In [31]:
```
class item:
    discount = 0.8
    everything = []
    def __init__(self, name, price, quantity):
        self.name = name
        self.price = price
        self.quantity = quantity

        item.everything.append(self)


    def calculator(self):
        return self.price * self.quantity

    def discounts(self):
        self.price = self.price * self.discount
        return self.price

    def __repr__(self):
        return f" item({'self.name'}, {self.price}, {self.quantity})"

item1 = item("phone", 246, 9)
item2 = item("charger", 340, 12)
item3 = item("head phone", 300, 30)
item4 = item("battery", 400, 22)
item4.discount = 0.75
item5 = item("casing", 350, 27)

print(item.everything)
```

```
[ item(self.name, 246, 9),  item(self.name, 340, 12),  item(self.name, 300, 3
0),  item(self.name, 400, 22),  item(self.name, 350, 27)]
```

In [26]:
```
print(item1.calculator())
```

```
2214
```

In [28]:
```
for instance in item.everything:
    print(instance.name)
```

```
phone
charger
head phone
battery
casing
```

In [32]:
```
import pandas as pd
read_file = pd.read_csv("oop.csv")
```

In [33]:
```
read_file
```

Out[33]:

	name	price	quantity
0	phone	100	1
1	laptop	250	5
2	pencil	300	20
3	book	20	7
4	charger	17	8

In [169]:
```
import csv

class item:
    discount = 0.8
    everything = []
    def __init__(self, name, price, quantity):
        self.name = name
        self.price = price
        self.quantity = quantity

        item.everything.append(self)


    def calculator(self):
        return self.price * self.quantity

    def discounts(self):
        self.price = self.price * self.discount
        return self.price

    @classmethod
    def instantiate_from_csv(cls):
        with open("oop.csv", "r", encoding="UTF-8") as f:
            reader = csv.DictReader(f)
            items = list(reader)

        for item in items:
            item = dict(item)
            item(
                name=item.get("name"),
                quantity=float(item.get("quantity")),
                price=int(item.get("price")),
            )

    def __repr__(self):
        return f" item({'self.name'}, {self.price}, {self.quantity})"
```

In [170]:
```
print(item.instantiate_from_csv())
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_7604\2268872508.py in <module>
----> 1 print(item.instantiate_from_csv())

~\AppData\Local\Temp\ipykernel_7604\3465272784.py in instantiate_from_csv(cls)
     30                 name=item.get("name"),
     31                 quantity=float(item.get("quantity")),
---> 32                 price=int(item.get("price")),
     33             )
     34

TypeError: 'dict' object is not callable
```

In [118]:
```
print(item.everything)
```

```
[]
```

In [108]:
```
import csv
with open('oop.csv', 'r') as file:
    reader = csv.DictReader(file)
    this = list(reader)
    for row in this:
        print(dict(row))
```

```
{'name': 'phone', 'price': '100', 'quantity': '1'}
{'name': 'laptop', 'price': '250', 'quantity': '5'}
{'name': 'pencil', 'price': '300', 'quantity': '20'}
{'name': 'book', 'price': ' 20', 'quantity': '7'}
{'name': 'charger', 'price': '17', 'quantity': '8'}
```

In [120]:
```
class MyClass:
    pass

my_instance = MyClass()

print(my_instance)
type(my_instance)
```

```
<__main__.MyClass object at 0x0000000010753388>
```

Out[120]:
```
__main__.MyClass
```

In [122]:
```
class MyClass:
    def first_method():
        return "This is my first method"

my_instance = MyClass()
```

In [133]:
```
class MyClass:
    def first_method(self):
        print("This is my first method")

my_instance = MyClass()
```

In [134]:
```
my_instance.first_method()
```

```
This is my first method
```

In [126]:
```
s = "MY STRING"

# call `str.title() directly
# instead of `s.title()`
result = s.title()
print(result)
```

```
My String
```

In [128]:
```
s = "MY STRING"

# call `str.title() directly
# instead of `s.title()`
result = str.title(s)
print(result)
```

```
My String
```

In [135]:
```
class MyClass:
    def return_list(self, input_list):
        return input_list

my_instance = MyClass()
my_instance.return_list([1, 2, 3])
```

Out[135]:
```
[1, 2, 3]
```

In [141]:
```
class MyClass:
    def return_list(self, input_list):
        return input_list

my_instance = MyClass()
my_instance.return_list([1, 2])
```

Out[141]:
```
[1, 2]
```

In [143]:
```
class MyClass:
    def return_list(self):
        print("here")

my_instance = MyClass()
my_instance.return_list()
```

```
here
```

In [144]:
```
class MyClass:
    def __init__(self):
        print("mine")

my_instance = MyClass()
```

```
mine
```

In [146]:
```
class MyClass:
    def __init__(self, string):
        print(string)

my_instance = MyClass("mine")
```

```
mine
```

In [148]:
```
class ExampleClass:

    def __init__(self, string):
        self.my_attribute = string

my_instance = ExampleClass("Hola!")
my_instance.my_attribute
```

Out[148]:
```
'Hola!'
```

In [154]:
```
class MyList:
    def __init__(self, initial_data):
        self.data = initial_data

    def append(self, new_item):


my_list = MyList([1, 2, 3, 4, 5])
my_list.data
```

Out[154]:
```
[1, 2, 3, 4, 5]
```

In [166]:
```
class MyList:

    def __init__(self, initial_data):
        self.data = initial_data
        # Calculate the initial length
        self.length = 0
        for item in self.data:
            self.length += 1

    def append(self, new_item):
        self.data = self.data + [new_item]
        # Update the length here

my_list = MyList([1, 2, 3, 4, 5])
my_list.append(6)
my_list.data
my_list.length
```

Out[166]: 5

In [158]:
```
a = my_list = [1, 2, 3]
b = new_item = [4]
print(a + b)
```

```
[1, 2, 3, 4]
```

In [ ]: