Material Complementar Classes

Classes proporcionam uma forma de organizar dados e funcionalidades juntos. Criar uma nova classe cria um novo "tipo" de objeto, permitindo que novas "instâncias" desse tipo sejam produzidas. Cada instância da classe pode ter atributos anexados a ela, para manter seu estado. Instâncias da classe também podem ter métodos (definidos pela classe) para modificar seu estado.

A forma mais simples de definir uma classe é:

```
class ClassName:
    <statement-1>
    .
    .
    .
    <statement-N>
```

Definições de classe, assim como definições de função (instruções def), precisam ser executadas antes que tenham qualquer efeito. (Você pode colocar uma definição de classe dentro do teste condicional de um if ou dentro de uma função.)

Na prática, as instruções dentro da definição de classe geralmente serão definições de funções, mas outras instruções são permitidas, e às vezes são bem úteis — voltaremos a este tema depois. Definições de funções dentro da classe normalmente têm uma forma peculiar de lista de argumentos, determinada pela convenção de chamada a métodos — isso também será explicado mais tarde.

Objetos classe suportam dois tipos de operações: referências a atributos e instanciação.

Referências a atributos de classe utilizam a sintaxe padrão utilizada para quaisquer referências a atributos em Python: obj.nome. Nomes de

atributos válidos são todos os nomes presentes dentro do espaço de nomes da classe, quando o objeto classe foi criado. Portanto, se a definição de classe tem esta forma:

```
class MyClass:
    """A simple example class"""
    i = 12345

    def f(self):
        return 'hello world'
```

então MyClass.i e MyClass.f são referências a atributo válidas, retornando, respectivamente, um inteiro e um objeto função. Atributos de classe podem receber valores, pode-se modificar o valor de MyClass.i num atribuição. __doc__ também é um atributo válido da classe, retornando a docstring associada à classe: "A simple example class".

Para instanciar uma classe, usa-se a mesma sintaxe de invocar uma função. Apenas finja que o objeto classe do exemplo é uma função sem parâmetros, que devolve uma nova instância da classe. Por exemplo (assumindo a classe acima):

```
x = MyClass()
```

cria uma nova instância da classe e atribui o objeto resultante à variável local x.

A operação de instanciação (“invocar” um objeto classe) cria um objeto vazio. Muitas classes preferem criar novos objetos com um estado inicial predeterminado. Para tanto, a classe pode definir um método especial chamado __init__(), assim:

```
def __init__(self):
    self.data = []
```

Quando uma classe define um método __init__(), o processo de instanciação automaticamente invoca __init__() sobre a instância recém criada. Em nosso exemplo, uma nova instância já inicializada pode ser obtida desta maneira:

```
x = MyClass()
```

Naturalmente, o método __init__() pode ter parâmetros para maior flexibilidade. Neste caso, os argumentos fornecidos na invocação da classe serão passados para o método __init__(). Por exemplo,

```
>>>
class Complex:
    def __init__(self, realpart, imagpart):
        self.r = realpart
        self.i = imagpart

x = Complex(3.0, -4.5)
x.r, x.i
(3.0, -4.5)
```

EXERCÍCIOS

1. **Classe Bola:** Crie uma classe que modele uma bola:
    1. Atributos: Cor, circunferência, material
    2. Métodos: trocaCor e mostraCor
2. **Classe Quadrado:** Crie uma classe que modele um quadrado:
    1. Atributos: Tamanho do lado
    2. Métodos: Mudar valor do Lado, Retornar valor do Lado e calcular Área;
3. **Classe Retangulo:** Crie uma classe que modele um retangulo:
    1. Atributos: LadoA, LadoB (ou Comprimento e Largura, ou Base e Altura, a escolher)

    2. Métodos: Mudar valor dos lados, Retornar valor dos lados, calcular Área e calcular Perímetro;
    3. Crie um programa que utilize esta classe. Ele deve pedir ao usuário que informe as medidades de um local. Depois, deve criar um objeto com as medidas e calcular a quantidade de pisos e de rodapés necessárias para o local.
4. **Classe Pessoa:** Crie uma classe que modele uma pessoa:
    1. Atributos: nome, idade, peso e altura
    2. Métodos: Envelhercer, engordar, emagrecer, crescer. Obs: Por padrão, a cada ano que nossa pessoa envelhece, sendo a idade dela menor que 21 anos, ela deve crescer 0,5 cm.
5. **Classe Conta Corrente:** Crie uma classe para implementar uma conta corrente. A classe deve possuir os seguintes atributos: número da conta, nome do correntista e saldo. Os métodos são os seguintes: alterarNome, depósito e saque; No construtor, saldo é opcional, com valor default zero e os demais atributos são obrigatórios.

fonte: https://docs.python.org/pt-br/3/tutorial/classes.html