THOUGHTS ON THE MIDTERM?

Using thumbs up/thumbs down...

What did everyone think of the midterm?

---

TODAY

- Review: Binary Search Trees
- Demo: Project 2
- Object-Oriented Programming
  - Defining our own data types!
  - Data with functions!
  - State!

---

REVIEW: BINARY TREES

Trees where each node has at most two children are known as binary trees.

---

REVIEW: BINARY SEARCH TREES

Binary search trees are binary trees where all of the items to the left of a node are smaller, and the items to the right are larger.
REVIEW: BST ADT
empty_bst = None
def make_bst(datum, left=empty_bst, right=empty_bst):
    return (left, datum, right)
def bst_datum(b):
    return b[1]
def bst_left(b):
    return b[0]
def bst_right(b):
    return b[2]

REVIEW: SEARCHING WITH BSTs
def bst_find(b, item):
    if b == empty_bst:
        return False
    elif bst_datum(b) == item:
        return True
    elif bst_datum(b) > item:
        return bst_find(bst_left(b), item)
    return bst_find(bst_right(b), item)

ANNOUNCEMENTS
• Homework 6 is due today, July 10.
  — Starting with the next homework, we will mark questions as core and reinforcement.
  — The core questions are ones that we suggest you work on to understand the idea better.
  — The reinforcement questions are extra problems that you can practice with, but are not as critical.
• Homework 7 is due Saturday, July 14.
  — It will be released this afternoon.
• Make sure you fill out a survey!
  — You must give us a completed survey to get back your exam once it has been graded.

PROJECT 2 DEMO
Get started if you haven’t already!

DEFINING OUR OWN DATA TYPES
So far, we’ve been defining new data types like this:
def make_account(owner, balance):
    """Makes a bank account for the given owner with balance dollars""
    return (owner, balance)
def account_balance(acct):
    """Return acct's balance""
    return acct[1]
def account_owner(acct):
    """Return acct's owner""
    return acct[0]

class Account:
    def __init__(self, owner, balance):
        self.__owner = owner
        self.__balance = balance
    def get_owner(self):
        return self.__owner
    def get_balance(self):
        return self.__balance

Define the new data type Account
Self refers to the specific instance of a class that we’re manipulating.
Methods are operations that are associated with a class or object.
The presence of an underscore before an instance variable means that the variable should only be accessed inside the class definition.

Constructor
Selectors
OBJECT-ORIENTED PROGRAMMING: CREATING OBJECTS

We say that `toms_account` is an instance of the Account class.

```python
>>> toms_account = Account("Tom", 50)
We call the process of creating an instance of a class instantiation.
```

DOT NOTATION

"Dot Notation" is used for referring to the data that is associated with a specific object.

```python
Object
  self.__balance
  toms_account.get_balance()
```

OPERATING ON OUR DATA

How would we have implemented withdraw and deposit operations before?

```python
def acct_withdraw(acct, amount):
    new_balance = acct.get_balance() - amount
    return Account(acct.get_owner(), new_balance)
def acct_deposit(acct, amount):
    new_balance = acct.get_balance() + amount
    return Account(acct.get_owner(), new_balance)
```

OPERATING ON OUR DATA

Using our operations for our accounts:

```python
>>> toms_account = Account("Tom", 50)
>>> toms_account.get_balance()
50
>>> toms_account.withdraw(25)
>>> toms_account.get_balance()
25
>>> toms_account.deposit(500)
>>> toms_account.get_balance()
525
```
OBJECT-ORIENTED PROGRAMMING
We now have a new *programming paradigm*!

Before – Functional Programming:
- Think of our program as a series of functions.
- Think in terms of inputs and outputs of each function.
- There is no change over time, no *state*.

Now – Object-Oriented Programming:
- Think of our program as a series of objects.
- Think in terms of the ways in which objects interact with each other using methods.
- Objects can change over time in our program. They have *state*.

CONCLUSION

- **Object-oriented programming**: a brand new paradigm.
- **Preview**: Classes that inherit traits from other classes and variables shared by all data of the same class.