プログラミング教室のテクノロ

Pythonの道

トレーニングドリル⑥

もくじ

・クラスの継承

食べ物と飲み物を分けよう

前回、作成した料理注文システムを改良し、食べ物と飲み物を別で管理できる方法を学習する。食べ物を扱う「Food」クラスと、飲み物を扱う「Drink」クラスをそれぞれ用意する。

Food

情報

- 名前: サンドイッチ
- 値段: ¥500
- カロリー: 330kcal

Drink

情報

- 名前: コーヒー
- 値段: ¥300
- 容量: 180mL

継承-既存のクラスを活用する

新たにFoodクラスとDrinkクラスをつくっていく。
1から新たにつくることもできるが、すでにあるMenuItemクラスを利用して、それをもとにFoodクラスとDrinkクラスを作る。
そうすることで、共通部分をまとめることができ、効率的にコードを書くことができる。

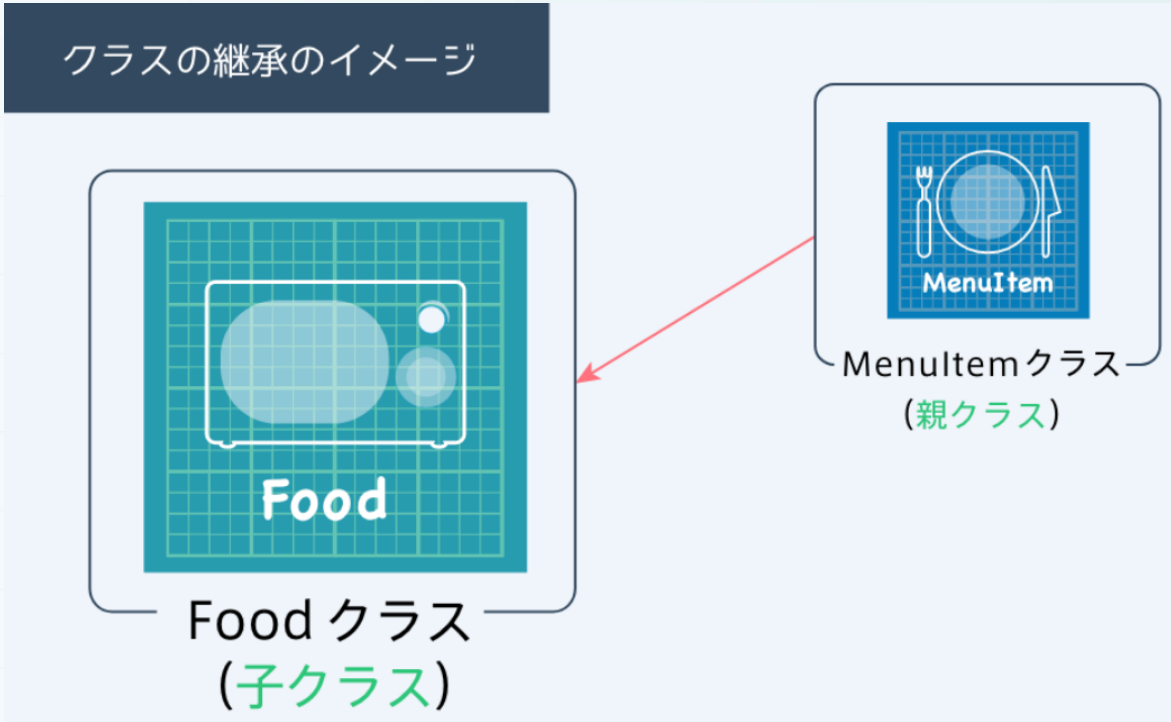
継承とは

あるクラスを元にして新たなクラスをつくることを「継承」と呼ぶ。「class 新しいクラス名(元となるクラス名):」とすることで他のクラスを継承して、新しいクラスを定義することができる。このとき、新しいクラスは「子クラス」、元となるクラスは「親クラス」と呼ぶ。

```
from menu_item import MenuItem

class Food(MenuItem):
    pass
```

子クラス　親クラス



練習

【問題】menu_item.py

```
class MenuItem:
    def __init__(self, name, price):
        self.name = name
        self.price = price

    def info(self):
        return self.name + ': ¥' + str(self.price)

    def get_total_price(self, count):
        total_price = self.price * count

        if count >= 3:
            total_price *= 0.9

        return round(total_price)
```

練習

【問題】food.py

from と import を用いて、 MenuItem クラスを読み込んでください

MenuItem クラスを継承して、 Food クラスを定義してください

練習

【問題】drink.py

from と import を用いて、 MenuItem クラスを読み込んでください

MenuItem クラスを継承して、 Drink クラスを定義してください

練習

【答え】menu_item.py

```
class MenuItem:
    def __init__(self, name, price):
        self.name = name
        self.price = price

    def info(self):
        return self.name + ': ¥' + str(self.price)

    def get_total_price(self, count):
        total_price = self.price * count

        if count >= 3:
            total_price *= 0.9

        return round(total_price)
```

練習

【答え】food.py

```
# from と import を用いて、 MenuItem クラスを読み込んでください
from menu_item import MenuItem

# MenuItem クラスを継承して、 Food クラスを定義してください
class Food(MenuItem):
    pass
```

