**Introdução da unidade**



**Objetivos da Unidade**

Ao longo desta Unidade, você irá:

* examinar a linguagem *Python;*
* identificar como ocorrem as estruturas de comando da linguagem;
* explicar as funções em *Python.*

**Introdução da Unidade**

print("hello world!")

hello world!

Diz a lenda dos programadores que, se você não imprimir o “[hello world](https://cienciacomputacao.com.br/curiosidade/a-origem-do-hello-world/)” quando começar a aprender uma linguagem de programação, você não conseguirá aprender nada da linguagem em questão. Mito ou verdade? Por via das dúvidas, essa foi nossa primeira linha de comando, de muitas que aprenderemos nessa disciplina.

Como você pode ver, um comando de sintaxe "limpa", sem ponto e vírgula, sem colchetes, sem importação de bibliotecas é a essência da linguagem de programação *Python*, que foi criada para ter comandos e estruturas simples, de fácil leitura e compreensão.

*Python*foi criado no início dos anos 1990 por Guido van Rossum no *Stichting Mathematisch Centrum*(CWI), na Holanda, como sucessor de uma linguagem chamada ABC. Guido é o principal autor do Python, embora haja muitas contribuições de outros pesquisadores [(](https://docs.python.org/3.0/license.html)*[Python Reference Manua](https://docs.python.org/3.0/license.html)*[l, 2020).](https://docs.python.org/3.0/license.html)

Guido passou por outras instituições, como a CNRI, em 1995, na Virginía (Corporação para Iniciativas Nacionais de Pesquisa) e na *BeOpen*, em 2000, onde formou a *BeOpen PythonLabs*. Em outubro do mesmo ano, a equipe do *PythonLabs*mudou-se para a *Digital Creations* (agora *Zope Corporation*).

Em 2001, a*Python Software Foundation* (PSF) foi formada, uma organização sem fins lucrativos criada especificamente para possuir a propriedade intelectual relacionada ao *Python*.

É sobre esta linguagem que vamos nos debruçar nesta unidade, está pronto para começar?

Então vamos lá!

**Introdução da aula**



**Qual é o foco da aula?**

Nesta aula, vamos conhecer o poder dessa linguagem de programação.

Objetivos gerais de aprendizagem

Ao longo desta aula, você irá:

* esclarecer o que é linguagem *Python*;
* examinar o papel do interpretador *Python*
* identificar ferramentas que auxiliam o desenvolvimento;

**Situação-problema**

No mercado de trabalho, existem diversas empresas que construíram seu modelo de negócio baseado na prestação de serviço especializado para outras empresas e são chamadas de "consultorias". Dentre essas empresas de consultoria, há uma grande procura pelas que desenvolvem *software*, assim a empresa contratante não precisa ter uma área de desenvolvimento interna, ela contrata a consultoria e faz a encomenda da solução que necessita.

Como seu primeiro trabalho de desenvolvedor em uma empresa de consultoria de *software*, você foi designado para atender um cliente que fabrica peças automotivas e criar um protótipo da solução que ele necessita. O cliente relata que tem aumentado o número de peças e que gostaria de uma solução que fosse capaz de prever quantas peças serão vendidas em um determinado mês. Esse resultado é importante para ele, uma vez que dependendo da quantidade, ele precisa contratar mais funcionários, reforçar seu estoque e prever horas extras.

O cliente enviou para você o relatório de vendas dos últimos 6 meses (figura abaixo). Agora você precisa analisar o gráfico, pensar no algoritmo que, a partir das informações no gráfico, seja capaz de prever quantas peças serão vendidas em um determinado mês. Por exemplo, considerando o mês de janeiro como o primeiro mês, ele vendeu x peças, em fevereiro (segundo mês) ele vendeu n peças, quantas peças ele vai vender no mês 10, e no mês 11 e no mês 32? Por se tratar de um protótipo, você deve utilizar somente as informações que lhe foram cedidas, não precisa, nesse momento, analisar o comportamento de fatores externos, por exemplo, comportamento da bolsa de valores, tendência de mercado, etc.

Relatório de vendas. Fonte: elaborada pela autora.

Você precisa escolher qual ferramenta de trabalho irá adotar para criar o protótipo, em seguida implementar o algoritmo que faça a previsão usando a linguagem de programação *Python*.

**Por que usar Python?**



Segundo o guia de desenvolvimento para iniciantes *Python*[(Beginners Guide Overview)](https://wiki.python.org/moin/BeginnersGuide/Overview), *Python*é uma linguagem de programação orientada a objetos, clara e poderosa, comparável a Perl, Ruby, Scheme ou Java. Podemos destacar algumas características que tornam essa linguagem notável:

* utiliza uma sintaxe elegante, facilitando a leitura dos programas que você escreve.
* é uma linguagem fácil de usar, o que torna o *Python*ideal para o desenvolvimento de protótipos e outras tarefas de programação ad-hoc, sem comprometer a manutenção.
* vem com uma grande biblioteca padrão que suporta muitas tarefas comuns de programação, como se conectar a servidores da *Web*, pesquisar texto com expressões regulares, ler e modificar arquivos.
* possui inúmeras bibliotecas que estendem seu poder de atuação.
* é uma linguagem interpretada, ou seja, uma vez escrito o código, este não precisa ser convertido em linguagem de máquina por um processo de compilação.
* permite atribuição mútlipla. Podemos atribuir valores a mais de uma variável em uma única instrução. Por exemplo, a, b = 2, 3.

