Course	
Term	
Week	
Date	
Chapter. Topic	10. Classes and Object-Oriented Programming

Classes and Objects

Siva R Jasthi

Computer Science and Cybersecurity

Metropolitan State University

Outline

- Classes
- Objects
- Abstract Thinking

Abstract Thinking: Examples

- Digital Music Player:
  - Play, Skip, volume +, volume -, pause, start (on), stop (off),
  - Browse, Select

- Uber Driver:
  - Request a ride, **take_me(“some place”)**

You care what it does! You don’t care how it is done!

Objects! Objects! Objects Everywhere!

- Look around! What objects do you see?
  - Table
  - Chair
  - Pen
  - Blackboard
  - Student
  - Book
  - Bench
  - Instructor
  - Shoes
  - Classroom
  - Course
  - Paper
  - Bag
  - Computer
  - Garbage Bin
  - Eye Glasses
  - Shirt
  - Door
- Imagine your room (or office) at your home (work)
  - What objects are there?
- Everything in the world can be seen as an object

Each Object has some properties

- Let us take an Object from the previous slide
  - A book (A java text book)
- Let us speak loud about that Book.
  - It is a book
  - It is written by two authors Bell and Parr.
  - It deals with the subject Java
  - It is used for two courses ICS140 and ICS141
  - It has 350 pages
  - Its ISBN is 0130 32377 2
  - It costs $70
  - This book is brand new (not used)
  - It weighs around 2 lbs.
  - I bought this from Barnes and Noble.

Each Object has some properties (contd.)

- Properties of the Book, Contd…
  - It is published in UK
  - It is published by McGrawHill Publishers
  - It is soft-cover bound.
  - It is owned by me
  - The book has 30 chapters.
  - It is first published in the year 1990.
  - The latest edition of the book is : Second
  - The title of the book is “Java For Students”
- Now, let us take a “Time” magazine and outline the properties of this book.
  - Do…..

Book → Properties → Mapping to python

You: I bought a book
Me: Can you tell me more about it? (properties = nouns =)

- title --> str
- authors --> list of strings
- price --> float
- page_count --> int
- cover_type --> str (hard cover or soft cover)
- is_hard_cover --> bool
- isbn_10 --> str
- isbn_13 --> str
- edition_no --> str
- publisher --> str
- pub_date --> str (~ Date object)
- avg_rating_on_amazon --> float
- language --> str
- item_weight --> float
- item_weight_type --> str (pound, gram, kg)
- dimensions --> tuple of 3 floats
- topic --> str

Come up with some properties for these classes/objects

- PHONE (model, price, brand, memory, screen_size, cost)
- COMPUTER (model, price, brand, memory, screen_size)
- STUDENT (name, grade, age, height, gpa, current_subjects, email, phone, parents_details, address, school, majors_list, is_double_major?)

Saw an object? Can you tell me about it?

- Student 1 (Vincent)

His name is Vincent.
He is a boy.
He is 16 years old.
He has black hair.
He likes reading.
He plays basketball.
Grade
Email
Phone
Tech_id
Social_security_number
address
Numer_of_credits_earned
Number_of_courses_completed
Current_classes_registered

- Student 2 (Alex)

His name is Alex.
He is a boy.
He is 17 years old.
He has white hair.
He likes reading.
He plays tennis.
Grade
Email
Phone
Tech_id
Social_security_number
address
Numer_of_credits_earned
Number_of_courses_completed
Current_classes_registered

Properties provide HALF-THE-STORY

- What is the other half?
- When we tell about an OBJECT, explaining about its properties reveals only partial picture of the object.
- We also need to tell about
  - how we can change these properties of the object
  - how we can know the properties of the object
  - some uses of the object
  - what do people do with that object

Operations on the Objects

- Let us recall the Book from the previous example.
- What are the operations?
  - Publish the book
  - Read the book
  - Buy the book
  - Loan the book
  - Revise the book
  - Set the price of the book
  - Know the number of pages in the book
  - Know the author of the book
  - Give a recommendation on the book
  - Trade it for a new edition of the book

In-Class Exercise on the Objects (10 min)

- Take Television object.
- What are the 5 properties?
- What are the some methods/operations you c an take that change those properties?

In-Class Exercise on the Objects (10 min)

- Television Properties:
  - model, volume, channel, is_on, location

- **Television Methods:**
  - Set_volume_up(50) -- where is the volume at right now?
  - Volume_up(5) -- increase the volume by 5 points
  - Volume_down(4)
  - Switch_on()
  - Switch_off()
  - set_channel(channel_no) -- get_channel( ) what is the channel you are watching?
  - set_location(location_details) -- get_location( )where is the TV located?