練習

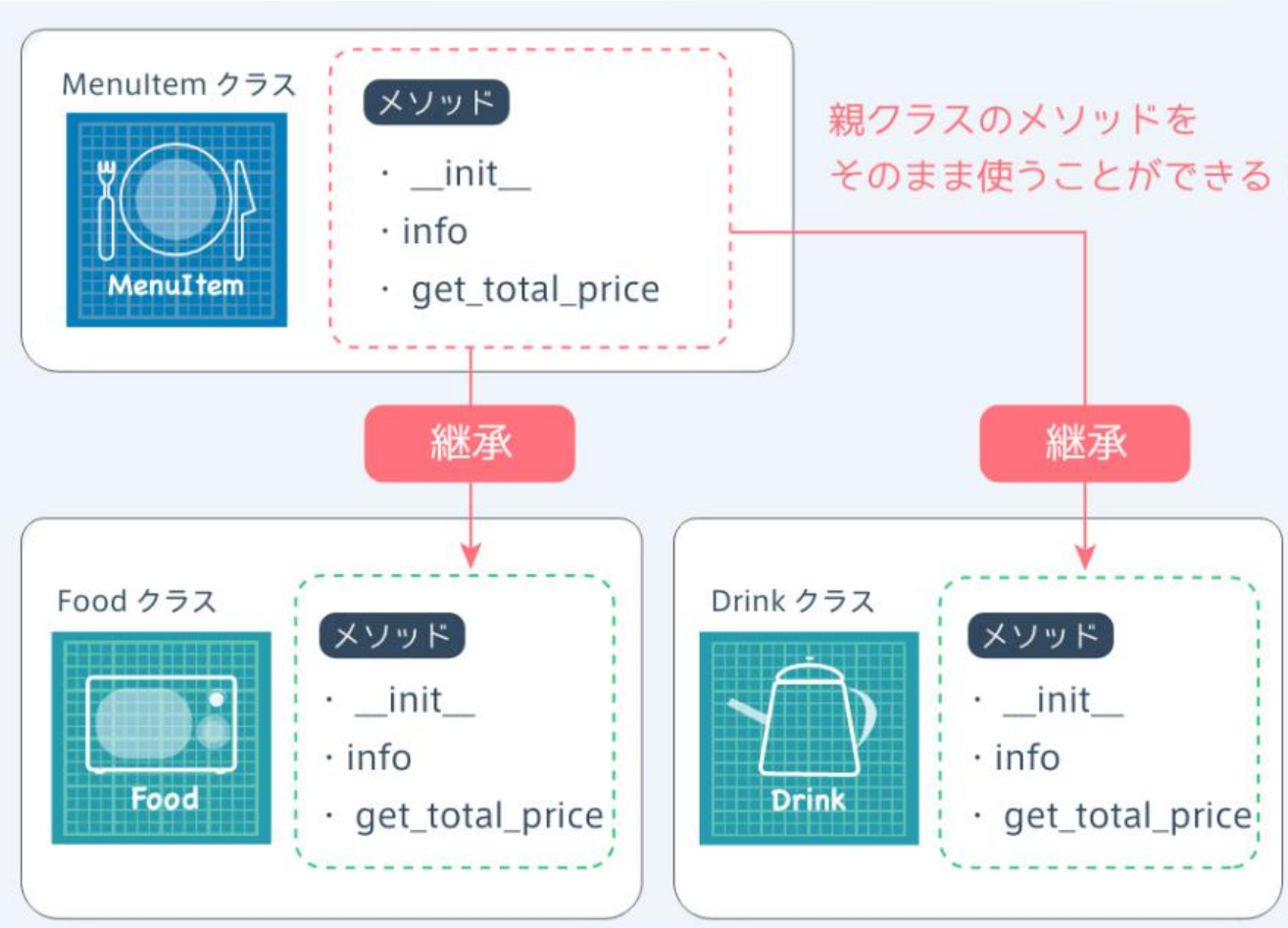
【答え】drink.py

```
# from と import を用いて、 MenuItem クラスを読み込んでください
from menu_item import MenuItem

# MenuItem クラスを継承して、 Drink クラスを定義してください
class Drink(MenuItem):
    pass
```

継承されるもの

MenuItemクラスをもとに、FoodクラスとDrinkクラスが作ることができた。継承を用いると子クラスには、親クラスのインスタンスメソッドが引き継がれる。Foodクラスのインスタンスも、MenuItemクラスの「info」メソッドが使えるようになる。



子クラスのインスタンス

継承をすると、子クラスは親クラスのインスタンスメソッドを引き継ぐ。Foodクラスのインスタンスは、MenuItemクラス内に定義してある「__init__」メソッドや「info」メソッドを使うことができる。

【script.py】

```
from food import Food

food1 = Food(“サンドイッチ”,500)

print(food1.info())
```

FoodクラスはMenuItemクラスを継承している

親クラスのインスタンスメソッド

【menu_item.py】

```
class MenuItem:
    :
    :
    def info(self):
        return self.name + “ :¥” + str(self.price)
    :
    :
```

練習

【問題】script.py

```
# Food クラスと Drink クラスをそれぞれ読み込んでください


# Food クラスのインスタンスを生成して変数 food1 に代入してください


# food1 に対して info メソッドを呼び出して戻り値を出力してください


# Drink クラスのインスタンスを生成して変数 drink1 に代入してください


# drink1 に対して info メソッドを呼び出して戻り値を出力してください
```

練習

【問題】menu_item.py

```
class MenuItem:
    def __init__(self, name, price):
        self.name = name
        self.price = price

    def info(self):
        return self.name + ': ¥' + str(self.price)

    def get_total_price(self, count):
        total_price = self.price * count

        if count >= 3:
            total_price *= 0.9

        return round(total_price)
```

練習

【問題】food.py

```
from menu_item import MenuItem

class Food(MenuItem):
    pass
```

練習

【問題】drink.py

```
from menu_item import MenuItem

class Drink(MenuItem):
    pass
```

練習

【答え】script.py

```
# Food クラスと Drink クラスをそれぞれ読み込んでください
from food import Food
from drink import Drink

# Food クラスのインスタンスを生成して変数 food1 に代入してください
food1 = Food("サンドイッチ",500)

# food1 に対して info メソッドを呼び出して戻り値を出力してください
print(food1.info())

# Drink クラスのインスタンスを生成して変数 drink1 に代入してください
drink1 = Drink("コーヒー",300)

# drink1 に対して info メソッドを呼び出して戻り値を出力してください
print(drink1.info())
```

練習

【答え】menu_item.py

```
class MenuItem:
    def __init__(self, name, price):
        self.name = name
        self.price = price

    def info(self):
        return self.name + ': ¥' + str(self.price)

    def get_total_price(self, count):
        total_price = self.price * count

        if count >= 3:
            total_price *= 0.9

        return round(total_price)
```

