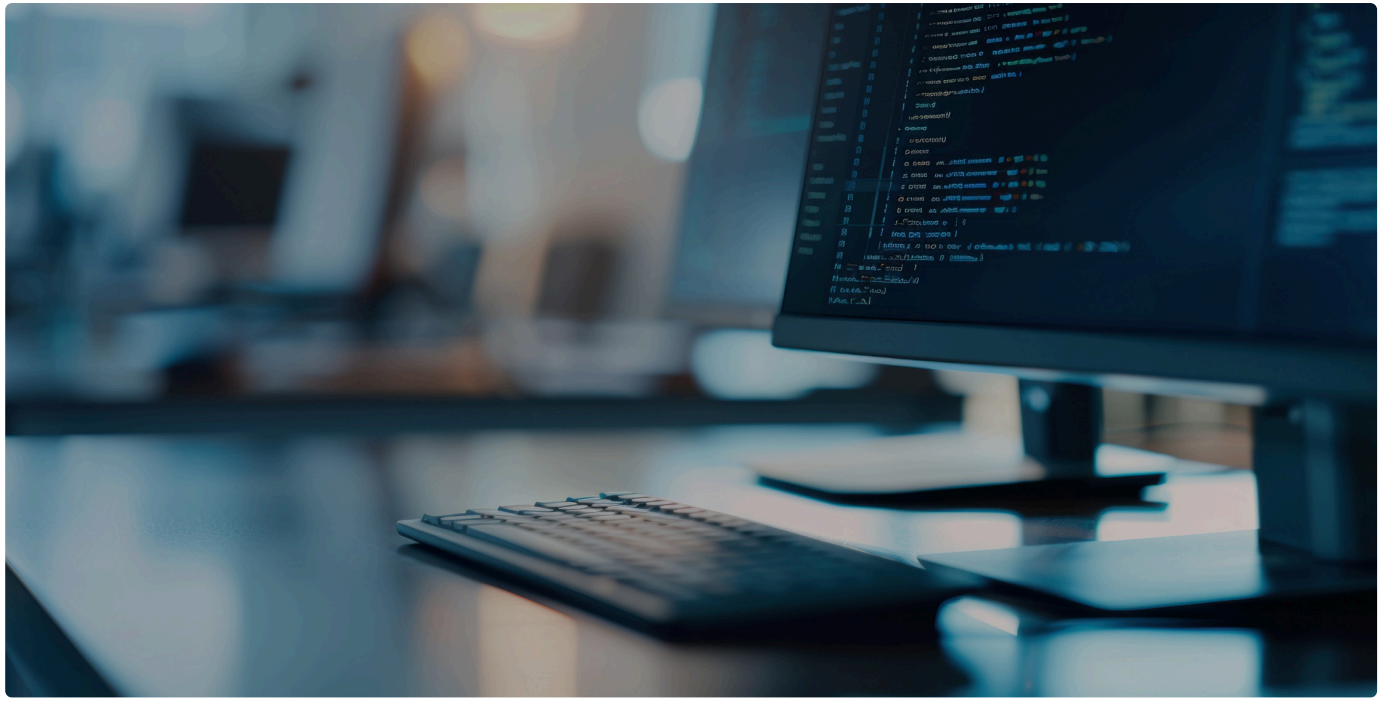


Python básico

Prof. Sérgio Monteiro

Apresentação

Neste estudo, vamos explorar a linguagem Python com o aprendizado de suas características. Conheceremos as variáveis, tipos de dados, entrada e saída de dados e lógica de programação. Dominar os fundamentos essenciais de Python é essencial para o profissional avançar no desenvolvimento de programas mais interessantes.

Propósito

Preparação

Será necessário baixar uma versão do Python, bem como fazer o download e instalar uma IDE, como o PyCharm.

Objetivos

Módulo 1

Linguagem Python

Descrever a linguagem Python e suas principais características.

Módulo 2

Variáveis

Reconhecer o uso de variáveis em Python.

Módulo 3

Tipos de dados e expressões

Identificar os tipos de dados e as expressões em Python.

Módulo 4

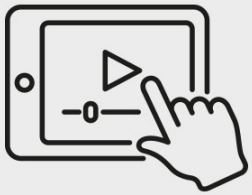
Atribuição, entrada e saída de dados

Identificar na linguagem Python as formas de atribuição, de entrada e saída de dados.

Introdução

Neste vídeo, apresentaremos o tema Python básico, incluindo o uso de variáveis, os tipos de dados, as expressões, as formas de atribuição e a entrada e a saída de dados.

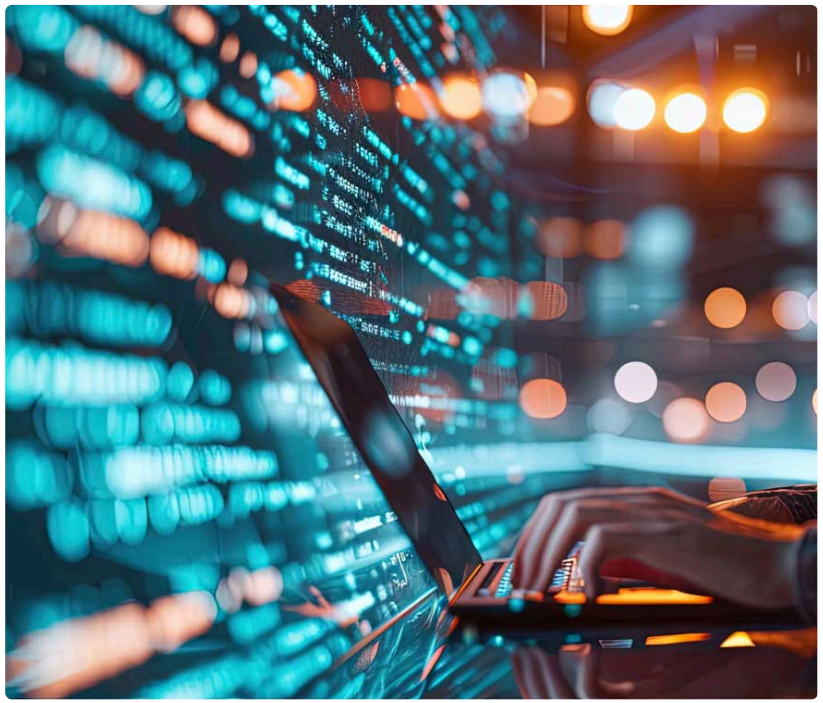
Para assistir a um vídeo sobre o assunto, acesse a versão online deste conteúdo.



Material para download

Clique no botão abaixo para fazer o download do conteúdo completo em formato PDF.

Download material



1 - Linguagem Python

Ao final deste módulo, você será capaz de descrever a linguagem Python e suas principais características.

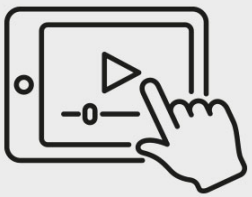
Visão geral

Python é uma linguagem de programação de alto nível que permite ao programador utilizar instruções de forma intuitiva, tornando seu aprendizado mais simples do que o aprendizado de uma linguagem de baixo nível.

Nas linguagens de baixo nível, o programador precisa se expressar de forma muito mais próxima do que o dispositivo “entende”, levando naturalmente a um distanciamento da linguagem utilizada para comunicação entre duas pessoas.

Neste vídeo, apresentaremos a linguagem de programação de alto nível Python, conhecida por sua sintaxe simples e versatilidade, amplamente utilizada em diversos domínios, incluindo desenvolvimento web, análise de dados e automação. Confira!

Para assistir a um vídeo sobre o assunto, acesse a versão online deste conteúdo.



Histórico

A linguagem Python, lançada pelo holandês **Guido van Rossum** em 1990, tem aumentado sua atuação na programação. Permite uma programação fácil e clara para escalas pequenas e grandes, além de enfatizar a legibilidade eficiente do código, notadamente usando espaços em branco significativos.

Guido van Rossum ao lado de logo Python.

Características da linguagem Python

A linguagem Python possui características específicas. Veja!

Ser multiparadigma

Apesar de suportar perfeitamente o paradigma de programação estruturada, também suporta programação orientada a objetos, tem características do paradigma funcional, com o amplo uso de bibliotecas, assim como permite recursividade e uso de funções anônimas.

Ser interativa

Permite que os usuários interajam com o interpretador Python diretamente para escrever os programas, utilizando o prompt interativo. Esse prompt fornece mensagens detalhadas para qualquer tipo de erro ou para qualquer comando específico em execução, suporta testes interativos e depuração de trechos de código.

Ser híbrida quanto ao método de implementação

Usa uma abordagem mista para explorar as vantagens do interpretador e do compilador. Assim como Java, utiliza o conceito de máquina virtual, permitindo a geração de um código intermediário, mais fácil de ser interpretado, mas que não é vinculado definitivamente a nenhum sistema operacional.

Ser portável

Tem a capacidade de rodar em uma grande variedade de plataformas de hardware com a mesma interface. Ele roda perfeitamente em quase todos os sistemas operacionais, como Windows, Linux, UNIX, e Mac OS, sem nenhuma alteração.

Ser extensível

Permite que os programadores adicionem ou criem módulos e pacotes de baixo nível/ alto nível ao interpretador Python. Esses módulos e pacotes de ferramentas permitem que os desenvolvedores tenham possibilidades amplas de colaboração, contribuindo para a popularidade da linguagem.

Suportar bancos de dados

Por ser uma linguagem de programação de uso geral, suporta os principais sistemas de bancos de dados. Permite escrever código com integração com MySQL, PostgreSQL, SQLite, ElephantSQL, MongoDB, entre outros.

Suportar interface com usuário

Permite escrever código de tal maneira que uma interface do usuário para um aplicativo possa ser facilmente criada, importando bibliotecas como Tkinter, Flexx, CEF Python, Dabo, Pyforms ou PyGUI wxPython.

Possibilidade de ser usada como linguagem de script

Permite fácil acesso a outros programas, podendo ser compilado para bytecode a fim de criar aplicativos grandes.

Permitir o desenvolvimento de aplicações Web

Devido à escalabilidade já citada, oferece uma variedade de opções para o desenvolvimento de aplicativos Web. A biblioteca padrão do Python incorpora muitos protocolos para o desenvolvimento da web, como HTML, XML, JSON, processamento de e-mail, além de fornecer base para FTP, IMAP e outros protocolos da Internet.

Permitir a criação de aplicações comerciais

É desenvolvido sob uma licença de código aberto aprovada pela OSI, tornando-o livremente utilizável e distribuível, mesmo para uso comercial.

Atividade 1

Imagine que você seja um desenvolvedor de software iniciante e esteja interessado em aprender uma nova linguagem de programação. Após pesquisar sobre as características da linguagem Python, você se depara com uma lista impressionante de recursos e capacidades. Animado com as possibilidades que Python oferece, você decide explorar mais a fundo e começar a desenvolver alguns projetos pessoais utilizando essa linguagem. Considerando o texto apresentado sobre as características da linguagem Python, analise as alternativas e assinale a opção correta:

A Python não suporta programação orientada a objetos.

B O prompt interativo do Python fornece mensagens detalhadas para erros.

C Python não é portável, pois requer alterações para rodar em diferentes sistemas operacionais.

D Python não permite a extensão com módulos e pacotes de baixo nível/alto nível.

E Python suporta apenas um banco de dados específico.

Parabéns! A alternativa B está correta.

A opção correta é a do prompt interativo do Python que fornece mensagens detalhadas para qualquer tipo de erro ou para qualquer comando específico em execução, conforme descrito no texto. As demais opções são incorretas, pois Python suporta programação orientada a objetos, é portátil e roda em diversas plataformas sem alterações. Além disso, permite a extensão com módulos e pacotes e suporta vários sistemas de bancos de dados.

Instalando o Python

A IDE PyCharm oferece um ambiente robusto e intuitivo para desenvolver aplicações em Python. Com recursos como autocompletar inteligente, depuração poderosa e integração com controle de versão, ela simplifica o processo de codificação, permitindo que os desenvolvedores foquem a criação de soluções inovadoras. Além disso, sua interface amigável e personalizável proporciona uma experiência de programação fluida e eficiente. Seja você um iniciante ou um profissional experiente, o PyCharm é uma ferramenta indispensável que eleva a produtividade e a qualidade do desenvolvimento de software em Python.

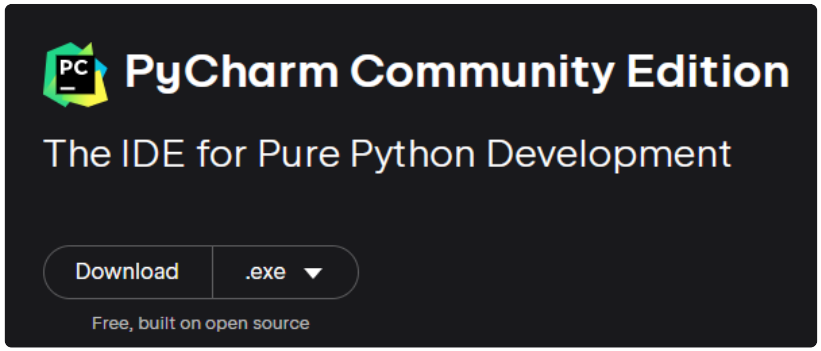
Neste vídeo, guiaremos você pela instalação do Python, um processo simples que pode ser feito baixando e executando o instalador disponível no site oficial ou utilizando gerenciadores de pacotes como o Anaconda. Acompanhe!

Para assistir a um vídeo sobre o assunto, acesse a versão online deste conteúdo.

Instalação do PyCharm

Existem várias formas de utilizarmos a linguagem de programação Python; uma delas é através do ambiente de desenvolvimento Pycharm. Para isso, acesse o site do Jet Brains para fazer o download do software PyCharm.

Em seguida, clique na opção Download da versão Community, conforme a imagem.



PyCharm Community.

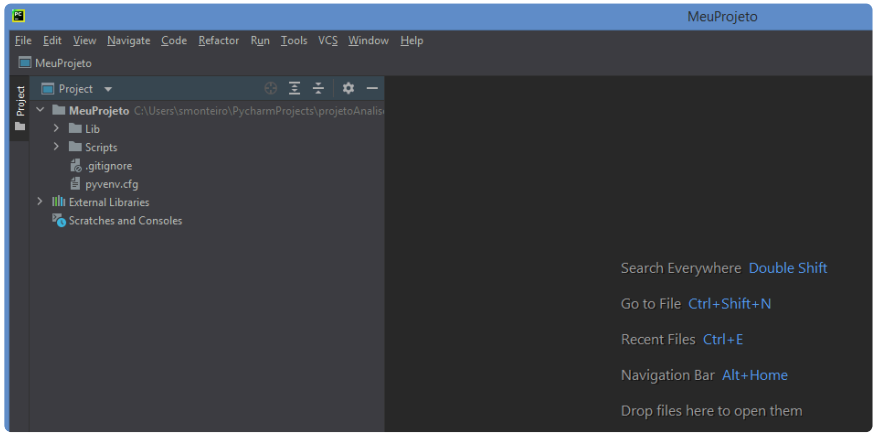
Depois de fazer o download no seu computador, basta dar dois cliques sobre o arquivo e seguir o padrão de instalação Next-Next do Windows.

Dica

Selecione o inglês como idioma durante a instalação, pois é mais fácil obter informações posteriormente nas diversas comunidades na Internet.

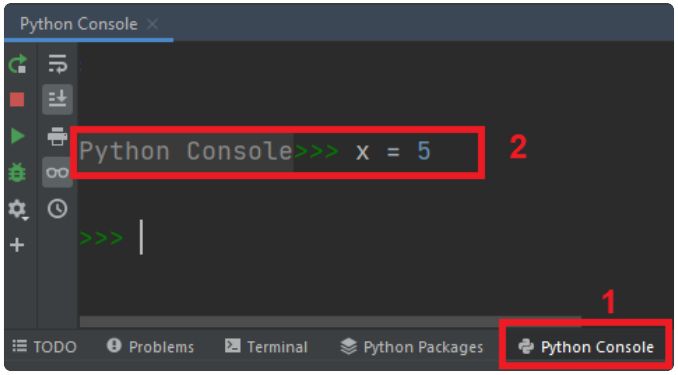
Teste do PyCharm

Depois de instalar o PyCharm, vamos testá-lo: abra a IDE, você verá uma imagem semelhante a esta.



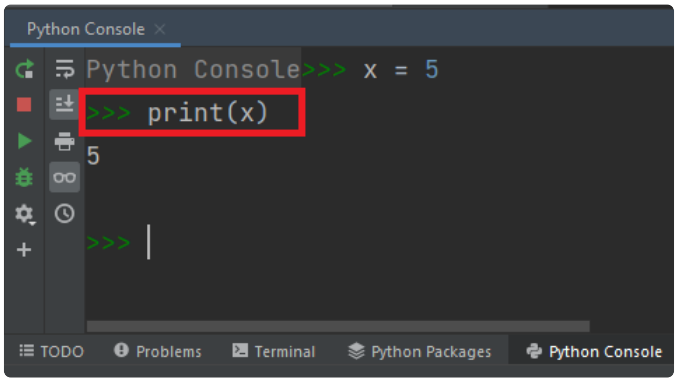
IDE PYCharm.

Agora, selecione a opção Python Console e escreva x=5 e pressione a tecla Enter, conforme a imagem.



Uso do Python Console.

Por fim, digite o comando print(x), para verificar o conteúdo da variável x. Veja!



Uso do Python Console.

Excelente! Agora você já consegue utilizar o PyCharm para realizar alguns comandos do Python. Neste momento, vamos realizar uma atividade para consolidar nossos conhecimentos.