Objects – Let us summarize

- We can give the complete description of an OBJECT by telling 3 things about it.
  - What is its type? (eg. It is a type of a BOOK)
  - What are its properties?
  - What are the operations?

- In Object Oriented Analysis/Design/Programming (OOA/D/P), these are termed as
  - What is its type? (CLASS)
  - What are its Properties (DATA)
  - What are its Operations (METHODS)

- Class = Data + Methods (is OO principle)

Class = Variables + Operations (is OO principle)

One class and many object

Class =
Template =
Model =
Master Copy =
Blueprint =
Idea =
Concept =
Generic class

- Student 1 (Vincent)

His name is Vincent.
He is a boy.
He is 16 years old.
He has black hair.
He likes reading.
He plays basketball.
Grade
Email
Phone
Tech_id
Social_security_number
address
Numer_of_credits_earned
Number_of_courses_completed
Current_classes_registered

- Student 2 (Alex)

Hix name is Alex.
He is a boy.
He is 17 years old.
He has white hair.
He likes reading.
He plays tennis.
Grade
Email
Phone
Tech_id
Social_security_number
address
Numer_of_credits_earned
Number_of_courses_completed
Current_classes_registered

Object =
Instance =
Concrete realization =
Specific instance

How do we think when we see objects?

- Consider the similar type of Objects.
- What do you think of the following?
  - Cell Phone
  - Corded Phone
  - Cordless Phone,
  - Internet Phone

- What do you think of the following?
  - Straight Line
  - Filled Square
  - Shallow Oval
  - Plastic Circle
  - Empty Cylinder
  - Mary's Cone
  - Red Triangle
  - Orange Rectangle
  - Small Ellipse
  - Metallic Star
  - Green Diamond
  - Solid Sphere

How do we think when we see objects?

- What do you think of the following?
  - Tourist Bus
  - Ambulance
  - Bicycle
  - Boat
  - Plane
  - MotorCycle
  - Bob's Truck
  - Used Car
  - New SUV
  - Family Van
  - Honda Civic
  - Dodge Caravan with 100K miles
  - Single-Owner Used Car
  - 2003 Toyotoa Corolla costs less than $16000

Let us think about 50 states of United States

Object = mn_state, ca_state
Class = State

Let us think about Some Countries

Abstract Thinking: Think higher. Think generic!

- It is possible to hide the details of one particular object.
- It is also possible to come with some “generic” template which can capture the details of all the objects.
- Take the case of “50 States of United States”. What is the template that can capture the details of all the states?
- Such a template is called as “**CLASS**”
- This type of thinking is called “Abstract Thinking” (where we are not hung up on a particular object, but thinking in terms of all objects)

Class: US_State (Link)

- Class US_State:
  - Name:
  - Capitol:
  - Flower:
  - Bird

```
1  # Defining the class for US_State
2  class US_State:
3
4      def __init__(self, name, capitol, flower, bird):
5          self.name = name
6          self.capitol = capitol
7          self.flower = flower
8          self.bird = bird
9
10     def getDetails(self):
11         print("Name : " + self.name)
12         print("Capitol : " + self.capitol)
13         print("Flower : " + self.flower)
14         print("Bird : " + self.bird)
15
16
17 # Creating an instance
18 us_minnesota = US_State("Minnesota", "St. Paul", "Common Loon",
19 us_minnesota.getDetails()
20
```

Class: US_State (Link)

```
1  # Defining the class for US_State
2  class US_State:
3
4      def __init__(self, name, capitol, flower, bird):
5          self.name = name
6          self.capitol = capitol
7          self.flower = flower
8          self.bird = bird
9
10     def getDetails(self):
11         print("Name : " + self.name)
12         print("Capitol : " + self.capitol)
13         print("Flower : " + self.flower)
14         print("Bird : " + self.bird)
15
16
17 # Creating an instance
18 us_minnesota = US_State("Minnesota", "St. Paul", "Common Lo
19 us_minnesota.getDetails()
20
```

- __init__ method is a special method.
- It is called “Constructor”
- This method is automatically called when we create an object
- Now, let us create some more US States.

