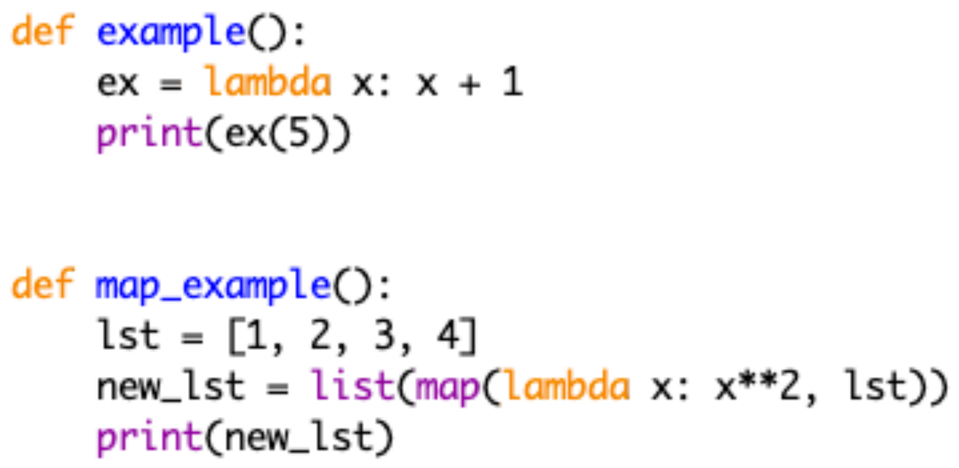
- **Predicates** are functions (or methods) that take a single argument & return either True or False
  - They are used to compose more complex and “higher order” function tests
  - *Predicate* in general meaning is a statement about something that is either true or false.



```
def less_than_ten(x): # predicate returns true or false
    return x < 10
```

LAMBDA

- **Lambda** are anonymous (unnamed) functions that can be disposed of after use
  - Create them in another function
  - Possibly pass them around as argument
  - Use them in one spot, then forget them

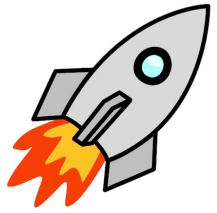


```python
def example():
    ex = lambda x: x + 1
    print(ex(5))


def map_example():
    lst = [1, 2, 3, 4]
    new_lst = list(map(lambda x: x**2, lst))
    print(new_lst)
```

LET'S WORK ON SOME EXAMPLES

- Functions as Objects

LAUNCH INTO LAB



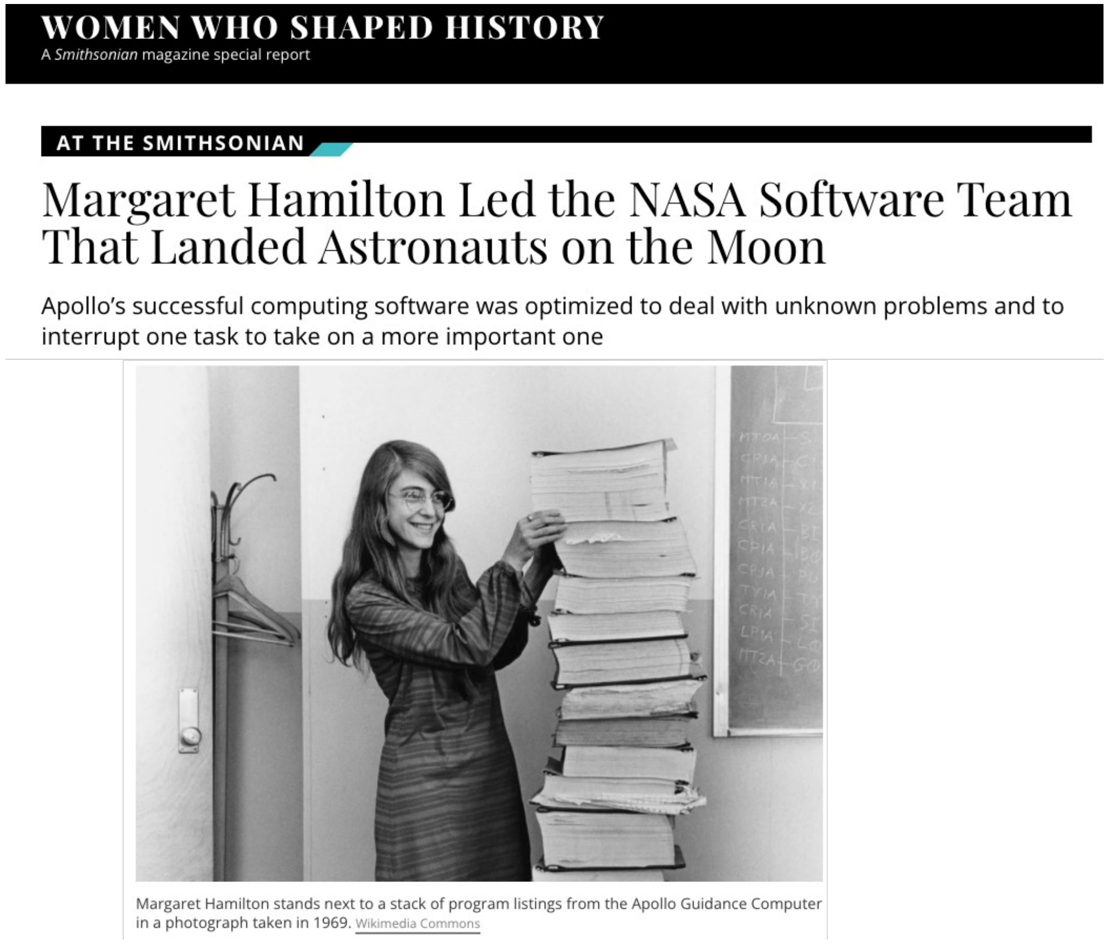
OPTIONAL LAB this week.
No formal requirement – Enjoy the Holiday Weekend

