Paradigmas de programação

Existem várias formas de pensar e organizar o código de um programa, cada uma com seus objetivos, vantagens e desvantagens. Normalmente, o uso de uma dessas abordagens depende do suporte da linguagem de programação à ferramentas ou técnicas específicas para permitir sua aplicação. Na literatura, essas diferentes formas são comumente denominadas paradigmas de programação.

O paradigma de programação mais "natural" para o computador é a chamada programação imperativa: o programa é uma série de instruções, uma embaixo da outra, sempre sendo executadas sequencialmente. O nome vem da ideia de que o programa pode ser descrito por verbos no modo imperativo.

Dentro da programação imperativa temos diversas outras estratégias, todas baseadas na ideia de um programa como uma coleção de instruções, mas dando diferentes abordagens.

Uma primeira tentativa de melhorar a programação imperativa foi a introdução de alguns desvios controlados de fluxo, como estruturas condicionais e malhas de repetição. Chamamos essa forma de programação imperativa de programação estruturada, e provavelmente foi a primeira forma de programação que você estudou.

O programa abaixo é um programa estruturado. Ele é constituído por um único bloco de instruções que são executadas sequencialmente. Apesar de ser possível pular uma ou outra instrução (através de if/else) ou repetir instruções dentro de um loop, a tendência geral do programa é sempre seguir sequencialmente, e uma vez encerrado um condicional ou loop, não retornamos mais para ele.

nota1 = float(input('Digite a primeira nota: '))

nota2 = float(input('Digite a segunda nota: '))

media = (nota1 + nota2)/2

if media >= 6:

print('Aprovado')

else:

print('Reprovado')

Outra forma de programação imperativa já estudada nesse curso é a programação procedural. Ela introduz um certo grau de modularização do programa.

Ao invés do programa ser um bloco único de instruções, nós podemos subdividir nossa lógica em "subprogramas": as funções. Uma função possui um nome, pode possuir dados de entrada (parâmetros) e de saída (retorno) e pode ser reutilizada múltiplas vezes.

Isso evita a repetição desnecessária de código, facilita o reaproveitamento de código e torna o processo de atualização e correção de bugs mais simples e seguro. O programa abaixo é procedural:

def soma(n1, n2):

return n1+n2

def subtracao(n1, n2):

return n1-n2

def multiplicacao(n1, n2):

return n1\*n2

def divisao(n1, n2):

return n1/n2

def obter\_numeros():

n1 = int(input('Digite um número: '))

n2 = int(input('Digite outro número: '))

return n1, n2

def menu():

opcao = 1

while opcao != 0:

print('1. Somar dois números')

print('2. Subtrair dois números')

print('3. Multiplicar dois números')

print('4. Dividir dois números')

print('0. Sair')

opcao = int(input('Digite a sua opcao: '))

if opcao == 1:

num1, num2 = obter\_numeros()

print(soma(num1, num2))

elif opcao == 2:

num1, num2 = obter\_numeros()

print(subtracao(num1, num2))

elif opcao == 3:

num1, num2 = obter\_numeros()

print(multiplicacao(num1, num2))

elif opcao == 4:

num1, num2 = obter\_numeros()

print(divisao(num1, num2))

elif opcao != 0:

print('Opção inválida!')

menu()

Nem todas as formas de programação são imperativas. Uma característica comum a todos os paradigmas derivados do imperativo é o foco em como o computador deve realizar uma tarefa. Existe uma forma bastante diferente de programar conhecida como programação declarativa. Os paradigmas declarativos colocam menos ênfase em como realizar uma tarefa e mais ênfase no resultado a ser obtido. Programas declarativos, como o nome sugere, são formados por declarações, não por instruções.

Uma forma de programação declarativa é a programação funcional, que tem como objetivo reduzir os efeitos colaterais de cada trecho de código e gerar um programa mais limpo, legível e determinístico, ou seja, previsível. Assim como na programação procedural, o programa será modularizado em funções. Porém, ao invés de técnicas de programação estruturada, como malhas de repetição, definimos nossas funções em termos de operações básicas e encadeamento de chamadas a outras funções, transformando nosso programa em uma árvore de expressões retornando valores.

Outra forma de programação declarativa é a programação lógica, onde o programador irá declarar uma série verdades lógicas, como relações entre elementos, e a partir daí o programa será capaz de inferir novas relações e verdades.

É possível que um mesmo programa incorpore conceitos ou técnicas de diferentes paradigmas. Além disso, linguagens de programação precisam oferecer recursos relacionados a cada um dos paradigmas para que seja possível utilizá-la daquela maneira.

O Python é uma linguagem multiparadigma. Já vimos exemplos procedurais em Python, ele possui suporte a vários recursos importantes da programação funcional e, como estudaremos ao longo desse módulo, ele apresenta suporte à programação orientada a objeto.

A programação orientada a objeto também é uma forma de programação imperativa: ainda teremos blocos de instruções. Porém, iremos aprofundar e melhorar a nossa modularização.

Ela foca em modelar o mundo real, ou seja, representar objetos do mundo real através de código de computação. Ao invés de apenas darmos ordens para o computador, iremos descrever entidades, mapeando suas características, habilidades e interações. O programa irá emergir das interações entre esses modelos.

Ao realizarmos uma boa modelagem orientada a objeto, podemos não apenas resolver um problema, mas tornar a solução fácil de ser adotada para resolver outros problemas ou ser incorporada a outros projetos. Por exemplo, uma boa modelagem dos comportamentos e operações envolvendo datas e horários permite que outros programadores possam apenas importar essa modelagem e utilizá-la, sem precisar saber como exatamente fazer operações como determinar se um ano é bissexto ou calcular "na mão" a diferença entre duas datas. Essa modelagem em particular já está pronta no módulo datetime do Python. Aprenderemos a criar nossos próprios módulos para condensar toda a lógica envolvendo algum objeto do mundo real e facilitar a nossa própria vida ou a de outros programadores.

O estudo dos paradigmas de programação agrega muito às habilidades de um programador. Conhecendo melhor as características e objetivos de cada forma de programação, seremos capazes de compreender e tirar melhor proveito das ferramentas oferecidas pelas linguagens.

Além disso, ele acelera bastante o aprendizado de novas linguagens. Você já sabe programar de maneira procedural em Python. Ao ter o primeiro contato com qualquer outra linguagem procedural, você já sabe que ela provavelmente oferecerá uma forma de criar variáveis, algum tipo de estrutura condicional, algum tipo de malha de repetição e uma forma de criar e chamar funções. É possível aprender todo esse básico em poucas horas, e depois concentrar os esforços apenas nos recursos particulares daquela linguagem.

Caso deseje se aventurar por essa área, um ótimo livro de referência é o Conceitos de Linguagens de Programação, por Robert W. Sebesta. Você pode encontrar a 11ª edição (2018) em português pela editora Bookman.