Curso Técnico Nível Médio Subsequente

# Informática Para Internet

## Fundamentos de Lógica e Algoritmos

**Aula 05**
Conceitos Fundamentais de Algoritmos e
Introdução à Programação em Python

*Bruno Emerson Gurgel Gomes*

Instituto Federal de Educação, Ciência e Tecnologia
do Rio Grande do Norte
Natal-RN
2015

Presidência da República Federativa do Brasil

Ministério da Educação

Secretaria de Educação a Distância

Este Caderno foi elaborado em parceria entre o Instituto Federal de Educação, Ciência e Tecnologia e o Sistema Escola Técnica Aberta do Brasil – e-Tec Brasil.


Equipe de Elaboração
**Cognitum**

Coordenação Institucional
**COTED**

Professor-autor
**Bruno Emerson Gurgel Gomes**

Projeto Gráfico
**Eduardo Meneses e Fábio Brumana**

Diagramação
**Georgio Nascimento**

Ficha catalográfica

G633c Gomes, Bruno Emerson Gurgel.

Curso Técnico Nível Médio Subsequente Informática para Internet : Fundamentos de Lógica e Algoritmos - Aula 05 : Conceitos fundamentais de algoritmos e introdução à programação em Python / Bruno Emerson Gurgel Gomes. – Natal : IFRN Editora, 2015.

22 f. : il. color.

1. Fundamentos de Lógica e Algoritmos - EaD. 2. Programação. 3. Linguagem de programação Python. 4. Algoritmos. I. Título.

RN/IFRN/EaD CDU 004.421

Ficha elaborada pela bibliotecária Edineide da Silva Marques, CRB 15/488

# Apresentação da disciplina

Prezado(a) aluno(a), na Unidade I você foi apresentado(a) aos fundamentos da lógica matemática. Lá, você aprendeu uma nova forma de organizar seu pensamento em um mundo de proposições, conectivos lógicos e regras bem estabelecidas. A partir deste momento, convido vocês a continuar nessa jornada, agora aplicando o raciocínio lógico para construir seus próprios programas de computador.

Imagine o quanto este aprendizado pode ser útil, não apenas no decorrer do seu curso, mas para a sua vida. Você já deve ter percebido o quanto os computadores estão presentes no nosso dia a dia, não é mesmo? Por exemplo, ao pagar uma conta em um restaurante, ao fazer pesquisas para um trabalho escolar em um *site* de busca e ao acessar nossas redes sociais e aplicativos favoritos usando o telefone celular.


Fonte: https://goo.gl/kOOYjl

**Figura 1**: Tirinha "O Grande Manual das Pequenas Ilusões"

Pois bem, caro(a) aluno(a). Em todo o nosso contato diário com os computadores, nós estamos na verdade lidando diretamente com programas, que também podem ser chamados de aplicativos ou software. Sendo assim, ao começar a compreender este universo a partir deste curso, você pode deixar de ser apenas um usuário leigo para, no mínimo, entender como funcionam os programas de computador. De fato, esperamos que este seja o pontapé inicial para que você possa desenvolver seus próprios aplicativos ou páginas de Internet. Tudo pronto, vamos começar?

# Aula 5 - Conceitos Fundamentais de Algoritmos e Introdução à Programação em Python

## Objetivos

Nesta aula são apresentados os conceitos básicos de algoritmos e de uma linguagem de programação. Ao término da aula, você deve criar e executar pequenos programas. Dessa forma, os objetivos dessa aula são:

Compreender as noções básicas de programação através de exemplos práticos do cotidiano;

Entender o conceito de algoritmos e a sua relação com programas de computador;

Criar e executar pequenos programas na linguagem de programação *Python*;

Aprender sobre o conceito de memória e como armazenar e modificar informações em um programa.

## Desenvolvendo o conteúdo

### Algoritmos e Programação de computadores

É importante destacar que construir um programa de computador real é uma tarefa que pode ser trabalhosa, a depender do fim a que se destina e da sua complexidade. Mas, assim como se constrói uma casa tijolo por tijolo, você vai aprender neste curso, de forma progressiva, as técnicas e ferramentas básicas para iniciar na criação de *software*. Com dedicação e o estudo de disciplinas complementares, você poderá em pouco tempo se tornar um programador.

Tipicamente, um programa é composto por centenas ou até milhares linhas de texto contendo a descrição passo-a-passo do problema que ele se propõe a resolver. Pois bem, a palavra "problema" aqui se aplica a uma necessidade de alguém que precisa ser atendida por meio de um programa de computador. Essa pessoa normalmente é um cliente que paga a uma outra pessoa (programador) ou empresa para adquirir o benefício que a resolução do problema irá trazer.

