**Apresentação - Lógica de Programação**

A disciplina de Lógica de Programação representa a entrada do aluno na área de desenvolvimento de sistemas e correlatos. Seu conteúdo abrange os conceitos e ferramentas básicas de programação, como variáveis e estruturas de controle, que capacitam o aluno a solucionar problemas computacionais e construir programas utilizando essas ferramentas básicas existentes em qualquer linguagem. É uma disciplina de vital importância, pois seu conteúdo serve de alicerce para uma série de outras disciplinas que abordam aspectos mais avançados de programação e que serão vistas na sequência do curso.

Acompanhe a apresentação da disciplina, elaborada pelo Professor Gilberto Aves Pereira:

[Apresentação Lógica de Programação](https://www.youtubeeducation.com/watch?v=L5l2bjfRi4g).

**Objetivos gerais**

Desenvolvimento do Raciocínio Lógico;

Aprender a descrever de forma gráfica uma solução lógica;

Entender os Conceitos Envolvidos em Lógica de Programação;

Ser capacitado a utilizar as ferramentas básicas de lógica de programação abaixo:

* Variáveis e Comandos de Entrada e Saída;
* Estruturas de Controle de Fluxo (Sequencial, Seleção e Repetição);
* Funções;
* Estruturas de Armazenamento - Vetores.

**Parte 1 -** [**Introdução - Algoritmos e Fluxogramas**](https://sites.google.com/faculdadeimpacta.com.br/lo-p1?pli=1&authuser=2) **-** [**Video**](https://www.youtubeeducation.com/watch?v=KWTQJxJ_ZPU)

**Resumo**

Nesse texto discutimos os conceitos básicos envolvidos quando falamos inicialmente sobre lógica de programação: Algoritmos, o funcionamento de um software, Linguagem de Programação e por fim falamos um pouco sobre a ferramenta que usaremos em nosso curso - o Fluxograma.

**Algoritmo**

O termo algoritmo pode ser visto desde o século IX. Foi nesta época que o cientista, astrônomo e matemático persa Abū ‘Abd Allāh Muhammad ibn Mūsā al-Khwārizmī usou pela primeira vez o termo para indicar regras de operações aritméticas utilizando algarismos indo-arábicos (DIERBACH 2012).

No século XII, Adelardo de Bath traduziu o termo para o latim Algorithmi. De lá para cá, o termo evoluiu bastante, incluindo todos os procedimentos definidos para resolver problemas ou realizar tarefas.

DIERBACH (2012) A formalização da noção de algoritmo ocorreu em 1936 com os trabalhos de Alan Turing e Alonzo Church, que desenvolveram independentemente os modelos de Máquinas de Turing e Cálculo Lambda.

Do ponto de vista computacional, um algoritmo pode ser visto como um conjunto de regras e procedimentos lógicos perfeitamente definidos que levam à solução de um problema em um número finito de passos.

DIERBACH (2012) Donald Knuth, um dos pesquisadores mais respeitados em algoritmos, indica uma lista

de cinco propriedades que são requisitos para algoritmos:

**Finitude**: um algoritmo deve sempre terminar após um número finito de etapas (ou passos).

**Definição**: cada passo de um algoritmo deve ser definido com precisão. As ações a serem executadas deverão ser especificadas rigorosamente e sem ambiguidades.

**Entrada**: valores que são dados ao algoritmo antes que ele inicie.

**Saída**: os valores resultantes das ações do algoritmo a partir de uma determinada entrada.

**Eficácia**: todas as operações a serem realizadas pelo algoritmo devem ser suficientemente básicas para poderem, em princípio, ser feitas com precisão e em um período de tempo finito por um homem usando papel e lápis.



A formalidade pode ser conseguida com o uso de lógica. Assim, vamos exigir que um algoritmo seja uma sequência lógica de passos com começo, meio e fim.

Comumente, esta lógica é conhecida como lógica de programação e isto ocupará grande parte de nossa disciplina.

**Funcionamento de um Software**

De uma maneira simplificada, um software para funcionar precisa de um ambiente chamado Sistema Digital. O sistema digital é composto minimamente por uma unidade de processamento de instruções (UCP), memória (volátil - RAM), e de dispositivos de entrada e saída. As informações dentro do sistema digital estão em formato binário (zeros e uns).