練習

【答え】food.py

```
from menu_item import MenuItem

class Food(MenuItem):
    pass
```

練習

【答え】drink.py

```
from menu_item import MenuItem

class Drink(MenuItem):
    pass
```

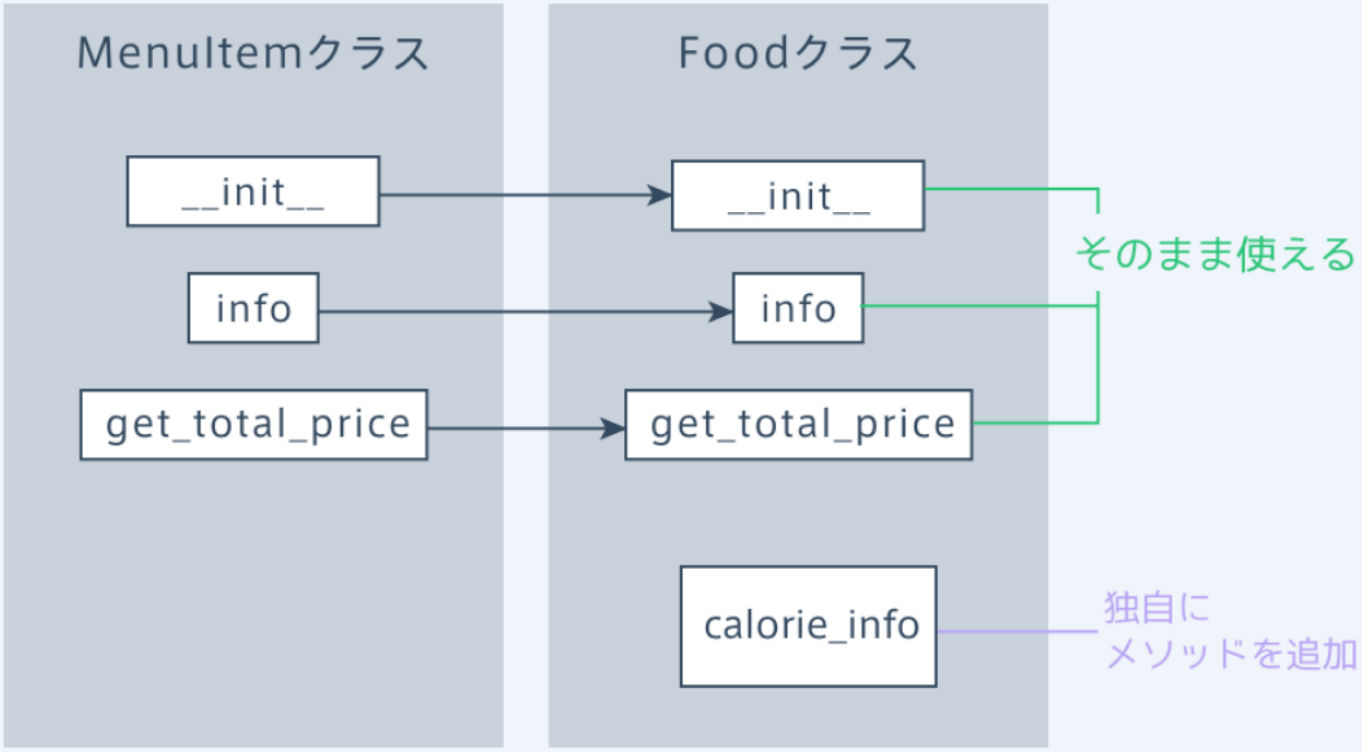
インスタンスメソッドを追加する

子クラスでは、親クラスから引き継いだメソッド以外にも、独自にメソッドを追加することができる。今回はFoodクラスのインスタンスにカロリーの情報を追加して、その値を出力するためのメソッドを追加してみよう。

【food.py】

```
class Food(MenuItem)
  def calorie_info(self):
        .
        .
```

Foodクラス内に独自のインスタンスメソッドを追加する



練習

【問題】script.py

```
from food import Food
from drink import Drink

food1 = Food('サンドイッチ', 500)

# food1 の calorie に 330 を代入してください

# food1 に対して calorie_info メソッドを呼び出してください
```

練習

【問題】menu_item.py

```
class MenuItem:
    def __init__(self, name, price):
        self.name = name
        self.price = price

    def info(self):
        return self.name + ': ¥' + str(self.price)

    def get_total_price(self, count):
        total_price = self.price * count

        if count >= 3:
            total_price *= 0.9

        return round(total_price)
```

練習

【問題】food.py

```
from menu_item import MenuItem

class Food(MenuItem):
    # calorie_info メソッドを定義してください
```

練習

【問題】drink.py

```
from menu_item import MenuItem

class Drink(MenuItem):
    pass
```

練習

【答え】script.py

```
from food import Food
from drink import Drink

food1 = Food('サンドイッチ', 500)

# food1 の calorie に 330 を代入してください
food1.calorie = 330

# food1 に対して calorie_info メソッドを呼び出してください
food1.calorie_info()
```

練習

【答え】menu_item.py

```
class MenuItem:
    def __init__(self, name, price):
        self.name = name
        self.price = price

    def info(self):
        return self.name + ': ¥' + str(self.price)

    def get_total_price(self, count):
        total_price = self.price * count

        if count >= 3:
            total_price *= 0.9

        return round(total_price)
```

練習

【答え】food.py

```
from menu_item import MenuItem

class Food(MenuItem):
    # calorie_info メソッドを定義してください
    def calorie_info(self):
        print(str(self.calorie) +"kcalです" )
```

