CS 106
Introduction to Computer Science I

07 / 28 / 2026

Instructor: Michael Eckmann

Today's Topics

- Questions / comments?
- Instance variables can be added to objects after they are created
- str format
- Start idea of inheritance in classes

Objects can add instance vars

- Instance variables can be added to an object
- They exist only for that object
- e.g.

```
c = playing_cards.Card(2,0)
d = playing_cards.Card(11,2)
c.color = ‘Red’
```

New instance variables can be added within methods other than the constructor.

string’s format method

- Let’s print results nicely, formatted using string’s format method (and using round to some number of decimal places)
- Let’s look at the format method in str
  - Number of figures, decimal places etc.
  - Left-aligned or right-aligned or center-aligned
  - Show , sep for thousands, millions

Is-a vs. Has-a

- A “has a” relationship is one that corresponds to a class and its instance variables (aka data members.)
- For example:
  - A Card **has a** suit and **has a** rank
  - A Deck **has** 52 Cards
  - A Car **has a** color and **has a** make and **has a** model, ...
  - An Employee **has a** hire date and **has a** salary, ...

  - Can we think of any more?

Is-a vs. Has-a

- A “has a” relationship is one that corresponds to a class and its data members.

```
class Date:
    # year, month and day instance variables
```

Is-a vs. Has-a

- An “is a” relationship is one that corresponds to a class and the class from which it inherits. Also can be said as “is a type of”.
- For example:
  - A Faculty member **is an** Employee
  - A Staff member **is an** Employee
  - A Car **is a** Vehicle
  - A Truck **is a** Vehicle
  - A Motorcycle **is a** Vehicle

  - Can we think of any more?

Other inheritance relationships

- Can you think of anything else that has a **“has a”** relationship?
  - Rooms have doors, windows, etc.

- Can you think of anything else that has an **“is a”** relationship?
  - Bedroom is a Room
  - Dining room is a Room

  - How about Animals?

Bank account Exercise

- What data members describe a bank account?
  - That is, what does a bank account **have**.
- What types are these?

- We want to handle Checking and Savings Accounts

Bank account

- Maybe we might want the following data members:
  - Account number
  - Balance
  - Overdrawn Fee
  - ATM Withdrawal Per Day Limit
  - Bounced Check Fee
  - Interest Rate

Bank account

- So we could have a class named Bank_Account, a class named Checking_Account and a class named Savings_Account.
- All Bank_Accounts have:
  - account_number
  - balance
  - overdrawn_fee
  - ATM_withdrawal_per_day_limit

Checking_Accounts additionally have:

  - bounced_check_fee

Savings_Accounts additionally have:

  - interest_rate

Bank account

- We could create a Bank_Account class that contained the common stuff among all accounts
- We then could create a Savings_Account class that inherits all the stuff about a Bank Account from Bank_Account class and adds the things that are specific to Savings Accounts.
- We also could create a Checking_Account class that inherits all the stuff about a Bank Account from Bank_Account class and adds the things that are specific to Checking Accounts.
- Checking_Account inherits from Bank_Account
- Savings_Account inherits from Bank_Account

Inheritance

- If class A inherits from class B, then class A is a **subclass** of B and class B is a **superclass** of A. We also talk of class A as being class B's **child** and class B being class A's **parent**.
- We can draw the inheritance relationships hierarchy by having superclass(es) at the top and subclass(es) lower, using lines to connect where there's an inheritance relationship.

Inheritance

- Class A may inherit from class B which can in turn, inherit from class C. (draw on the board.)
- However, this is not considered multiple inheritance.
- Multiple inheritance is an object-oriented concept that allows a subclass to inherit from more than one class.
- e.g. Class D can inherit from both class E and F directly. (draw on the board.)

Inheritance

- A subclass inherits all the instance variables in its superclass and has access to (can call) any public methods.
- Even if there is private instance data in a superclass, the subclass inherits that data but can't refer directly to the variables.
- e.g. a Savings_Account will inherit account_number from Bank_Account, so an object of type Savings_Account will have an account_number.

Code for inheritance

```
class Bank_Account:


class Savings_Account(Bank_Account):


class Checking_Account(Bank_Account):
```

Calling the superclass's constructor

- The first think to do in the constructor of a subclass is to call the constructor of the superclass.
- Two ways to do it:

```
super(subclass_name,self).__init__(parameters for superclass)
    # or
    superclass_name.__init__(parameters for superclass)
```

e.g.

```
super(Savings_Account,self).__init__(acct_num, balance, ...)
    # or
    Bank_Account.__init__(acct_num, balance, …)
```

Lab time

- Use the remaining class time to work on the determine_hand method in PokerHand.
- Tomorrow I will write a complete example of code for a class hierarchy with inheritance so you can see how child classes can inherit from a parent class.