O software é um conjunto de instruções para o processador que fica armazenado na memória. Esse conjunto de instruções basicamente fazem o seguinte:

1 - Obtém dados de algum dispositivo de entrada (teclado, mouse, HD, placa de rede,etc) e armazena na memória;

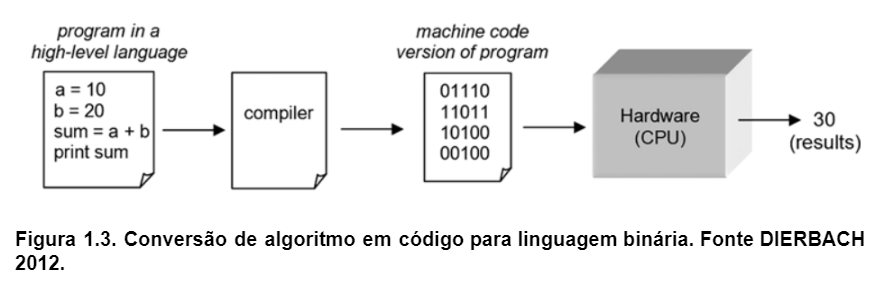
2 - Com as informações na memória efetua cálculos - o que chamamos de processamento, armazenando esses resultados na memória;

3 - Por fim, envia esses resultados desses cálculos da memória para algum dispositivo de saída (impressora, tela, placa de rede,etc).



**Sobre Linguagens de Programação e Algoritmos**

A Linguagem de Programação auxilia no processo de conversão para a linguagem binária do processador, através de uma série de regras que eliminam as ambiguidades.



Algoritmos resolvem qualquer problema. As Linguagens resolvem problemas utilizando os recursos do Sistema Digital (dispositivos de entrada, memória, processamento e dispositivos de saída). Nem todos Algoritmos podem ser implementados por Linguagens.

**O processo para resolver problemas computacionais**

A resolução de um problema computacional não envolve simplesmente o ato de programar um computador. Ele é um processo, sendo que a programação é apenas um dos passos. Antes de escrever um programa é preciso entender e pensar a solução do problema, depois desenvolver o projeto, escrever e testar a solução encontrada.

Uma vez que o algoritmo tenha sido definido, podemos simulá-lo com ferramentas como Scratch, VisuAlg ou Raptor, representá-los em Fluxograma ou implementá-lo diretamente em alguma linguagem de programação (Python, C, C++, Java, Pascal, PHP, dentre outras). Nossa disciplina será focada em Fluxograma.

**Fluxograma**

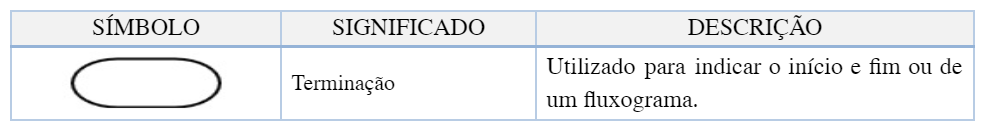
Fluxograma é uma forma padronizada e eficaz para se representar os passos lógicos de um determinado processo (algoritmo), é uma técnica de representação gráfica na qual se utilizam símbolos previamente convencionados, permitindo a descrição clara e precisa do fluxo, ou sequência de um processo, bem como sua análise e redesenho.