Atividade 2

Após instalar o PyCharm, uma IDE popular para desenvolvimento em Python, você decide testá-lo seguindo algumas instruções básicas. Ao abrir a IDE, você é orientado a acessar o Python Console e somar o conteúdo de duas variáveis x e y, com valores, respectivos, de 5 e 10 e, depois, imprimir o resultado. Considerando o texto apresentado sobre o uso do PyCharm, analise as alternativas e assinale a opção correta:

A Você deve digitar "resultado = x + y" no Python Console e pressionar Enter.

B Ao abrir a IDE, clique na opção Terminal em vez de Python Console.

C A soma das variáveis x e y não pode ser realizada no Python Console.

D Não é possível imprimir o resultado da soma no PyCharm.

E

No Python Console, Você deve digitar "print(x + y)" (sem as aspas) e pressionar Enter.

Parabéns! A alternativa E está correta.

A opção correta é digitar "print(x + y)" (sem as aspas) no Python Console e pressionar Enter. O texto menciona claramente que o objetivo é somar o conteúdo das variáveis x e y e imprimir o resultado. As demais opções são incorretas, pois não seguem as instruções fornecidas no texto: uma das opções, não imprime o resultado da soma (apenas atribui à variável "resultado"); a outra opção sugere usar o Terminal em vez do Python Console. Ainda há a opção que afirma que a soma não pode ser realizada no Python Console (o que não é verdade); por fim, há a opção que afirma que não é possível imprimir o resultado da soma no PyCharm (o que também não é verdade).

Principais características

Em Python, a simplicidade é essencial para a eficácia. Com comandos de controle de fluxo intuitivos, estruturas de repetição flexíveis e a capacidade de adicionar comentários que melhoram a compreensão do código, a linguagem se destaca pela acessibilidade e clareza. Programadores de todos os níveis podem criar e manter programas eficientes, aproveitando essas características fundamentais. No complexo mundo da programação, Python oferece uma base sólida e acolhedora para expressar ideias e resolver problemas de forma eficaz.

No vídeo a seguir, falaremos sobre as principais características do Python, incluindo sua sintaxe simples e legível, tipagem dinâmica, vasta biblioteca padrão e capacidade de suportar múltiplos paradigmas de programação, tornando-o uma escolha popular para uma gama de aplicações. Confira!

Para assistir a um vídeo sobre o assunto, acesse a versão online deste conteúdo.

Blocos

Em Python, os blocos são definidos pela indentação. Diferente de C e Java, que usam as chaves { e } para delimitar os blocos, em Python, todos os blocos são iniciados com o símbolo : (dois pontos) na linha superior e representados pelo acréscimo de 4 (quatro) espaços à esquerda.

Sem se preocupar por enquanto com o significado das expressões for, if, else ou range, observe o código a seguir.

Python

A partir disso, podemos destacar alguns pontos. Acompanhe!

1

Linha 1

Está mais à esquerda, assim como as linhas 2 e 11.

2

Linha 2

Todas as linhas de 3 a 10 estão dentro do bloco do for da linha 2.

3

Linha 3

Observe que a linha 3 tem um if abrindo um bloco, dentro do qual estão as linhas 4 e 5.

4

Linha 6

Por sua vez, a linha 6 tem um else abrindo outro bloco, composto pelas linhas de 7 a 10. Os blocos do if (linha 3) e do else (linha 6) estão no mesmo nível.

5

Linha 7

Mostra outro if abrindo outro bloco – composto apenas pela linha 8 – que está no mesmo nível do bloco do else da linha 9 – composto apenas pela linha 10.

6

Linha 11

Como a linha 11 está no mesmo nível da linha 2, ela não faz parte do bloco do for.

Comentários

Em Python, os comentários podem ser de uma linha ou de várias linhas. A tabela a seguir mostra as formas de limitar um comentário, além de comparar essas formas em Python e C. Observe!

	Python	C
Comentários com uma linha	Iniciados com #	Iniciados com //
Comentários com várias linhas	Limitados por """ (três aspas duplas) no início e no fim	Iniciados com /* e encerrados com */

Tabela: Formas de limitar comentários.
Humberto Henriques de Arruda.

Os comentários não são instruções a serem executadas. Então, você pode escrever de forma simples e objetiva, sem obedecer às regras de sintaxe da linguagem.

Boas práticas de programação

Ao começar sua jornada como programador, é importante perceber que existem algumas práticas que não são obrigatórias, mas podem ajudar muito no seu aprendizado. Além disso, podem permitir que você corrija mais rapidamente erros que podem surgir no futuro e tornam seu código mais fácil de ser compreendido por outro programador, favorecendo o trabalho em equipe.

Pessoa programando em computador.

Vamos conhecer algumas dessas **boas práticas**. Confira!

Inclusão de comentários

Uma prática muito importante é utilizar comentários no seu programa, explicando o que aquele trecho resolve.

PEP

Uma característica marcante da comunidade de desenvolvedores Python é manter uma lista de propostas de melhorias, chamadas PEP.

PEP

Sigla de Python Enhancement Proposals, dentre as PEPs, destaca-se a PEP8, que estabelece um guia de estilo de programação.

Atividade 3

Em Python, boas práticas incluem o uso de nomes de variáveis claros, a escrita de funções e classes modulares, a documentação do código com docstrings, o uso de ferramentas de controle de versão como o Git e a preferência por bibliotecas padrão para tarefas comuns, promovendo a reutilização e a legibilidade. Nesse contexto de melhores práticas, analise as alternativas e assinale a opção correta:

A Em Python, os comentários podem ser iniciados com "//".

B Os comentários em Python devem seguir rigorosamente as regras de sintaxe da linguagem.

C Utilizar comentários no código não é uma prática recomendada pela comunidade de desenvolvedores Python.

D Uma das boas práticas de programação é explicar trechos de código por meio de comentários.

E As propostas de melhoria Python (PEP) são usadas apenas para correções de erros na linguagem.

Parabéns! A alternativa D está correta.

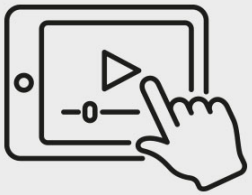
O texto enfatiza a importância de utilizar comentários para explicar o que cada trecho do código resolve, favorecendo a compreensão por outros programadores e facilitando o trabalho em equipe. As demais opções são incorretas: em Python, os comentários são iniciados com "#", os comentários em Python não precisam seguir as regras de sintaxe, o texto destaca a importância do uso de comentários e as propostas de melhoria Python (PEP) não se limitam a correções de erros na linguagem.

Python básico na prática

Aprender a programar em Python através de prática é essencial para iniciantes. A prática constante não só fortalece o entendimento dos conceitos fundamentais, como também desenvolve habilidades de resolução de problemas. Programar envolve mais do que apenas entender a teoria; é sobre aplicar esse conhecimento para criar soluções tangíveis. Fazendo projetos práticos, como criar pequenos programas ou automatizar tarefas simples, os iniciantes podem ganhar confiança, consolidar seu aprendizado e descobrir o poder transformador da programação. A prática contínua é a chave para se tornar um programador habilidoso e criativo em Python.

No vídeo a seguir, praticaremos a implementação de elementos básicos de Python, incluindo variáveis, estruturas de controle de fluxo (como loops e condicionais), funções e manipulação de dados, proporcionando uma base sólida para o desenvolvimento de programas funcionais e eficientes.

Para assistir a um vídeo sobre o assunto, acesse a versão online deste conteúdo.



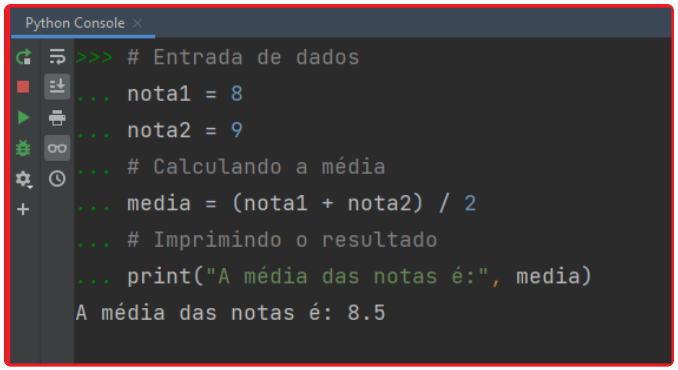
Roteiro de prática

Nesta etapa, vamos criar um programa para calcular a média de duas notas. Você precisa seguir os seguintes passos:

1. Abra o Pycharm.
2. Pressione a opção Python Console.
3. Agora, digite o código a seguir.

Python

4. O seu programa deve ficar conforme a imagem.

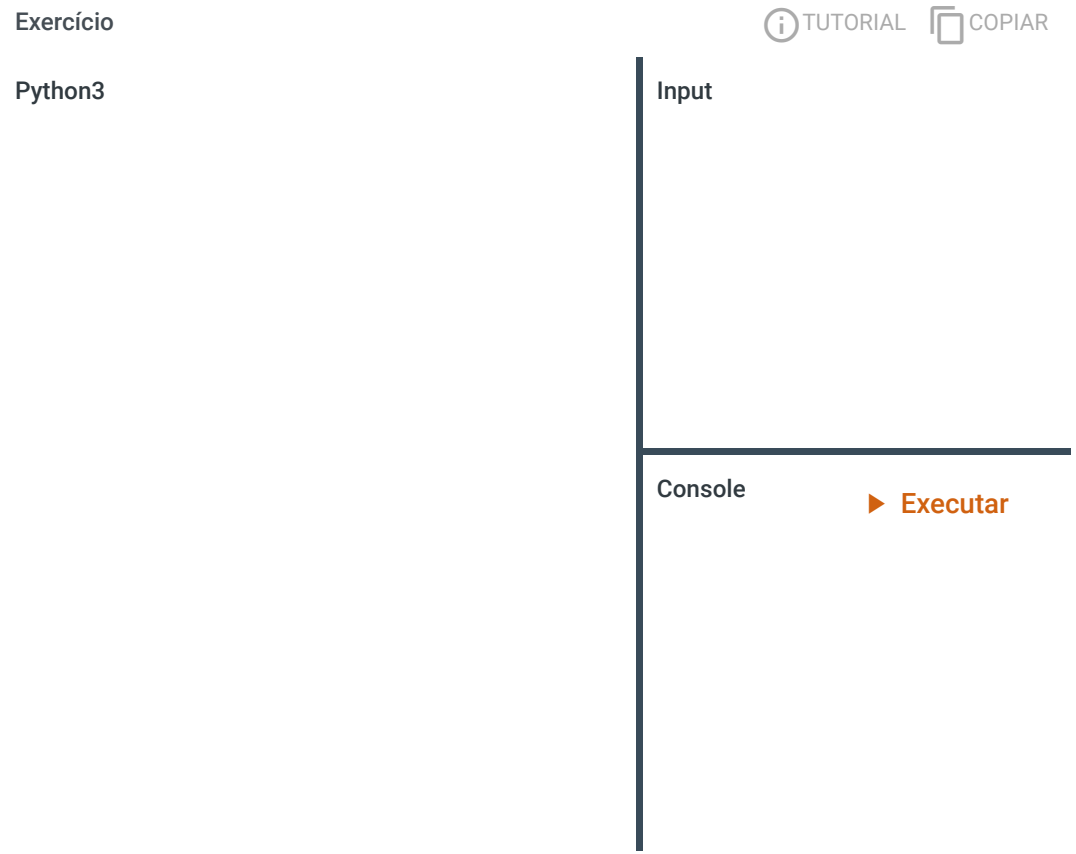


Programa para calcular a média de dois números.

Agora, é a sua vez! Reveja o que estudou até aqui e pratique o que aprendeu!

Atividade 4

Elabore um programa em Python para calcular média de quatro números e exiba o resultado para o usuário. Use as seguintes notas como entrada de dados do seu programa: 8, 9, 10 e 5.



Digite sua resposta aqui

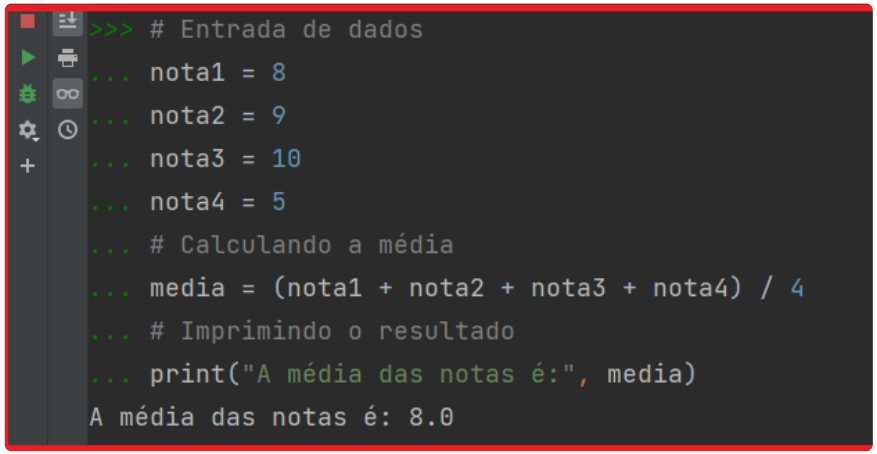
Chave de resposta

O resultado esperado deve seguir o passo a passo a seguir.

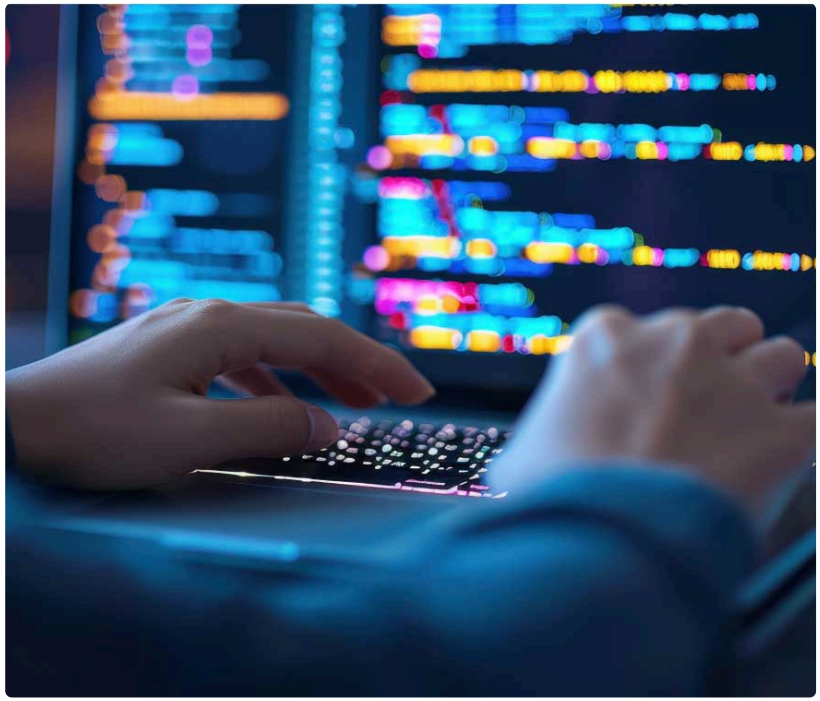
1. Execute o Pycharm.
2. Pressione a opção Python Console.
3. Digite o programa a seguir.

Python

4. O resultado final deve ser este:



Solução da questão.



2 - Variáveis em Python

Ao final deste módulo, você será capaz identificar o módulo caso haja alteração de conteúdo.

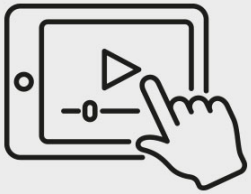
Variáveis

Variáveis são os blocos fundamentais da programação em Python, desempenhando um papel crucial na criação de código dinâmico e flexível. Elas permitem armazenar e

manipular dados, tornando possível a criação de programas que respondem a diferentes entradas e condições. O uso correto de variáveis não apenas organiza o código de forma eficiente, mas também facilita a compreensão e manutenção dele. Ao entender e dominar o uso de variáveis, os programadores ganham a capacidade de criar soluções complexas de forma mais clara, impulsionando o desenvolvimento de aplicações mais robustas e eficazes em Python.

Neste vídeo, abordaremos os conceitos e usos de variáveis, responsáveis por armazenar dados temporários durante a execução de um programa e permitir a manipulação e a reutilização de valores ao longo do código. Acompanhe!

Para assistir a um vídeo sobre o assunto, acesse a versão online deste conteúdo.



Conceitos

As variáveis são abstrações para endereços de memória que permitem que os programas fiquem mais fáceis de codificar, entender e depurar. Ao nomear uma variável com o identificador **x**, determinado espaço em memória passará a ter esse apelido. Em outras palavras, será possível acessar esse espaço de memória sabendo o seu

Pessoa digitando em teclado de computador.

apelido e, consequentemente, recuperar
o valor guardado nele, que no nosso
exemplo é **10**.

Uma analogia possível com o mundo real é com aquelas caixas de correio que ficam em frente às casas.

Em Python, o operador de atribuição é o = (símbolo de igual). A instrução x = 10 atribui o valor 10 à variável x.

Digite x = 10 e na linha seguinte digite print(x) no emulador. Em seguida, clique em Executar (no Python Console do PyCharm, digite x = 10 e pressione a tecla [ENTER] ou [RETURN] e depois digite print(x) e pressione [ENTER] ou [RETURN]). Veja:

Exercício

TUTORIAL COPIAR

Python3

Input

Console

▶ Executar

Observe que o retorno no console foi 10.