A essa altura, é possível que você pergunte "como eu posso escrever essas linhas de texto que formam um programa de computador?". Muito bem, caro(a) aluno(a), você deve escrever seus programas usando uma *linguagem de programação*. Essa linguagem deve ser de fácil compreensão pelo programador, mas deve ser estruturada de modo que possa ser traduzida em instruções que são entendidas e executadas pelo computador.

Assim sendo, você não deve escrever um texto em língua portuguesa e sim um texto que segue um conjunto bem definido de regras e instruções de uma determinada *linguagem de programação.* Observe que a escrita deve seguir esse padrão rigoroso, pois o computador só é capaz de entender sentenças bem formatadas, seguindo as regras da linguagem. Ele não tem a habilidade do ser humano de compreender e interpretar sentenças com ambiguidades ou erros de escrita.

Nesse contexto, é possível pensar em um programa de computador como sendo um conjunto de instruções descritas usando uma linguagem de programação. Não cabe aqui discorrer sobre os diversos tipos de linguagens. Na seção de leituras complementares, sugerimos referências para quem quiser se aprofundar no assunto. O importante aqui é entender que existem centenas delas disponíveis, cada uma criada pensando em resolver de forma adequada um determinado conjunto de problemas.

Nesse sentido, é possível dizer que existem linguagens que são melhores para a criação de *software* para computadores pessoais (*desktop*), outras para dispositivos móveis (celulares, tablets, etc.), e aquelas mais voltadas aos sistemas para Internet. No entanto, todas elas possuem diversos elementos em comum, como instruções para obter as informações que o programa necessita e um conjunto de estruturas para manipular essas informações e exibir um resultado para o usuário do programa.

## Conceito de Algoritmos

Inicialmente, devemos observar que um algoritmo não está relacionado apenas à ciência da computação ou à programação de computadores. De forma geral, um algoritmo é uma forma de descrever passo-a-passo a solução de algum problema. Desse modo, você está fazendo um algoritmo ao explicar a solução de um problema de física a um colega ou quando anota os ingredientes e o procedimento para fazer uma receita da sua comida favorita.

O interessante, ao se fazer um algoritmo, é que ele pode ser reproduzido por qualquer outra pessoa que tenha interesse no problema que ele se propõe a solucionar. Se você fizer um algoritmo, e ele estiver correto, qualquer pessoa que segui-lo fielmente deve obter como resultado a solução do problema. Isso é semelhante a você realizar o preparo de uma comida a partir da sua receita. Pois bem, caro(a) aluno(a), vamos apresentar nosso primeiro algoritmo através de uma receita simples e saborosa. Espero que você esteja com fome, pois vamos aprender o algoritmo para preparar pipoca.

### Algoritmo 1 - Preparo de pipoca

**Ingredientes**

- 1 (uma) xícara de chá de milho para pipoca
- 5 colheres de sopa de óleo
- Sal

Fonte: http://goo.gl/pAQovs

**Figura 2**: Pipoca

**Modo de preparo**

1. Despeje o conteúdo da xícara em uma panela fina e alta, cobrindo o fundo da panela, formando uma camada sem espaços;

2. Em seguida, acrescente o óleo de soja ou outro de sua preferência;

3. O milho deve ficar bem brilhante, mas não encoberto no óleo;

4. Em fogo alto, observe que as pipocas começam a estourar em até 2 minutos;

5. Assim que parar de fazer barulho (parar de estourar) a pipoca está pronta;

6. Retire com cuidado a pipoca da panela e despeje em uma bacia;

7. Acrescente um pouco de sal e prove;

8. Caso necessário, acrescente mais um pouco de sal até ficar ao seu gosto.

Fonte: adaptado de http://www.tudogostoso.com.br/receita/32450-pipoca-salgada.html]

No Algoritmo 1, duas partes estão em destaque, os ingredientes e o modo de preparo. Os ingredientes enfatizam o que é necessário para preparar nossa pipoca, ou seja, as "informações" que você precisa ter em mãos antes de começar a resolver o problema. O modo de preparo descreve, em um conjunto de passos, no caso 8 (oito), a solução do problema. Nesse caso, tem-se o algoritmo propriamente dito. Ao seguir esses passos na ordem em que são listados, usando os ingredientes nas quantidades recomendadas, é esperado que você obtenha ao final uma deliciosa pipoca.