Also: NO PLA this week! ☺

FACES OF C.S.: MARGARET HAMILTON

Margaret Hamilton is an American computer scientist, systems engineer, and business owner. She was director of the Software Engineering Division of the MIT Instrumentation Laboratory, which developed on-board flight software for NASA's Apollo program.

She is one of the people credited with coining the term "software engineering”. She received the Presidential Medal of Freedom from President Barack Obama for her work leading to the development of on-board flight software for NASA's Apollo Moon missions.



WOMEN WHO SHAPED HISTORY
A *Smithsonian* magazine special report

AT THE SMITHSONIAN

Margaret Hamilton Led the NASA Software Team That Landed Astronauts on the Moon

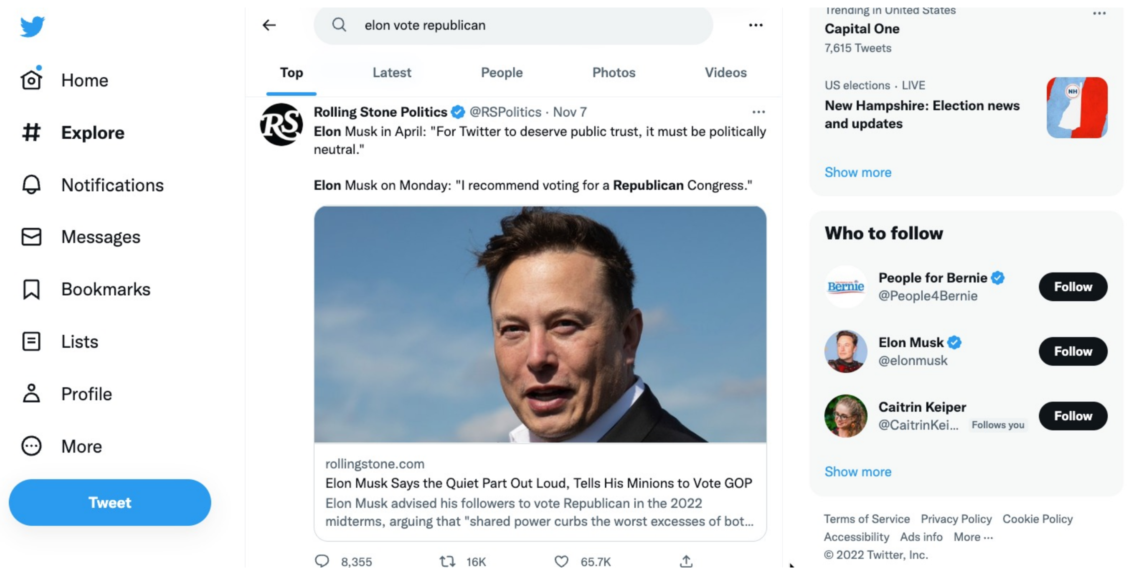
Apollo’s successful computing software was optimized to deal with unknown problems and to interrupt one task to take on a more important one

Margaret Hamilton stands next to a stack of program listings from the Apollo Guidance Computer in a photograph taken in 1969. Wikimedia Commons

https://en.wikipedia.org/wiki/Margaret_Hamilton_(software_engineer)

PROVOKING THOUGHTS FOR THE WEEK?

To the extent that Social Media can sway opinion, policy, votes, etc… Should there be greater regulation and a “firewall” of sorts, especially with ownership conflict-of interest?



Rolling Stone Politics @RSPolitics · Nov 7
Elon Musk in April: "For Twitter to deserve public trust, it must be politically neutral."

Elon Musk on Monday: "I recommend voting for a **Republican** Congress."

rollingstone.com
Elon Musk Says the Quiet Part Out Loud, Tells His Minions to Vote GOP
Elon Musk advised his followers to vote Republican in the 2022 midterms, arguing that "shared power curbs the worst excesses of bot…

8,355 16K 65.7K

Trending in United States
Capital One
7,615 Tweets

US elections · LIVE
New Hampshire: Election news and updates

Show more

Who to follow
People for Bernie @People4Bernie Follow
Elon Musk @elonmusk Follow
Caitrin Keiper @CaitrinKei... Follows you Follow
Show more

Terms of Service Privacy Policy Cookie Policy Accessibility Ads info More ···
© 2022 Twitter, Inc.

Q & A

- Questions?

THANKS!

- Stay safe, be encouraged, & see you next week!