Uma das grandes características da linguagem é sua sintaxe. Uma das principais ideias de Guido é que o código é lido com muito mais frequência do que está escrito *[(PEP 8 - Style Guide for Python Cod](https://www.python.org/dev/peps/pep-0008/" \l "introduction" \t "_blank)*[e)](https://www.python.org/dev/peps/pep-0008/" \l "introduction" \t "_blank). Tal aspecto é tão relevante, que um código que segue as regras do idioma *Python*é chamado de “*pythonic code*”. Essas regras são definidas pelo PEP 8 (*Python Enhancement Proposal*) e dizem respeito a formatação, indentação, parâmetros em funções e tudo mais que possa estar relacionado à sintaxe do código.

*Python*tem se mostrado uma linguagem muito poderosa e está sendo amplamente adotada por profissionais na área de dados.

O interpretador *Python*3 utiliza *unicode*por padrão, o que torna possível usar nomes de variáveis com acento e até outros caracteres especiais, porém não é uma boa prática. Tal flexibilidade mostra porque a linguagem tem sido adotada por pessoas que não são da área de exatas, pois a leitura está mais próxima da interpretação humana do que da máquina

**Instalação do interpretador Python**

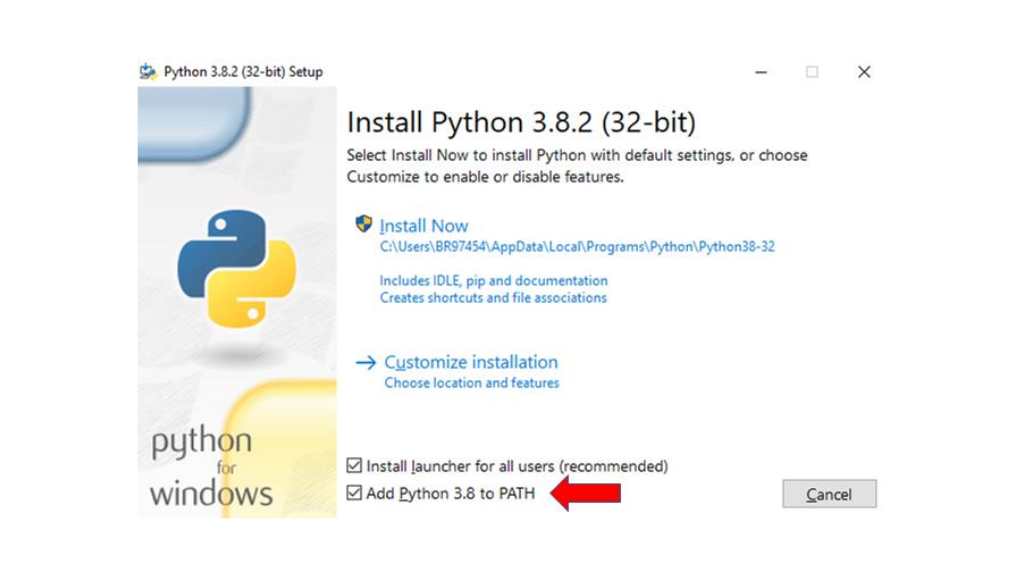


Agora que já conhecemos um pouco dessa importante linguagem e já sabemos que existe até um “código” de escrita determinado pelo PEP 8, podemos nos perguntar: mas onde eu escrevo os códigos em *Python*e vejo os resultados? A implementação de códigos em *Python*pode ser feita tanto em ferramentas instaladas no seu computador quanto em ambientes em nuvem.

Nesse universo de possiblidades, o primeiro ponto que você precisa entender é que, para obter os resultados de um código, precisamos ter um interpretador *Python*. Vamos começar analisando as opções que podem ser instaladas.

Na[página oficial da linguagem](https://www.python.org/downloads/) , podemos encontrar as várias versões do interpretador *Python*puro (sem bibliotecas de terceiros), disponíveis para diversos sistemas operacionais. Para instalar, basta escolher o sistema operacional, fazer o *download*da versão mais atual do interpretador 3.X e, após concluído o *download*, clique no arquivo executável que foi descarregado do repositório oficial.

No processo de instalação do interpretador *Python*, é preciso marcar a opção *Add Python 3.X to PATH* (figura a seguir). Ao marcar essa opção, a variável de ambiente *Path*é configurada para entender o comando “*python*”. Caso não seja marcada a opção, você precisará configurar manualmente depois.

Processo de instalação do interpretador. Fonte: captura de tela do instalador do Python.

Ao concluir a instalação do interpretador, se você marcou a opção “*Add Python 3.X to PATH*”, já podemos começar a programar através do modo iterativo. Para isso, abra o prompt de comando do *Windows*e digite “*python*” (figura abaixo). Após aberto o modo iterativo, já podemos digitar comandos *python*e, ao apertar “enter”, o comando imediatamente é interpretado e o resultado é exibido.

Modo de programação iterativo. Fonte: captura de tela do prompt de comando.

Caso não queira navegar até o *prompt*de comando, você também pode utilizar o aplicativo *IDLE*, já pré-instalado no *Windows*.

