**Texto base**

**2**

**Estrutura Sequencial - Variáveis, Comandos Básicos e Operadores Aritméticos**

Gilberto Alves Pereira

***Resumo***

*Para elaborar um programa precisamos entender algumas ferramentas básicas. Como vimos na aula 1, um software recebe informações de dispositivos de entrada, armazena na memória, efetua um processamento (cálculo) com as informações que estão na memória, armazena o resultado na memória e por fim envia o resultado desse cálculo que está na memória para um dispositivo de saída. São esses os componentes que iremos ver nesse texto: Variável (memória) e os tipos de informações que são armazenadas na memória, Comandos de Entrada e Saída, processamento (atribuição =). Por fim, como embasamento para se fazer processamento vamos detalhar melhor os operadores aritméticos.*

**1.1. Variáveis e Identificadores**

Uma variável tem um nome (identificador) que está associado a um espaço em memória que armazena um valor. Uma variável pode receber diferentes valores durante a execução de um programa, por isso o nome “variável”.

Para atribuir um valor a uma variável, utiliza-se o símbolo de **Processo** e o **operador de atribuição “=”**



**Figura 2.1. Exemplo do uso do símbolo Processo no fluxograma. Fonte Autor.**

Neste exemplo o processo atribui o valor 1 à variável de nome num. Note que a atribuição é feita aqui com o símbolo de igual, mas ele não tem o mesmo significado de equivalência da matemática (veremos mais a respeito disso nas próximas aulas).

Para mostrar um determinado valor na tela, usamos o símbolo de **Exibir**.



**Figura 2.2. Exemplo do uso do símbolo Exibir no fluxograma. Fonte Autor.**

**1.2. Como nomear as variáveis**

Um **nome** ou **identificador** de uma variável é formado por uma sequência de um ou mais caracteres. Como exemplo, as regras adotadas pelo Python são::

* Pode conter apenas letras, dígitos e underscores (sublinhado);
* Pode começar por uma letra ou underscore;
* Não é permitido o uso de outros caracteres especiais (por exemplo, espaço);
* Não é permitido o uso de palavras reservadas da linguagem;
* O identificador deve ser conciso, porém descritivo (idade é melhor que i, tamanho\_nome é melhor que tamanho\_do\_nome\_da\_pessoa).

Vejamos alguns exemplos de identificadores inválidos:

**Tabela 2.1. Exemplos da nomeação inválida de variáveis.**

| **Válidos** | **Inválidos** | **Motivo** |
| --- | --- | --- |
| totalVenda | ‘totalVenda’ | Uso de caractere especial ‘ aspas simples |
| total\_venda | total venda | Uso de espaço |
| venda2016 | 2016Venda | Começar com número |

Saber qual é o tipo de dado mais adequado para ser armazenado em uma variável é muito importante para garantir a resolução de um problema. Há um conjunto predefinido de tipos de dados chamados de tipos embutidos. Vejamos alguns deles:

**1.2.1. Números**

Um dado numérico é composto por uma sequência de dígitos (0 a 9), um sinal opcional (+ ou –) e um possível ponto decimal (usa-se o ponto e não a vírgula para separar a parte inteira da fracionária). São classificados como **inteiro** ou de **ponto flutuante**.

Exemplos:

**Tabela 2.2. Exemplos da escrita de dados numéricos.**

| **Corretos** | | | | **Incorretos** | |
| --- | --- | --- | --- | --- | --- |
| **Inteiro** | **Ponto flutuante** | | |
| 5 | 5. | 5.0 | 0.125 | 5,03 | |
| 2500 | 2500. | 2500.0 | 2500.125 | 2,500 | 2,500.125 |
| +2500 | +2500. | +2500.0 | +2500.125 | +2,500 | +2,500.125 |
| -2500 | -2500. | -2500.0 | -2500.125 | -2,500 | -2,500.125 |

**1.2.2. Texto - letras**

Um texto (string**)** representa uma sequência de caracteres (letras, dígitos, símbolos especiais). As *strings* podem ser delimitadas por um par de aspas simples ( ’ ) ou duplas ( ” ).

Exemplos:

“olá”

‘Faculdade Impacta de Tecnologia’

‘Avenida Rudge, 315’

Os dados para serem armazenados na memória devem ser convertidos para números. Com a finalidade de padronizar essa conversão de letras e símbolos para informações em formato numérico para ser armazenada foi criada a tabela ASCII. Veja que por exemplo a letra A (a maiúsculo) é armazenada na memória como o número 65.

