

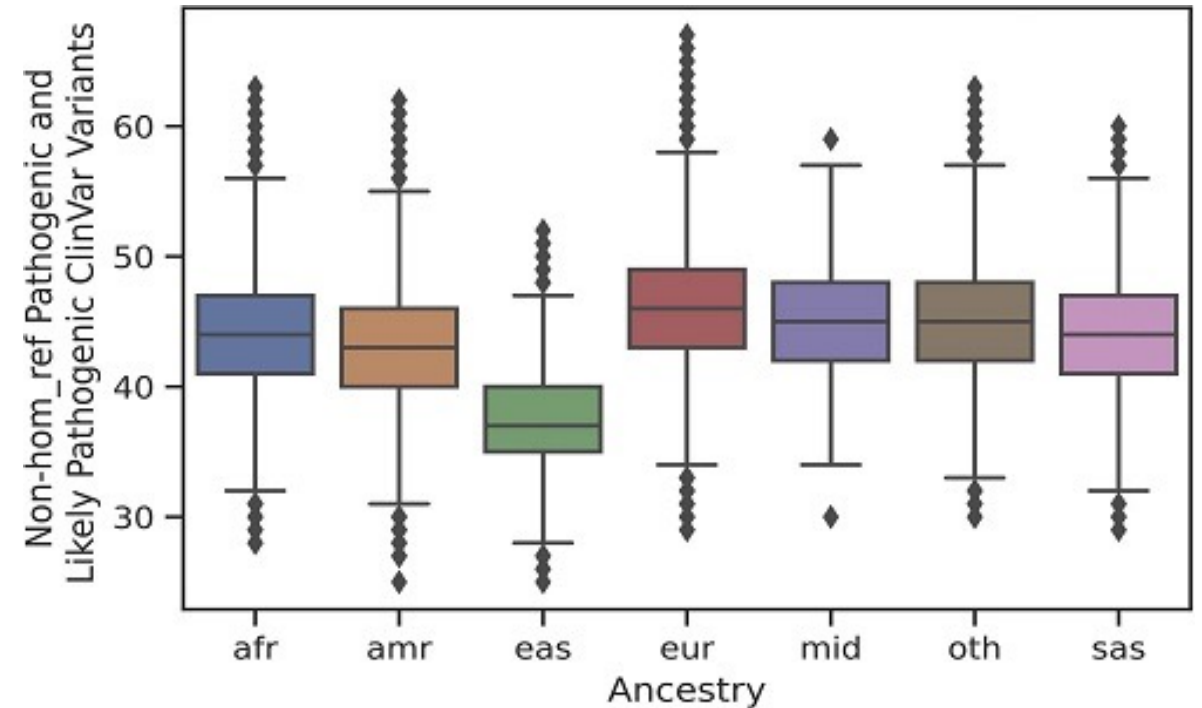
Announcements

- Maps project out!
- Remember to test locally!
- `python3 ok -u ... # to unlock tests`
- `python3 ok -i ... # open interactive Python when tests fail.`
- Use `print("DEBUG: ", ...)` or `print(f"DEBUG: {...} ")`
 - These will not cause the autograder to fail.
- Today:
 - Wrap up 1 mutable data example, then ADTs

Human Diversity Survey Yields Undiscovered Genetic Variants

[Nature; Max Kozlov \(February 19, 2024\)](#)

Analyses of up to 245,000 genomes from diverse populations in the U.S gathered by the U.S. National Institutes of Health's All of Us program uncovered more than 275 million new genetic markers. The data will help produce more accurate "polygenic risk scores," which rate a person's risk of developing a disease as a result of their genetics. To calculate such a score, researchers develop an algorithm trained on the genomic data of people who either do or do not have a particular disease; an individual's score can then be calculated by feeding their genetic data into the algorithm.



Computational Structures in Data Science

Mutable Functions



Learning Objectives

- Remember: Each function gets its own new frame
- Inner functions can access data in the parent environment
- Use an inner function along with a mutable data type to capture changes

Making Functions that Capture and change state

- We want to make a function, which returns a function that can change the state.
- [Python Tutor Link](#)

```
def make_counter():  
    counter = [0]  
    def count_up():  
        counter[0] += 1  
        return counter  
    return count_up
```

```
c = make_counter()  
print(c)  
c()  
c()  
c()
```

Functions with Changing State

- Goal: Use a function to repeatedly withdraw from a bank account that starts with \$100.

