

빅 데이터 혁신 공유 대학

파이썬으로 배우는 데이터 구조

한동대학교 전산전자공학부

김영섭 교수



교육부



한국연구재단



Data Structures in Python

Chapter 1 - 2

- Object-Oriented Programming
- OOP in Python
- OOP - Fraction Example
- OOP - Classes
- OOP - In-Place Operators
- Exceptions
- Exception Clauses

내 아들들을 먼 곳에서 이끌며 내 딸들을 땅 끝에서 오게 하며 내 이름으로 불려지는 모든 자 곧 내가 내 영광을 위하여 창조한 자를 오게 하라 그를 내가 지었고 그를 내가 만들었노라 (사43:6-7)

그런즉 너희가 먹든지 마시든지 무엇을 하든지 다 하나님의 영광을 위하여 하라 (고전10:31)

Agenda

- **Object-Oriented Programming**
 - Objects - State and Behavior
 - Classes
- OOP in Python
 - Constructors
 - Methods & Self
 - Point class
 - Saving a class file and the module Geometry.py
 - Data Field Encapsulation
- References:
 - [Problem Solving with Algorithms and Data Structures using Python](#)
 - Chapter 1.13 Object-Oriented Programming in Python

Exercise

- What is the output of the following code fragment?

```
x = ['a', 'b', 'c']  
y = x  
z = ['a', 'b', 'c']  
print (x == y)  
print (x is y)  
print (x == z)  
print (x is z)
```

```
x = 'Hello'  
y = x  
z = 'Hello'  
print (x == y)  
print (x is y)  
print (x == z)  
print (x is z)
```

Object Oriented Programming(OOP)

- An **object** represents an entity in the real world that can be distinctly identified, e.g., students, dogs, cars, cats, books.



- Object Oriented Programming(OOP) involves the use of objects to create programs.

Objects

- Cars may have:
 - **information:** color, current speed, current gear, etc.
 - **function:** accelerate, brake, change gear, reverse, etc.



Car A

color: red
speed: 50
doors: 2
gear: 4th

Car B

color: white
speed: 5
doors: 4
gear: 1st

accelerate

color: white
speed: 10
doors: 4
gear: 1st

Object State and Behavior

- Every real world object has:
 - State — information that the object stores.
 - Behavior — functionality of the object, i.e., what the object can do.
- Example:
 - Consider a system managing university students.
 - A student object has:
 - State — id, name, age, contact number, address, stage, grade, completed courses, current courses, advisor, faculty, ...
 - Behavior — enroll in a new course, change contact number, change enrollment, choose degree, ...
 - A person object has:
 - State - id, name, age, contact number, address, ...
 - Behavior - eat, drink, wear, talk, work, meet, swim, run, drive, ...

Object is state + behavior

- A software object's **state** is represented by its variables, called **data fields**.
- A software object implements its **behavior** with **methods**.
 - Every object is a bundle of variables and related methods.
 - We make an object perform actions by invoking the methods on that object.
- Example:

```
my_list = [ 1, 2, 3 ]  
my_list.reverse()
```

In a Program

- Our program consists of many different objects.
- **Two objects of the same kind would have the same set of behaviors, but independent state information.**
 - Two string objects store different words, but can perform same methods, e.g., lower(), split(), index(), etc.
- For an object in our program
 - State — is defined by variables (data fields).
 - Behaviors — is defined by methods (actions).
- The definition of a particular kind of objects is called a **class**. Once created, **an object is an instance of a class**.

Python Class

- A **class** is the structure we use to define a category of objects. It defines the **state and behavior** of a category of objects.
- A class is a **template or blueprint defining** the data fields and actions (methods) that any instance (object) of that class can have.
- For an object in our program
 - State — is defined by variables (data fields).
 - Behaviors — is defined by methods (actions).
- Analogies for class and object:
 - Factory mold and products produced from that mold
 - Blueprint and apartment building units
 - Cookie cutter and cookies



Classes

- Python has a number of built-in classes
 - list, dict, set, int, float, boolean, str
- We can define our own classes
 - creates a new type of object in Python

```
class name_of_the_class:  
    # definition of the class goes here  
    # initializer  
    # methods
```

- Classes consist of:
 - **state** variables (sometimes called instance variables)
 - **methods** (functions that are linked to a particular instance of the class)

Example

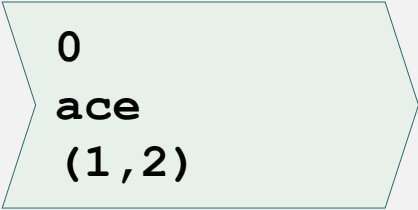
- An example:

```
class foo:  
    a, b, c = 0, "ace", (1,2)
```

← multiple assignments

- Instantiating Classes
 - A class is instantiated by calling the class object:

```
obj = foo()  
print(obj.a)  
print(obj.b)  
print(obj.c)
```