**Ide's Pycharm E Visual Studio Code**



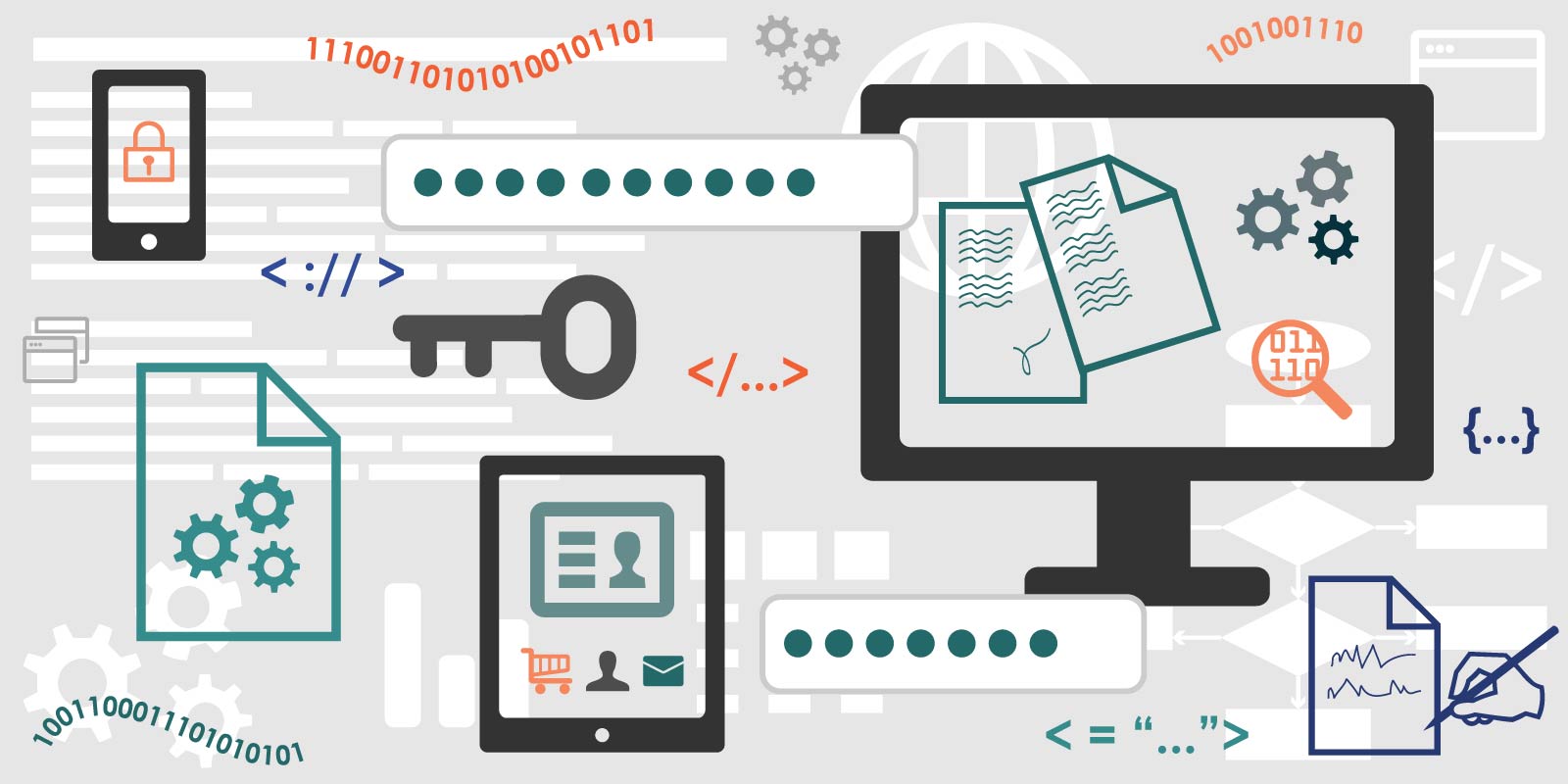
Utilizar o modo iterativo é muito simples e recomendado para testes rápidos, porém essa forma de implementação não é aplicável quando precisamos criar um código que será executado mais de uma vez. Isso acontece porque no modo iterativo, ao fechar o *prompt*, todo o código é perdido.

Para implementação de soluções (e não somente testes rápidos), nós programadores usamos uma IDE, (*Integrated Development Environment*) que traduzindo significa Ambiente de Desenvolvimento Integrado.

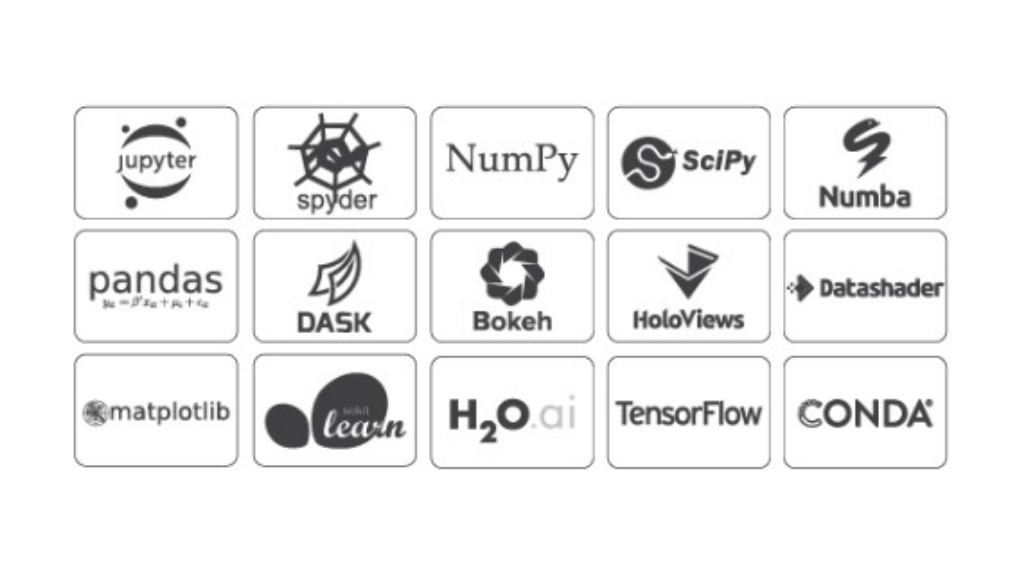
Esse tipo de *software*possui uma série de ferramentas que auxiliam o desenvolvimento, como integração com sistemas de versionamento de código, refatoração de código, debug, etc. No mercado, duas IDE’s disputam a preferência dos desenvolvedores *Python, o [PyCharm](https://www.jetbrains.com/pycharm/" \t "_blank)* e o *[Visual Studio Code](https://visualstudio.microsoft.com/pt-br/" \t "_blank)* (VSCode)[.](https://visualstudio.microsoft.com/pt-br/)