- Build our account: `withdraw = make_withdraw_account(100)`

- First call to the function:

```
withdraw(25)          # 75
```

- Second call to the function:

```
withdraw(25)          # 50
```

- Third call to the function:

```
withdraw(60)          # 'Insufficient funds'
```

How Do We Implement Bank Accounts?

- A mutable value in the parent frame can maintain the local state for a function.
- [View in PythonTutor](#)

```
def make_withdraw_account(initial):  
    balance = [initial]  
  
    def withdraw(amount):  
        if balance[0] - amount < 0:  
            return 'Insufficient funds'  
        balance[0] -= amount  
        return balance[0]  
    return withdraw
```

Implementing Bank Accounts

- A mutable value in the parent frame can maintain the local state for a function.

```
def make_withdraw_account(initial):  
    balance = [initial]  
  
    def withdraw(amount):  
        if balance[0] - amount < 0:  
            return 'Insufficient funds'  
        balance[0] -= amount  
        return balance[0]  
    return withdraw
```

[View in PythonTutor](#)

Computational Structures in Data Science

Abstract Data Types



Abstract Data Type

- Uses pure functions to encapsulate some logic as part of a program.
- We rely on built-in types (int, str, list, etc) to build ADTs
- This is a contrast to object-oriented programming
 - Which is coming soon!