練習

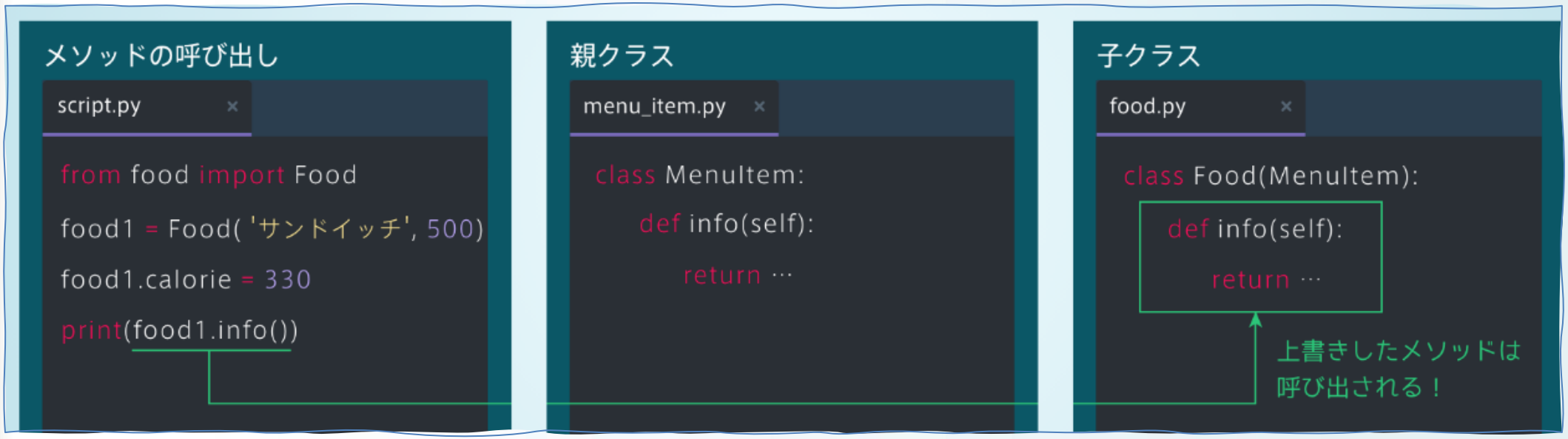
【答え】drink.py

```
from menu_item import MenuItem

class Drink(MenuItem):
    pass
```

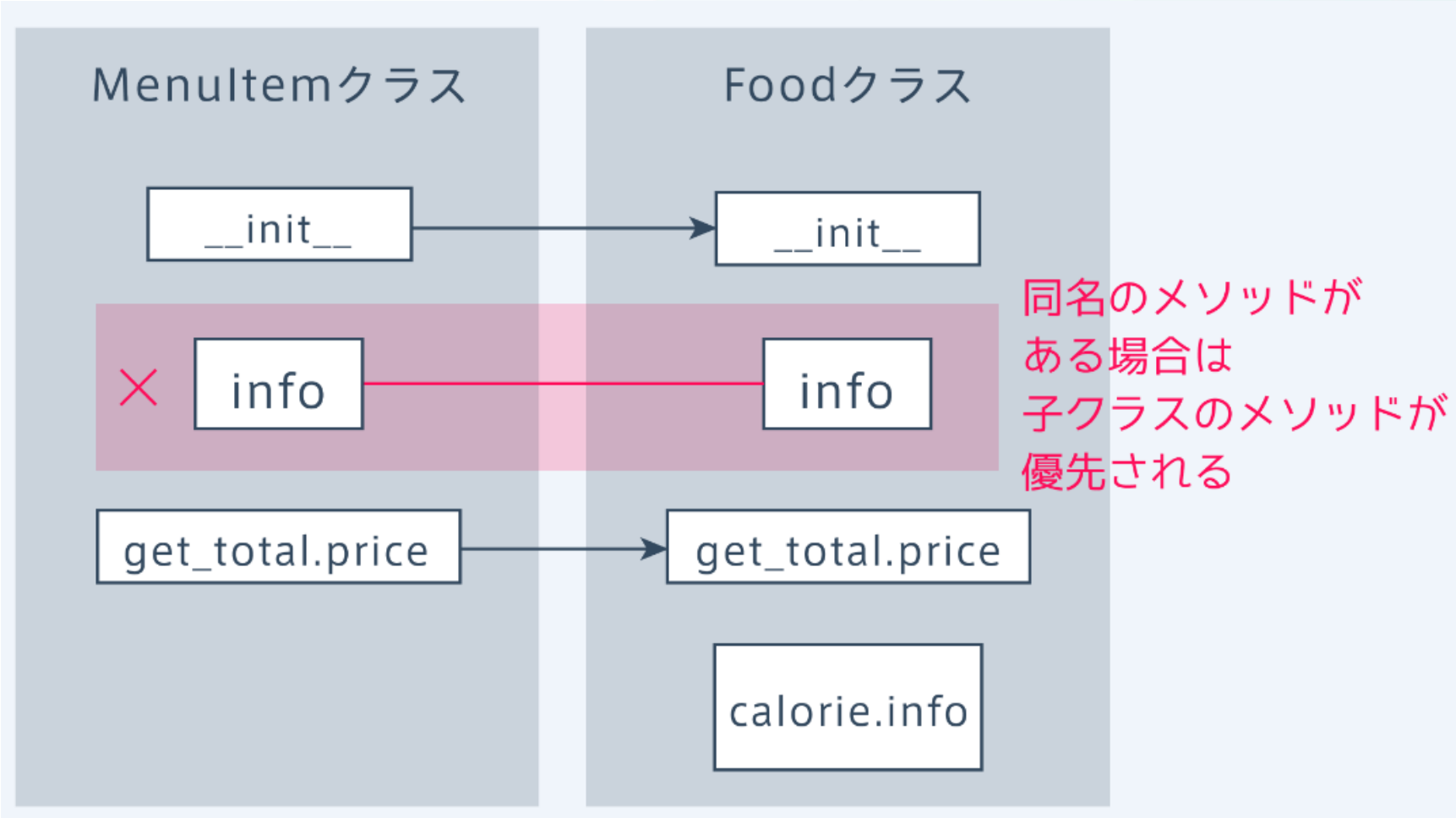
オーバーライド

親クラスにあるメソッドと同じ名前のメソッドを子クラスで定義すると、メソッドを上書きすることができる。これをメソッドの「オーバーライド」と呼ぶ。オーバーライドすると、子クラスのインスタンスは親クラスのメソッドではなく、子クラスで定義したメソッドを呼び出すようになる。