O *PyCharm* é oferecido em duas opções: *Professional e Community*, sendo a primeira paga e a segunda gratuita. Portanto, caso opte por essa IDE, você deverá fazer o *[download](https://www.jetbrains.com/pycharm/download/" \t "_blank)*da versão *Community*no site . Já o *[VSCode](https://visualstudio.microsoft.com/pt-br/" \t "_blank)*é gratuito e pode ser encontrado no site.

**Projeto Python Anaconda**



Para quem está começando a aprender *Python*, uma opção de ferramenta de trabalho é o projeto *[Python](https://www.anaconda.com/distribution/" \t "_blank)*[Anaconda](https://www.anaconda.com/distribution/" \t "_blank) . O projeto Anaconda consiste na união de diversas ferramentas *Python*, compostas por bibliotecas e IDE’s figura abaixo).

Ferramentas disponíveis no projeto Anaconda. Fonte: captura de tela de Anaconda (2020).

Fazendo a instalação do projeto Anaconda, você passa a ter tanto o interpretador *Python*quanto várias bibliotecas, além de duas interfaces de desenvolvimento: a IDE *spyder*e o projeto *Jupyter*. A IDE *spyder*, é similar ao *PyCharm*e ao *VSCode*, ou seja, é um ambiente de desenvolvimento com várias funcionalidades integradas.

Um dos grandes diferenciais do projeto Anaconda é ter o *[Jupyter Notebook](https://jupyter.org/" \t "_blank)*  integrado na instalação.

O *Jupyter Notebook* é um ambiente de computação interativa, que permite aos usuários criar documentos de notebook que incluem: código ativo, gráficos, texto narrativo, equações, imagens e vídeo. Esses documentos fornecem um registro completo e independente de uma computação (um código) que pode ser convertida em vários formatos e compartilhada com outras pessoas usando email, *Dropbox*, sistemas de controle de versão (como git / GitHub). Uma das grandes vantagens do *Jupyter Notebook* é funcionar em um navegador de internet.

No endereço *[Jupyter](https://jupyter.org/try" \t "_blank)*, você pode experimentar a ferramenta, sem precisar de instalação. Porém, instalando o projeto *Python*Anaconda, essa ferramenta já é instalada e você pode acessar fazendo a busca por *Jupyter Notebook* nos aplicativos do seu sistema operacional.

Em um *Jupyter Notebook*, o programador pode escrever trechos de códigos em células e, assim que executa a célula, o trecho é interpretado e o resultado aparece logo abaixo, da mesma forma como fizemos no primeiro comando *print*("hello world").

\_\_\_\_\_

**➕ Pesquise mais**

No canal do Caio Dallaqua, você encontra um vídeo “[Curso](https://www.youtube.com/watch?v=m0FbNlhNyQ8" \t "_blank)*[Python](https://www.youtube.com/watch?v=m0FbNlhNyQ8" \t "_blank)*[3 | Aula 1 -](https://www.youtube.com/watch?v=m0FbNlhNyQ8" \t "_blank)*[Notebook Jupyter](https://www.youtube.com/watch?v=m0FbNlhNyQ8" \t "_blank)*” de 2 minutos, com uma rápida introdução ao uso dessa ferramenta.

**Google Colaboratory (colab)**



[*Colaboratory*](https://research.google.com/colaboratory/faq.html), ou "*Colab*", é um produto da Pesquisa do Google. O *Colab*permite que qualquer pessoa escreva e execute código *Python*através do navegador. Além disso, é especialmente adequado para aprendizado de máquina, análise de dados e educação. Mais tecnicamente, o *Colab*é um serviço de *notebook Jupyter* hospedado que não requer configuração para ser usado, além de fornecer acesso gratuito a recursos de computação, incluindo GPUs.

O *Colab*é baseado no projeto de código aberto *Jupyter*. Portanto, o *Colab*permite que você use e compartilhe os notebooks *Jupyter*com outras pessoas sem precisar baixar, instalar ou executar nada. Ao criar um *Jupyter Notebook*no *Colab*, você pode compartilhar o link com pessoas específicas, ou então, compartilhar o trabalho no *GitHub*.

Para começar a usar o *Colab*, basta [acessar o endereço](https://colab.research.google.com/notebooks/intro.ipynb?utm_source=scs-index" \t "_blank)e desfrutar das vantagens dessa ferramenta.

Devido a facilidade e vantagens do *Colab*, indicamos essa ferramenta como principal meio de trabalho para essa disciplina.

Faça o link do seu *Colab*com seu *Google drive*, copie os códigos que serão apresentados, altere-os, explore!