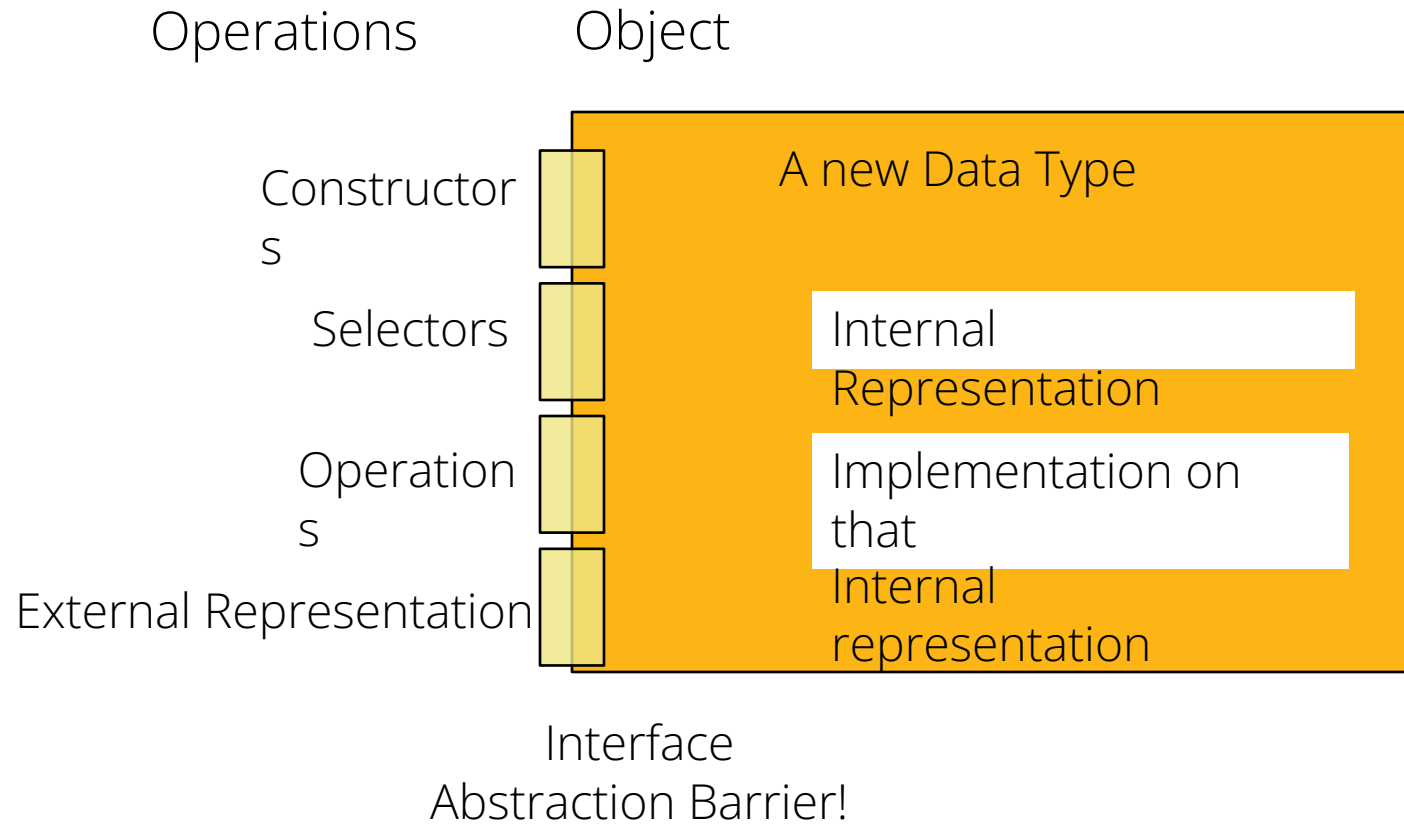
Creating Abstractions

- Compound values combine other values together
 - date: a year, a month, and a day
 - geographic position: latitude and longitude
 - a game board
- Data abstraction lets us manipulate compound values as units
- Isolate two parts of any program that uses data:
 - How data are represented (as parts)
 - How data are manipulated (as units)
- **Data abstraction:** A methodology by which functions enforce an abstraction barrier between *representation* and *use*

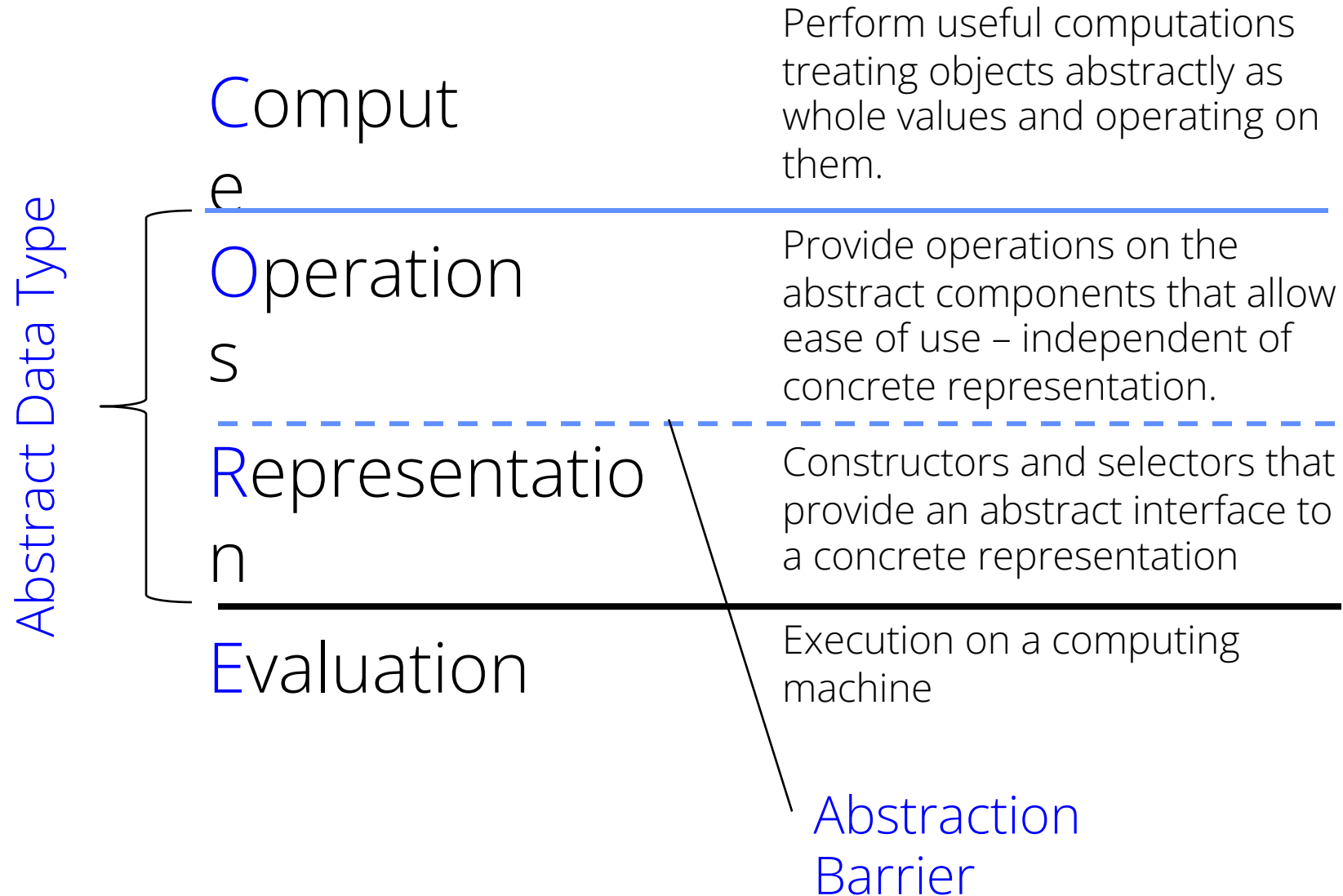
Why Abstract Data Types?