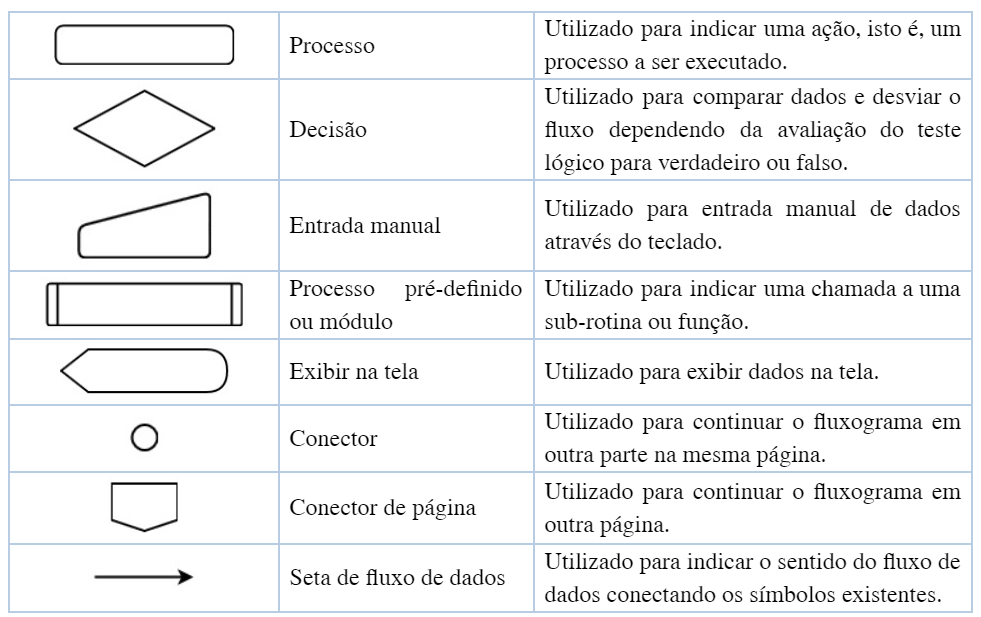
Vantagens: maior clareza na visualização do fluxo de execução (“imagens valem mais do que mil palavras”).

Desvantagens: requer conhecimento de convenções gráficas e dificuldade para fazer correções.

O fluxograma é representado por símbolos e cada um é utilizado para uma finalidade específica, o que torna o fluxograma mais fácil de compreender, pois a representação segue um padrão pré-estabelecido.

Neste curso, usaremos apenas uma pequena parcela dos símbolos disponíveis para uso nos fluxogramas, mostrados na tabela a seguir:

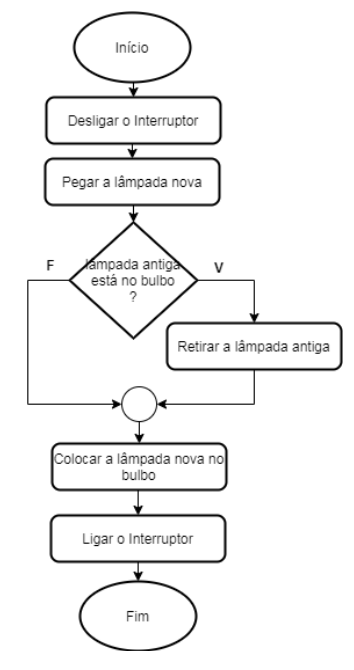




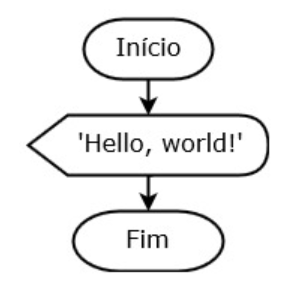
Fluxograma para troca de uma lâmpada. Nesse exemplo, estamos usando um símbolo diferente e novo em relação ao exemplo anterior, o símbolo de decisão.



Este símbolo nos permite criar dois caminhos em nosso fluxograma. Nesse caso, um dos caminhos não fazemos nada e o outro retiramos a lâmpada do bulbo. Tomamos o caminho dependendo da resposta a pergunta - Verdadeiro ou Falso.



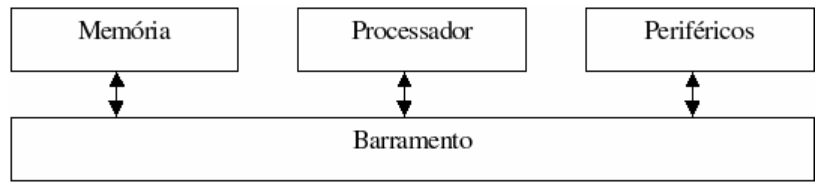
Para exibir uma mensagem na tela, utilizamos o símbolo Exibir. Esse símbolo é uma Saída de Dados, que pode ser utilizado para mostrar o resultado de um processo, um nome ou até mesmo para pedir para que o usuário digite alguma coisa.



Ferramenta para realização dos fluxogramas. - <https://www.drawio.com/>

**Introdução à Arquitetura de Computadores**

**Componentes básicos de um computador**



**Processador (ou microprocessador)**

É um circuito integrado (ou chip). É considerado o cérebro do computador. É ele que executa os programas, faz os cálculos e toma as decisões, de acordo com as instruções armazenadas na memória.