**Variáveis e tipos básicos de dados em Python**



Variáveis são espaços alocados na memória RAM para guardar valores temporariamente. Em *Python*, esses espaços não precisam ser tipados, ou seja, a variável pode ser alocada sem especificar o tipo de dado que ela aguardará. O interpretador *Python*é capaz de determinar o tipo de dado da variável com base no seu valor, ou seja, as variáveis são tipadas dinamicamente nessa linguagem.

Veja alguns exemplos:

IN[2]: x **=** 10

 nome **=** 'aluno'

 nota **=** 8.75

 fez\_inscricao **=** True

Para saber o tipo de dado que uma variável guarda, podemos imprimir seu tipo usando a função type() , veja como:

IN[3]: print(type(x))

 print(type(nome))

 print(type(nota))

 print(type(fez\_inscricao))

<class 'int'>  
  
<class 'str'>  
  
<class 'float'>  
  
<class 'bool'>

A célula de entrada 3 (In [3]) tem os comandos para imprimir na tela, os tipos das quatro variáveis que criamos anteriormente. Veja que foram impressas quatro diferentes classes de tipos de dados. A variável "x" é do tipo inteira (int). A variável "nome" é do tipo string (str). A variável "nota" é do tipo ponto flutuante (float). A variável "fez\_inscricao" é do tipo booleana (bool).

Em *Python*, tudo é objeto! Por isso os tipos de dados aparecem com a palavra "class", que significa classe. Teste você mesmo os códigos no [emulador](https://trinket.io/python3/f4697c31a7) e aproveite para explorar outras variáveis!

Agora que já sabemos como criar variáveis, vamos aprimorar nosso *hello world*. Vamos solicitar que o usuário digite um nome, para que possamos fazer uma saudação personalizada. A função input() faz a leitura de um valor digitado. Veja como usar:

IN[4]: nome **=** input("Digite um nome: ")

 print(nome)

Digite um nome: João  
  
João

Veja que dentro da função input(), colocamos a mensagem que irá aparecer na tela, o que o usuário digitar será capturado e guardado dentro da variável "nome". Na linha em seguida, imprimos o que o usuário digitou. Mas como podemos combinar a frase "*hello world*" com o nome digitado? Como vamos imprimir variáveis e textos juntos? Como podemos imprimir: "Olá João! Bem-vindo à disciplina de programação. Parabéns pelo seu primeiro *hello world*"?

Temos uma variedade de formas de imprimir texto e variável em *Python*. Vejamos algumas: podemos usar formatadores de caracteres (igual em C), podemos usar a função format() e podemos criar uma string formatada. Vejamos cada uma delas:

IN[5]: # Modo 1 - usando formatadores de caracteres (igual na linguagem C) para imprimir variável e texto

  print("Olá %s, bem vindo a disciplina de programação. Parabéns pelo seu primeiro hello world" **%** (nome))

Olá João, bem vindo a disciplina de programação. Parabéns pelo seu primeiro hello world

IN[6]: # Modo 2 - usando a função format() para imprimir variável e texto

 print("Olá {}, bem vindo a disciplina de programação. Parabéns pelo seu primeiro hello world"**.**format(nome))

Olá João, bem vindo a disciplina de programação. Parabéns pelo seu primeiro hello world

IN[7]: # Modo 3 - usando strings formatadas

  print(f"Olá {nome}, bem vindo a disciplina de programação. Parabéns pelo seu primeiro hello world")

Olá João, bem vindo a disciplina de programação. Parabéns pelo seu primeiro hello world

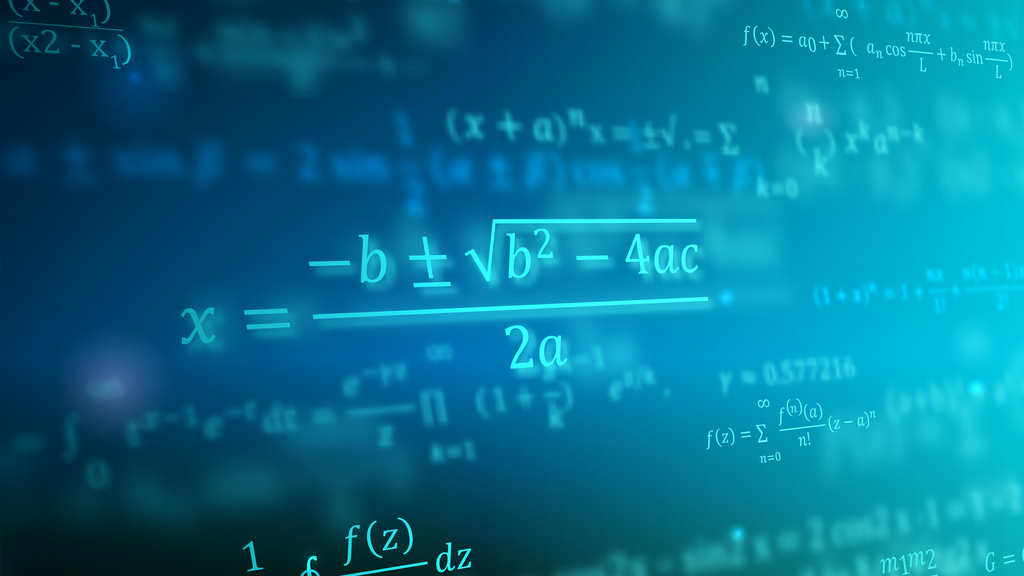
Em primeiro lugar, como vocês podem observar, usamos o hash # para criar comentários de uma linha. Lembrando que em qualquer linguagem de programação, os comentários não são nem interpretados, nem compilados.

