MC102-2018s1-Aula01-180226

Arthur J. Catto, PhD

27 de fevereiro de 2018

1 Introdução

1.1 O caminho do programa

Um cientista da computação **resolve problemas**.

Resolver um problema envolve:

- Entender o problema
- Formular e responder questões
- Pensar criativamente sobre possíveis soluções
- Expressar uma solução de forma clara e precisa

1.2 Algoritmos

Um **algoritmo** é a descrição de uma maneira de se resolver um certo problema.

Para que um algoritmo possa ser executado por um computador ele precisa ser expresso numa **linguagem de programação**.

Um **programa de computador** é um algoritmo expresso numa linguagem de programação.

1.3 Linguagens formais e naturais

Linguagens naturais são as linguagens que as pessoas falam, como Português, Inglês e Espanhol. Elas não são projetadas, mas evoluem naturalmente.

Linguagens formais são linguagens projetadas para serem usadas em determinadas aplicações. Matemáticos, químicos e programadores, todos usam linguagens formais especificamente desenvolvidas.

Linguagens de programação são linguagens formais desenhadas especificamente para expressar algoritmos de forma a poderem ser interpretados e executados por computadores.

Linguagens naturais e linguagens formais têm muitas características comuns.
Por exemplo: elementos básicos (*tokens*), estrutura, sintaxe e semântica.

Tokens são os elementos básicos da linguagem, como as palavras, números e símbolos.

A estrutura define como as tokens são arranjadas para formar elementos mais complexos como frases, parágrafos e discurso.

A **sintaxe** é a parte da gramática que estuda a disposição das tokens na frase e a das frases no discurso, bem como a relação lógica das frases entre si. - Ao contrário das linguagens naturais, as linguagens formais, em geral, têm regras de sintaxe bastante rígidas.

Finalmente, a **semântica** trata do significado das frases, parágrafos e discursos.

Para que seja possível "entender"uma frase, é necessário reconhecer sua estrutura.

Este processo é chamado **parsing** (análise sintática).

Uma vez analisada uma frase é possível entender a semântica, ou o significado da frase.

Embora as linguagens formais e as linguagens naturais tenham muitas características em comum, existem muitas diferenças:

- **Ambiguidade**
- A mãe pegou o filho correndo na rua.

- **Redundância**
- Vi com meus próprios olhos.

- **Literalidade**
- A vaca foi pro brejo.

1.4 A linguagem de programação Python

Python é uma **linguagem de programação de alto nível**.

Uma linguagem de alto nível é mais abstrata, mais próxima de uma linguagem natural.

Na sua grande maioria, programas de computador são escritos em linguagens de alto nível porque

- O trabalho é mais fácil, mais rápido, mais conciso, mais legível, mais fácil de alterar, mais fácil de corrigir, menos sujeito a erros

- Os programas são portáteis, isto é podem ser executados em diferentes ambientes computacionais sem qualquer modificação.

Antes que um programa escrito em uma linguagem de alto nível possa ser executado por um computador ele precisa ser traduzido para uma **linguagem de baixo nível**.

Um programa escrito numa linguagem de alto nível é chamado **código fonte** e sua tradução em uma linguagem de baixo nível é chamada **código objeto** ou **código executável**.

A tradução de uma linguagem de alto nível para uma linguagem de baixo nível é feita por um programa que pode ser um **interpretador** ou um **compilador**.

- Um **interpretador** lê, traduz e executa uma ou mais linhas de um programa de cada vez, alternadamente.

- Um **compilador** lê e traduz um programa inteiro, deixando-o pronto para ser executado.

Muitas linguagens de programação modernas usam esses dois processos: - O código fonte é compilado gerando um código objeto chamado
código em bytes (byte code). - O código em bytes é interpretado por um programa chamado **máquina virtual**.

Embora Python também use os dois processos, ela é geralmente considerada uma linguagem interpretada.

O interpretador Python pode ser usado de dois modos: - **Linha de comando** (*shell mode*) -- que permite a interação entre o usuário e o interpretador. - **Script** (*program mode*) -- no qual um arquivo contendo um programa fonte completo é submetido ao interpretador para ser executado

Este é um exemplo do uso do interpretador no modo **linha de comando**.