オーバーライドの仕組み

オーバーライドの仕組みとして、子クラスのインスタンスは子クラスで定義したメソッドを優先して呼び出すようになっている。そのため、子クラスと親クラスに同名のメソッドがある場合は、メソッドの内容が上書きされたようになる。



練習

【問題】script.py

```
from food import Food
from drink import Drink

food1 = Food('サンドイッチ', 500)
food1.calorie = 330

# food1 に対して info メソッドを呼び出して戻り値を出力してください
```

練習

【問題】menu_item.py

```
class MenuItem:
    def __init__(self, name, price):
        self.name = name
        self.price = price

    def info(self):
        return self.name + ': ¥' + str(self.price)

    def get_total_price(self, count):
        total_price = self.price * count

        if count >= 3:
            total_price *= 0.9

        return round(total_price)
```

練習

【問題】food.py

```
from menu_item import MenuItem

class Food(MenuItem):
    # info メソッドを定義してください

    def calorie_info(self):
        print(str(self.calorie) + 'kcalです')
```

練習

【問題】drink.py

```
from menu_item import MenuItem

class Drink(MenuItem):
    pass
```

練習

【答え】script.py

```
from food import Food
from drink import Drink

food1 = Food('サンドイッチ', 500)
food1.calorie = 330

# food1 に対して info メソッドを呼び出して戻り値を出力してください
print(food1.info())
```

練習

【答え】menu_item.py

```
class MenuItem:
    def __init__(self, name, price):
        self.name = name
        self.price = price

    def info(self):
        return self.name + ': ¥' + str(self.price)

    def get_total_price(self, count):
        total_price = self.price * count

        if count >= 3:
            total_price *= 0.9

        return round(total_price)
```

練習

【答え】food.py

```
from menu_item import MenuItem

class Food(MenuItem):
    # info メソッドを定義してください
    def info(self):
        return self.name + ': ¥' + str(self.price) + ' (' 
+ str(self.calorie) + 'kcal)'


    def calorie_info(self):
        print(str(self.calorie) + 'kcalです')
```

練習

【答え】drink.py

```
from menu_item import MenuItem

class Drink(MenuItem):
    pass
```

オーバーライドの仕組み

Foodクラスでinfoメソッドをオーバーライドして、
カロリーの情報を表示できるようになった。次にDrinkクラスで
もinfoメソッドをオーバーライドしてみよう。Drinkクラスでは、
飲み物の量を表示するようする。

```
サンドイッチ: ¥500 (330kcal)
コーヒー: ¥300 (180mL)
```

練習

【問題】script.py

```
from food import Food
from drink import Drink

food1 = Food('サンドイッチ', 500)
food1.calorie = 330
print(food1.info())

# Drink クラスのインスタンスを生成して変数 drink1 に代入してください

# drink1 の amount に 180 を代入してください

# drink1 に対して info メソッドを呼び出して戻り値を出力してください
```

練習

【問題】menu_item.py

```
class MenuItem:
    def __init__(self, name, price):
        self.name = name
        self.price = price

    def info(self):
        return self.name + ': ¥' + str(self.price)

    def get_total_price(self, count):
        total_price = self.price * count

        if count >= 3:
            total_price *= 0.9

        return round(total_price)
```

練習

【問題】food.py

```
from menu_item import MenuItem

class Food(MenuItem):
    def info(self):
        return self.name + ': ¥' + str(self.price) + ' ('
+ str(self.calorie) + 'kcal)'

    def calorie_info(self):
        print(str(self.calorie) + 'kcalです')
        print(str(self.calorie) + 'kcalです')
```

練習

【問題】drink.py

```
from menu_item import MenuItem

class Drink(MenuItem):
    # info メソッドを定義してください
```

練習

【答え】script.py

```
from food import Food
from drink import Drink

food1 = Food('サンドイッチ', 500)
food1.calorie = 330
print(food1.info())

# Drink クラスのインスタンスを生成して変数 drink1 に代入してください
drink1 = Drink("コーヒー",300)

# drink1 の amount に 180 を代入してください
drink1.amount = 180

# drink1 に対して info メソッドを呼び出して戻り値を出力してください
print(drink1.info())
```

練習

【答え】menu_item.py

```
class MenuItem:
    def __init__(self, name, price):
        self.name = name
        self.price = price

    def info(self):
        return self.name + ': ¥' + str(self.price)

    def get_total_price(self, count):
        total_price = self.price * count

        if count >= 3:
            total_price *= 0.9

        return round(total_price)
```

練習

【答え】food.py

```
from menu_item import MenuItem

class Food(MenuItem):
    def info(self):
        return self.name + ': ¥' + str(self.price) + ' ('
+ str(self.calorie) + 'kcal)'

    def calorie_info(self):
        print(str(self.calorie) + 'kcalです')
        print(str(self.calorie) + 'kcalです')
```

練習

【答え】drink.py

```
from menu_item import MenuItem

class Drink(MenuItem):
    # info メソッドを定義してください
    def info(self):
        return self.name + ': ¥' + str(self.price) + ' (' + str(self.amount) + 'mL)'
```

__init__メソッドのオーバーライド

Foodクラス内で、MenuItemクラスの__init__メソッドを上書きする。図のように、オーバーライドする場合には、引数の数などを変えることもできる。

【script.py】

```
from food import Food

food1 = Food(“ サンドイッチ” ,500, 330)
```

引数でカロリーの情報を渡す

【food.py】

```
class Food(Menuitem):

    def__init__(self, name, price, calorie)
```

引数でカロリーの情報を渡す

練習

【問題】script.py

```
from food import Food
from drink import Drink

# Food() に引数を追加してください

print(food1.info())

drink1 = Drink('コーヒー', 300)
drink1.amount = 180
print(drink1.info())
```

