Python super

Summary: in this tutorial, you will learn how to use the Python `super()` to delegate to the parent class when overriding methods.

Introduction to the Python super

First, define an `Employee` class (https://www.pythontutorial.net/python-oop/python-class/):

```
class Employee:
    def __init__(self, name, base_pay, bonus):
        self.name = name
        self.base_pay = base_pay
        self.bonus = bonus

    def get_pay(self):
        return self.base_pay + self.bonus
```

The `Employee` class has three instance variables (https://www.pythontutorial.net/python-oop/python-instance-variables/) `name`, `base_pay`, and `bonus`. It also has the `get_pay()` method that returns the total of base_pay and bonus.

Second, define the `SalesEmployee` class that inherits from the `Employee` class:

```
class SalesEmployee(Employee):
    def __init__(self, name, base_pay, bonus, sales_incentive):
        self.name = name
        self.base_pay = base_pay
        self.bonus = bonus
        self.sales_incentive = sales_incentive

    def get_pay(self):
        return self.base_pay + self.bonus + self.sales_incentive
```

The `SalesEmployee` class has four instance variables `name`, `base_pay`, `bonus`, and `sales_incentive`. It has the `get_pay()` method that overrides (https://www.pythontutorial.net/python-oop/python-overriding-method/) the `get_pay()` method in the `Employee` class.

super().__init__()

The `__init__()` method of the `SalesEmployee` class has some parts that are the same as the ones in the `__init__()` method of the `Employee` class.

To avoid duplication, you can call the `__init__()` method of `Employee` class from the `__init__()` method of the `SalesEmployee` class.

To reference the `Employee` class in the `SalesEmployee` class, you use the `super()`. The `super()` returns a reference of the parent class from a child class.

The following redefines the `SalesEmployee` class that uses the `super()` to call the `__init__()` method of the `Employee` class:

```
class SalesEmployee(Employee):
    def __init__(self, name, base_pay, bonus, sales_incentive):
        super().__init__(name, base_pay, bonus)
        self.sales_incentive = sales_incentive
```

```
    def get_pay(self):
        return self.base_pay + self.bonus + self.sales_incentive
```

When you create an instance of the `SalesEmployee` class, Python will execute the `__init__()` (https://www.pythontutorial.net/python-oop/python-__init__/) method in the `SalesEmployee` class. In turn, this `__init__()` method calls the `__init__()` method of the `Employee` class to initialize the `name`, `base_pay`, and `bonus`.

Delegating to other methods in the parent class

The `get_pay()` method of the `SalesEmployee` class has some logic that is already defined in the `get_pay()` method of the `Employee` class. Therefore, you can reuse this logic in the `get_pay()` method of the `SalesEmployee` class.

To do that, you can call the `get_pay()` method of the `Employee` class in the `get_pay()` method of `SalesEmployee` class as follows:

```
class SalesEmployee(Employee):
    def __init__(self, name, base_pay, bonus, sales_incentive):
        super().__init__(name, base_pay, bonus)
        self.sales_incentive = sales_incentive

    def get_pay(self):
        return super().get_pay() + self.sales_incentive
```

The following calls the `get_pay()` method of the `Employee` class from the `get_pay()` method in the `SalesEmployee` class:

```
super().get_pay()
```

When you call the `get_pay()` method from an instance of the `SalesEmployee` class, it calls the `get_pay()` method from the parent class (`Employee`) and return the sum of the result of the `super().get_pay()` method with the `sales_incentive`.

Put it all together

The following shows the complete code:

```
class Employee:
    def __init__(self, name, base_pay, bonus):
        self.name = name
        self.base_pay = base_pay
        self.bonus = bonus

    def get_pay(self):
        return self.base_pay + self.bonus


class SalesEmployee(Employee):
    def __init__(self, name, base_pay, bonus, sales_incentive):
        super().__init__(name, base_pay, bonus)
        self.sales_incentive = sales_incentive

    def get_pay(self):
        return super().get_pay() + self.sales_incentive


if __name__ == '__main__':
    sales_employee = SalesEmployee('John', 5000, 1000, 2000)
    print(sales_employee.get_pay())  # 8000
```

Summary

- Use `super()` to call the methods of a parent class from a child class.