O usuário interage com o interpretador por meio do **terminal Python** (*Python shell*). $python3Python3.6.4|Anacondacustom(64-bit)|(default,Dec212017,15:39:08)[GCC4.2.1CompatibleClang4.0.1(tags/RELEASE_401/final)]ondarwinType"help","copyright","credits"or"lic$ $2+35>>>Osmbolo$**>>>**$chamado$**prompt**$eindicaqueoterminalPythonestprontopararecebberumcomando.$

Este é um exemplo do uso do interpretador no modo **script**.

Primeiro, o usuário criou um arquivo chamado `soma.py`: print(2 + 3) E depois submeteu esse arquivo ao interpretador Python. $python3soma.py5$ Os programas fonte em Python têm extensão `.py`

1.5 Como executar programas Python

Um programa Python pode ser executado de diversas maneiras, entre as quais: - Interativamente no terminal Python - Submetendo a Python um arquivo com o código fonte - Num ambiente IDE como Spyder, PyCharm, Pythonista (iPad), etc. - Nas janelas **ActiveCode** do livro interativo - Numa célula de um Notebook Jupyter, como abaixo...

```
In [1]: print(2+3)
```

```
5
```

1.6 Um pouco mais sobre programas

Nesta disciplina, um *programa* será sempre entendido como uma *sequência de instruções* que especificam como executar uma determinada tarefa.

Em quase todas as linguagens de programação, as instruções essenciais pertencem a 5 grupos:

- Entrada
- Obter dados do teclado, de um arquivo ou de algum outro dispositivo.
- Saída
- Enviar resultados para a tela, um arquivo ou algum outro dispositivo.
- Matemática e lógica
- Executar operações matemáticas e lógicas
- Execução condicional
- Verificar se certas condições são satisfeitas e, nesse caso, executar uma ou mais instruções.
- Repetição
- Executar uma ou mais instruções repetidamente, enquanto certas condições forem satisfeitas.

1.7 O que é depurar um programa?

Depuração é o trabalho de encontrar e corrigir os erros que um programa apresenta.

Os erros que um programa pode apresentar são geralmente classificados em três grupos: - Erros de sintaxe - Erros de execução - Erros de semântica

1.7.1 Erros de sintaxe

Ocorrem quando a disposição das palavras na frase ou das frases no discurso, bem como a relação lógica das frases entre si, desobedecem a sintaxe da linguagem.

Por exemplo, em português, uma frase deve começar com uma letra maiúscula e terminar com um ponto. - herrar é umano - esta frase contém **um** erro de sintaxe. - Esta também

```
In [5]: print('Vou somar 2 e 3...)
        print(2 + 3)


          File "<ipython-input-5-7a8269fa1654>", line 1
        print('Vou somar 2 e 3...)
                                  ^
    SyntaxError: EOL while scanning string literal
```

1.7.2 Erros de execução

Erros de execução ou **exceções** só são percebidos quando um programa está sendo rodado e, em geral, interrompem a execução.

```
In [6]: print('Vou dividir 2 por 3...')
        print(2 / 0)

Vou dividir 2 por 3...


        ---------------------------------------------------------------------------

        ZeroDivisionError                         Traceback (most recent call last)

        <ipython-input-6-7df3fc0b8ef7> in <module>()
          1 print('Vou dividir 2 por 3...')
    ----> 2 print(2 / 0)


        ZeroDivisionError: division by zero
```

1.7.3 Erros de semântica

Um **erro de semântica** é também chamado **erro de lógica**.

Um programa com um erro semântico não falha ou termina abruptamente,
mas não produz o resultado desejado.

Erros semânticos são muito mais difíceis de identificar do que erros de sintaxe ou de execução, porque um programa com erros semânticos ainda é um programa válido na linguagem em que está escrito.

- Um erro de sintaxe torna o programa inválido e é facilmente percebido pelo interpretador.
- Uma exceção interrompe a execução de um programa.
- No entanto, cabe a você identificar os erros de semântica em seus programas.

```
In [4]: print('Vou dividir 2 por 3...')
        print(2 / 8)
```

```
Vou dividir 2 por 3...
0.25
```