Se, posteriormente, você digitar x = 20 no emulador anterior e pressionar Executar (ou pressionar [ENTER] no Python Console do PyCharm), você alterará o valor da variável x. Ou seja, você mudará o valor armazenado naquele espaço de memória, mas sem alterar seu apelido. Faça esse teste!

Comentário

Diferentemente de outras linguagens, como C ou Java, não é necessário declarar uma variável antes de utilizá-la em Python. Basta atribuir um valor inicial à variável e utilizá-la dali em diante. Embora não seja necessário declarar uma variável para utilizá-la, não é possível utilizar uma variável que não tenha recebido alguma atribuição de valor.

No emulador a seguir, digite **b** e pressione Executar (no Python Console do PyCharm, pressione a tecla [ENTER] ou [RETURN]). Confira!

Exercício

TUTORIAL COPIAR

Python3

Input

Console

Executar

Veja no emulador que a mensagem de erro informa que o nome b não foi definido. Ou seja, não é possível determinar o valor atribuído a esse nome.

Identificadores de variáveis

Os identificadores das variáveis podem ser compostos por letras, o **underline (_)** e, com exceção do primeiro caractere, números de 0 a 9. Veja os exemplos.

1

MinhaVariavel, _variavel, salario1 e salario1_2

São válidos.

2

1variavel e salario-1

Não são válidos.

MinhaVariavel e minhavariavel

São identificadores de duas variáveis distintas.

Mesmo que seja um identificador permitido, nem sempre um identificador é bom para uma variável. Tente utilizar nomes que ajudem a entender o significado da variável para ganhar tempo quando for entender o código posteriormente.

Exemplo

salario é um nome de variável melhor que **s**.

Algumas palavras são consideradas reservadas e **não podem ser usadas como identificadores de variáveis em Python**. São elas: and, as, assert, break, class, continue, def, del, elif, else, except, exec, finally, for, from, global, if, import, in, is, lambda, not, or, pass, print, raise, return, try, while, with e yield.

Atividade 1

(MS CONCURSOS/2016/Creci 1° Região (RJ)/Analista de TI) Qual alternativa representa a declaração de uma variável na linguagem de programação Python?

A var valor = 3

B Boolean inicio = falso

C Texto = 'texto de exemplo’

D Int i = 1

E not = falso

Parabéns! A alternativa C está correta.

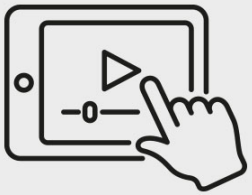
Lembre-se de que, em Python, as variáveis não são declaradas com o tipo vinculado. Assim, basta atribuir um valor inicial à variável para que ela possa ser usada. Isso ocorre com a variável texto, que recebe o valor inicial “texto de exemplo”.

Conceito de amarração (Binding)

A amarração, ou binding, é fundamental na estrutura e no funcionamento das linguagens de programação, como Python. Trata-se da associação entre entidades de programação, como variáveis e funções, a valores ou identificadores específicos em tempo de execução ou compilação. Essa associação é muito importante para garantir que o código funcione corretamente, permitindo que as operações sejam realizadas de maneira consistente e previsível. Em Python, entender a amarração é essencial para compreender como as variáveis são atribuídas, os objetos são referenciados e os valores são vinculados. Isso contribui para a escrita de um código mais claro, eficiente e livre de erros.

Neste vídeo, falaremos sobre o conceito de amarração (binding), que se refere à associação dinâmica entre um nome e um objeto, permitindo que variáveis sejam vinculadas a valores durante a execução do programa. Confira!

Para assistir a um vídeo sobre o assunto, acesse a versão online deste conteúdo.



Amarrações

Chamamos de amarração (binding) a associação entre entidades de programação. Veja alguns exemplos!

- Variável amarrada a um valor.

- Operador amarrado a um símbolo.
- Identificador amarrado a um tipo.

O tempo em que a amarração ocorre é chamado de **tempo de amarração**. Veja a seguir os tempos de amarração que cada linguagem pode ter.

Projeto da linguagem

Os símbolos são amarrados ao operador, como * (multiplicação), ou à definição das palavras reservadas.

Implementação

Ocorre, em geral, nos compiladores, como a definição de faixa de valores para determinado tipo.

Compilação

Corresponde à associação da variável ao seu tipo. Lembre-se de que Python associa a variável ao tipo.

Ligação

A ligação de vários módulos compilados previamente, como a chamada a uma função de um módulo importado. Em C, utilizamos a diretiva #include para termos permissão de utilizar as funções de determinada biblioteca. Em Python, utilizamos o import para isto.

Carga

Quando o programa é carregado, por exemplo, endereços de memória relativos são substituídos por endereços absolutos.

Execução

Corresponde à associação de valores a variáveis que dependam de entradas do usuário, por exemplo. A variável é vinculada ao valor apenas durante a execução do programa.

O momento em que ocorre a ligação pode ser classificado como cedo (early binding) ou tardio (late binding). Quanto mais cedo ocorre a ligação, maior a eficiência de execução do programa, porém menor a flexibilidade das estruturas disponibilizadas.

Amarrações de tipo

As amarrações de tipo vinculam a variável ao tipo do dado. Elas podem ser estáticas ou dinâmicas. Observe a diferença.

Estáticas

Ocorrem antes da execução e permanecem inalteradas. Em C, declaramos **int a**.

×

Dinâmicas

Ocorrem durante a execução e podem ser alteradas. É o caso do Python.

Veja o código a seguir.

Python

- A chamada **type** (parâmetro) retorna o tipo do parâmetro informado entre parênteses.
- A variável **valor** recebeu 10 e, com isso, ficou vinculada ao tipo **int**. Porém, ao receber o valor ‘a’, passou a estar vinculada ao tipo **str** (string).

Atividade 2

Um dos elementos de grande importância na programação é a amarração (binding). Ela representa a associação entre diferentes entidades de programação, como variáveis, operadores e identificadores, a determinados valores, símbolos ou tipos. Além disso, os diferentes tempos de amarração em um programa incluem o tempo de projeto de linguagem, implementação, compilação, ligação, carga e execução, destacando a diferença entre o binding cedo e tardio e seu impacto na eficiência e flexibilidade do programa. Considerando o texto sobre a amarração (binding) na programação, analise as alternativas e assinale a opção correta:

A O tempo de amarração ocorre apenas durante o tempo de execução de um programa.

B O binding tardio (late binding) proporciona maior eficiência de execução do programa.

C O tempo de compilação envolve a associação da variável ao seu tipo.

D A ligação de vários módulos compilados previamente ocorre durante o tempo de projeto de linguagem.

E A associação de valores a variáveis que dependem de entradas do usuário ocorre durante o tempo de carga.

Parabéns! A alternativa C está correta.

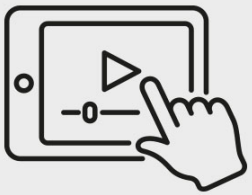
O tempo de compilação é quando a variável é associada ao seu tipo, o que ocorre antes da execução do programa. As demais opções são incorretas: o tempo de amarração ocorre em diferentes fases do programa, não apenas durante o tempo de execução; o binding tardio (late binding) geralmente proporciona menor eficiência de execução do programa em comparação com o binding cedo; já a ligação de módulos compilados previamente ocorre durante o tempo de ligação, não durante o tempo de projeto de linguagem; a associação de valores a variáveis que dependem de entradas do usuário ocorre durante o tempo de execução, não durante o tempo de carga.

Escopo e tempo de vida

Entender o escopo das variáveis é essencial para qualquer programador, especialmente para iniciantes em Python. O escopo determina onde uma variável pode ser acessada e modificada dentro de um programa. Em Python, existem diferentes níveis de escopo, como o escopo local, global e de função, que refletem diretamente no tempo de vida das variáveis. Compreender essas diferenças permite que os programadores escrevam código mais organizado, evitando conflitos e garantindo que as variáveis sejam utilizadas de maneira adequada em diferentes partes do programa. Dominar o escopo das variáveis é essencial para desenvolver aplicativos Python eficientes e livres de erros.

Neste vídeo, falaremos sobre o conceito de escopo, que se refere à visibilidade de variáveis em diferentes partes do código. Abordaremos também o tempo de vida, que indica a duração em que as variáveis permanecem na memória durante a execução do programa. Acompanhe!

Para assistir a um vídeo sobre o assunto, acesse a versão online deste conteúdo.



Escopo de visibilidade

O escopo define em quais partes do programa uma variável é visível. Cada nome de variável em Python tem seu escopo e, fora desse escopo, o nome não existe, gerando um erro quando se tenta referenciar esse nome. Quanto ao escopo, chamamos as variáveis de **globais** ou **locais**. A seguir, detalharemos cada tipo de variável. Acompanhe!

Código Python em tela de computador.

Variáveis globais

Todos os nomes atribuídos no prompt interativo do Python (prompt do emulador, Python Console do PyCharm etc.) ou em um módulo fora de qualquer função são considerados como de escopo global. Por exemplo, ao executar a instrução da caixa de código a seguir, a variável x é uma variável global. Veja!