Class: US_State (Link)

```
1  # Defining the class for US_State
2  class US_State:
3
4      def __init__(self, name, capitol, flower, bird):
5          self.name = name
6          self.capitol = capitol
7          self.flower = flower
8          self.bird = bird
9
10     def getDetails(self):
11         print("Name : " + self.name)
12         print("Capitol : " + self.capitol)
13         print("Flower : " + self.flower)
14         print("Bird : " + self.bird)
15
16
17 # Creating an instance
18 us_minnesota = US_State("Minnesota", "St. Paul", "Common Loon",
19 us_minnesota.getDetails()
20
```

- getDetails( ) is a method in US_State class.
- You can have as many methods as you want

Class: US_State (Link)

```
# Creating an instance

us_minnesota = US_State("Minnesota", "St. Paul",
"Common Loon", "Pink Lady Slipper")


us_minnesota.getDetails()
```

- We create an instance by invoking the class definition.
- This automatically calls the __init__ method.

In-Class Exercise: Student (5 min)

Create a new instance of your_state and print its details.

In-Class Exercise: Student (10 min)

Create a class called "Student" with two variables - name and marks.

Then create two student instances.

Print (getDetails) of those two instances.

Class: Book (Link)

- Class Book:
  - Name
  - Author
  - ISBN

Let us write it in the class

Class: US_State (Link)

```
# Creating an instance

us_minnesota =
US_State("Minnesota", "St. Paul",
"Common Loon", "Pink Lady Slipper")

# call getDetails method
us_minnesota.getDetails()
```

- Using the instance name (remote), you call the methods.

- Analogy: You use your remote
  - to switch on TV
  - To shutoff TV
  - To change channels
  - To change the volume

Class is 1; Objects are many

- Class: Represents the abstraction of a number of similar (or identical) objects. Describes the properties and operations of the object.
- Object: represents a specific instance of a Class. It incorporates some data (properties) and the actions (operations) that are associated with that data.

- For a given Class, there can be many Objects.
- There is one BOOK class, there can be millions of books (objects)
- There is one USSTATE class, there are 50 states.

Terminology: Let us fix the names

- Class:
  - Though the following names are less common, you can form your own mental image of what a “Class” represents
  - Template, Model, Blue Print, Design, Master, Skeleton.
- Object:
  - This is also called Instance.
  - There is also a Class called “Object” in Java.
  - So, when I say “Object”, what do I mean? Do I mean an instance of a class? Or Do I mean “Object” class?
  - For this purpose, we use “Instance” in this class to represent a specific instance of a class.

- So, Class is 1. Objects are many!

Terminology: Let us fix the names (contd.)

- Properties (variables)
  - also called “Data” or “Attributes” or “Fields” or “Characteristics” or “Descriptors” or “Variables”
  - let us use “**Data**” in concepts & “**Variable**” in specifics.
  - Tip: When you describe something, some descriptors (usually all **Nouns**) become “data” or “variables”.

- Operations (methods)
  - also called “actions”, “ methods”, “messages” or functions
  - Tip: When you describe something, some descriptors (usually all **Verbs**) become “methods”

Terminology: What do we have so far?

- Class = is a model of (Data + Methods)
  - Class represents a “generic” concept
- Object = is an incorporation of (Data + Methods)
  - Instance represents a “specific” object of a given Class.
- Here is an example of a “Book” class
  - “name” and “author” are the data (variables) of the Book Class.
  - “Java For Students” is an instance of Book Class.
  - This instance has the following values. Name = “Java For Students”, Author = “Bell and Parr”
  - getName and getAuthor are two methods.

Class & Instance: Master & Copy – An analogy

- Imagine that
  - I have today's lecture notes as master copy
  - I made 30 copies of the master copy and distributed to all of you.
- We can make an analogy as
  - master copy = Class
  - copies = Instance

Q1. Can I make another copy from the same Master? **YES**
Q2. Can I create another instance from the same Class? **YES**.
Q3. How many copies can I make? **MANY** (until you are out of paper)
Q4. How many instances can I create? **MANY** (until you are out of memory)

Class & Instance: Master & Copy – An analogy

Imagine that I have the following box on the master copy

Name	
Student ID	
Email	

Each student will have his/her own set of values.

Now I made 30 copies of that master copy and asked you to fill in the details. And you did!

