9.1 – Restaurante: Crie uma classe chamada `Restaurant`. O método `__init__()` de `Restaurant` deve armazenar dois atributos: `restaurant_name` e `cuisine_type`. Crie um método chamado `describe_restaurant()` que mostre essas duas informações, e um método de nome `open_restaurant()` que exiba uma mensagem informando que o restaurante está aberto.

Crie uma instância chamada `restaurant` a partir de sua classe. Mostre os dois atributos individualmente e, em seguida, chame os dois métodos.

9.2 – Três restaurantes: Comece com a classe do Exercício 9.1. Crie três instâncias diferentes da classe e chame `describe_restaurant()` para cada instância.

9.3 – Usuários: Crie uma classe chamada `User`. Crie dois atributos de nomes `first_name` e `last_name` e, então, crie vários outros atributos normalmente armazenados em um perfil de usuário. Escreva um método de nome `describe_user()` que apresente um resumo das informações do usuário. Escreva outro método chamado `greet_user()` que mostre uma saudação personalizada ao usuário.

Crie várias instâncias que representem diferentes usuários e chame os dois métodos para cada usuário.

9.4 – Pessoas atendidas: Comece com seu programa do Exercício 9.1 (página 225). Acrescente um atributo chamado `number_served` cujo valor default é 0. Crie uma instância chamada `restaurant` a partir dessa classe. Apresente o número de clientes atendidos pelo restaurante e, em seguida, mude esse valor e exiba-o novamente.

Adicione um método chamado `set_number_served()` que permita definir o número de clientes atendidos. Chame esse método com um novo número e mostre o valor novamente.

Acrescente um método chamado `increment_number_served()` que permita incrementar o número de clientes servidos. Chame esse método com qualquer número que você quiser e que represente quantos clientes foram atendidos, por exemplo, em um dia de funcionamento.

9.5 – Tentativas de login: Acrescente um atributo chamado `login_attempts` à sua classe `User` do Exercício 9.3 (página 226). Escreva um método chamado `increment_login_attempts()` que incremente o valor de `login_attempts` em 1. Escreva outro método chamado `reset_login_attempts()` que reinicie o valor de `login_attempts` com 0.

Crie uma instância da classe `User` e chame `increment_login_attempts()` várias vezes. Exiba o valor de `login_attempts` para garantir que ele foi incrementado de forma apropriada e, em seguida, chame `reset_login_attempts()`. Exiba `login_attempts` novamente para garantir que seu valor foi reiniciado com 0.

9.6 – Sorveteria: Uma sorveteria é um tipo específico de restaurante. Escreva uma classe chamada `IceCreamStand` que herde da classe `Restaurant` escrita no Exercício 9.1 (página 225) ou no Exercício 9.4 (página 232). Qualquer versão da classe funcionará; basta escolher aquela de que você mais gosta. Adicione um atributo chamado `flavors` que armazene uma lista de sabores de sorvete. Escreva um método para mostrar esses sabores. Crie uma instância de `IceCreamStand` e chame esse método.

9.7 – Admin: Um administrador é um tipo especial de usuário. Escreva uma classe chamada `Admin` que herde da classe `User` escrita no Exercício 9.3 (página 226), ou no Exercício 9.5 (página 232). Adicione um atributo `privileges` que armazene uma lista de strings como `"can add post"`, `"can delete post"` `"can ban user"`, e assim por diante. Escreva um método chamado `show_privileges()` que liste o conjunto de privilégios de um administrador. Crie uma instância de `Admin` e chame seu método.

9.8 – Privilégios: Escreva uma classe `Privileges` separada. A classe deve ter um atributo `privileges` que armazene uma lista de strings conforme descrita no Exercício 9.7. Transfira o método `show_privileges()` para essa classe. Crie uma instância de `Privileges` como um atributo da classe `Admin`. Crie uma nova instância de `Admin` e use seu método para exibir os privilégios.

9.9 – Upgrade de bateria: Use a última versão de *electric_car.py* desta seção. Acrescente um método chamado `upgrade_battery()` na classe `Battery`. Esse método deve verificar a capacidade da bateria e definir-la com 85 se o valor for diferente. Crie um carro elétrico com uma capacidade de bateria default, chame `get_range()` uma vez e, em seguida, chame `get_range()` uma segunda vez após fazer um upgrade da bateria. Você deverá ver um aumento na distância que o carro é capaz de percorrer.

9.10 – Importando Restaurant: Usando sua classe `Restaurant` mais recente, armazene-a em um módulo. Crie um arquivo separado que importe `Restaurant`. Crie uma instância de `Restaurant` e chame um de seus métodos para mostrar que a instrução `import` funciona de forma apropriada.

9.11 – Importando Admin: Comece com seu programa do Exercício 9.8 (página 241). Armazene as classes `User`, `Privileges` e `Admin` em um módulo. Crie um arquivo separado e uma instância de `Admin` e chame `show_privileges()` para mostrar que tudo está funcionando de forma apropriada.

9.12 – Vários módulos: Armazene a classe `User` em um módulo e as classes `Privileges` e `Admin` em um módulo separado. Em outro arquivo, crie uma instância de `Admin` e chame `show_privileges()` para mostrar que tudo continua funcionando de forma apropriada.

9.13 – Reescrevendo o programa com OrderedDict: Comece com o Exercício 6.4 (página 155), em que usamos um dicionário-padrão para representar um glossário. Reescreva o programa usando a classe `OrderedDict` e certifique-se de que a ordem da saída coincida com a ordem em que os pares chave-valor foram adicionados ao dicionário.

9.14 – Dados: O módulo `random` contém funções que geram números aleatórios de várias maneiras. A função `randint()` devolve um inteiro no intervalo especificado por você. O código a seguir devolve um número entre 1 e 6:

```
from random import randint
x = randint(1, 6)
```

Crie uma classe `Die` com um atributo chamado `sides`, cujo valor default é 6. Escreva um método chamado `roll_die()` que exiba um número aleatório entre 1 e

o número de lados do dado. Crie um dado de seis dados e lance-o dez vezes.

Crie um dado de dez lados e outro de vinte lados. Lance cada dado dez vezes.

9.15 – Módulo Python da semana: Um excelente recurso para explorar a biblioteca-padrão de Python é um site chamado *Python Module of the Week* (Módulo Python da semana). Acesse *http://pymotw.com/* e observe a tabela de conteúdo. Encontre um módulo que pareça ser interessante e leia a sua descrição ou explore a documentação dos módulos `collections` e `random`.