O exemplo é bastante simples, porém útil para nos ajudar a entender diversos conceitos importantes. Primeiro, um algoritmo de fato descreve a solução de um problema, ou de parte dele, em um ou mais passos. Esses passos devem seguir uma ordem lógica, do primeiro passo até o último. O algoritmo pode ficar incorreto caso um passo seja adiantado ou suprimido. Por exemplo, se você esquecer de colocar o óleo (passo 3) boa parte das pipocas irá queimar.

Outra observação importante é que, após ser iniciado, o algoritmo deve sempre terminar. Ou seja, deve existir um passo final, que pode até mesmo ser uma mensagem indicando algum erro previsível no decorrer do processo. Essa parte de tratar e reportar um erro ao usuário será detalhada em aulas posteriores.

A partir do que vimos, é interessante definir o que é um algoritmo aplicado à solução de um problema computacional, como sendo:

### Algoritmo

Um algoritmo é a descrição da solução de um problema em uma sequência ordenada e finita de passos.

Um programa de computador geralmente é composto por um ou vários algoritmos. É possível, e desejável, inicialmente descrever a ideia de forma geral por meio de um algoritmo e só depois traduzir esse algoritmo em um programa. Desse modo, você pode raciocinar melhor sobre o problema e a solução pode sair mais rápido ou ser de melhor qualidade. O algoritmo também é uma forma de comunicar a sua solução para outros programadores de modo independente de linguagem.

É importante destacar que, muitas vezes, existe mais de um caminho que leva à solução de um mesmo problema. Ou seja, é possível ter mais de um algoritmo que descreva o problema. Nesse caso, um algoritmo pode ser mais ou menos eficiente que outro em termos de tempo de conclusão da tarefa ou recursos consumidos pelo computador. No contexto desta disciplina, não importa se você fez um algoritmo melhor ou pior, com mais ou menos passos. O importante é chegar, de maneira correta, à solução proposta. Sugerimos leituras no material complementar para quem quiser se aprofundar no estudo dos algoritmos.

## ATIVIDADE

**1.** Altere o algoritmo de preparação de pipoca, deixando-o mais detalhado. Nesse caso, insira ao menos mais 3 (três) passos no algoritmo.

**2.** Faça um algoritmo que descreva a troca do pneu de um carro. Se você nunca trocou um pneu, pesquise na internet os passos necessários.

**a)** a. Faça o algoritmo considerando que você tem certeza que o pneu já está furado e, a partir daí, precisa trocá-lo.

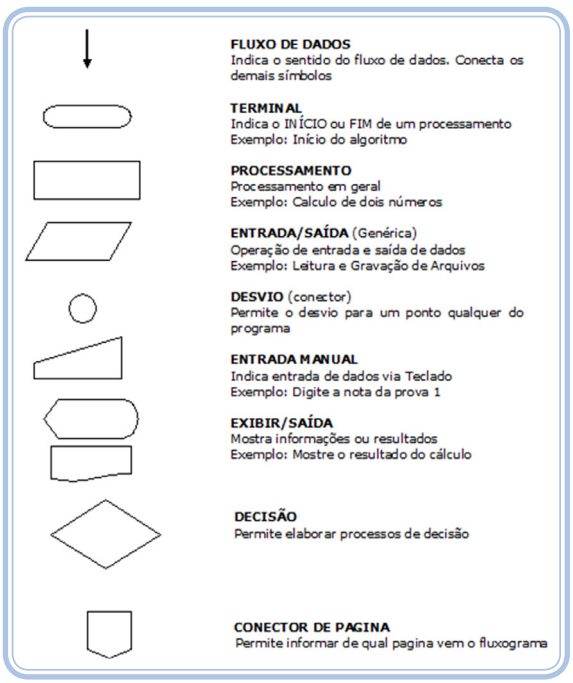
**b)** b. Considere incluir um ou mais passos para verificar se o pneu realmente está furado ou apenas precisando ser calibrado.

## Formas de Representação de Algoritmos

Há diversas formas de representar um algoritmo. Inicialmente, no exemplo da preparação de pipoca (Código 1), usamos a narrativa em linguagem natural. Outras formas são a descrição por meio de figuras e fluxos de dados usando os diagramas de blocos e o uso do pseudocódigo ou português estruturado.