Exercício

TUTORIAL COPIAR

Python3

Input

Console

Executar

Variáveis locais

Para exemplificar o uso de variáveis com escopo local, vamos utilizar uma função definida pelo desenvolvedor. Não se preocupe com esse tipo de função por enquanto, você aprenderá mais detalhes posteriormente. Por enquanto, observe o emulador a seguir.

Exercício

TUTORIAL COPIAR

Python3

Input

Console

Executar

Perceba que:

- As linhas 2, 3 e 4 compõem o bloco interno à função chamada **multiplicador()**.
- Embora as variáveis das linhas 2 e 6 tenham o mesmo nome, elas são abstrações a endereços de memória diferentes.
- Dentro da função **multiplicador()**, a chamada ao nome **a** recupera o valor 2.
- Fora da função **multiplicador()**, a chamada ao nome **a** recupera o valor 3.

Execute o código no emulador e veja a saída que será produzida.

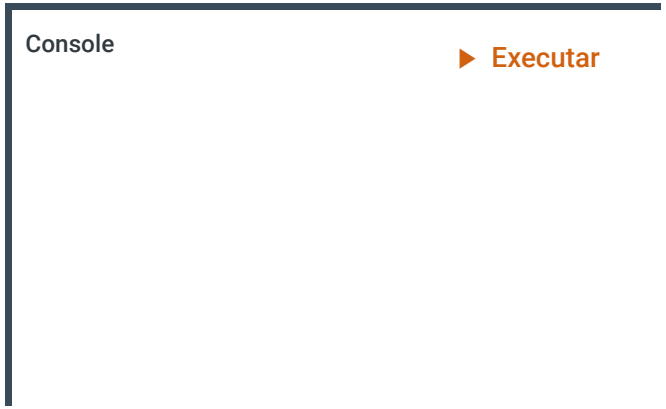
Agora, observe a função **multiplicador()** com uma pequena alteração, em que retiramos a inicialização da variável a dentro da função.

Exercício

TUTORIAL COPIAR

Python3

Input



Perceba que, na linha 5, ao se chamar a função do **multiplicador()**, a variável **a** será procurada. Como não existe uma variável **a** no bloco interno da função, ela é procurada como variável global. Uma vez encontrada, o valor recuperado é 3. Ao executar esse código, a saída obtida será *A variável b vale 15*.

Agora, confira por si mesmo, clique em Executar no emulador anterior.

Usamos o exemplo para mostrar que o interpretador Python pode procurar o mesmo nome de variável em diferentes escopos. A ordem utilizada para a procura é:

1. A chamada da função delimitadora.
2. As variáveis globais.
3. O módulo builtins.

Perceba que, se a variável **a** é inicializada na função **multiplicador()**, qualquer chamada a esse nome dentro da função resultará na referência a essa variável local. Mas seria possível alterar a variável **a** global com uma instrução dentro da função **multiplicador()**? Sim, utilizando-se a palavra reservada **global**.

Agora é a sua vez! Veja no emulador como isso poderia ser feito. Clique em Executar e obtenha o resultado do código.

Exercício

TUTORIAL COPIAR

Python3

Input

Console

Executar

Escopos

Os escopos podem ser estáticos ou dinâmicos. Observe a diferença entre os dois tipos.

Estático

O escopo é baseado na descrição textual do programa e as amarrações são feitas em tempo de compilação. É o caso de C, C++ e Java, por exemplo.

Dinâmico

O escopo é baseado na sequência de chamada dos módulos (ou funções). Por isso, as amarrações são feitas em tempo de execução. É o caso do Python.

O fato de Python ser de escopo dinâmico traz alguns problemas, como a perda de eficiência – uma vez que os tipos precisam ser verificados em tempo de execução – e a redução na legibilidade – porque é difícil determinar a sequência exata de todas as chamadas de função.

Tempo de vida

Embora escopo e tempo de vida tenham uma relação próxima, eles são conceitos diferentes. Observe!

Escopo

Tempo de vida

É um conceito textual.

É um conceito temporal.

As variáveis globais têm o tempo de vida que é o de execução do programa, ao passo que as variáveis locais somente existem no intervalo de duração da função ou do bloco a que se limitam.

Atividade 3

(IF-CE/2017/Técnico de Laboratório Informática) Considere o trecho do programa Python a seguir.

Python

Os valores impressos, ao se executar o programa, são, respectivamente:

A 1 e 1

B 10

C 1 e 10

D 10 e 10

E 10 e 1

Parabéns! A alternativa C está correta.

A variável **x** da linha 2 é local da função **func()**, sendo visível para a chamada **print()** da linha 3. Por sua vez, a variável **x** da linha 4 é global, sendo visível para a chamada **print()** da linha 6.

Constantes

Em Python, não existe um tipo de dado específico para constantes como em algumas outras linguagens de programação. No entanto, é comum utilizar variáveis com nomes em letras maiúsculas para representar valores que não devem ser alterados ao longo do programa, seguindo uma convenção de nomenclatura para indicar que são constantes. Embora essas variáveis ainda possam ser reatribuídas, a prática de nomeá-las em letras maiúsculas ajuda a indicar sua intenção de serem tratadas como constantes. Apesar de Python não impor a imutabilidade das constantes, essa convenção contribui para tornar o código mais claro e legível, facilitando a manutenção e compreensão do programa.

No vídeo a seguir, falaremos sobre constantes em Python. Nessa linguagem, não existem constantes no sentido estrito, mas convenções de nomenclatura que são frequentemente utilizadas para indicar que um valor deve ser tratado como constante, embora ainda seja possível modificá-lo. Acompanhe!

Para assistir a um vídeo sobre o assunto, acesse a versão online deste conteúdo.

Definição

Em Python, não existe o conceito de constante. Se você precisar de uma constante ao longo de sua jornada como programador, atribua o valor a uma variável e tome cuidado para não mudar esse valor.

Inicie o nome dessa variável com c_ ou utilize todas as letras maiúsculas, o que vai diferenciar essa variável das outras.

Exemplo

É possível utilizar a expressão c_PI = 3.141592 para armazenar o valor de PI e agilizar o cálculo de área e perímetro de um círculo, ou utilizar a expressão PRECISION = 0.001 para armazenar a precisão a ser utilizada em qualquer cálculo matemático no seu programa.

Veja outro exemplo no código a seguir.

Exercício

TUTORIAL COPIAR

Python3

Input

Console

Executar

Aqui, CONSTANTE_NUMERICA é definida com o valor 10 e é utilizada na multiplicação. Note que, apesar de ser uma constante, seu valor não é imutável. Se desejar criar uma constante verdadeiramente imutável, você pode usar a biblioteca constant ou o decorador @final do módulo typing (disponível a partir do Python 3.8).

> **Fique atento ao uso correto das variáveis, especialmente observando as questões de escopo e visibilidade, para evitar que algum cálculo seja realizado corretamente, mas com resultado diferente do esperado por você ao programar.**

Agora, vamos realizar uma atividade para consolidar nossos conhecimentos.

Atividade 4

É um fato que não há um tipo de dado específico para constantes numéricas em Python. Considerando o texto sobre o uso de constantes em Python, analise as alternativas e assinale a opção correta:

A

Em Python, as constantes numéricas são declaradas usando a palavra-chave `const`.

B A convenção de nomenclatura para constantes numéricas em Python sugere o uso de letras minúsculas.

C Não há nenhuma forma para criar constantes verdadeiramente imutáveis em Python.

D A declaração de CONSTANTE_NUMERICA não se constitui em uma constante, apesar do nome.

E Python oferece suporte nativo para a definição de constantes numéricas.

Parabéns! A alternativa D está correta.

Apesar do nome CONSTANTE_NUMERICA, não se trata de uma constante imutável, pois é possível reatribuir seu valor. No entanto, a convenção de nomenclatura em letras maiúsculas sugere que ela seja tratada como uma constante, ou seja, um valor fixo que não deve ser alterado ao longo do programa. As demais opções são incorretas: em Python, as constantes numéricas não são declaradas usando a palavra-chave `const`; a convenção de nomenclatura sugere o uso de letras maiúsculas para constantes. Além disso, não há suporte nativo para criar constantes verdadeiramente imutáveis em Python. É necessário instalar a biblioteca constant ou o decorador @final do módulo typing (disponível a partir do Python 3.8).

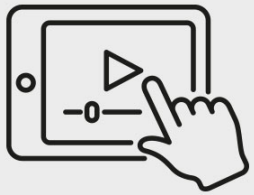
Variáveis na prática

Não há dúvidas de que a prática é essencial para solidificar os conceitos aprendidos e desenvolver habilidades eficazes na programação. Nesta fase, vamos criar um exemplo simples para demonstrar o conceito de binding em Python, envolvendo tanto tipos numéricos quanto strings.