Os microprocessadores formam uma parte importantíssima do computador chamada de UCP (Unidade Central de Processamento), ou em inglês CPU (Central Processing Unit). Antes da existência dos microprocessadores, as CPUs dos computadores eram formadas por um grande número de chips, distribuídos ao longo de uma ou diversas placas. Um microprocessador nada mais é que uma CPU inteira, dentro de um único chip.

Ligando-se um microprocessador a alguns chips de memória e alguns outros chips auxiliares, construiu-se um computador inteiro em uma única placa de circuito, chamada placa mãe dos microcomputadores.

A CPU realiza as seguintes tarefas:

1 - Busca e executa as instruções existentes na memória. Os programas e os dados que ficam gravados no disco (rígido ou disquete) são transferidos para a memória. Uma vez estando na memória a CPU pode executar os programas e processar os dados.

2 - Comanda todos os outros chips do computador

A CPU é composta por:

**Unidade lógica e aritmética (ULA) -**  Assume todas as tarefas relacionadas às operações lógicas (and, or, not, etc.) e aritméticas (adições, subtrações, ...)

**Unidade de controle (UC) -** Assume toda a tarefa de controle das ações a serem realizadas pelo computador, comandando todos os demais componentes de sua arquitetura. É a UC que deve garantir a correta execução dos programas e a utilização dos dados corretos nas operações que as manipulam.

**Os registradores -** A CPU contém internamente uma memória de alta velocidade que permite o armazenamento de valores intermediários ou informação de comando. Esta memória é composta por registradores (ou registros) na qual cada registro possui uma função própria. Um registro memoriza um número limitado de bits, geralmente uma palavra de memória. Os registros mais importantes:

* contador de programa (PC) que aponta para a próxima instrução a executar;
* registro de instrução (IR) que armazena a instrução em execução;
* outros registros que permitem o armazenamento de resultados intermediários.

**Clock**

Clock é um circuito oscilador que tem a função de sincronizar e ditar a medida de velocidade de transferência de dados entre duas partes essenciais de um processamento, por exemplo, entre o processador e a memória principal. Esta frequência é medida em ciclos por segundo, ou hertz .

**Memória**

Constitui de um conjunto de circuitos capazes de armazenar os dados e os programas a serem executados pela máquina. Temos as seguintes categorias de memória:

a ) **A memória principal (ou memória de trabalho)**

É onde normalmente devem estar armazenados os programas e dados a serem manipulados pelo processador. Este tipo de memória aparece como um conjunto de chips que são inseridos na placa mãe do computador.

b) **A memória secundária (ou memória de massa)**

Não é acessada diretamente pela CPU. O acesso é feito através de interfaces ou controladoras especiais. É uma memória do tipo permanente. Possui alta capacidade de armazenamento e um custo menor que o da memória principal. A memória secundária não é formada por chips, e sim por dispositivos que utilizam outras tecnologias de armazenamento (magnética ou óptica). Exemplos: disco rígido, disquete, fita magnética e cd-rom.

**Tipos de memória**

Os chips de memória podem ser divididos em duas grandes categorias:

**RAM (memória de leitura e escrita)**

São chips de memória que podem ser gravados pela CPU a qualquer instante. A CPU usa a RAM para armazenar e executar programas vindos do disco, para ler e gravar os dados que estão sendo processados. É uma memória volátil ( quando o computador é desligado, todos os seus dados são apagados). Por esta razão, os dados e programas devem ficar gravados no disco, que é uma memória permanente.

**Memórias não voláteis**

São memórias cujas informações mantidas não são perdidas caso o computador seja desligado. Exemplo: BIOS (basic input-output system – sistema básico de entrada e saída). Está gravado em uma memória permanente localizada na placa mãe. Tipos de memórias permanentes:

**ROM**

São chips que podem ser lidos pela CPU a qualquer instante, mas não podem ser gravados pela CPU. A gravação é feita pelo fabricante. Este tipo de memória foi usada para armazenar a BIOS.

**PROM**

É uma ROM programável. A gravação pode ser feita apenas uma vez, pois utiliza um processo irreversível.

