**Definindo o problema**

Vamos adotar como problema inicial a relação *perímetro/área* de uma geometria. Se adotarmos o paralelepípedo como geometria teríamos como possíveis parâmetros, sua largura, altura e sua profundidade.

Cada parâmetro – largura, altura e profundidade – deve possuir um *nome* e um *valor*. Este valor, por sua vez, deve ser percorrido dentro de um *valor mínimo* e um *valor máximo*. (o *slide* do *Grasshopper*)

Fazendo analogia com um organismo vivo, o paralelepípedo seria um individuo constituído por um conjunto de genes – altura, largura e profundidade – que formaria seu DNA, ou seja, o composto orgânico cujas informações contêm as instruções genéticas que coordenam o desenvolvimento e funcionamento de sua forma.

Todo problema de otimização precisa ter um função-objetivo. A função-objetivo é o objeto da otimização. Para este nosso problema, a função-objetivo é a relação perímetro/área. Poderia ser qualquer coisa ou, como dizem nesta área, poderia ser uma "caixa preta" onde sabemos apenas o formato das entradas e nos retorna um valor que queremos otimizar. A grande vantagem dos algoritmos genéticos esta no fato de não precisarmos saber como funciona esta função objetivo, apenas tê-la disponível para ser aplicada aos indivíduos e comparar os resultados.

Definido a função-objetivo, precisaremos definir se queremos maximizar ou minimizar nossa função-objetivo.

[...]

**1 - Criando a classe produto**

Segundo a documentação do Python[[1]](#footnote-1), uma classe associa dados (atributos) e operações (métodos) numa só estrutura. Um objeto é uma instância de uma classe. Classes proporcionam uma forma de organizar dados e funcionalidades juntas. 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.

[001]**class Parametro**():

[002] **def \_\_init\_\_**(self, nome, minimo, maximo):

[003] self.nome = nome

[004] self.minimo = minimo

[005] self.maximo = maximo

[007]**if** Toggle:

[008] lista\_parametros = []

[009] lista\_parametros.**append**(**Parametro**('Largura', 0.00, 11.00))

[010] lista\_parametros.**append**(**Parametro**('Altura', 2.00, 22.00))

[011] lista\_parametros.**append**(**Parametro**('Profundidade', 2.00, 22.00))

[012] **for** parametro **in** lista\_parametros:

[013] **print**(parametro.nome)

Em [001] definimos uma classe chamada **Parametro**. Por convenção, nomes com a primeira letra maiúscula referem-se a classes em Python. Os parênteses na definição da classe estão vazios porque estamos criando essa classe do zero. Em v escrevemos uma **docstring** que descreve o que essa classe faz.

**1.1 - Método \_\_init\_\_()**

Uma função que faz parte de uma classe é um método. O método **\_\_init\_\_()** em [002] é um método especial que Python executa automaticamente sempre que criamos uma nova instância baseada na classe **Parametro**. Esse método tem dois *underscores* no início e dois no final – uma convenção que ajuda a evitar que os nomes *default* de métodos Python entrem em conflito com nomes de métodos criados pelo usuário.

Definimos o método **\_\_init\_\_()** para que tenha três parâmetros: **self**, **maximo** e **minimo**. O parâmetro self é obrigatório na definição do método e deve estar antes dos demais parâmetros. Deve estar incluído na definição, pois, quando Python chama esse método **\_\_init\_\_()** depois (para criar uma instância de **Parametro**), a chamada do método passará o argumento **self** automaticamente.

Toda chamada de método associada a uma classe passa **self**, que é uma referência à própria instância, de modo automático; ele dá acesso aos atributos e métodos da classe à instância individual. Quando criamos uma instância de **Parametro**, Python chamará o método **\_\_init\_\_()** da classe **Parametro**.

Passaremos um valor mínimo e um máximo como argumentos para **Parametro()**; **self** é passado automaticamente, portanto não é preciso especificá-lo.

Sempre que quisermos criar uma instância da classe **Parametro** forneceremos valores apenas para os três últimos parâmetros, que são **nome**, **minimo** e **maximo**.

As três variáveis definidas em [003], [004] e [005] têm o prefixo **self**. Qualquer variável prefixada com **self** está disponível a todos os métodos da classe; além disso, podemos acessar essas variáveis por meio de qualquer instância criada a partir da classe. **self.nome = nome** usa o valor armazenado no parâmetro nome e o armazena na variável nome, que é então associada à instância criada. O mesmo processo ocorre com **self.minimo = minimo** e **self.maximo = maximo**. Variáveis como essas, acessíveis por meio de instâncias, são chamadas de atributos.

Em [007] definimos um ***main*** e, caso ***Toggle*** seja ***True***, significa que qualquer comando que esteja depois desta linha e clicarmos em executar, ele será executado.

|  |
| --- |
| Esta linha é o equivalente de:  **if \_\_name\_\_ == '\_\_main\_\_':**  Não há uma função **main()** em Python, pelo menos não da forma explícita. O que temos é o '**\_\_main\_\_**' que é o nome do escopo do código executado em top-level, sendo que é definido em **\_\_name\_\_** o nome do modulo. O '**\_\_main\_\_**' é definido quando o script é executado do terminal interativo ou chamado pelo interpretador da linguagem. |

**1.2 Criando uma instância a partir de uma classe**

Pense em uma classe como um conjunto de instruções para criar uma instância. A classe **Parametro** é um conjunto de instruções que diz a Python como criar instâncias individuais que representem parâmetros específicos.

Na linha [009], [010] e [011] criamos instancias que representam os parâmetros específicos e os adicionamos a lista **lista\_parametros** [008].

**1.3 Acessando atributos**

Para acessar os atributos de uma instância utilizamos a notação de ponto. Em [013] acessamos o valor do atributo nome de **parametro** escrevendo: **parametro.nome**.

A notação de ponto é usada com frequência em Python. Essa sintaxe mostra como Python encontra o valor de um atributo. Nesse caso, o interpretador olha para a instância criada e encontra o atributo nome associado a ela. É o mesmo atributo referenciado como **self.nome** na classe **Parametro**. Podemos usar a mesma abordagem para trabalhar com o atributo **minimo** e **maximo**.

**2 – Classe individuo I**

1. <https://docs.python.org/pt-br/3/tutorial/classes.html> [↑](#footnote-ref-1)