Os três modos usados obtiveram o mesmo resultado e você poderia adotar o que achar mais conveniente. Entretanto, lembra que existe um código para sintaxe do "*pythonic code*"? Pois bem, a [PEP 498](https://www.python.org/dev/peps/pep-0498/" \t "_blank) fala sobre a criação de *strings*com interpolação (mistura de variáveis com texto". Nessa PEP, o autor destaca o uso do "modo 3" como a melhor opção chamando-a de "*f-strings*". Portanto, em nossos códigos, iremos adotar essa sintaxe para a criação de *strings*com interpolação.

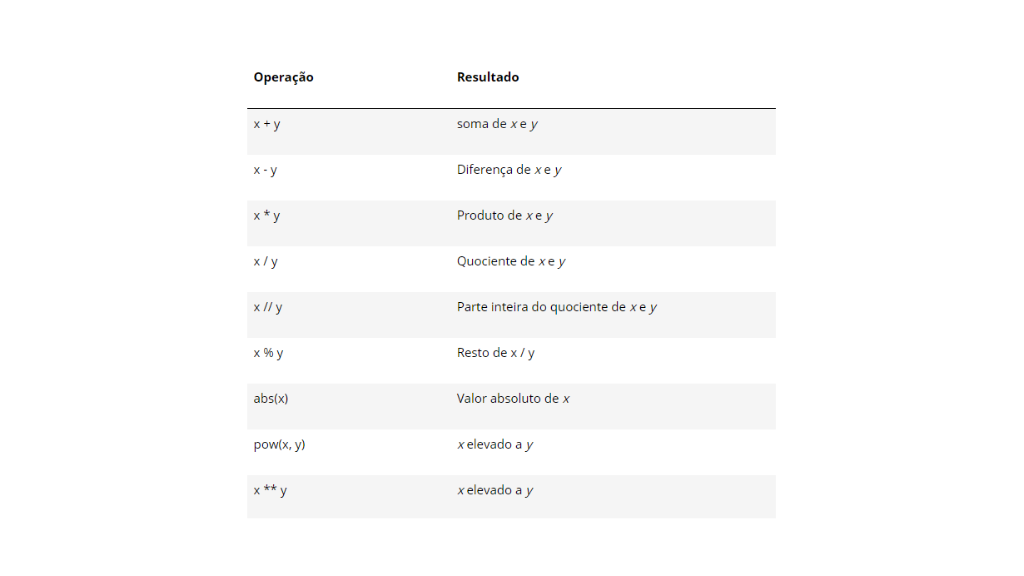
As *strings*formatadas com "*f-strings*" só podem ser compiladas com o interpretador *Python*na versão 3.6. Se tentar usar essa sintaxe em um interpretador em versões anteriores, será dado erro de sintaxe.

O uso do interpretador na versão 2 é fortemente desencorajado pela equipe *Python*, pois em 2020 ele será descontinuado. Para saber mais, leia a seção[Update do endereço](https://www.python.org/dev/peps/pep-0373/" \t "_blank).

**Operações matemáticas em Python**



O quadro abaixo apresenta um resumo das operações matemáticas suportadas por *Python*. Com exceção das funções abs() e pow() e da notação de potência \*\*, as outras operações e sintaxe são similares a diversas linguagens de programação.

Operações matemáticas em Python. Fonte: adaptada de Docs Python.

Com relação às operações matemáticas, seja na programação, seja na resolução analítica, o mais importante é lembrar a ordem de precedências das operações.

1. primeiro resolvem-se os parênteses, do mais interno para o mais externo.
2. exponenciação.
3. multiplicação e divisão.
4. soma e subtração.

Vejamos alguns exemplos. Repare como é fundamental conhecer a ordem de procedência das operações para não criar cálculos errados durante a implementação de uma solução. Teste você mesmo os códigos no [emulador](https://trinket.io/python3/2a97351693" \t "_blank) e aproveite para explorar outras operações.

**IN[8]:**# Qual o resultado armazendo na variável operacao\_1: 25 ou 17?

 operacao\_1 **=** 2 **+** 3 **\*** 5

 # Qual o resultado armazendo na variável operacao\_2: 25 ou 17?

 operacao\_2 **=** (2 **+** 3) **\*** 5

 # Qual o resultado armazendo na variável operacao\_3: 4 ou 1?

 operacao\_3 **=** 4 **/** 2 **\*\*** 2

 # Qual o resultado armazendo na variável operacao\_4: 1 ou 5?

 operacao\_4 **=** 13 **%** 3 **+** 4

 print(f"Resultado em operacao\_1 = {operacao\_1}")

 print(f"Resultado em operacao\_2 = {operacao\_2}")

 print(f"Resultado em operacao\_3 = {operacao\_3}")

       print(f"Resultado em operacao\_4 = {operacao\_4}")

Resultado em operacao\_1 = 17  
  
Resultado em operacao\_2 = 25  
  
Resultado em operacao\_3 = 1.0  
  
Resultado em operacao\_4 = 5

Agora vamos juntar tudo o que aprendemos até o momento. Uma equação do segundo grau possui a fórmula: y = a\*x\*\*2 + b\*x + c, onde a, b, c são constantes. O valor de y (resultado) depende do valor de x, ou seja, x é a variável independente e y a dependente. Considerando os valores a = 2, b = 0.5 e c = 1, vamos solicitar para o usuário um valor de x e retornar o valor de y correspondente ao x que ele digitou. Veja como deve ser implementado:

**IN[9]:**a **=** 2

 b **=** 0.5

 c **=** 1

 x **=** input("Digite o valor de x: ")

 y **=** a **\*** x **\*\*** 2 **+** b **\*** x **+** c

 print(f"O resultado de y para x = {x} é {y}.")