Cada forma tem seus prós e contras. No caso da linguagem natural, o lado positivo é que nós já conhecemos a linguagem que iremos trabalhar, que vem a ser a língua portuguesa escrita. A desvantagem é que as sentenças escritas em português podem levar a interpretações ambíguas ou deixar lacunas no entendimento. Por exemplo, no caso do algoritmo de preparação de pipoca, o que seria exatamente "acrescentar um pouco de sal"?. O pouco para uma pessoa pode significar muito para outra.

Os diagramas blocos usam figuras para representar processos comuns em algoritmos, como entrada de dados, fluxo e processamento de informações, tomadas de decisão e saída de dados (Figura 2). Essa forma de representação é interessante para algoritmos pequenos por facilitar a visualização do fluxo de informações. No entanto, para problemas mais complexos, pode se tornar difícil acompanhar o fluxo. Modificar o algoritmo também pode exigir certo esforço para reorganizar o diagrama.




Fonte: http://goo.gl/OzLYbR

**Figura 3**: Formas utilizadas em um diagrama de blocos

A forma mais comum de se representar algoritmos é por meio de pseudocódigo, também conhecido como *português estruturado* ou *portugol*. Ela estabelece uma linguagem bem estruturada, intermediária entre a linguagem natural e uma linguagem de programação. As instruções que são comumente encontradas nessas linguagens estão de certo modo presentes no pseudocódigo. Assim, ela se torna um modelo genérico que você programador pode traduzir no código final de uma linguagem de programação de sua preferência.

A desvantagem em usar o pseudocódigo é que não há um modelo específico que seja usado por todos. Embora, em geral, o código seja compreensível, cada um estabelece sua notação de pseudocódigo. Assim, é preferível usar essa forma de representação quando você deseja comunicar um algoritmo de forma independente de linguagem ou quando você quer rascunhar uma ideia antes de partir para a programação. Esse comportamento de esboçar uma ideia é bem interessante, pois permite a você pensar sobre os passos do algoritmo, a entender melhor o que você quer alcançar.

Como exemplo, temos um algoritmo que recebe dois números e diz se o primeiro número é maior que o segundo (Algoritmo 2).

### Algoritmo 2 – Recebe dois números e diz se o primeiro número é maior que o segundo

```
1. início
2.     leia (n1, n2)
3.     se n1 > n2 então
4.         escreva (n1, " é maior que", n2)
5.     fimse
6. fim
```

Em um primeiro momento, peço a você que não se assuste com o código do Algoritmo 2. Vamos entendê-lo em todos os seus detalhes em poucas aulas. As linhas do algoritmo foram numeradas para facilitar a citação no texto, um procedimento que iremos seguir daqui em diante. Na *linha 1*, temos

a palavra "início", ela demarca o começo do algoritmo. A seguir, na *linha 2*, estamos colhendo os dados que serão usados no algoritmo, no caso, o primeiro número (n1) e o segundo número (n2). Na *linha 3*, é feito um teste para saber se *n1* é maior que *n2*. Caso esse teste seja verdadeiro, o texto da *linha 4* é impresso. A linha 6 demarca o fim do algoritmo.

## Linguagem de Programação Python

Neste curso, nós decidimos por iniciar a atividade de programação diretamente em uma linguagem de programação. Uma motivação para isso é que você poderá visualizar, na prática, o resultado dos seus programas. O ambiente de programação que iremos usar destaca em cores as palavras que são parte da linguagem, permite executar o programa e aponta erros no código.

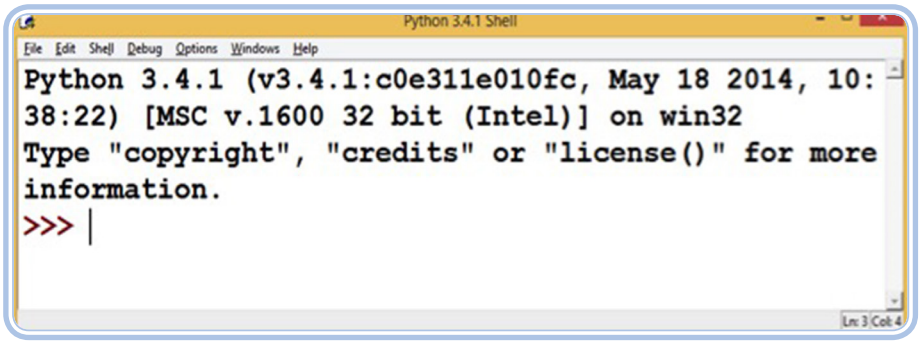
Por motivos didáticos, a linguagem escolhida para uso foi o *Python* (PYTHON, 2015), uma linguagem bastante simples, poderosa e popular. Ela pode ser aplicada tanto no desenvolvimento de programas para *desktop* quanto para a Internet. *Python* é excelente como uma primeira linguagem por possuir uma sintaxe simples, direta e de fácil aprendizado. Uma vez que as instruções nessa linguagem são formadas por palavras em Inglês, sempre que necessário essas instruções serão explicadas a partir dos termos equivalentes em Português.