Ao criar uma função que recebe dois argumentos – um número e uma string – e imprimir esses valores, os programadores podem visualizar na prática como os valores são vinculados aos parâmetros durante a chamada da função. Isso não apenas ajuda a compreender melhor o conceito de binding, mas também a aplicá-lo de forma eficiente em situações reais de desenvolvimento de software.

Neste vídeo, praticaremos a implementação de variáveis utilizando a linguagem de programação Python. Acompanhe!

Para assistir a um vídeo sobre o assunto, acesse a versão online deste conteúdo.



Roteiro de prática

Nesta etapa, vamos criar um exemplo simples para demonstrar o conceito de binding em Python, envolvendo tipos numéricos e strings. Vamos criar uma função que recebe dois argumentos: um número e uma string. Em seguida, a função irá imprimir o número e a string fornecidos como argumentos.

Siga o passo a passo:

1. Defina uma função chamada imprimir_dados que aceita dois parâmetros: um número e uma string.
2. Dentro da função, imprima o número e a string usando a função print().
3. Forneça dois valores para a função imprimir_dados: um número e uma string.
4. Chame a função imprimir_dados com os valores fornecidos.

Confira a seguir o código Python correspondente ao passo a passo. Depois, observe o resultado da execução do exemplo de binding do código no Python Console do PyCharm.

Exercício

TUTORIAL COPIAR

Python3

Input

Console

Executar

O exemplo demonstrou como os argumentos (um número e uma string) são vinculados aos parâmetros (número e texto) da função imprimir_dados. Quando a função é chamada com os valores 10 e "Olá, mundo!", esses valores são vinculados aos parâmetros dentro da função, resultando na impressão do número e da string fornecidos. Isso exemplifica o conceito de binding em Python.

Agora é a sua vez! Reveja o que estudou até aqui e pratique o que aprendeu!

Atividade 5

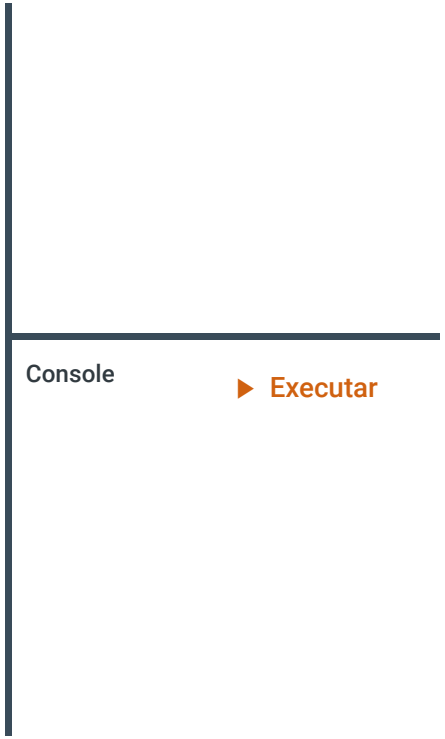
Elabore um programa em Python para criar uma função simples em Python que recebe dois argumentos – um número e uma string – e imprime esses valores, destacando o conceito de binding.

Exercício

TUTORIAL COPIAR

Python3

Input



Digite sua resposta aqui

Chave de resposta

O resultado esperado deve seguir o passo a passo a seguir.

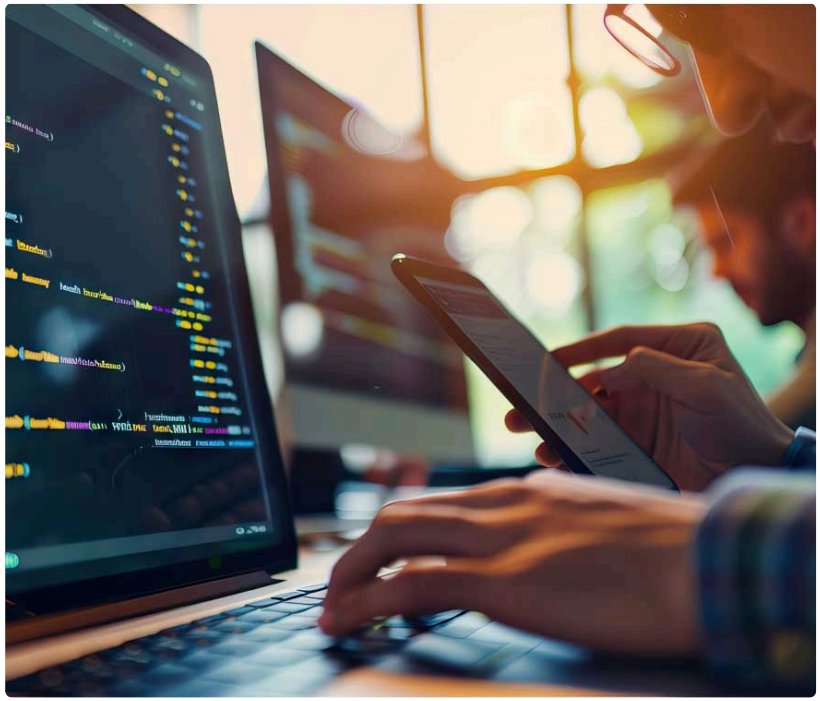
1. Execute o Pycharm.
2. Pressione a opção Python Console.
3. Defina uma função chamada imprimir_dados que aceita dois parâmetros: um número e uma string.
4. Dentro da função, imprima o número e a string usando a função print().
5. Forneça dois valores para a função imprimir_dados: um número e uma string.
6. Chame a função imprimir_dados com os valores fornecidos e observe como os valores são vinculados aos parâmetros.
7. Digite o programa a seguir.

Python

8. Confira o resultado da execução do programa!

```
Fora da funcao:
Número antes da chamada da funcao: 42
Texto antes da chamada da funcao: Python e incrivel!
Dentro da funcao:
Número vinculado ao parametro 'numero': 42
Texto vinculado ao parametro 'texto': Python e incrivel!
Fora da funcao apos a chamada:
Numero apos a chamada da funcao: 42
Texto apos a chamada da funcao: Python e incrivel!
```

Resultado da execução do Programa.



3 - Tipos de dados e expressões em Python

Ao final deste módulo, você será capaz de identificar os tipos de dados e as expressões em Python.

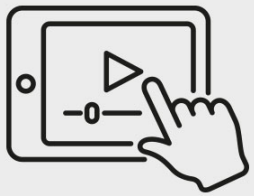
Tipos de dados padrão

Devemos compreender os tipos numéricos e sequenciais, pois esse conhecimento é essencial para explorar todo o potencial da linguagem Python. Os tipos numéricos

incluem inteiros (int) e números de ponto flutuante (float), que possibilitam realizar operações matemáticas e representar valores numéricos de diferentes formas. Dominar esses tipos de dados abre caminho para a criação de programas mais complexos e eficientes, permitindo representar e manipular uma gama de informações de forma flexível e poderosa.

Neste vídeo, falaremos sobre os tipos de dados padrão na linguagem Python, incluindo tipos numéricos e booleanos, bem com os operadores numéricos. Confira!

Para assistir a um vídeo sobre o assunto, acesse a versão online deste conteúdo.



Conceitos

Agora, você será apresentado aos tipos de dados padrão incorporados ao interpretador Python. Veja os principais **tipos internos**.

Numéricos

Sequenciais

Dicionários

Classes, instâncias e exceções também são tipos padrão, mas não entraremos em detalhes aqui. Para ter nosso primeiro contato com expressões em Python, use o prompt interativo >>>.

Ao digitar uma expressão algébrica no Python Console do PyCharm (ou em qualquer outro prompt interativo do Python) e pressionar a tecla [ENTER], o resultado da expressão é obtido. Isso ocorre porque o Python, através de seu prompt interativo, permite que você calcule expressões algébricas como uma calculadora, além de executar instruções básicas em Python. Confira!

Prompt

Tipos numéricos

Existem três tipos numéricos distintos em Python. Conheça!

1

Números inteiros

2

Números de ponto flutuante

3

Números complexos

Lembrando que os booleanos são um subtipo dos números inteiros.

O tipo int

É o tipo usado para manipular números inteiros. Fazendo uma analogia com a Matemática, o tipo int é usado para elementos do conjunto dos inteiros (**Z**).

> **Diferentemente de outras linguagens, como C ou Java, a linguagem Python não limita o tamanho de uma variável de qualquer tipo, logo, não existe um valor inteiro máximo definido. O limite depende da quantidade de memória disponível no computador.**

Novamente, fazendo uso do prompt interativo do Python, veja algumas coisas interessantes: digite 1_000_000 e pressione a tecla [ENTER], como no box a seguir.

Prompt

Python permite que você utilize o **underline** (_) como separador de milhar. Isso ajuda a visualizar números com muitos dígitos.

Para encerrarmos este primeiro contato com o tipo int, no emulador seguinte, digite print(type(1_000_000)), clique em Executar e verifique que o valor do exemplo anterior é um inteiro. Esse mesmo teste pode ser feito no PyCharm, basta digitar em seu Python Console o seguinte: type(1_000_000) e apertar a tecla [ENTER]. Veja!

