Key Concepts: Inheritance in Python

Definition and Importance

- **Inheritance**: Allows a class (child/subclass) to acquire attributes and behaviors (methods) from another class (parent/superclass).
- Reduces redundancy, promotes code reuse, and keeps the codebase organized.
- Enables:
  - Adding new properties or methods to the child class.
  - Overriding or modifying inherited properties/methods without affecting the parent class.

Example 1: Simple Inheritance

1. **Parent Class `P`**:
   - Holds a variable `a` with a value of `7`.

```
class P:
    a = 7
```

2. **Child Class `C`**:
   - Inherits from `P` but is itself empty.

```
class C(P):
    pass
```

3. **Usage**:
   - Creating an instance of `C` gives access to `a` inherited from `P`.

```
c = C()
print(c.a)  # Output: 7
```

Example 2: Practical Use Case with Employees

Step 1: Create Parent Class

- **Class `Employees`**:
  - Initializes `name` and `last` variables.

```
class Employees:
    def __init__(self, name, last):
        self.name = name
        self.last = last
```

Step 2: Create Child Classes

1. **Child Class `Supervisors`**:
   - Inherits from `Employees`.
   - Adds a new variable `password` using the `super()` method to initialize the parent class attributes.

```
class Supervisors(Employees):
    def __init__(self, name, last, password):
        super().__init__(name, last)
        self.password = password
```

2. **Child Class `Chefs`**:
   - Inherits from `Employees`.
   - Introduces a new method `leave_request()` to request leave for a specific number of days.

```
class Chefs(Employees):
    def leave_request(self, days):
        return f"May I take a leave for {str(days)} days?"
```

Step 3: Instantiate and Use Classes

1. Create instances:

```
adrian = Supervisors("Adrian", "A", "apple")
emily = Chefs("Emily", "E")
juno = Chefs("Juno", "J")
```

2. Call methods and access instance variables:

```
print(emily.leave_request(3))  # Output: May I take a leave for 3 days?
print(adrian.password)        # Output: apple
print(emily.name)             # Output: Emily
```

Key Features of Inheritance

1. **Reusability**:
    - Common functionality resides in the parent class.
    - Unique features are added in child classes.
2. **Extensibility**:
    - Parent class can be extended by adding new methods or attributes in child classes.
3. **Modifiability**:
    - Overriding parent methods in child classes does not affect the original parent method.

Advantages of Inheritance

- Simplifies code structure by promoting modular design.
- Avoids duplication, ensuring consistency across similar functionalities.
- Provides flexibility for extending base class functionality.

In this example, inheritance streamlines the management of employees, supervisors, and chefs by reusing and extending shared functionalities.