Na seção de leituras complementares você encontra um tutorial que lhe ensina a instalar e usar o ambiente *Python* para começar a programar. Para os mais apressados, é possível seguir as instruções de download e instalação do ambiente no *site* oficial da linguagem, <http://www.python.org/downloads/>. É importante observar que este livro usa a versão 3 do *Python*. Esse detalhe merece destaque, pois essa versão apresenta algumas diferenças em relação a versão 2. Portanto, quando for fazer o *download* do ambiente, não esqueça de baixar a versão 3 ou uma posterior.

## Introdução ao interpretador Python

O nosso primeiro contato com a linguagem *Python* será através de um programa chamado *idle*, que vem a ser um interpretador para a linguagem. Esse programa, como o próprio nome sugere, é responsável por interpretar cada instrução que você digitar e, caso ela esteja correta, realizar alguma ação no programa.

Inicialmente, vamos imaginar o ***Python*** como sendo uma calculadora. Após ter instalado o *Python*, abra o programa *idle*. Irá aparecer uma janela semelhante à Figura 3.



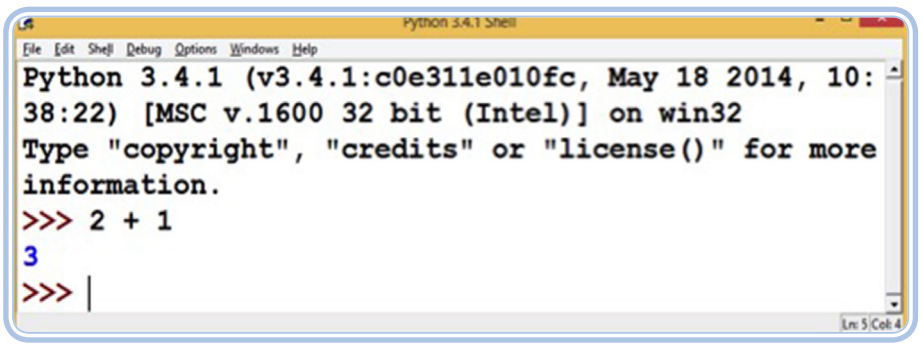

Fonte: Autoria própria.

**Figura 4**: Janela do interpretador *Python* (*idle*)

Na janela da Figura 3, o sinal ">>>" no interpretador Python indica que ele está aguardando que você digite um comando. Após digitá-lo, ao apertar o botão ENTER, o comando vai ser processado pelo interpretador e o resultado será exibido para você logo abaixo.

Observe que o símbolo (">>>") será utilizado no texto sempre que for necessário representar uma entrada de um valor no interpretador Python. Esse símbolo não é um comando da linguagem, apenas indica que você está no interpretador.

Tudo certo, agora vamos digitar um comando. No caso, vamos realizar uma soma entre dois números. No interpretador, digite 2 + 1 (O número 2, seguido de "+" e do número 1). Agora aperte o botão ENTER e veja o resultado, conforme a Figura 4. Nesse exemplo, 2 e 1 são valores constantes e o símbolo "+" é o operador para adição entre dois números. O interpretador lê o seu comando, calcula o resultado e exibe para você na linha abaixo (na cor azul). A partir daí ele fica pronto para receber outro comando (veja o símbolo ">>>").

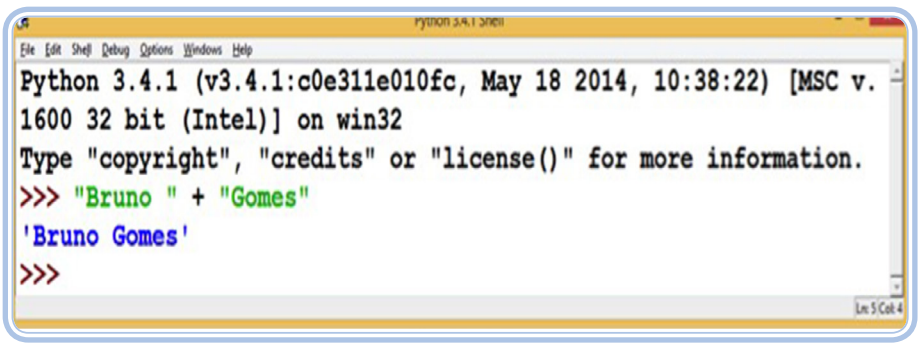



Fonte: Autoria própria.

