02

Interpretado vs Compilado

Transcrição

Para executar um arquivo Python, por exemplo o **adivinhacao.py**, fazemos:

```
python3 adivinhacao.py
```

Foi isso que fizemos o treinamento inteiro, seja pela linha de comando ou pelo PyCharm. E com o C? Será que existe um comando, algo como um `cexecuter`, para executarmos um arquivo C? Não, não existe.

O ambiente do C exige que primeiramente devemos passar o código fonte (o arquivo .c) para um **compilador**. O compilador lê o código fonte e faz uma análise da sintaxe, se esquecemos algum ponto e vírgula, ou de tipar alguma variável, etc. E feita essa análise, o compilador cria um outro arquivo, e é esse arquivo que podemos executar. Então primeiro vamos compilar o arquivo, vamos utilizar o compilador **gcc** (novamente, não é necessária a sua instalação, estamos usando-o somente para mostrar a diferença entre ambientes que usam o conceito de compilação e ambientes que usam o conceito de interpretação):

```
gcc adivinhacao.c o adivinhacao
```

Ou:

```
gcc -std=c99 adivinhacao.c -o adivinhacao
```

Esse comando compila o arquivo **adivinhacao.c** e se tudo estiver correto, criará o arquivo executável **adivinhacao**. Agora é só executar o arquivo gerado. Em UNIX, fazemos:

```
./adivinhacao
```

Essa é a diferença de um ambiente que usa o conceito de compilação, no qual o código fonte, que não é executável, deve ser compilado para criar um arquivo executável; e um ambiente que usa o conceito de interpretação, no qual o código fonte é executado diretamente.

Transferindo código

Em Python, conseguimos executar um arquivo em qualquer sistema operacional, inclusive os códigos aqui feitos são disponibilizados para vocês, alunos, e ele poderá ser executado seja qual for o seu sistema operacional, basta ter o Python 3 instalado. Já o arquivo executável do C, gerado pelo compilador, não é executável em um sistema operacional diferente. É preciso compilar novamente o código fonte no sistema operacional desejado, para ter um executável funcional. E muitas vezes o código fonte (não é o nosso caso) utiliza algo específico do sistema operacional, passando a depender dele, então nem adiantaria compilá-lo em um SO diferente.

Logo, o Python tem uma portabilidade maior que o C.

O Python é realmente uma linguagem estritamente interpretada?

Para finalizar, falamos que o Python utiliza o conceito de interpretação, ou seja, passamos o código fonte e ele é interpretado, mas não é bem assim. Podemos executar o arquivo **jogos.py** e reparar na pasta que é criada dentro do diretório, a **pycache**. Se formos ver o que tem dentro da sua pasta, vemos que há dois arquivos referentes aos módulos importados no **jogos.py**, ou seja, um arquivo referente à `adivinhacao` e outro à `forca`. Mas o que são esses arquivos?

O que o Python faz ao vivo é ler os módulos importados e os **compila para bytecode**. Esse código foi criado ao mesmo tempo em que executamos o arquivo **jogos.py**. Apesar do Python ter um ambiente de interpretação, ele compila os módulos importados para melhorar o desempenho, a execução do ambiente, apesar de não ter esse processo de compilação explícito.

Do ponto de vista do Python, ele considera que esses módulos não serão modificados, então na próxima execução, para melhorar o desempenho, esse código compilado é que será utilizado.

Com isso, terminamos aqui o nosso curso. No próximo implementaremos o jogo da forca, aprendendo mais sobre funções, coleções, outras funções *built-in*, e muito mais! Muito obrigado por assistirem esse curso e nos encontramos no próximo treinamento!