Objectives

- Defining our own classes

Review: Dictionaries

- What is a dictionary in Python?
- What is the syntax for creating a new dictionary?
- How do we access a key’s value from a dictionary?
    - What happens if there is no mapping for that key?
- How do we create a key → value mapping in a dictionary?
- How can we iterate through a dictionary?

ABSTRACTIONS

Abstractions

- Provide ways to think about program and its data
  - Get the jist without the details
- Examples we've seen
  - Functions and methods

    ```
    encodeFile(filename, key)
    ```

    - Used to perform some operation but we don't need to know how they're implemented
  - Dictionaries
    - Know they map keys to values
    - Don't need to know how the keys are organized/stored in the computer's memory
  - Just about everything we do in this class…

Classes and Objects

- Provide an abstraction for how to organize and reason about data
- Example: `GraphWin` class
  - Had ***attributes*** (i.e., data or state) background color, width, height, and title
  - Each `GraphWin` object had these attributes
    - Each `GraphWin` object had its own values for these attributes
  - Used methods (API) to modify the object's state, get information about attributes

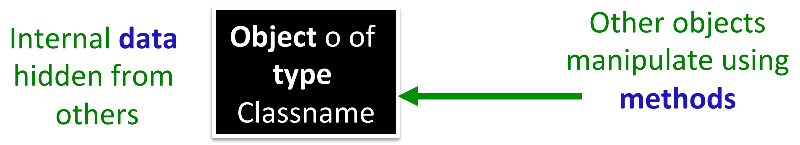
Defining Our Own Classes

- Often, we want to represent data or information that we do ***not*** have a way to represent using *built-in types* or *libraries*

- Classes provide way to *organize* and *manipulate* data
  - Organize: data structures used
    - E.g., ints, lists, dictionaries, other objects, etc.
  - Manipulate: methods

What is a Class?

- Defines a new ***data type***
- Defines the class's ***attributes*** (i.e., data or state) and ***methods***
  - ➢ Methods are like **functions *within*** a class and are the class's **API**


Internal data hidden from others
Object o of type Classname
Other objects manipulate using methods

Defining a Card Class

- Create a class that represents a playing card
  - ➢ How can we represent a playing card?
  - ➢ What information do we need to represent a playing card?

Representing a Card object

- Every card has two attributes:
  - Suite (one of “hearts”, “diamonds”, “clubs”, “spades”)
  - Rank
    - 2-10: numbered cards
    - 11: Jack
    - 12: Queen
    - 13: King
    - 14: Ace

Defining a New Class

- Syntax:

Keyword

Typically starts with a capital letter

```
class ClassName:
    <method definitions>
```

Card Class (Incomplete)

Doc String

```
class Card:
    """ A class to represent a standard playing card.
    The ranks are ints: 2-10 for numbered cards, 11=Jack,
12=Queen, 13=King, 14=Ace.
    The suits are strings: 'clubs', 'spades', 'hearts',
'diamonds'."""
    def __init__(self, rank, suit):
        """Constructor for class Card takes int rank and
             string suit."""
        self._rank = rank
        self._suit = suit

    def getRank(self):
        "Returns the card's rank."
        return self._rank

    def getSuit(self):
        "Returns the card's suit."
        return self._suit
```

Methods

card.py

Card Class (Incomplete)

Doc String

```
class Card:
    """ A class to represent a standard playing card.
    The ranks are ints: 2-10 for numbered cards, 11=Jack,
12=Queen, 13=King, 14=Ace.
    The suits are strings: 'clubs', 'spades', 'hearts',
'diamonds'."""
    def __init__(self, rank, suit):
        """Constructor for class Card takes int rank and
             string suit."""
        self._rank = rank
        self._suit = suit

    def getRank(self):
        "Returns the card's rank."
        return self._rank

    def getSuit(self):
        "Returns the card's suit."
        return self._suit
```

Methods

Methods are like *functions* defined in a *class*

card.py

Defining the Constructor

- `__init__` method is like the ***constructor***
- In constructor, define ***instance variables***
  - **Data** contained in every object
  - Also called **attributes** or **fields**

Convention: named with _

- Constructor **never *returns*** anything

First parameter of *every* method is `self`
- pointer to the object that method acts on

```
def __init__(self, rank, suit):
    """Constructor for class Card takes int rank
     and string suit."""
     self._rank = rank
     self._suit = suit
```

Instance variables

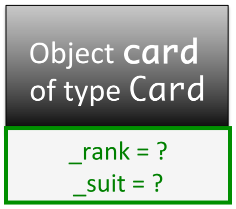
Review

- How do we use the constructor for an object?