**Figura 5**: Resultado da interpretação da expressão (2 + 1)

Você pode tentar uma expressão mais elaborada, como 2 * (10 / 3) + 1. As expressões são abordadas em detalhes na aula 2. Na linguagem, é possível também trabalhar com textos. No caso, se você digitar um texto qualquer, o interpretador irá gerar como resultado o próprio texto. A próxima seção descreve como salvar um valor, seja texto ou número para que possamos usar posteriormente no programa.

Perceba que, em *Python*, o operador "+" também pode ser usado com textos. Certo, mas como assim? Ele irá "somar" o texto? Não, ao ser aplicado a um ou mais textos, o símbolo de "+" tem o papel de juntá-los. Para ver como isso funciona, você pode tentar juntar o seu nome e sobrenome, como "Bruno " + "Gomes" (Figura 5). Observe que foi deixado, propositalmente, um espaço em branco no primeiro nome. Isso foi feito para que os nomes ficassem separados por um espaço.




Fonte: Autoria própria.

**Figura 6**: Operador "+" aplicado a dois textos

## Variáveis e Atribuição de Valores

Na seção anterior vimos como inserir valores e instruções básicas no interpretador *Python*. Conforme explicamos, ao terminar de digitar no interpretador *Python* e clicar com o botão ENTER, o interpretador executa a instrução digitada e devolve o resultado. Ocorre que, nesse caso, o valor lido não fica "salvo" em nenhum lugar.

É importante destacar, caro(a) aluno(a), que na maioria das vezes, você está lendo um valor para usar posteriormente no programa. Esse valor pode ser, por exemplo um nome, uma data, ou um ou mais números para se usar em um cálculo matemático.

Pois bem, então como faremos para salvar um valor qualquer no nosso programa? Nesse caso, podemos salvar o valor em uma variável. Mas, o que

vem a ser uma variável? Inicialmente, vamos pensar em uma variável como uma gaveta, ou seja, um lugar onde guardamos algum objeto, como uma escova de cabelo. Sempre que for preciso pentear os cabelos, podemos abrir a nossa gaveta e pegar a escova. Caso a escova se quebre, é possível adquirir uma nova e substituir a anterior na gaveta.

No caso dos nossos programas de computador, as "gavetas" são representadas por espaços na memória do computador. Ou seja, ao criar uma variável, na verdade, estamos reservando uma gaveta na memória para guardar uma informação que o nosso programa irá precisar. Criar uma variável é bastante simples. Na verdade, basta apenas que você dê um nome a ela e armazene um valor.


Fonte: https://goo.gl/eKmyTj

**Figura 7**: Variável

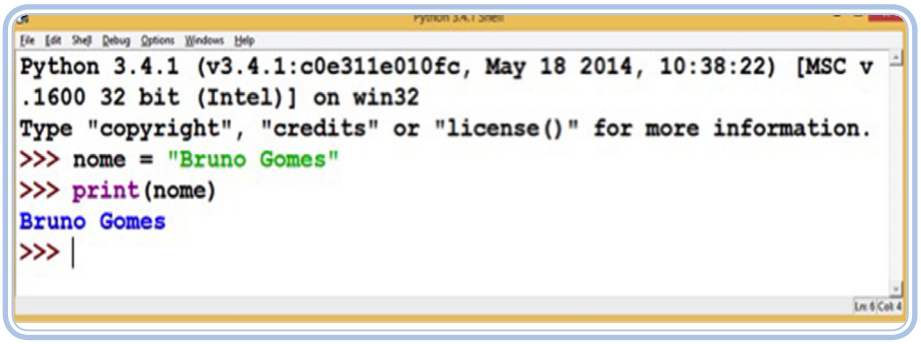
No nosso exemplo, podemos criar uma variável denominada "nome" para receber o um valor de texto com o nome de uma pessoa. Assim, o que for digitado no teclado ficará armazenado na variável *nome*, conforme pode ser visto no Código 1.

## Código 1 – Salva um texto na variável nome

```
>>> nome = "Bruno Gomes"
```

Observe que o símbolo de "=" (igualdade), denominado de *atribuição*, tem uma função especial na linguagem. É com esse símbolo que dizemos que estamos querendo salvar (ou atribuir) o valor que está à direita dele, no caso o texto "Bruno Gomes", na variável que está à esquerda.