練習

【問題】menu_item.py

```
class MenuItem:
    def __init__(self, name, price):
        self.name = name
        self.price = price

    def info(self):
        return self.name + ': ¥' + str(self.price)

    def get_total_price(self, count):
        total_price = self.price * count

        if count >= 3:
            total_price *= 0.9

        return round(total_price)
```

練習

【問題】food.py

```
from menu_item import MenuItem

class Food(MenuItem):
    # __init__ メソッドを定義してください


    def info(self):
        return self.name + ': ¥' + str(self.price) + ' ('
+ str(self.calorie) + 'kcal)'

    def calorie_info(self):
        print(str(self.calorie) + 'kcalです')
```

練習

【問題】drink.py

```
from menu_item import MenuItem

class Drink(MenuItem):
    def info(self):
        return self.name + ': ¥' + str(self.price) + ' ('
+ str(self.amount) + 'mL)'
```

練習

【答え】script.py

```
from food import Food
from drink import Drink

# Food() に引数を追加してください
food1 = Food('サンドイッチ', 500,330)

print(food1.info())

drink1 = Drink('コーヒー', 300)
drink1.amount = 180
print(drink1.info())
```

練習

【答え】menu_item.py

```
class MenuItem:
    def __init__(self, name, price):
        self.name = name
        self.price = price

    def info(self):
        return self.name + ': ¥' + str(self.price)

    def get_total_price(self, count):
        total_price = self.price * count

        if count >= 3:
            total_price *= 0.9

        return round(total_price)
```

練習

【答え】food.py

```
from menu_item import MenuItem

class Food(MenuItem):
    # __init__ メソッドを定義してください
    def __init__(self,name,price,calorie):
        self.name = name
        self.price = price
        self.calorie = calorie

    def info(self):
        return self.name + ': ¥' + str(self.price) + ' ('
+ str(self.calorie) + 'kcal)'

    def calorie_info(self):
        print(str(self.calorie) + 'kcalです')
```

練習

【答え】drink.py

```
from menu_item import MenuItem

class Drink(MenuItem):
    def info(self):
        return self.name + ': ¥' + str(self.price) + ' ('
+ str(self.amount) + 'mL)'
```

メソッド内の重複

Foodクラスで__init__メソッドをオーバーライドしたが、nameとpriceについては共通の処理で、MenuItemクラスの__init__メソッドにも含まれている。このメソッド内の重複をまとめる方法を学んでみよう。

menu_item.py

```
class MenuItem:
    def __init__(self, name, price):
        self.name = name
        self.price = price
```

food.py

```
class Food(MenuItem):
    def __init__(self, name, price, calorie):
        self.name = name
        self.price = price
        self.calorie = calorie
```

MenuItemクラスの__init__メソッドでも同じ内容のことをしている！

super()

オーバーライドしたメソッドの中で「super()」とすることで、親クラスを呼び出すことができる。また、図のように「super().メソッド名()」とすることで、親クラス内に定義されたインスタンスメソッドをそのまま利用することができる。

menu_item.py

```
class MenuItem:
    def __init__(self, name, price):
        self.name = name
        self.price = price
```

food.py

```
class Food(MenuItem):
    def __init__(self, name, price, calorie):
        super( ).__init__(name, price)
        親クラスの__init__メソッドを利用
        self.calorie = calorie
```

練習

【問題】script.py

```
from food import Food
from drink import Drink

food1 = Food('サンドイッチ', 500, 330)
print(food1.info())

drink1 = Drink('コーヒー', 300)
drink1.amount = 180
print(drink1.info())
```

練習

【問題】menu_item.py

```
class MenuItem:
    def __init__(self, name, price):
        self.name = name
        self.price = price

    def info(self):
        return self.name + ': ¥' + str(self.price)

    def get_total_price(self, count):
        total_price = self.price * count

        if count >= 3:
            total_price *= 0.9

        return round(total_price)
```

練習

【問題】food.py

```
from menu_item import MenuItem

class Food(MenuItem):
    def __init__(self, name, price, calorie):
        # super() を用いて、親クラスの __init__() を呼び出してください


        # 以下の2行をコメントアウトして削除してください


        self.calorie = calorie

    def info(self):
        return self.name + ': ¥' + str(self.price) + ' (' + str(self.calorie)
+ 'kcal)'

    def calorie_info(self):
        print(str(self.calorie) + 'kcalです')
```

練習

【問題】drink.py

```
from menu_item import MenuItem

class Drink(MenuItem):
    def info(self):
        return self.name + ': ¥' + str(self.price) + ' ('
+ str(self.amount) + 'mL)'
```

練習

【答え】script.py

```
from food import Food
from drink import Drink

food1 = Food('サンドイッチ', 500, 330)
print(food1.info())

drink1 = Drink('コーヒー', 300)
drink1.amount = 180
print(drink1.info())
```

練習

【答え】menu_item.py

```
class MenuItem:
    def __init__(self, name, price):
        self.name = name
        self.price = price

    def info(self):
        return self.name + ': ¥' + str(self.price)

    def get_total_price(self, count):
        total_price = self.price * count

        if count >= 3:
            total_price *= 0.9

        return round(total_price)
```

練習

【答え】food.py

```
from menu_item import MenuItem

class Food(MenuItem):
    def __init__(self, name, price, calorie):
        # super() を用いて、親クラスの __init__() を呼び出してください
        super().__init__(name, price)

        # 以下の2行をコメントアウトして削除してください
        # self.name = name
        # self.price = price

        self.calorie = calorie

    def info(self):
        return self.name + ': ¥' + str(self.price) + ' (' + str(self.calorie)
+ 'kcal)'

    def calorie_info(self):
        print(str(self.calorie) + 'kcalです')
```

練習

【答え】drink.py

```
from menu_item import MenuItem

class Drink(MenuItem):
    def info(self):
        return self.name + ': ¥' + str(self.price) + ' ('
+ str(self.amount) + 'mL)'
```