Using the Constructor

```
def __init__(self, rank, suit):
```

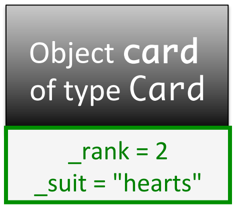
- As defined, constructor is called using `Card(<rank>,<suit>)`
  - Do not *pass* anything for the `self` parameter
  - Python handles for us
    - Passes the parameter *automatically*



Using the Constructor

```
def __init__(self,
        rank, suit):
```

- As defined, constructor is called using `Card(<rank>,<suit>)`
  - Do not *pass* anything for the `self` parameter
  - Python handles, passing the parameter for us automatically
- Example:
  - `card = Card(2, "hearts")`
  - Creates a 2 of Hearts card
  - Python passes `card` as `self` for us



Review

- How do we call a method on an object?

Accessor Methods

- Need to be able to get information about the object
  - Have `self` parameter
  - Return data/information

```
def getRank(self):
   "Returns the card’s rank."
   return self._rank

def getSuit(self):
   "Returns the card’s suit."
   return self._suit
```

```
card = Card(…, …)
```

- These methods will get called as `card.getRank()` and `card.getSuit()`
  - Python plugs `card` in for `self`

Another Special Method: `__str__`

- Returns a *string* that describes the object
- Whenever you **print** an object, Python checks if the object's `__str__` method is defined
  - Prints result of calling `__str__` method
- `str(<object>)` also calls `__str__` method

```
def __str__(self):
    """Returns a string
describing the card as 'rank of
suit'."""
    result = ""
    if self._rank == 11:
        result += "Jack"
    elif self._rank == 12:
        result += "Queen"
    elif self._rank == 13:
        result += "King"
    elif self._rank == 14:
        result += "Ace"
    else:
        result += str(self._rank)
    result += " of " + self._suit
    return result
```

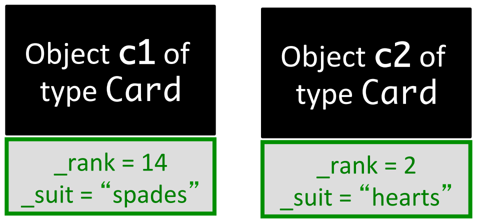
Using the Card Class

Invokes the `__str__` method

```
def main():
    c1 = Card(14, "spades")
    print(c1)
    c2 = Card(2, "hearts")
    print(c2)
```

Displays:

```
Ace of spades
2 of hearts
```



Example: Card Color

- Problem: Add a method to the `Card` class called `getCardColor` that returns the card's suit's color ("red" or "black")

- Procedure for defining a method (similar to functions)
  - What is the input?
  - What is the output?
  - What is the method signature/header?
  - What does the method do?

- How do we call the method?

`card2.py`

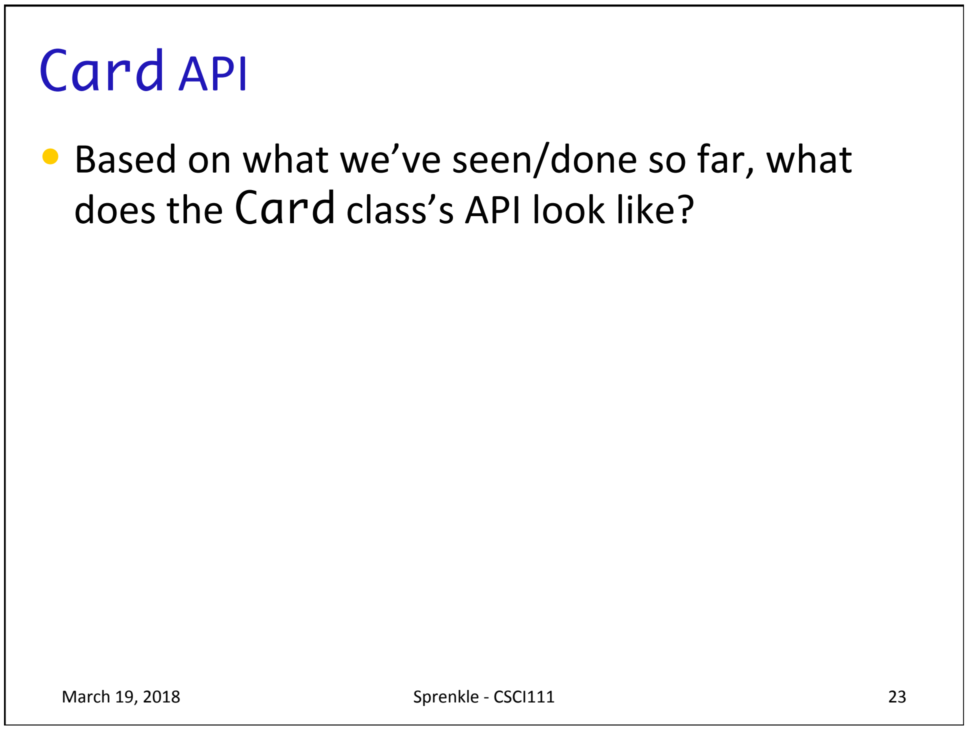
Example: Rummy Value

- Problem: Add a method to the `Card` class called `getRummyValue` that returns the value of the card in the game of Rummy

- Procedure for defining a method (similar to functions)
  - What is the input?
  - What is the output?
  - What is the method signature/header?
  - What does the method do?