Se você der um *print* na variável nome, conforme a Figura 6, você tem como resultado o conteúdo que se encontra armazenado na variável. *Print* é um comando do **Python** utilizado para gerar uma saída na tela para o usuário do programa.




Fonte: Autoria própria.

**Figura 8**: Salvando um texto em uma variável e imprimindo o resultado

Observe que você pode fornecer qualquer nome a uma variável, desde que ele comece com uma letra ou *underscore* (_) e seja seguido por letra, número ou *underscore*. Um nome de variável não pode começar com um número, nem possuir caracteres especiais ou espaços em branco. Outra restrição é que não é permitido usar um nome já definido na linguagem, como *print* ou *input*.

Embora uma variável possa receber qualquer nome dentro desses requisitos, é uma boa prática que você dê um nome coerente com o que a variável irá armazenar. Por exemplo, se a variável armazena um nome, eu não vou nomeá-la de "casa", ou de "computador", e sim de "nome" ou "pessoa", a depender do contexto em que ela será utilizada.

Para facilitar a manipulação no programa, geralmente não colocamos acentos nos nomes de variáveis, embora seja permitido em *Python*. Uma outra observação é que nomes de variáveis são sensíveis à caixa. Ou seja, a variável "*pessoa*" é diferente das variáveis "*Pessoa*" e "*PESSOA*".

Sugiro que você mantenha um padrão de escrita de nomes de variáveis, possivelmente colocando todos os nomes em caixa baixa (letras minúsculas). No caso de uma variável possuir mais de um nome, você pode começar a letra da segunda palavra em diante em maiúscula ou separar as palavras com *underscore* como "*nomePessoa*" ou "*nome_pessoa*".

## ATIVIDADE

1. Com a ajuda do interpretador Python, verifique qual desses nomes de variáveis é válido. Para testar, atribua um valor qualquer a cada nome de variável.

**a.** cidade

**b.** __Cidade

**c.** 2nome

**d.** idade pessoa

**e.** idadePessoa

**f.** CPF

**g.** len

**h.** &endereço

**i.** _rua_

**j.** x

Como o próprio nome sugere, e você já deve ter percebido, uma variável é um valor que pode mudar sempre que necessário. Desse modo, se você quiser trocar o valor armazenado na variável "*nome*", é só modificar o texto da atribuição. Por exemplo, se você digitar no interpretador a instrução do Código 2, o valor armazenado na variável "nome" será atualizado para o texto "Thiago Medeiros".

## Código 2 – Atualizando o valor da variável "nome"

```
>>> nome = "Thiago Medeiros"
```

No caso, usamos um valor constante do tipo texto, que é representado por qualquer valor entre aspas (" ") ou apóstrofo (' '). Tanto faz você inserir "Thiago Medeiros" ou 'Thiago Medeiros', desde que você seja coerente. Se começou com aspas, termine com aspas. O mesmo vale para o apóstrofo.

**LEMBRE-SE**

O símbolo ">>>" (três sinais de maior seguidos) neste caso não faz parte do comando. Ele indica apenas que você está no interpretador *Python*.

Nesta seção, definimos como exemplo uma variável "nome" que recebe um texto. A próxima aula apresenta outros tipos de dados básicos que podem ser usados nos nossos programas. Na aula seguinte, também vamos detalhar as instruções de entrada e saída de dados, *input* e *print*, respectivamente, e aprender a criar e interpretar expressões mais complexas, formadas por variáveis e operadores de diversos tipos.

## RESUMINDO

Nesta aula, vimos os conceitos básicos da programação de computadores. Apresentamos conceito de algoritmos e as suas formas de representação em linguagem natural, diagrama de blocos e pseudocódigo. Iniciamos o estudo da programação com a linguagem *Python* usando seu interpretador para o desenvolvimento e execução dos nossos primeiros códigos. Em um primeiro momento, aprendemos a usar o interpretador como se fosse uma calculadora. Posteriormente, vimos que através do uso de variáveis, é possível armazenar informações no nosso programa. As variáveis são nomes que podem receber valores e guardá-los para uso posterior no programa.

## LEITURAS COMPLEMENTARES

**Iniciação aos algoritmos e à programação**

*Code.org* (https://code.org) – Esta página apresenta diversos materiais para de introdução a programação. Sugiro que você faça a "hora do código", um jogo bem interessante que exercita a lógica de programação.

