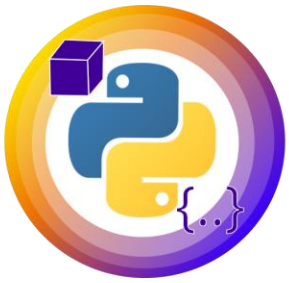
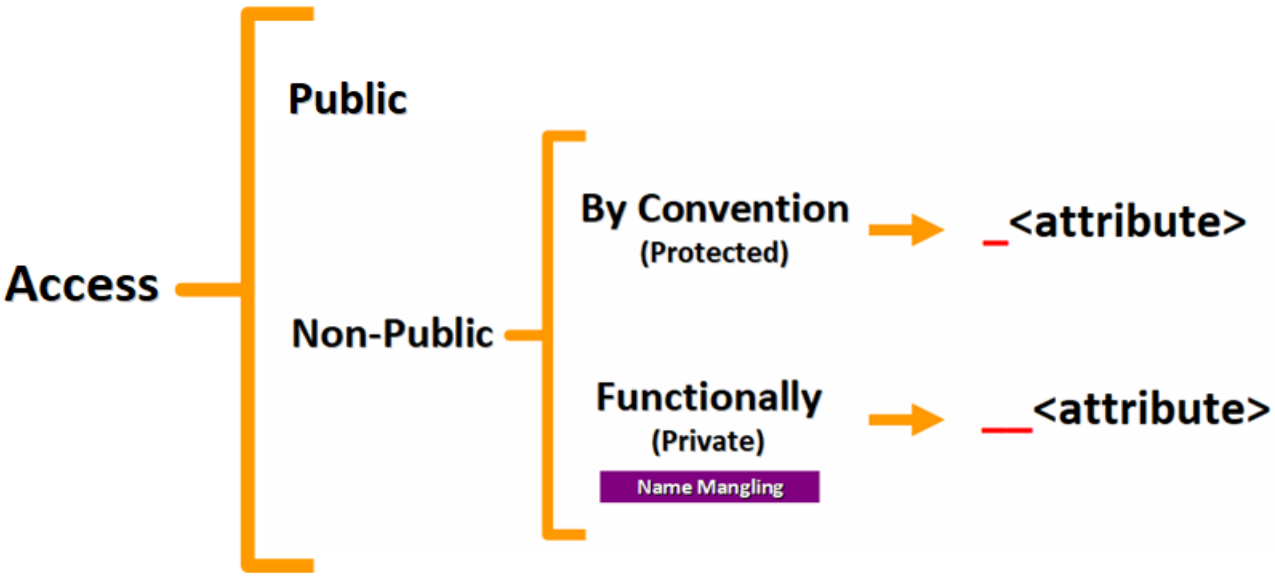


Python OOP: Encapsulation & Abstraction

Key Takeaways

- **Encapsulation**
  - "Bundling" of data and actions into a single unit (class).
  - Applied through the principle of information hiding.
  - You should restrict direct access to your data unless there is an important reasons not to do so.
  - To do this, you can define non-public attributes in Python.



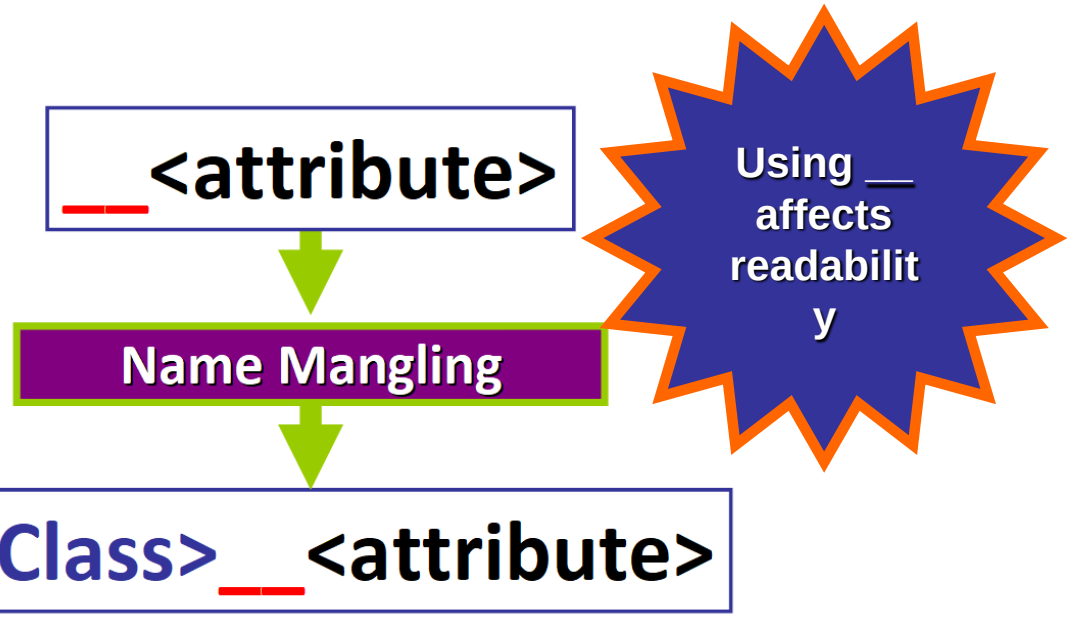
Note: The terms "private" and "protected" are symbolic in Python. No attribute is completely private.

Public vs. Non-Public

Key Takeaways

- **Public and Non-Public Attributes. Example:**

```
class Player:
    def __init__(self, username, rank, time_played):
        self.username = username
        self._rank = rank
        self.__time_played = time_played
```



The **recommended way** to indicate that an attribute is "protected" and should not be accessed outside of the class, is to use a **single leading underscore**.

Key Takeaways

- **Abstraction**
  - Different facets and expressions:
    - The interface of a component should be independent of the implementation.
    - Relying on more general or “abstract” types of objects to avoid code repetition with the use of inheritance.