 Digite o valor de x: 3

**---------------------------------------------------------------------------**

**TypeError**                                 Traceback (most recent call last)

**<ipython-input-9-42060e5b5536>** in <module>

       4 x **=** input**("Digite o valor de x: ")**

       5

**----> 6**y **=** a **\*** x **\*\*** **2** **+** b **\*** x **+** c

       7

       8 print**(f"O resultado de y para x = {x} é {y}.")**

**TypeError**: unsupported operand type(s) for \*\* or pow(): 'str' and 'int'

Erro? E agora? Calma, erros fazem parte da vida do desenvolvedor. Vamos ler a mesagem de erro. Primeiro o erro foi do tipo "TypeError", isso quer dizer que alguma variável não está com o tipo adequado para a situação.

A mensagem nos diz que não é permitida a operação de potenciação (\*\*) entre um tipo string e inteiro. Portanto, o erro é porque estamos tentando fazer uma operação matemática entre string e um tipo numérico.

Então, dentre as variáveis deve ter uma com o tipo errado. Vamos usar a função type() para verificar o tipo das variáveis usadas.

**IN[10]:**print(type(a))

 print(type(b))

 print(type(c))

print(type(x))

<class 'int'>  
  
<class 'float'>  
  
<class 'int'>  
  
<class 'str'>

Olhando o resultado da entrada 10 (In [10]), vemos que o tipo da variável x é string (str), isso acontece porque ao usar a função input(), ela retorna uma *string*, independente do que o usuário digitou, sempre será string. Portanto, para corrigir o erro, precisamos converter o resultado da entrada em um tipo numérico. Como não sabemos se o usuário vai digitar um número inteiro ou decimal, vamos fazer a conversão usando a função float(). Veja como fica o código:

**IN[11]:**a **=** 2

  b **=** 0.5

  c **=** 1

  x **=** input("Digite o valor de x: ")

  x **=** float(x) # aqui fazemos a conversão da string para o tipo numérico

  y **=** a **\*** x **\*\*** 2 **+** b **\*** x **+** c

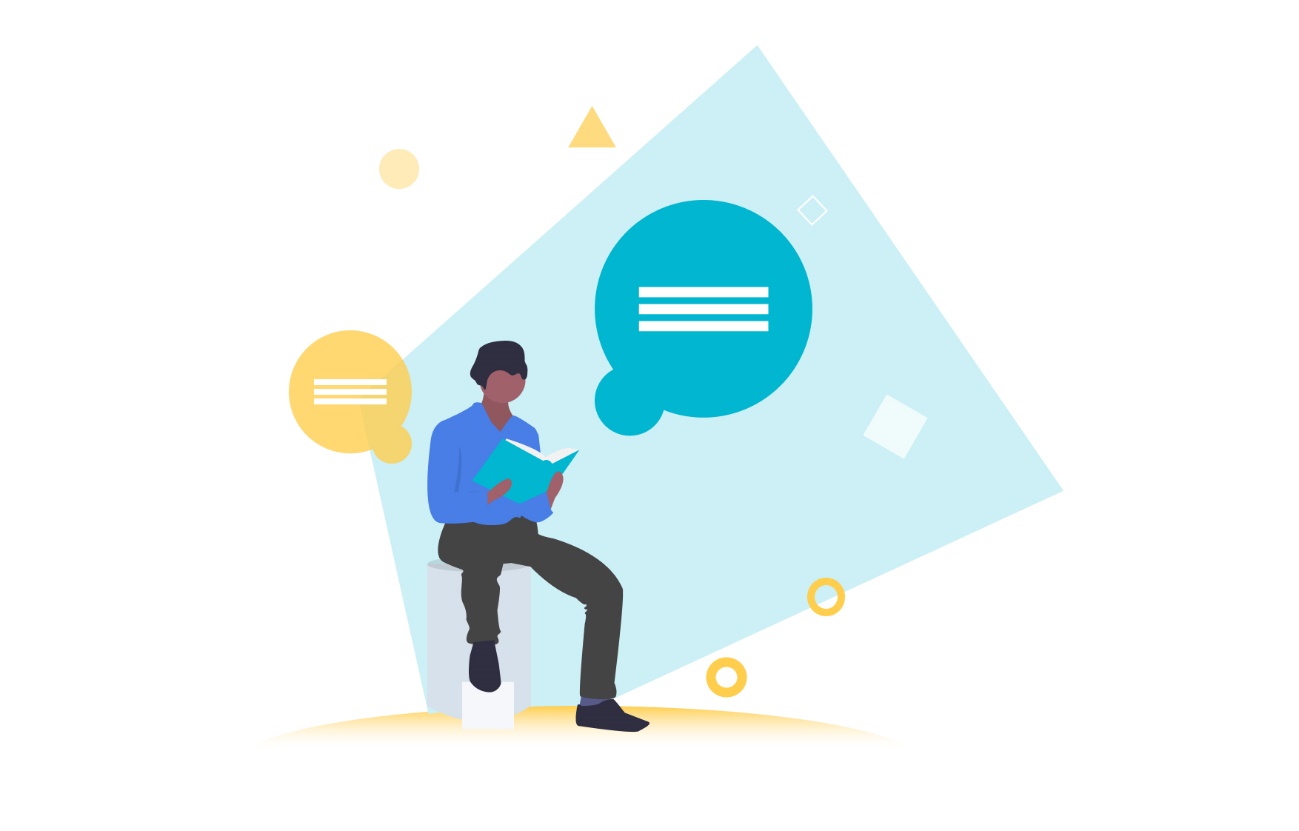
  print(f"O resultado de y para x = {x} é {y}.")