**EPROM**

É uma ROM programável e apagável. Pode ser programada comportando-se com o uma ROM. A EPROM pode ser apagada com raios ultravioletas de alta potência.

**EEPROM**

É um tipo de memória ROM mais flexível. Pode ser apagada sob controle de software. Utilizada para armazenar as BIOS atuais.

**Memória fora da placa mãe**

A placa mãe contém quase toda a memória de um microcomputador, mas outras placas também podem conter memórias, do tipo RAM e do tipo ROM. As placas de vídeo contém uma ROM com a sua própria BIOS e uma RAM chamada de memória de vídeo

**Placas controladoras**

SCSI (small compact system interface) permite a conexão de diversos periféricos;

IDE (intelligent drive electronics).

**A memória cache**

É uma área reservada de memória que possui duas funções:

* Aumentar o desempenho do computador
* Aumentar o tempo de vida das unidades de disco

Temos dois tipos de memória cache:

* A que vem incorporada à máquina, dessa forma é mais rápida que a memória RAM;
* A que é implementada via software na memória RAM, aumentando o desempenho do acesso ao disco.

**Barramentos**

Um barramento ou bus, é um caminho comum pelo qual os dados trafegam dentro do computador. Este caminho é usado para comunicação e pode ser estabelecido entre dois ou mais elementos do computador.

O tamanho do barramento determina quantos dados podem ser transferidos em uma única vez (16 bits, 32bits, ...)

Um PC possui muitos barramentos, que incluem:

**Barramento do processador**

É o barramento que o chipset (chips de suporte adjacentes contidos na placa mãe) usa para enviar/receber informações do processador.

**Barramento de cache**

É um barramento dedicado para acessar a cache. Usado pelos Pentium Pro e Pentium III.

**Barramento de memória**

Conecta o sub-sistema da memória ao chipset e ao processador.

**Barramento local de entrada/saída (E/S)**

Usado para conectar periféricos de alto desempenho à memória, chipset e processador. Exemplo: placas de vídeo, interface de redes de alta velocidade. Os mais comuns:

* Vesa local bus (VLB)
* Peripheral component inter connect bus (PCI)

**Barramento padrão de entrada/ saída(E/S):**

Usado para periféricos lentos (mouses, placas de som) e também para compatibilidade com dispositivos antigos.

Todos os barramentos possuem duas partes: um barramento de endereçamento (que transfere a informação de onde o dado se encontra) e um barramento de dados (que transfere os dados em si, ou seja, o valor de memória).

**Dispositivos de Entrada e Saída ou Periféricos**

São equipamentos utilizados como portadores das informações que o computador irá processar. Através desses dispositivos, o computador pode armazenar, ler, transferir e receber dados.

Dispositivos de entrada:

Teclado - Mouse - Drive de CD-ROM - Microfone - Scanner

Dispositivos de saída:

Vídeo - Impressora - Alto-falante

Dispositivos de entrada e saída:

Disco rígido - Drive de disquete - Unidade de fita magnética - Modem

**Tipos de comunicação com os dispositivos**

A CPU comunica-se com os periféricos através de circuitos chamados interfaces ou portas de E/S, que implementam a transmissão de dados segundo duas políticas:

* comunicação paralela: impressora;
* comunicação serial: mouse, modem.

**Parte 2 -** [**Estrutura Sequencial - Variáveis, Comandos Básicos e Operadores Aritméticos**](https://sites.google.com/faculdadeimpacta.com.br/lo-p2?pli=1&authuser=2) **-** [**Video**](https://www.youtubeeducation.com/watch?v=cr0xtsc85PI)

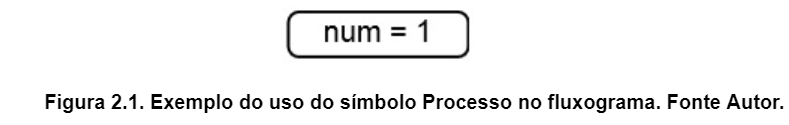
**Resumo**

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.

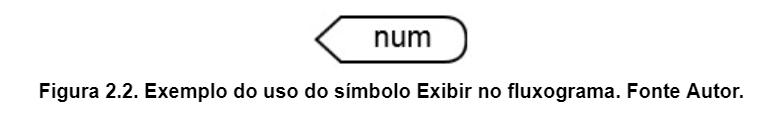
Variáveis e Identificadores

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 “=”.