**Tabela 2.3. Tabela ASCII (parcial).**

| **Caractere** | **Cód ASCII** | **Caractere** | **Cód ASCII** | **Caractere** | **Cód ASCII** |
| --- | --- | --- | --- | --- | --- |
| **Espaço** | **32** | **A** | **65** | **a** | **97** |
| **!** | **33** | **B** | **66** | **b** | **98** |
| **"** | **34** | **C** | **67** | **c** | **99** |
| **#** | **35** | **D** | **68** | **d** | **100** |
| **$** | **36** | **E** | **69** | **e** | **101** |
| **%** | **37** | **F** | **70** | **f** | **102** |
| **&** | **38** | **G** | **71** | **g** | **103** |
| **'** | **39** | **H** | **72** | **h** | **104** |
| **(** | **40** | **I** | **73** | **i** | **105** |
| **)** | **41** | **J** | **74** | **j** | **106** |
| **\*** | **42** | **K** | **75** | **k** | **107** |
| + | **43** | **L** | **76** | **l** | **108** |
| **,** | **44** | **M** | **77** | **m** | **109** |
| **-** | **45** | **N** | **78** | **n** | **110** |
| **.** | **46** | **O** | **79** | **o** | **111** |
| **/** | **47** | **P** | **80** | **p** | **112** |

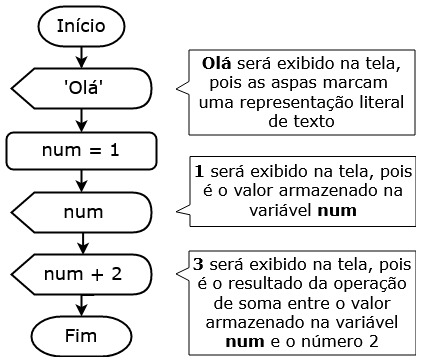
**1.2.3. Entrada e saída de dados**

Os programas que iremos escrever poderão obter e apresentar dados aos usuários. Para isso, já vimos em exemplos anteriores o símbolo **Exibir** (saída de dados), que é usado para exibir informações na tela. O exemplo a seguir exibe a mensagem: **Meu primeiro fluxograma**.



**Figura 2.3. Exemplo de saída de dados. Fonte Autor.**

É possível com isso exibir na tela uma *string*, o conteúdo de uma variável, o resultado de uma expressão matemática ou booleana, etc.



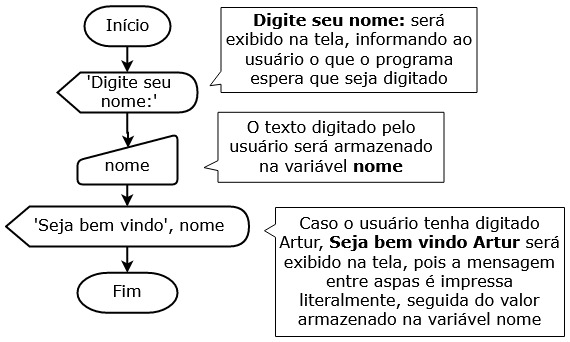
**Figura 2.4. Exemplo do uso dos símbolos do fluxograma. Fonte Autor.**

Se quisermos pedir ao usuário que forneça um valor, via teclado, usaremos o símbolo de **Entrada Manual**. Esse valor deverá ser armazenado em uma variável para que possa ser reutilizado posteriormente.

Para que o usuário saiba o que deve ser informado, podemos exibir uma mensagem antes de utilizar o símbolo de entrada manual, conforme no exemplo a seguir, no qual pedimos para o usuário digitar seu nome e em seguida exibimos uma mensagem de boas-vindas.



**Figura 2.5. Símbolo de entrada manual. Fonte Autor.**



**Figura 2.6. Exemplo de fluxograma com o símbolo de entrada manual. Fonte Autor.**

**1.3. Operadores e Operadores Aritméticos**

Agora que já conhecemos os conceitos e comandos básicos (Variável, Comando de Entrada, Comando de Saída e Atribuição), vamos detalhar alguns conceitos importantes em relação ao comando de atribuição “=” usado em processos.

Um operador é um símbolo que representa a operação que pode ser realizada em um ou mais operandos. Operadores que atuam sobre um operando são chamados operadores unários, e operadores que atuam sobre dois operandos são chamados de operadores binários.