*Code Academy* (http://www.codecademy.com) – Site que se propõe a ensinar programação em diversas linguagens, incluindo Python. O *site* fornece uma interface para você praticar online enquanto aprende cada lição.

*Algoritmos* (MANZANO; OLIVEIRA, 2014) – Este livro apresenta uma introdução interessante dos conceitos iniciais de algoritmos e linguagens de programação e sobre os diversos paradigmas de programação.

**Linguagem de programação Python**

*Instalação do ambiente Python* – Você pode obter ajuda para instalar o ambiente de programação Python de diversas fontes. No Youtube, o vídeo https://www.youtube.com/watch?v=wpqkZJ10Gmo ensina a instalação do Python 3 para o Windows 7. O *site Python Club* (http://pythonclub.com.br/instalacao-python-django-windows.html) explica a instalação de forma textual. Observe que ele demonstra a instalação do *Python 2.7*, mas você deve obter a versão 3, mais recente.

*Site oficial da linguagem* (https://www.python.org) – Apresenta a linguagem, sua documentação oficial e permite o download do ambiente de programação.

*Python para Zumbis* (http://pycursos.com/python-para-zumbis) – Curso online gratuito da linguagem Python através de vídeo-aulas.

# AVALIANDO SEUS CONHECIMENTOS

1. O que é um programa de computador?

2. Qual(is) a diferença (as) entre um programa de computador e um algoritmo?

3. Resolva o seguinte problema de lógica por meio de um algoritmo em linguagem natural:

> Três jesuítas e três canibais precisam atravessar um rio. Para tanto, dispõem de um barco com capacidade para apenas duas pessoas. Por medidas de segurança, não se deve permitir que em alguma margem do rio a quantidade de jesuítas seja inferior a de canibais. Elabore um algoritmo indicando os passos necessários para que a travessia seja feita de forma segura.

4. Imagine que você acionou o interruptor da lâmpada do seu quarto e ela não ligou. Faça um algoritmo que descreva o procedimento para realizar a troca da lâmpada.

5. Verifique o que está errado nos códigos a seguir e faça as devidas correções.

**a.** >>> print 'Instituto Federal"

**b.** >>> nome$ = 10

6. Qual o valor da variável "x" ao final da execução do código a seguir?

```
>>> x = 37

>>> x = x + 5

>>> x  = x – 2
```

# Referências


CODE.ORG. Disponível em: <https://code.org/>. Acesso em: 05 fev. 2015.

CODE ACADEMY. Disponível em <http://www.codecademy.com/>. Acesso em: 09 fev. 2015.

CORMEN, T. H. et al. **Algoritmos**: Teoria e Prática. 3. ed. Rio de Janeiro: Campus, 2012.

CORÔA, T. **Instalando e Configurando o Python e Django no Windows**. 2014. Disponível em: <http://pythonclub.com.br/instalacao-python-django-windows.html>. Acesso em: 29 abr. 2015.

FORBELLONE, A. L. V.; EBERSPACHER, F. H. **Lógica de Programação**: A construção de Algoritmos e Estruturas de Dados. 3. ed. São Paulo: Pearson, 2005.

MANZANO, J. A. N. G.; OLIVEIRA, J. F. **Algoritmos**: Lógica para Desenvolvimento de Programação de Computadores. São Paulo: Érica, 2014.

MASANORI, F. **Python para Zumbis**. [2013]. Disponível em: <http://pycursos.com/python-para-zumbis/>. Acesso em: 05 fev. 2015.

MEDINA, M.; FERTIG, C. **Algoritmos e Programação**: Teoria e Prática. 2. ed. São Paulo: Novatec, 2006.

PYTHON. Disponível em: <https://www.python.org/>. Acesso em: 05 fev. 2015.

PYTHON BRASIL. Disponível em: <http://wiki.python.org.br/>. Acesso em: 05 fev. 2015.

REIS, F. **Lógica de programação**: Vídeo 03: Instalação do ambiente Python 3.3 e IDLE no Windows 7. [2013]. Disponível em: <https://www.youtube.com/watch?v=wpqkZJ10Gmo>. Acesso em: 29 abr. 2015.

SEVERANCE, C. **Python for Informatics**: Exploring Information. 2013. Disponível em: <http://www.pythonlearn.com/book.php>. Acesso em: 05 fev. 2015.

SZWARCFITER, J. L.; MARKENZON, L. **Estruturas de Dados e Seus Algoritmos**. 3. ed. São Paulo: LTC, 2010.