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

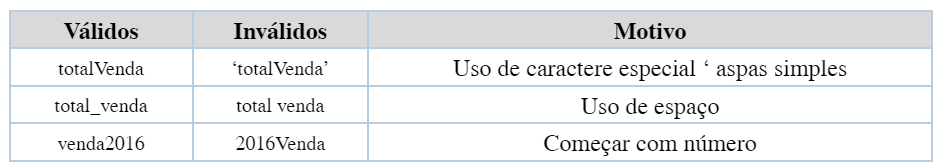


**Como nomear as variáveis**

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).



**Tipos de dados**

**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.

**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’

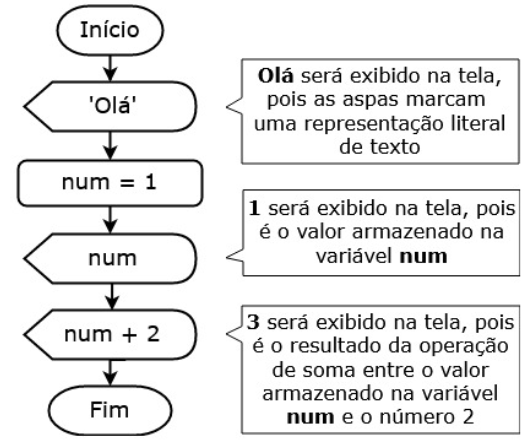
‘Avenida Rudge, 315’

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.

**Entrada e saída de dados**

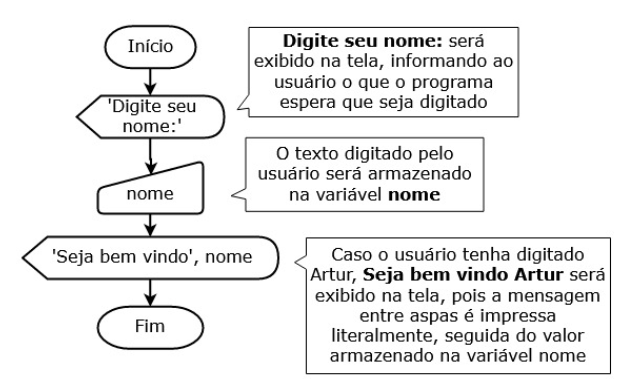
Os programas que iremos escrever poderão obter e apresentar dados aos usuários. Para isso, já vimos em exemplos anteriores o símbolo Exibir (saída de dados).

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



Se quisermos pedir ao usuário que forneça um valor, via teclado, usaremos o símbolo de Entrada Manual.



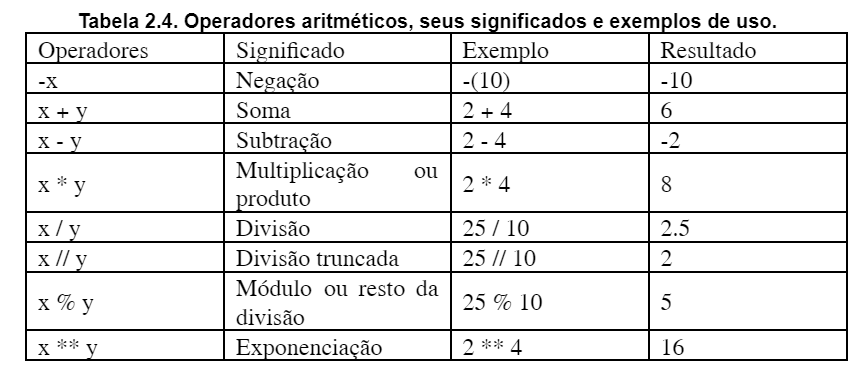


**Operadores e Operadores Aritméticos**

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.

**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).



**Expressões e tipos de dados**

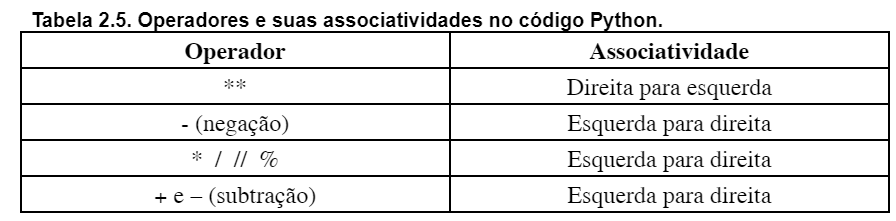
Uma expressão é uma combinação de símbolos (operandos e operadores) que serão avaliados. Assim como na matemática, em programação uma expressão pode conter variáveis, como por exemplo: 4 + 3\*k.