Drinkクラスの__init__メソッド

super()を用いると同じコードを何回も書かなくてすむ。Drinkクラスでも同じように、__init__メソッドをオーバーライドしてみよう。

```
サンドイッチ: ¥500 (330kcal)

コーヒー: ¥300 (180mL)
```

練習

【問題】script.py

```
from food import Food
from drink import Drink

food1 = Food('サンドイッチ', 500, 330)
print(food1.info())

# Drink() に引数を追加してください


# 以下の1行はコメントアウトして削除してください


print(drink1.info())
```

練習

【問題】menu_item.py

```
class MenuItem:
    def __init__(self, name, price):
        self.name = name
        self.price = price

    def info(self):
        return self.name + ': ¥' + str(self.price)

    def get_total_price(self, count):
        total_price = self.price * count

        if count >= 3:
            total_price *= 0.9

        return round(total_price)
```

練習

【問題】food.py

```
from menu_item import MenuItem

class Food(MenuItem):
    def __init__(self, name, price, calorie):
        super().__init__(name, price)
        self.calorie = calorie

    def info(self):
        return self.name + ': ¥' + str(self.price) + ' ('
+ str(self.calorie) + 'kcal)'

    def calorie_info(self):
        print(str(self.calorie) + 'kcalです')
```

練習

【問題】drink.py

```
from menu_item import MenuItem

class Drink(MenuItem):
    # __init__ メソッドを定義してください


    def info(self):
        return self.name + ': ¥' + str(self.price) + ' ('
+ str(self.amount) + 'mL)'
```

練習

【答え】script.py

```
from food import Food
from drink import Drink

food1 = Food('サンドイッチ', 500, 330)
print(food1.info())

# Drink() に引数を追加してください
drink1 = Drink('コーヒー', 300, 180)

# 以下の1行はコメントアウトして削除してください
# drink1.amount = 180

print(drink1.info())
```

練習

【答え】menu_item.py

```
class MenuItem:
    def __init__(self, name, price):
        self.name = name
        self.price = price

    def info(self):
        return self.name + ': ¥' + str(self.price)

    def get_total_price(self, count):
        total_price = self.price * count

        if count >= 3:
            total_price *= 0.9

        return round(total_price)
```

練習

【答え】food.py

```
from menu_item import MenuItem

class Food(MenuItem):
    def __init__(self, name, price, calorie):
        super().__init__(name, price)
        self.calorie = calorie

    def info(self):
        return self.name + ': ¥' + str(self.price) + ' ('
+ str(self.calorie) + 'kcal)'

    def calorie_info(self):
        print(str(self.calorie) + 'kcalです')
```

練習

【答え】drink.py

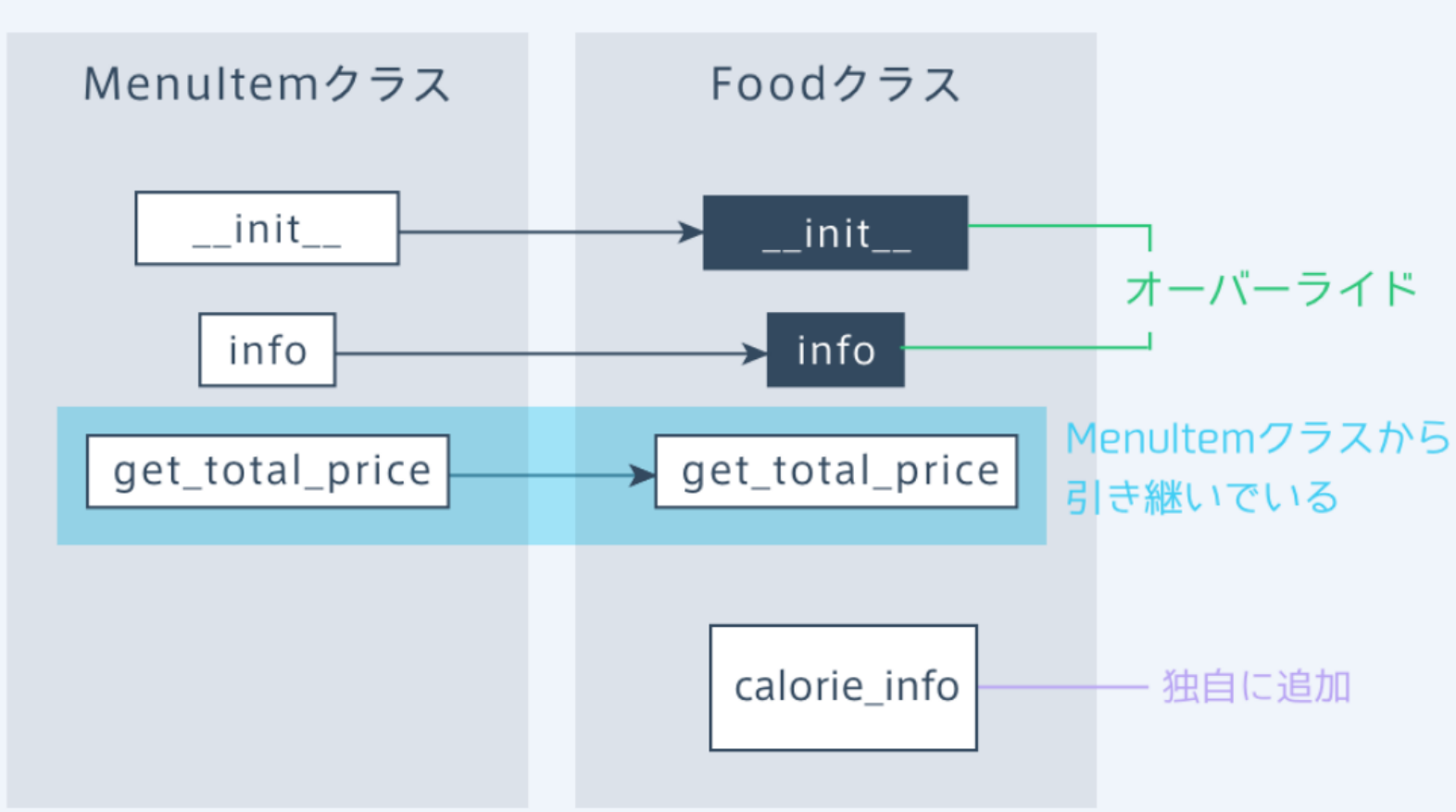
```
from menu_item import MenuItem

class Drink(MenuItem):
    # __init__ メソッドを定義してください
    def __init__(self, name, price, amount):
        super().__init__(name, price)
        self.amount = amount

    def info(self):
        return self.name + ': ¥' + str(self.price) + ' ('
+ str(self.amount) + 'mL)'
```

