Lecture





Restrict Access

Information-hiding







Public

Can be accessed anywhere

No access restriction

Private (non-public)

Can't (or shouldn't)
 be accessed outside
 the class

Two levels of access restriction

```
class Car:

def __init__(self, model, year):
    self.model = model
    self.year = year

Public instance attributes
```



```
class Car:

def __init__(self, model, year):
    self.model = model
    self.year = year

Public instance attributes
```

```
>>> my_car = Car("Escape", 2006)
>>> my_car.model
'Escape'
>>> my_car.year
2006
```



```
class Car:

def __init__(self, model, year):
    self.model = model
    self.year = year

Public instance attributes
```

```
>>> my_car = Car("Escape", 2006)
>>> my_car.model
'Escape'
>>> my_car.year
2006
```



```
class Car:

def __init__(self, model, year):
    self.model = model
    self.year = year

Public instance attributes
```

```
>>> my_car = Car("Escape", 2006)
>>> my car.model
'Escape'
>>> my_car.year
2006
```

```
class Car:

def __init__(self, model, year):
    self.model = model
    self.year = year

Public instance attributes
```

```
>>> my_car.year = 5600
>>> my_car.year
5600
```



```
class Car:

def __init__(self, model, year):
    self.model = model
    self.year = year

Public instance attributes
```



```
class Car:

def __init__(self, model, year):
    self.model = model
    self.year = year

Public instance attributes
```

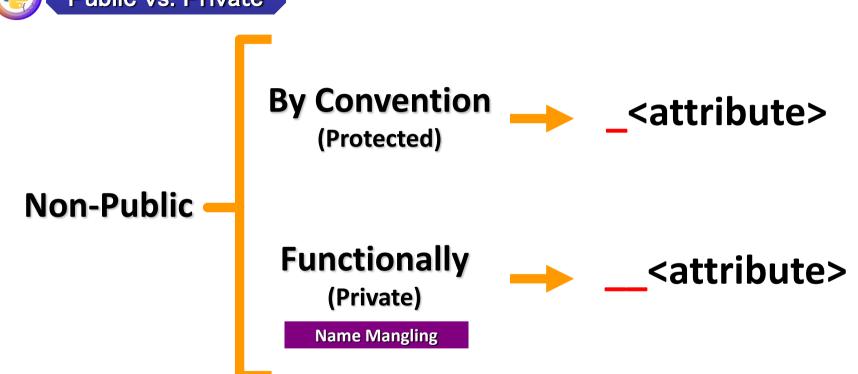
```
>>> my_car.year = 5600
>>> my_car.year
5600
```



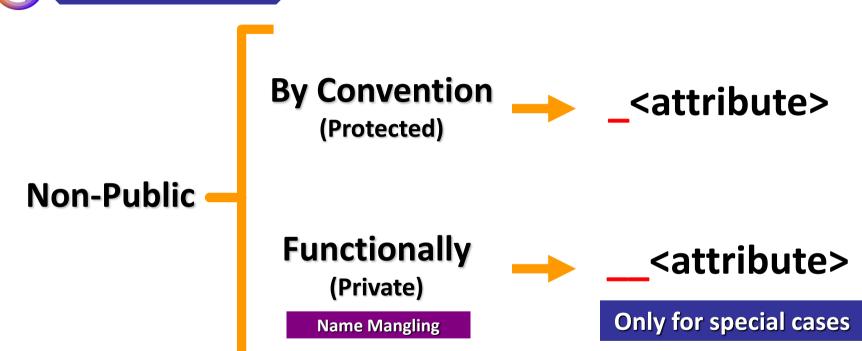
Risk of Public Attributes













Method Names and Instance Variables

Use the function naming rules: lowercase with words separated by underscores as necessary to improve readability.

Use one leading underscore only for non-public methods and instance variables.

To avoid name clashes with subclasses, use two leading underscores to invoke Python's name mangling rules.

Python mangles these names with the class name: if class Foo has an attribute named __a, it cannot be accessed by Foo.__a. (An insistent user could still gain access by calling Foo._Foo__a.) Generally, double leading underscores should be used only to avoid name conflicts with attributes in classes designed to be subclassed.