**1.3.1 Operadores Aritméticos**

Operadores aritméticos são aqueles cujos operandos são dados de tipo numérico (inteiros ou em ponto flutuante). A tabela a seguir lista os operadores básicos de acordo com a sintaxe do Python, e de maneira geral, a maioria das linguagens de programação tem operadores equivalentes (ou funções que os substituem).

**Tabela 2.4. Operadores aritméticos, seus significados e exemplos de uso.**

| Operadores | Significado | Exemplo | Resultado |
| --- | --- | --- | --- |
| -x | Negação | -(10) | -10 |
| x + y | Soma | 2 + 4 | 6 |
| x - y | Subtração | 2 - 4 | -2 |
| x \* y | Multiplicação ou produto | 2 \* 4 | 8 |
| x / y | Divisão | 25 / 10 | 2.5 |
| x // y | Divisão truncada | 25 // 10 | 2 |
| x % y | Módulo ou resto da divisão | 25 % 10 | 5 |
| x \*\* y | Exponenciação | 2 \*\* 4 | 16 |

Observações:

* O operador (-) pode ser um operador unário (negação) ou um operador binário (subtração). Todos os outros operadores são sempre binários.
* A exponenciação pode ser utilizada tanto para números inteiros quanto para números em ponto flutuante na base e no expoente.
* Exemplos:
  + O resultado de 2\*\*4 é 16; e
  + O resultado de 2.5\*\*1.2 é 3.00281108.

**1.4. Expressões e tipos de dados**

Uma expressão é uma combinação de símbolos (operandos e operadores) que serão avaliados. Assim como na matemática, em programação uma expressão pode conter variáveis, como por exemplo: 4 + 3\*k.

As expressões que avaliam valores do tipo numérico são chamadas de **expressões aritméticas**.

Para avaliarmos corretamente as expressões acima, é importante sabermos em que ordem devemos realizar as operações, ou seja, é necessário conhecer a precedência dos operadores. O natural seria considerar a precedência da matemática, porém cada linguagem de programação tem sua própria regra, que pode ser igual ou não à da matemática.

Além da precedência dos operadores, devemos conhecer também como cada linguagem trata os operadores que tenham o **mesmo nível de prioridade**. À ordem em que isso ocorre damos o nome de **associatividade** dos operadores.

A tabela a seguir traz a associatividade adotada pelo Python:

**Tabela 2.5. Operadores e suas associatividades no código Python.**

| **Operador** | **Associatividade** |
| --- | --- |
| \*\* | Direita para esquerda |
| - (negação) | Esquerda para direita |
| \* / // % | Esquerda para direita |
| + e – (subtração) | Esquerda para direita |

As expressões que avaliam valores do tipo numérico são chamadas de expressões aritméticas.

**1.5. Funções Matemáticas**

Podemos usar um conjunto variado de funções como as trigonométricas (ex: sin, cos, tan, etc) e logarítmicas (ex: log) e também constantes matemáticas (ex: pi). Esse conjunto de funções pode variar dependendo da linguagem de programação utilizada. No nosso curso, utilizaremos as funções disponíveis na biblioteca de funções matemáticas da linguagem Python.

**Tabela 2.6. Exemplos de funções matemáticas do código Python.**

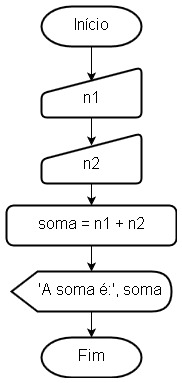
| **Sintaxe** | **Descrição** |
| --- | --- |
| acos(x) | retorna o arco cosseno de x |
| asin(x) | retorna o arco seno de x |
| atan(x) | retorna o arco tangente de x |
| ceil(x) | retorna o menor inteiro maior ou igual a x como int |
| cos(x) | retorna o cosseno de x |
| degrees(r) | converte radianos para graus |
| exp(x) | retorna o exponencial de x (*e*x) |
| fabs(x) | retorne o valor absoluto de x |
| factorial(x) | retorna x! |
| floor(x) | retorna o maior inteiro menor ou igual a x como um int |
| log(x,b) | retorna logb x (se b for omitido, retorna log x na base *e*) |
| log10(x) | retorna log10 x |
| modf(x) | retorna a parte fracionária e a parte inteira como dois floats |
| pow(x,y) | retorna xy |
| radians(g) | converte graus para radianos |
| sin(x) | retorna o seno de x |
| sqrt(x) | retorna a raiz quadrada de x |
| tan(x) | retorna a tangente de x |
| trunc(x) | retorna a parte inteira de x como um int; igual a int(x) |

**1.6. Estrutura Sequencial**

Nos exercícios construídos anteriormente, sempre foi utilizada a mesma sequência de instruções: receber dados iniciais, realizar operações matemáticas e apresentar resultados finais.