  Digite o valor de x: 3

  O resultado de y para x = 3.0 é 20.5.

Aprendemos bastante até aqui, agora é hora de colocar em prática. Não deixe de explorar os links apresentados, quanto mais você buscar informações, mais fluente ficará na linguagem de programação.

**Conclusão**

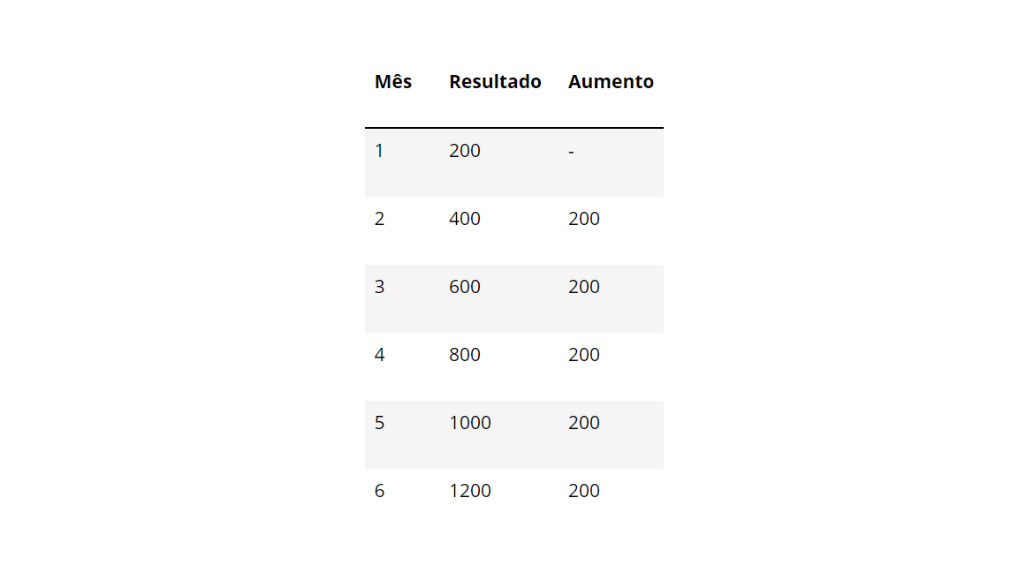


Foi lhe dada a missão de escolher uma ferramenta para desenvolver um protótipo para o cliente que fabrica peças automotivas. Uma opção é usar o *Colab*, pois nessa ferramenta você consegue implementar seu algoritmo usando a linguagem *Python*.

Outra grande vantagem em utilizá-lo é o fato de ser on-line e não precisar de instalação. Uma vez pronto o protótipo, você pode enviar o link, tanto para seu gerente ver o resultado do seu trabalho, quanto para o cliente testar a solução.

Uma vez decidida a ferramenta, é hora de começar a pensar na solução. Tudo que você tem de informação está em um gráfico, portanto é preciso interpretá-lo.

Vamos extrair as informações de venda do gráfico e escrever em forma de tabela, apresentada a seguir:

Venda de peças. Fonte: elaborada pela autora.

Ao tabular os dados do gráfico, aparece um valor interessante na coluna que mostra o aumento mês após mês. De acordo com as informações o aumento tem sido constante.

Se o aumento é constante, podemos usar uma função do primeiro grau para prever qual será o resultado em qualquer mês. A função será*r = c \* mes*, onde, r é o resultado que queremos, c é a constante de crescimento e *mes*é a variável de entrada. Dessa forma, ao obter um mês qualquer (2, 4, 30, etc) podemos dizer qual o resultado.

Vamos testar nossa fórmula:

* mes = 2; c = 200 -> r = 200 \* 2 = 400 (Valor correto para o mês 2.
* mes = 3; c = 200 -> r = 200 \* 3 = 600 (Valor correto para o mês 3.
* mes = 4; c = 200 -> r = 200 \* 4 = 800 (Valor correto para o mês 4.
* mes = 5; c = 200 -> r = 200 \* 5 = 1000 (Valor correto para o mês 5.

Agora que já sabemos como resolver, vamos implementar usando a linguagem *Python*. Veja a seguir o código.

**IN[1]:**c **=** 200 # valor da constante

 mes **=** input("Digite o mês que deseja saber o resultado: ") # Função para captura o mês que o cliente digitar

 mes **=** int(mes) # Não esqueça de converter para numérico o valor captura pela função input()

 r **=** c **\*** mes # Equação do primeiro grau, também chamada função do primeiro grau ou de função linear.

 print(f"A quantidade de peças para o mês {mes} será {r}") # Impressão do resultado usando string interpolada "f-strings" (PEP 498)

 Digite o mês que deseja saber o resultado: 30

 A quantidade de peças para o mês 30 será 6000

\_\_\_\_\_

**➕ Pesquise mais**

Que tal treinar um pouco mais de programação e conhecer novas fontes de informações? Você, estudante, tem acesso à Biblioteca Virtual, um repositório de livros e artigos que pode ser acessado no endereço:

Na página 47 (capítulo 2: Objetos e Comandos de Entrada e Saída em Python) da obra:**Python 3 - conceitos e aplicações: uma abordagem didática**. você encontra o exercício proposto 1.