- How do we call the method?

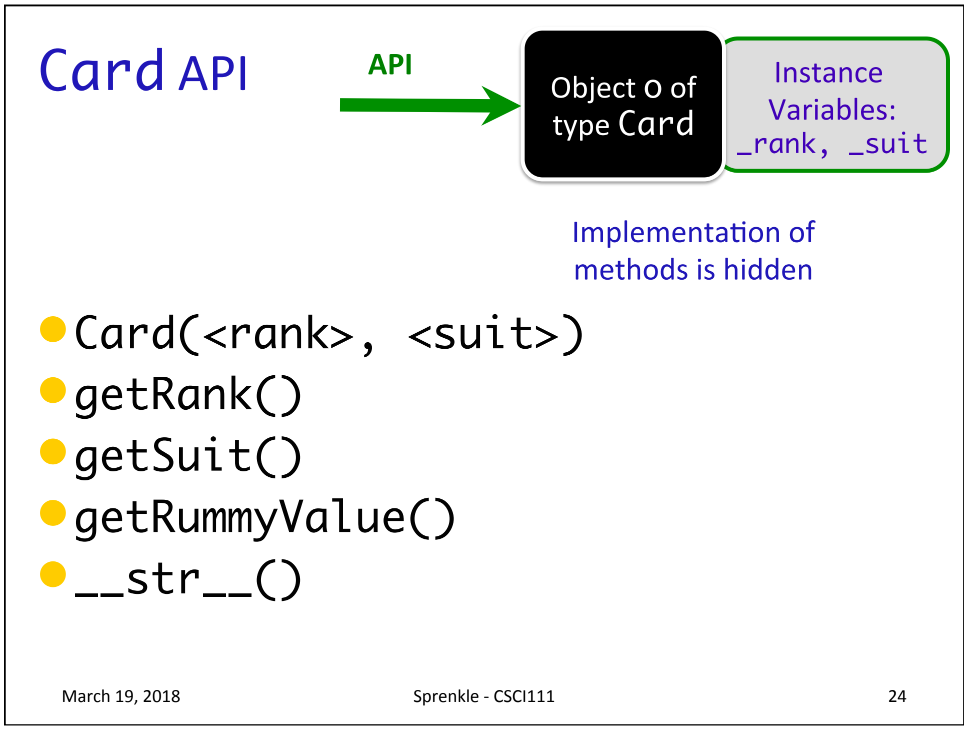
`card2.py`



Card API

- Based on what we've seen/done so far, what does the Card class's API look like?

March 19, 2018 Sprenkle - CSCI111 23



Card API

API

Object o of type Card

Instance Variables: `_rank, _suit`

Implementation of methods is hidden

- `Card(<rank>, <suit>)`
- `getRank()`
- `getSuit()`
- `getRummyValue()`
- `__str__()`

March 19, 2018 Sprenkle - CSCI111 24

Defining a Card Class

(not covered in class)

- Create a class that represents a playing card
  - How can we represent a playing card?
  - What information do we need to represent a playing card?
- Do we ***need*** a class to represent a card?
  - Does any built-in data type naturally represent a card?
  - What are the tradeoffs to those approaches?

Using the Card class

- Having the Card class means that we can represent a Card in code

Now that we have the Card class, how can we **use** it?

Using the Card class

Now that we have the Card class, how can we **use** it?

- Let's write a simplified version of the game of War
  - Basically just part of a round
- What are the rules of a round of War?

war.py

Review

```
from graphics import *

win = GraphWin("Picture")
win.setBackground("black")
```

```
from card import *

c = Card(7, "diamonds")
print(c.getRank())
```

- Same programming as before
- Just defining our own classes

Using the `Card` class

Now that we have the `Card` class, how can we **use** it?

- Can make a `Deck` class
  - What data should a `Deck` contain?
  - How can we represent that data?
- To start: write methods `__init__` and `__str__`
  - What do the method headers look like?

Looking Ahead

- Prelab 9 for tomorrow
  - Engaged in the object-oriented reading
- Lab 9 due Friday
- Exam Friday
  - Discussion on Wednesday