- How do you represent the *idea* of a game board, a "course", a person, a student, anything?
 - Programming languages allow you to do just about anything!
- "Self-Documenting"
 - `contact_name(contact)`
 - vs `contact[0]`
 - "0" may seem clear now, but what about in a week? 3 months?
- Change your implementation
 - Maybe today it's just a Python List
 - Tomorrow: It could be a file on your computer; a database in web

Abstract Data Type



C.O.R.E concepts



Reminder: Lists

- Lists
 - Constructors:
 - `list(...)`
 - `[<expr>, ...]`
 - `[<exp> for <var> in <list> [if <exp>]]`
 - Selectors: `<list> [<index or slice>]`
 - Operations: `in`, `not in`, `+`, `*`, `len`, `min`, `max`
 - Mutable ones too (but not yet)
- Tuples
 - A lot like lists, but you cannot edit them. We'll revisit on Monday.

A Small ADT

```
def point(x, y): # constructor
    return [x, y]
```

```
x = lambda point: point[0] # selector
y = lambda point: point[1]
```

```
def distance(p1, p2): # Operator
    return ((x(p2) - x(p1))**2 + (y(p2) -
y(p1))**2) ** 0.5
```

```
origin = point(0, 0)
my_house = point(5, 5)
campus = point(25, 25)
distance_to_campus = distance(my_house, campus)
```


Creating an Abstract Data Type

- Constructors & Selectors
- Operations
 - Express the behavior of objects, invariants, etc
 - Implemented (abstractly) in terms of Constructors and Selectors for the object
- Representation
 - Implement the structure of the object

Defining The Abstraction Barrier

- An abstraction barrier violation occurs when a part of the program that can use the "higher level" functions uses "lower level" ones instead
 - At either layer of abstraction
 - e.g. Should your function be aware of the implementation?
 - Be consistent!
- Abstraction barriers make programs easier to get right, maintain, and modify
 - Fewer changes when representation changes

A Layered Design Process – Button Up

- Start with "What do you want to do?"
- Build the application based entirely on the ADT interface
 - Focus first on Operations, then Constructors and Selectors
 - Do not implement them! Your program won't work.
 - You want to capture the "user's" point of view
- Build the operations in ADT on Constructors and Selectors
 - Not the implementation representation
 - This is the end of the abstraction barrier.
- Build the constructors and selectors on some concrete representation

Example: Tic Tac Toe and Phone Book

- See the companion notebook.
- Download the file "ipynb"
 - Go to datahub.berkeley.edu
 - Log in, then select "Upload"

Question: The Abstraction Barrier

Which of these *violates* a board ADT?

- A) `diag_left = diagonal(board, 0)`
- B) `board[0][2] = 'x'`
- C) `all_rows = rows(board)`
- D) `board = empty_board()`
- E) None of the above