PROGRAMAÇÃO POR OBJETOS

Tópico 7
Versatilidade na programação em Python, usando bibliotecas

Leonel Morgado, Universidade Aberta, 2024

As bibliotecas, essa porta para o mundo

No vasto universo da programação, não iríamos longe se tivéssemos de programar de raiz todas as funcionalidades de que precisássemos. As bibliotecas são o que permite aos programadores partilhar algoritmos e funcionalidades, para construir novas soluções. Atuam, por isso, como catalisadores de eficiência e inovação. Cada biblioteca é uma coleção organizada de componentes, como classes, funções e outros elementos, prontos a usar. Esses componentes são, nas bibliotecas de programação, como os livros nas bibliotecas físicas. Através deles podemos estender as nossas capacidades de programação para novas fronteiras, mais ambiciosas.

Em concreto, as bibliotecas em Python permitem-nos integrar o código desenvolvido por outros programadores nos nossos próprios projetos. Este processo ocorre através da importação da biblioteca no nosso código. Ao usarmos o comando `import` seguido do nome da biblioteca,

estamos a criar uma ponte entre o nosso código e o conjunto de classes e métodos definidos naquela biblioteca. Este simples ato de importação torna acessíveis todos os elementos da biblioteca. Podemos, assim, chamar as suas funções e utilizar as suas classes como se fossem parte do nosso próprio código-fonte.

Em Python temos bibliotecários digitais para nos ajudar

Em linguagens mais antigas, este processo consistia em obter os ficheiros de código-fonte das bibliotecas ou a sua versão já compilada, geralmente em formatos como .lib ou .dll. Depois era necessário colocar esses ficheiros em diretórios específicos, muitas vezes dentro da estrutura do projeto, outras vezes em pastas específicas do sistema operativo. Em seguida, era necessário configurar o ambiente de programação para usar as bibliotecas, o que podia incluir a criação ou atualização de variáveis de ambiente, para assegurar que o compilador e o ambiente de execução pudessem localizar esses ficheiros das bibliotecas e vincular a eles o código do nosso projeto. Em linguagens mais modernas como o Python, o Perl ou o R, tira-se partido da existência da Internet e da possibilidade de haver serviços permanentemente disponíveis, para simplificar estas tarefas (atualmente, em linguagens mais clássicas também surgiram serviços similares). Através dos próprios ambientes de desenvolvimento ou de programas de apoio, chamados "gestores de pacotes", basta indicar qual a biblioteca desejada para que os ficheiros necessários sejam transferidos, as configurações efetuadas no sistema e se possa simplesmente fazer `import` no nosso código.

No caso do Python, o serviço chama-se Python Package Index (PyPI): além de simplificar a obtenção dos ficheiros das bibliotecas e a configuração do ambiente de desenvolvimento, este serviço também gere dependências e compatibilidades de versões entre bibliotecas, tornando o processo de integração de bibliotecas externas significativamente mais eficiente e menos propenso a erros. Por exemplo, se uma biblioteca X precisar de usar a versão 4.3 de uma biblioteca Y, o PyPI é capaz de identificar e gerir esta dependência automaticamente. Quando instalamos a biblioteca X através do

PyPI, usando um gestor de pacotes (sendo o mais comum o **pip**, que fica disponível quando se instala o Python no sistema), o sistema verifica as dependências necessárias. Assim, se a biblioteca Y ainda não estiver instalada ou se a versão disponível for incompatível com a biblioteca X, o PyPI deteta-o e o gestor de pacotes descarrega não apenas a biblioteca Y, mas também a versão correta da biblioteca X. Isto elimina a necessidade de o programador gerir manualmente as versões de cada biblioteca, um processo que em ambientes de programação mais antigos era complexo e propenso a erros. Além disso, também permite rapidamente atualizar todas as bibliotecas instaladas, ficando disponíveis as suas versões mais recentes. Isto garante aos programadores um acesso facilitado às versões mais atualizadas e seguras, o que facilita a manutenção e a atualização dos projetos de software.
Já temos vindo a usar bibliotecas nos tópicos anteriores, sem grandes preocupações. Por exemplo, no tópico 3, o exemplo fornecido começava logo com estas linhas:

```
# Biblioteca para mostrar imagem e som (instale usando o pip: "pip
install pygame").
import pygame
```

Este processo de executar o pip na linha de comandos, como indica a linha de comentários (`pip install pygame`) tira partido do serviço PyPI para assegurar que a biblioteca pygame fica instalada no sistema. Depois, quando se executa o comando de importação (neste caso, import pygame) o interpretador Python procura carrega esse código para a sessão de trabalho atual. Todas as funções e classes da biblioteca importada ficam então acessíveis e podem ser utilizadas como se fossem parte do código-fonte do projeto. É graças a isso que as linhas seguintes funcionam:

```
pygame.mixer.init()
miau_sound = pygame.mixer.Sound('miau.wav')
pygame.mixer.Sound.play(miau_sound)
```

Mãos à obra: usar bibliotecas em Python

Agora é preciso dar um salto em frente e abraçar a diversidade de bibliotecas disponíveis – é enorme, temos o mundo aos nossos pés. Podemos ser imaginativos, ambiciosos, criativos. É um desafio que devemos colocar a nós mesmos enquanto programadores. Mais do que usar "esta" ou "aquela" biblioteca, a aptidão essencial é sentir a autoconfiança de abraçar bibliotecas novas, tal como não hesitamos em abrir um novo livro ou iniciar um novo jogo.

Eis algumas listas com sugestões de bibliotecas em Python para iniciarem a vossa exploração:

Recomendações de Vinay Khatri: doze bibliotecas Python muito conhecidas e usadas, acessíveis a programadores com pouca experiência, abarcando áreas diversificadas (e sim, uma delas é a Pygame).

Navegar diretamente a lista do PyPI: podemos usar o Python para ter visualizações tridimensionais interativas, processar áudio, gravar vídeo e muitas outras ideias. Navegando pelo filtro "Topic", podemos encontrar bibliotecas (e projetos) para temas tão diversos como artes, comunicações, jogos, domótica, criptografia, computação distribuída, etc.

Falar com grandes modelos de linguagem: por que não experimentar usar bibliotecas para integrar nos vossos projetos os grandes modelos de inteligência artificial generativa, como os por trás do ChatGPT e outros, inclusivamente alguns de acesso aberto? Vale a pena experimentar bibliotecas como a Marvin que facilitam o processo discursivo ou ir diretamente aos guias da OpenAI.

A capacidade para abraçar bibliotecas em Python pode transformar a vossa capacidade de programação. Seja na visualização de dados complexos, na interação com tecnologias de ponta como o metaverso ou a inteligência artificial generativa, ou no manuseamento eficiente de grandes conjuntos de dados, as bibliotecas são ferramentas indispensáveis no cinturão dos programadores. Não se trata apenas de ter soluções para problemas específicos: trata-se também de ter artefactos inspiradores de novas maneiras de pensar e resolver desafios.