Esta estrutura de controle, caracterizada pela execução das instruções na ordem em que foram escritas, é denominada de Estrutura Sequencial.

Podemos observar esta estrutura no exemplo já trabalhado nas aulas anteriores em que construímos um fluxograma que exibe a soma de dois números informados pelo usuário.



**Figura 2.7. Exemplo de estrutura sequencial no fluxograma. Fonte Autor.**

**1.7. Exemplo de Fluxograma**

Desenhar um fluxograma que calcula e apresenta o dobro, o triplo e o quadrado de um número digitado pelo usuário.

Solução:

Precisamos inicialmente entender o objetivo do exercício. Uma forma interessante de saber onde devemos chegar é identificar no enunciado do exercício o resultado final que está sendo solicitado (**saída**), os valores que devemos fornecer para conseguirmos fazer o cálculo (**entradas**) e finalmente elaborar o cálculo necessário (**processamento**).

Podemos usar algumas palavras-chave para identificarmos a saída (exibir, apresentar, imprimir). Do texto podemos observar que temos 3 saídas (apresentar dobro, triplo e quadrado). Vamos guardar essas informações em variáveis e chamá-las de **dobro**, **triplo** e **quadrado**.

As entradas também podem ser identificadas por palavras-chave como (dado, digitado, informado, obtido). Nesse caso temos como entrada um único número digitado pelo usuário. Vamos atribuí-lo à variável **num**.

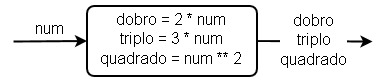
Agora precisamos identificar qual o cálculo que devemos fazer com a entrada (num) para obtermos as saídas pedidas (dobro, triplo e quadrado). Seguem os cálculos:

dobro = 2\*num → dobro do número digitado pelo usuário

triplo = 3\*num → triplo do número digitado pelo usuário

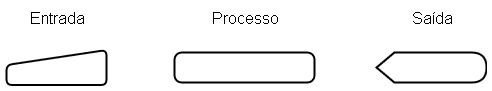
quadrado = num\*\*2 → quadrado do número digitado pelo usuário

Em seguida, podemos desenhar o seguinte diagrama de entrada/processamento/saída:



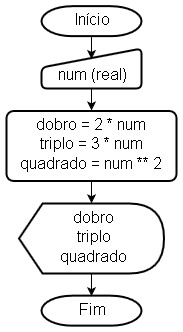
**Figura 2.8. Representação do processamento dos dados. Fonte Autor.**

E fazendo uso dos seguintes símbolos:



**Figura 2.9. Símbolos de entrada, processo e saída usados no fluxograma. Fonte Autor.**

Podemos finalmente desenhar o fluxograma:



**Figura 2.10. Exemplo de fluxograma da solução. Fonte Autor.**

Exemplo: Fluxograma e Conversão em Python

Nesse exemplo, vamos criar um fluxograma para mostrar a soma entre dois números inteiros. Esses números serão digitados pelo usuário e iremos armazená-los nas variáveis **n1** e **n2**. Em seguida, vamos realizar o processo de soma, que consiste em recuperar os valores salvos, somá-los e atribuir o resultado à variável **soma**. Ao final, este resultado deverá ser apresentado ao usuário.

**Tabela 2.7. Correspondência dos elementos do fluxograma e seu respectivo código Python.**

| **Fluxograma** | **Python** | **Descrição** |
| --- | --- | --- |
|  | n1=int (input ( ) ) | **Entrada de número inteiro via teclado** |
|  | soma = n1 + n2 | **Processo - soma de n1 e n2 e armazenamento em soma** |
|  | print (‘A soma é: ’ , soma) | **Exibição na tela de mensagem e conteúdo de uma variável** |

**1.8. Referências**

DIERBACH, C. “Introduction to Computer Science Using Python: A Computational Problem-Solving Focus”1st Edition, New York: Wiley, 2012.