料理注文システムを完成させよう

ここまでの知識を使って料理注文システムを完成させよう。コンソールに入力された数値を用いて、合計金額を出力できるようにする。MenuItemクラスに定義した「get_total_price」メソッドは、子クラスであるFoodクラスやDrinkクラスのインスタンスでも使うことができる。



練習

【問題】script.py

```
from food import Food
from drink import Drink

food1 = Food('サンドイッチ', 500, 330)
food2 = Food('チョコケーキ', 400, 450)
food3 = Food('シュークリーム', 200, 180)

foods = [food1, food2, food3]

drink1 = Drink('コーヒー', 300, 180)
drink2 = Drink('オレンジジュース', 200, 350)
drink3 = Drink('エスプレッソ', 300, 30)

drinks = [drink1, drink2, drink3]

print('食べ物メニュー')
index = 0
```

次のページに続く

練習

```
for food in foods:
    print(str(index) + '. ' + food.info())
    index = index + 1

print('飲み物メニュー')
index = 0
for drink in drinks:
    print(str(index) + '. ' + drink.info())
    index = index + 1

print('--------------------')

food_order = int(input('食べ物の番号を選択してください: '))
selected_food = foods[food_order]

drink_order = int(input('飲み物の番号を選択してください: '))
selected_drink = drinks[drink_order]
```

次のページに続く

練習

コンソールから入力を受け取り、変数 count に代入してください

selected_food と selected_drink のそれぞれに対して、get_total_price メソッドを呼び出してください

「 合計は〇〇円です 」となるように出力してください

練習

【問題】menu_item.py

```
class MenuItem:
    def __init__(self, name, price):
        self.name = name
        self.price = price

    def info(self):
        return self.name + ': ¥' + str(self.price)

    def get_total_price(self, count):
        total_price = self.price * count

        if count >= 3:
            total_price *= 0.9

        return round(total_price)
```

練習

【問題】food.py

```
from menu_item import MenuItem

class Food(MenuItem):
    def __init__(self, name, price, calorie):
        super().__init__(name, price)
        self.calorie = calorie

    def info(self):
        return self.name + ': ¥' + str(self.price) + ' ('
+ str(self.calorie) + 'kcal)'

    def calorie_info(self):
        print(str(self.calorie) + 'kcalです')
```

練習

【問題】drink.py

```
from menu_item import MenuItem

class Drink(MenuItem):
    def __init__(self, name, price, amount):
        super().__init__(name, price)
        self.amount = amount

    def info(self):
        return self.name + ': ¥' + str(self.price) + ' ('
+ str(self.amount) + 'mL)'
```

練習

【答え】script.py

```
from food import Food
from drink import Drink

food1 = Food('サンドイッチ', 500, 330)
food2 = Food('チョコケーキ', 400, 450)
food3 = Food('シュークリーム', 200, 180)

foods = [food1, food2, food3]

drink1 = Drink('コーヒー', 300, 180)
drink2 = Drink('オレンジジュース', 200, 350)
drink3 = Drink('エスプレッソ', 300, 30)

drinks = [drink1, drink2, drink3]

print('食べ物メニュー')
index = 0
```

次のページに続く

練習

```
for food in foods:
    print(str(index) + '. ' + food.info())
    index = index + 1

print('飲み物メニュー')
index = 0
for drink in drinks:
    print(str(index) + '. ' + drink.info())
    index = index + 1

print('--------------------')

food_order = int(input('食べ物の番号を選択してください: '))
selected_food = foods[food_order]

drink_order = int(input('飲み物の番号を選択してください: '))
selected_drink = drinks[drink_order]
```

次のページに続く

練習

```
# コンソールから入力を受け取り、変数 count に代入してください
count = int(input('何セット買いますか？(3つ以上で1割引): '))

# selected_food と selected_drink のそれぞれに対して、get_total_price メソッドを呼び出してください
result = selected_food.get_total_price(count) + selected_drink.get_total_price(count)

# 「 合計は〇〇円です 」となるように出力してください
print('合計は' + str(result) + '円です')
```

練習

【答え】menu_item.py

```
class MenuItem:
    def __init__(self, name, price):
        self.name = name
        self.price = price

    def info(self):
        return self.name + ': ¥' + str(self.price)

    def get_total_price(self, count):
        total_price = self.price * count

        if count >= 3:
            total_price *= 0.9

        return round(total_price)
```

練習

【答え】food.py

```
from menu_item import MenuItem

class Food(MenuItem):
    def __init__(self, name, price, calorie):
        super().__init__(name, price)
        self.calorie = calorie

    def info(self):
        return self.name + ': ¥' + str(self.price) + ' ('
+ str(self.calorie) + 'kcal)'

    def calorie_info(self):
        print(str(self.calorie) + 'kcalです')
```

練習

【答え】drink.py

```
from menu_item import MenuItem

class Drink(MenuItem):
    def __init__(self, name, price, amount):
        super().__init__(name, price)
        self.amount = amount

    def info(self):
        return self.name + ': ¥' + str(self.price) + ' ('
+ str(self.amount) + 'mL)'
```

復習＆チャレンジ

ここまで習ったことをScratchでもできるかチャレンジしてみよう。
その過程でScratchでできること、Pythonでないとできないことを整理してみよう。
例えば、データの型という概念はscratchにはあっただろうか？

メモ

プログラミング教室のテクノロ

なまえ：