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.

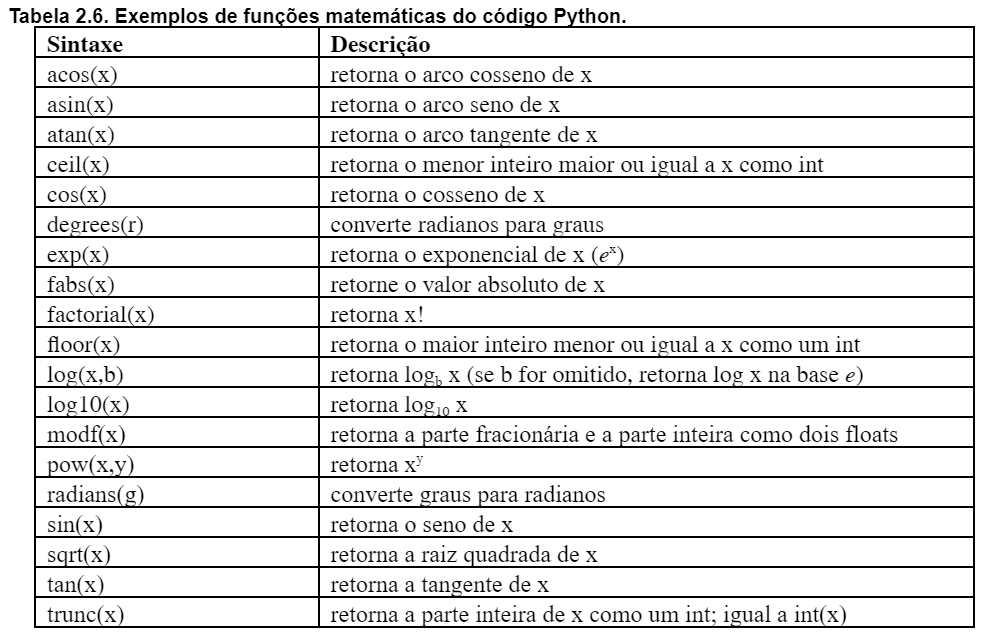
Além da precedência dos operadores, devemos conhecer também como cada linguagem trata os operadores que tenham o mesmo nível de prioridade. À ordem em que isso ocorre damos o nome de associatividade dos operadores.

A tabela a seguir traz a associatividade adotada pelo Python:



**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.

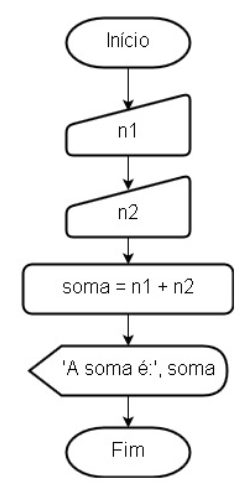


**Estrutura Sequencial**

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.



**Exemplo de Fluxograma**

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

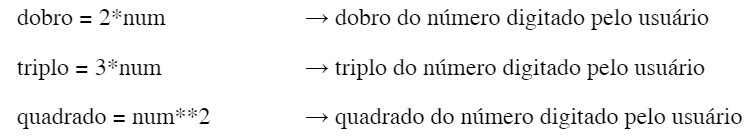
Solução:

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).

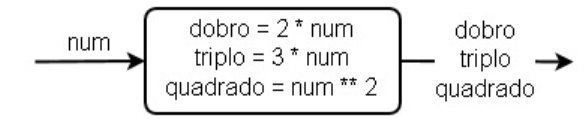
Podemos usar algumas palavras-chave para identificarmos a saída (exibir, apresentar, imprimir). Do texto podemos observar que temos 3 saídas (apresentar dobro, triplo e quadrado). Vamos guardar essas informações em variáveis e chamá-las de dobro, triplo e quadrado.

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.

