Objectives

- Wrap up dictionaries
- Defining our own classes

Review: Dictionaries

- What is a dictionary in Python?
  - What is it helpful for representing?
- What is the syntax for creating a new dictionary?
- How do we access a key's value from a dictionary? (2 ways)
  - What happens if there is no mapping for that key in each case?
- How do we create a key → value mapping in a dictionary?
- How do we iterate through a dictionary?

- Using objects
  - How do we know what we can do to objects?
  - How do we create objects?
  - How do we perform operations on an object?

Problem

Example file:

```
Name1 2026
Name2 2028
Name3 2026
Name4 2027
Name5 2028
…
```

- Same form of the file as before
  - <firstname> <classyear>
- Goal: Report the *number* of students in each graduation year
  - How do we want to model the data?
  - What is the key? What is the value?

Example output:

```
2026 4
2027 8
2028 5
2029 7
```

- Problem-solving approach:
  - Pretend you are the computer, how would you solve this problem?
- Revision: How would program change if used the CSV file format instead?

Equivalent Solutions

```
if key not in dictionary :
   dictionary[key] = 1
else:
   value = dictionary[key] + 1
   dictionary[key] = value
```

```
if key not in dictionary :
   dictionary[key] = 1
else:
   dictionary[key] += 1
```

ABSTRACTIONS

Abstractions

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

```
encrypt_file(filename, key)
```

    - 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

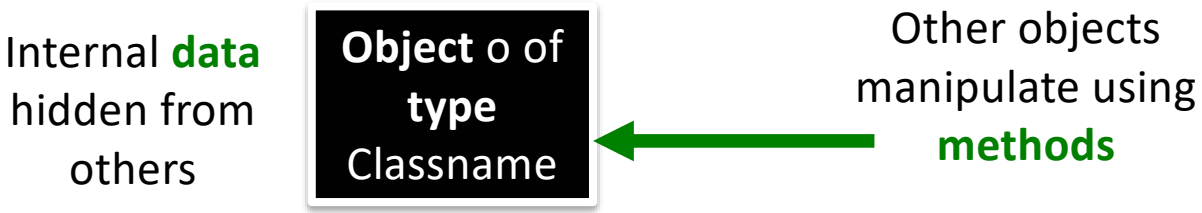
- Provide an abstraction for how to organize and reason about data
- Example: `GraphWin` class
  - Had ***attributes*** (i.e., data or state) background color, width, height, and title
  - Each `GraphWin` object has these attributes
    - Each `GraphWin` object has its own values for these attributes
  - 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 a 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**



Object o is an *instance* of Classname

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:
  - Suit (one of “hearts”, “diamonds”, “clubs”, “spades”)
  - Rank
    - 2-10: numbered cards
    - 11: Jack
    - 12: Queen
    - 13: King
    - 14: Ace

Pattern of Presentation Today

How We Define It

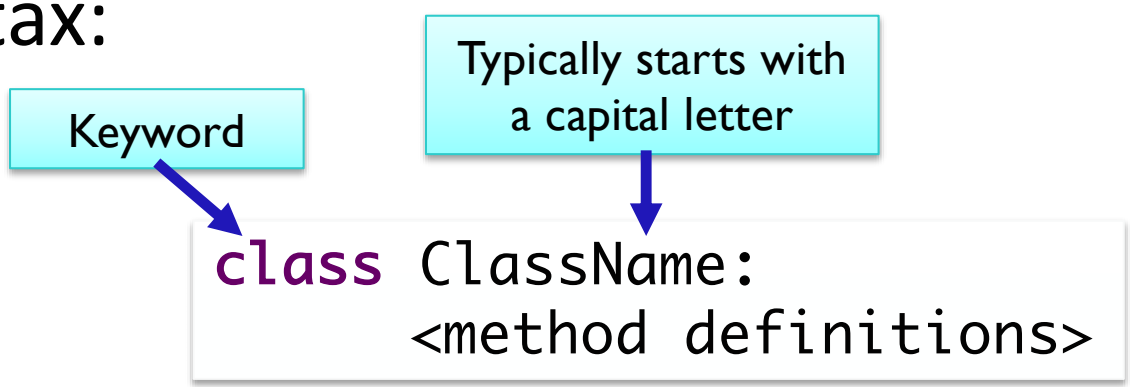
- The code we write so that someone else can use this code

How Someone Calls/Uses It

- How someone uses/leverages our code

Defining a New Class

- Syntax:



Card Class (Incomplete)

```
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
```

card.py

Card Class (Incomplete)

Class 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
```

Method Doc String

Methods

card.py

Card Class (Incomplete)

Class 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
```

Method Doc String

Methods

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

Defining the Constructor: `__init__`

- `__init__` method is like the ***constructor***
- In constructor, define ***instance variables***
  - **Instance variables:** the data contained in every object
  - Also called **attributes** or **fields**
- Constructor **never *returns*** anything

First parameter of *every* method is `self`
- reference 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

Convention: named with _

Review

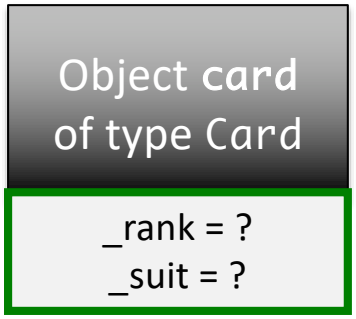
- How do we call/use the constructor for a class?

Using the Constructor

Method Signature

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

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

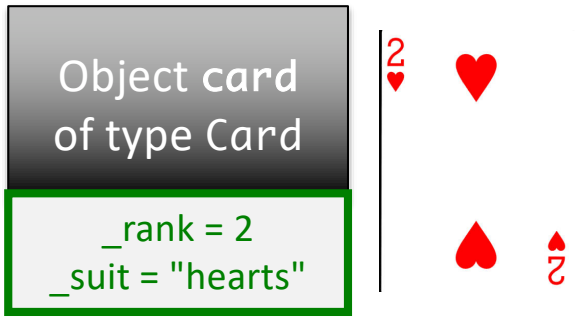


Using the Constructor

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

Method Signature

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



Looking Ahead

- Pre Lab 9 due before lab on Tuesday
- Exam next Friday
  - See preparation guide