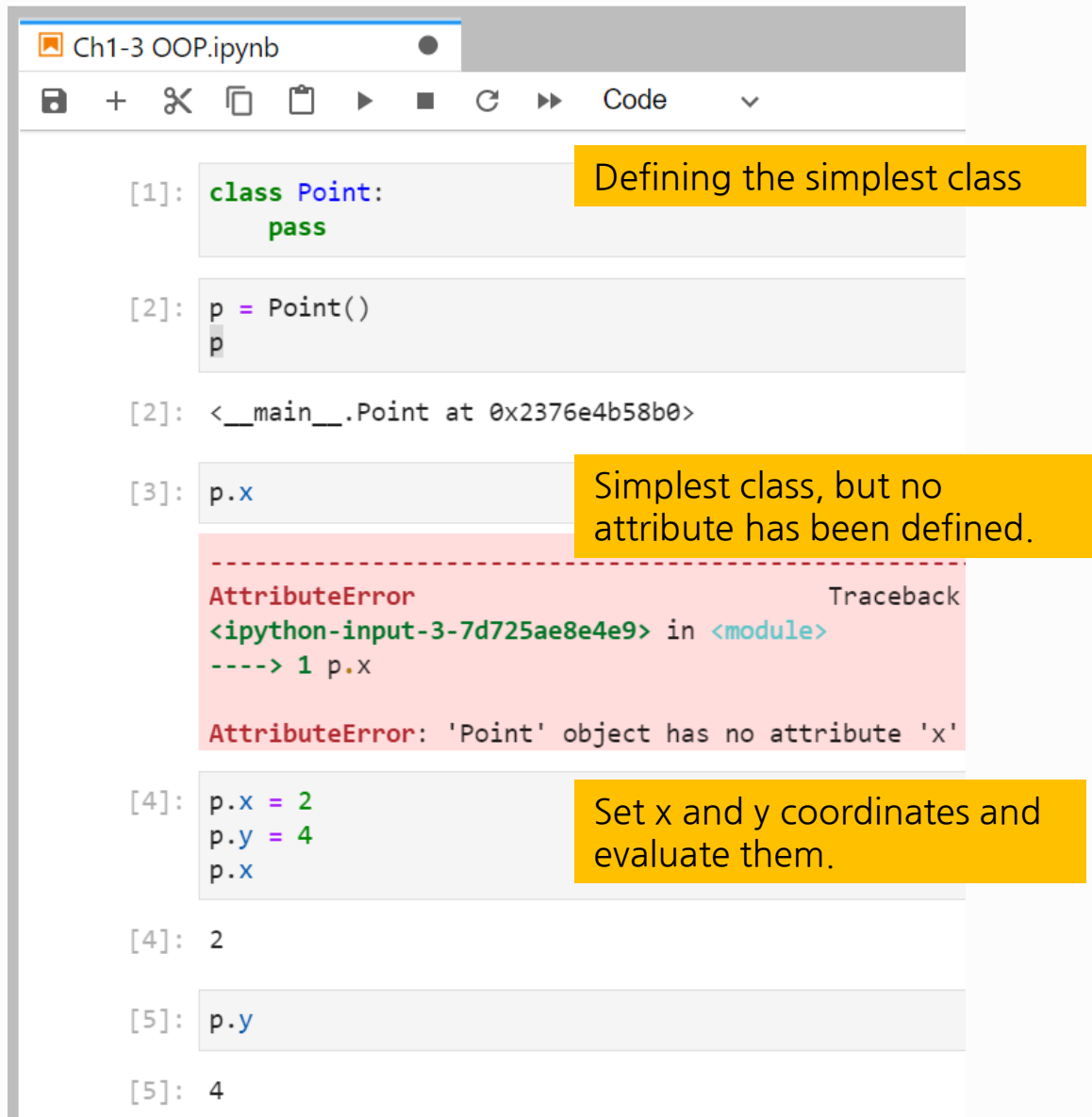


The simplest class possible

- A simple example:

```
class Point:  
    pass
```

- “pass” is a statement that does nothing. It is often used as a placeholder when developing code



The screenshot shows a Jupyter Notebook interface with the following code and output:

```
[1]: class Point:  
      pass
```

Defining the simplest class

```
[2]: p = Point()  
      p
```

```
[2]: <__main__.Point at 0x2376e4b58b0>
```

```
[3]: p.x
```

Simplest class, but no attribute has been defined.

```
-----  
AttributeError                                Traceback  
<ipython-input-3-7d725ae8e4e9> in <module>  
----> 1 p.x  
AttributeError: 'Point' object has no attribute 'x'
```

```
[4]: p.x = 2  
      p.y = 4  
      p.x
```

Set x and y coordinates and evaluate them.

```
[4]: 2
```

```
[5]: p.y
```

```
[5]: 4
```

Data Structures in Python

Chapter 1 - 2

- Object-Oriented Programming
- OOP in Python
- OOP - Fraction Example
- OOP - Classes
- OOP - In-Place Operators
- Exceptions
- Exception Clauses