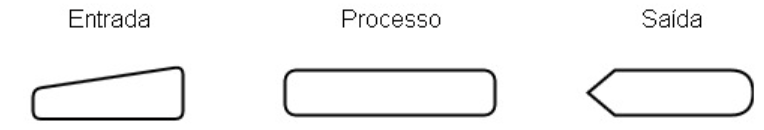
Agora precisamos identificar qual o cálculo que devemos fazer com a entrada (num) para obtermos as saídas pedidas (dobro, triplo e quadrado). Seguem os cálculos:



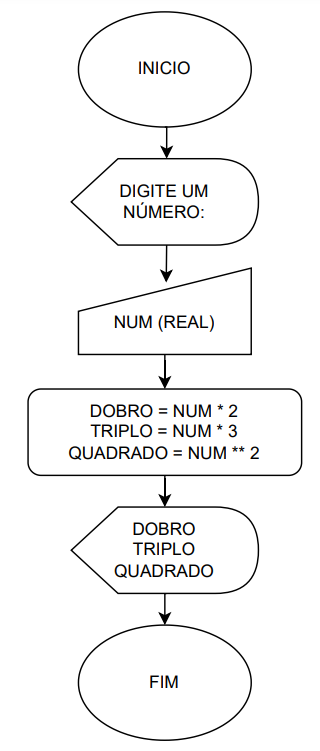
Em seguida, podemos desenhar o seguinte diagrama de entrada/processamento/saída:



E fazendo uso dos seguintes símbolos:

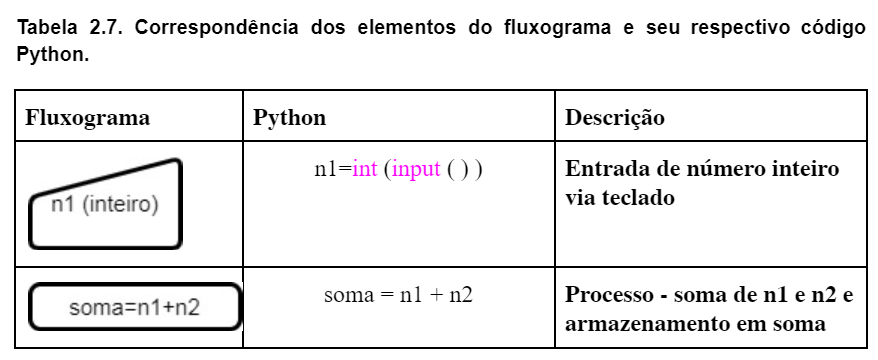


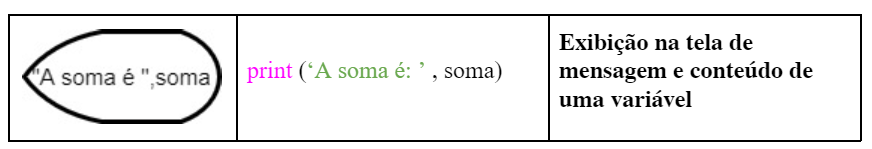
Podemos finalmente desenhar o fluxograma:



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.





Faça um fluxograma que:

Defina 3 variáveis com valores que armazenem:

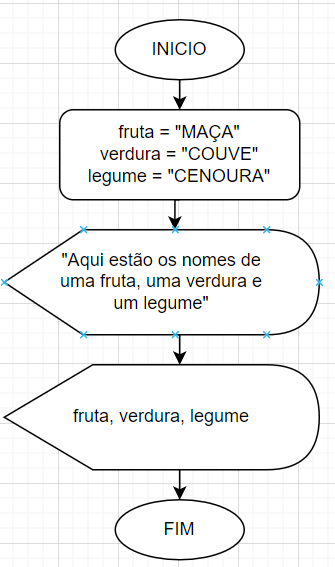
O nome de uma fruta;

O nome de verdura; e

O nome de um legume.

Mostre uma mensagem “Aqui estão os nomes de uma fruta, uma verdura e um legume:”;

Mostre, após essa mensagem, o nome da fruta, da verdura e do legume, respectivamente.



Altere o fluxograma anterior para conter mais três variáveis, com o preço unitário da fruta, verdura e legume. Depois, mostre o preço de cada produto após o nome (nome ao lado do preço, por exemplo: “Maçã: 10”).