Note: there is some controversy about the use of __names (see below).



```
class Car:

def __init__(self, model, year, id_num, engine_serial_num):
    self.model = model
    self.year = year
    self._id_num = id_num
    self.__engine_serial_num = engine_serial_num
```

Non-Public Attributes



```
class Car:

def __init__(self, model, year, id_num, engine_serial_num):
    self.model = model
    self.year = year
    self.__id_num = id_num
    self.__engine_serial_num = engine_serial_num
```

Non-Public Attributes



```
class Car:

def __init__(self, model, year, id_num, engine_serial_num):
    self.model = model
    self.year = year
    self._id_num = id_num
    self.__engine_serial_num = engine_serial_num
```

Non-Public Attributes





```
>>> class Car:
   def init (self, model, year, id num, engine serial num):
       self.model = model
        self.year = year
        self. id num = id num
       self. engine serial num = engine serial num
>>> my car = Car("Escape", 2006, "44542", "201109048934242")
>>> my car. engine serial num
Traceback (most recent call last):
 File "<pyshell#3>", line 1, in <module>
   my car. engine serial num
AttributeError: 'Car' object has no attribute ' engine serial num'
```



```
>>> class Car:
   def init (self, model, year, id num, engine serial num):
       self.model = model
        self.year = year
        self. id num = id num
       self. engine serial num = engine serial num
>>> my car = Car("Escape", 2006, "44542", "201109048934242")
>>> my car. engine serial num
Traceback (most recent call last):
 File "<pyshell#3>", line 1, in <module>
   my car. engine serial num
AttributeError: 'Car' object has no attribute ' engine serial num'
```



```
>>> class Car:
   def init (self, model, year, id num, engine serial num):
        self.model = model
        self.year = year
        self. id num = id num
        self. engine serial num = engine serial num
>>> my car = Car("Escape", 2006, "44542", "201109048934242")
>>> my car. engine serial num
Traceback (most recent call last):
  File "<pyshell#3>", line 1, in <module>
    my car. engine serial num
AttributeError: 'Car' object has no attribute ' engine serial num'
```

Non-Public?



We don't use the term "private" here, since no attribute is really private in Python





```
>>> class Car:
   def init (self, model, year, id num, engine serial num):
        self.model = model
        self.year = year
        self. id num = id num
        self. engine serial num = engine serial num
>>> my car = Car("Escape", 2006, "44542", "201109048934242")
>>> my car. id num
'44542'
>>> my car. Car engine serial num
'201109048934242'
```



```
>>> class Car:
    def init (self, model, year, id num, engine serial num):
        self.model = model
        self.year = year
        self. id num = id num
        self. engine serial num = engine serial num
>>> my car = Car("Escape", 2006, "44542", "201109048934242")
>>> my car. id num
'44542'
>>> my car. Car engine serial num
'201109048934242'
```

```
>>> class Car:
   def init (self, model, year, id num, engine serial num):
        self.model = model
        self.year = year
        self. id num = id num
        self. engine serial num = engine serial num
>>> my car = Car("Escape", 2006, "44542", "201109048934242")
>>> my car. id num
'44542'
>>> my car. Car engine serial num
'201109048934242'
```

You can still access them (technically), but you shouldn't



```
public String model;
public int year;
protected String idNum;
private String EngineSerialNum;
```



```
public String model;
public int year;
protected String idNum;
private String EngineSerialNum;
```

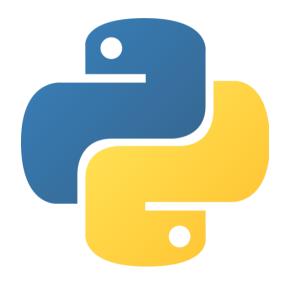


```
public String model;
public int year;
protected String idNum;
private String EngineSerialNum;
```



```
public String model;
public int year;
protected String idNum;
private String EngineSerialNum;
```





No Access Modifiers



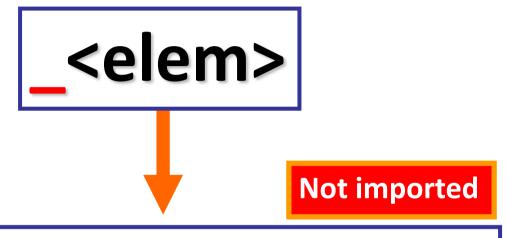


No Access Modifiers

Conventions







from <module> import *