Name	Al Gore Retire
Student ID	3607
Email	agore@aol.com

Name	George Bush
Student ID	4509
Email	gbush@msn.com

Class & Instance: Master & Copy – An analogy

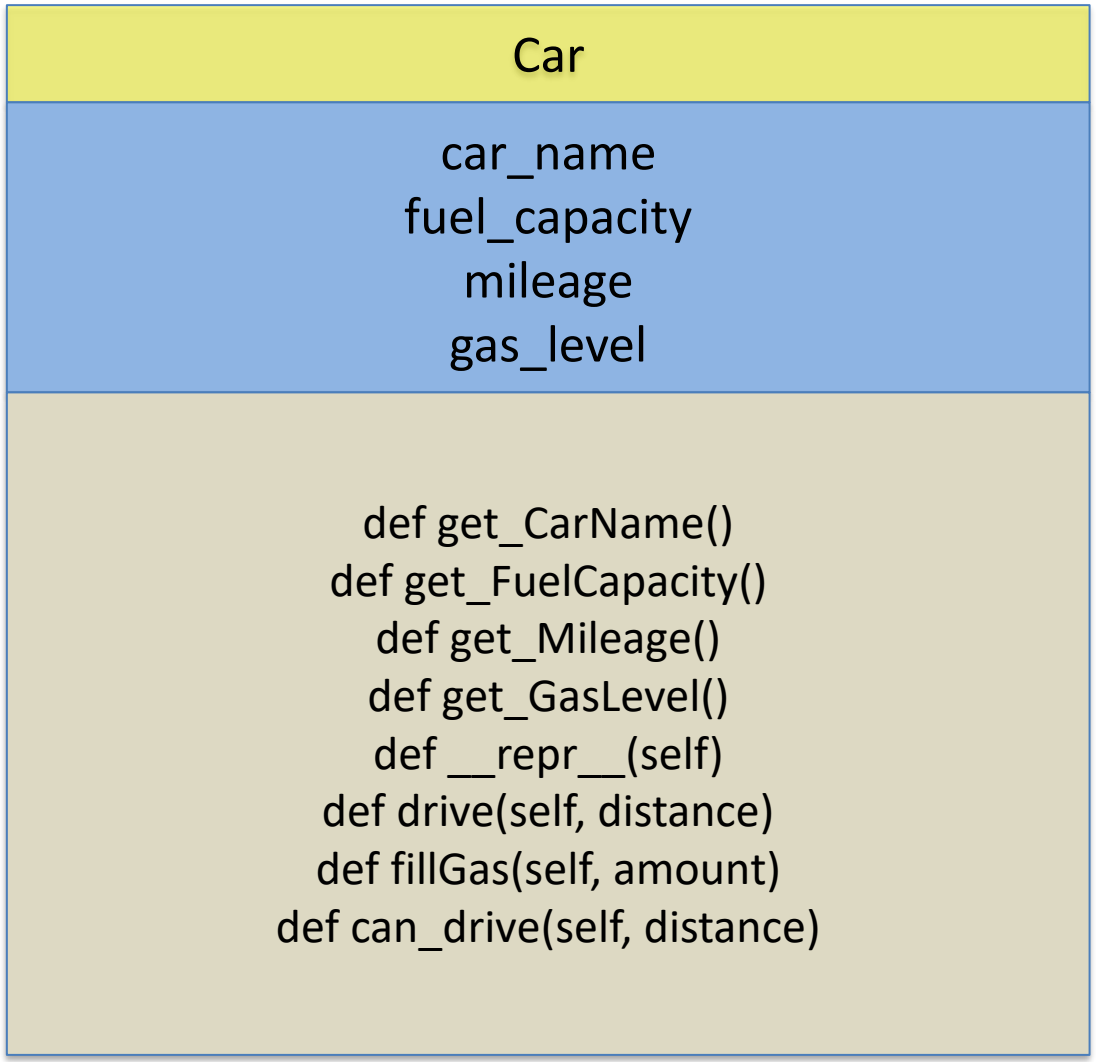
Q5. What do I have on my master copy? Nothing.
The class on its own doesn't have much value

Q6. What do you on your copy? Some unique values.
Your instance has its own set of values

Q7. What happens to Student A's Email if Student B's Email changes? Nothing. Changes to one instance does not affect another instance. Each instance is separate.

Q8. What happens to other copies (instances) if Student A dumps his copy (instance) in the garbage? Nothing happens to the other instances. They are still valid.

UML Notation



This is UML notation for the “Car” class.

UML = Unified Modeling Language

Summary

“Object Oriented Thinking” – the concepts we talked today are not specific to Python.

These are valid and applicable in any programming language.

The syntax might differ between the languages. However, the foundational concepts are common.

Get started with solid understanding on “Objects and Classes”!

We just covered C of Objects in CRUD. We will review R,U, and D in the next class.