Exercício

TUTORIAL COPIAR

Python3

Input

Console

Executar

O tipo float

É o tipo usado para manipular números com parte inteira e parte decimal, chamados de números de ponto flutuante. Fazendo uma analogia com a matemática, o tipo **float** é usado para elementos do conjunto dos reais (**R**).

Para diferenciar um número real de um inteiro, é possível utilizar a parte decimal zerada. No emulador a seguir, digite print(type(50.0)) e pressione Executar (No Python Console do PyCharm, digite type(50.0) e pressione [ENTER]). Veja o resultado.

Exercício

TUTORIAL COPIAR

Python3

Input

Console

Executar

Devemos usar o **ponto** para separar a parte inteira da parte decimal, e **não a vírgula**.

No emulador, digite x = 50.0. Em seguida, digite y = 50,0, depois print(x) e por fim, print(y). Após isso, clique em Executar.

> **Ao usar a vírgula como separador em Python, o que ocorre, na verdade, é a criação de uma tupla de dois elementos, e não o tipo**

float.

Embora os tipos **int** e **float** sejam semelhantes, por tratarem de números, eles têm propriedades um pouco diferentes. Em expressões algébricas, sempre que somamos, subtraímos ou multiplicamos apenas elementos do tipo int, o resultado é int. Porém, basta um operando do tipo **float** para que o resultado seja **float**.

Agora é a sua vez! Clique em Executar no emulador a seguir e observe o resultado no console.

Exercício

TUTORIAL COPIAR

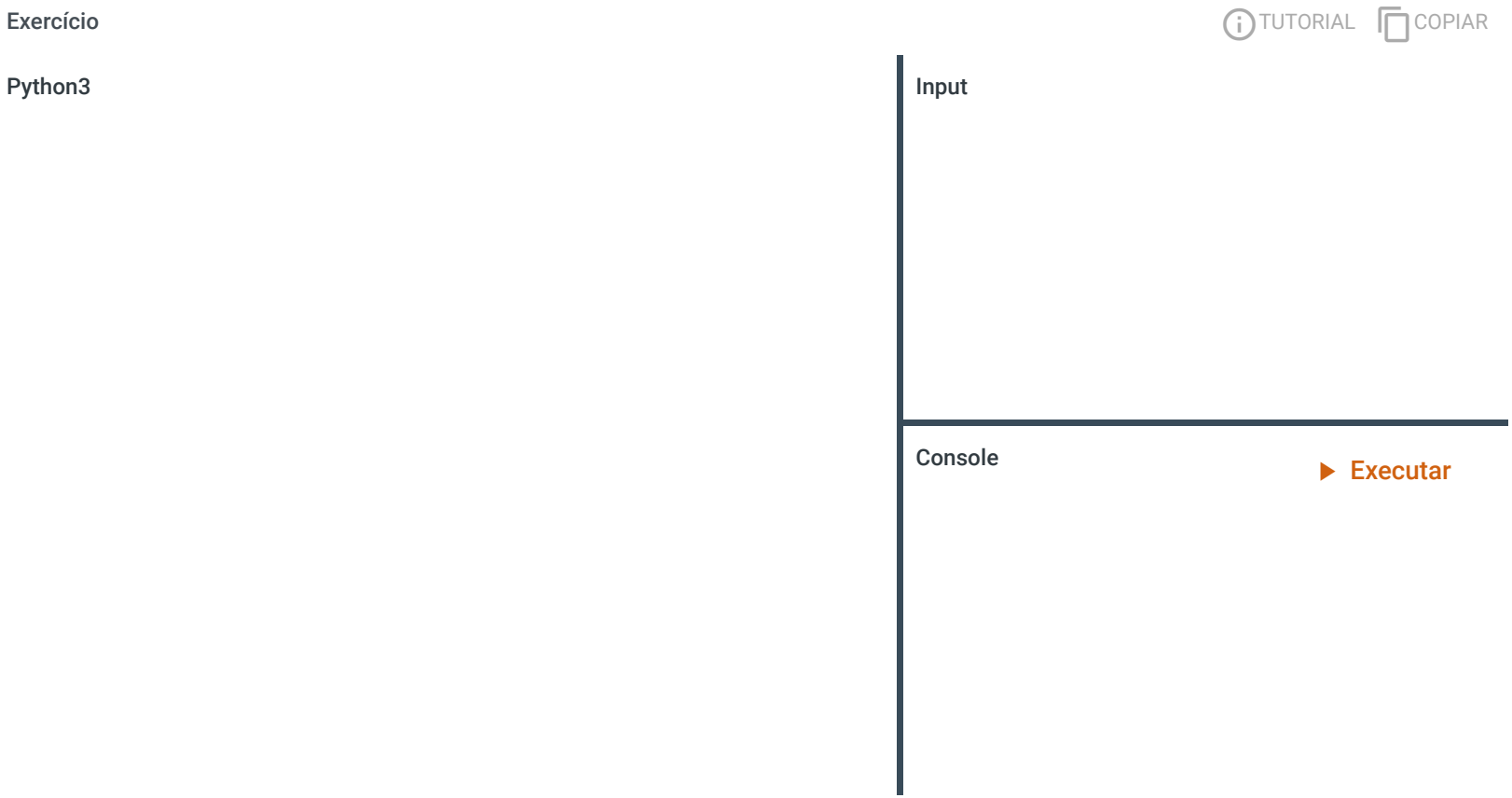
Python3

Input

Console

Executar

Agora, vamos analisar a exponenciação. Para realizar essa operação matemática, utilizamos o operador (**). Clique em Executar no emulador a seguir e observe!



Veja que basta que a base seja **float** para que o resultado também o seja.

> **Diferentemente de outras linguagens, como C, a divisão de dois números inteiros não necessariamente tem resultado inteiro.**

No emulador, digite as linhas de código a seguir, clique em Executar e observe o resultado.

```
x = 5/2
print(x)
```

Exercício

TUTORIAL COPIAR

Python3

Input

Console

Executar

No PyCharm, em seu Python Console, basta digitar 5/2 e pressionar [ENTER].

Para obter o quociente inteiro e resto, quando dois inteiros são divididos, é necessário utilizar os operadores // e %, respectivamente. Ao dividir 21 por 2, temos quociente 10 e resto 1. Observe no box a seguir.

Python

O tipo complex

É o tipo utilizado para manipular números complexos, na forma **x + yj**, sendo **x** a parte real e y a parte imaginária do complexo.

Veja dois exemplos de variáveis do tipo complex nas imagens seguintes, em que a parte real é 2 e a parte imaginária é 5.

```
>>> r = complex (2,5)
>>> r
(2+5j)
```

Complex 1.

```
>>> w=2+5j
>>> type (w)
<class 'complex'>
```

Complex 2.

A chamada **r.conjugate()** retorna o conjugado do número complexo r, em que a parte real é mantida e a parte imaginária tem o seu sinal trocado.

O tipo bool

Uma expressão algébrica, como vimos nos exemplos dos tipos **int** e **float**, é avaliada como um número, seja desses tipos ou de outro tipo numérico admitido em Python. Porém, utilizar expressões não algébricas também é bastante comum. E uma boa notícia é que Python pode avaliar expressões desse tipo também. Essa é uma diferença entre Python e outras linguagens, como C, por exemplo, em que não existe o tipo **bool**.

No prompt interativo (PyConsole do PyCharm, por exemplo), digite a expressão 2 < 3 e pressione [ENTER]. Ou, no emulador a seguir, digite print(2 < 3) e clique em Executar. Observe o resultado!

Exercício

TUTORIAL COPIAR

Python3

Input

Console

Executar

Repare que o resultado dessa expressão não é um número, mas sim a palavra True. Caso você colocasse a expressão 2 > 3, o resultado seria False. Faça o teste no PyCharm ou no emulador anterior.

> **As expressões que você viu nos dois exemplos são chamadas de expressões booleanas. Trata-se de expressões que podem ser avaliadas com um dos dois valores booleanos, True ou False. Assim, em Python, existe o tipo bool, utilizado para permitir o tratamento de expressões como essas.**

Agora, vamos ver o operador **not**, que é um operador unário, ou seja, só precisa de um operando. Esse operador inverte o valor booleano, ou seja, se o valor original for **True, not**(valor) terá o valor **False**. E vice-versa.

No prompt interativo, digite a expressão not(2 < 3) e pressione [ENTER]. Ou, no emulador, digite print(not(2 < 3)) e clique em Executar. Verifique que o resultado a ser obtido é o contrário do obtido anteriormente.

É possível também escrever expressões booleanas compostas, utilizando conectivos como **E OU**. Vamos ver mais detalhes sobre essas expressões ainda neste módulo. Fique atento!

Operadores numéricos

Operadores matemáticos

São muito semelhantes àqueles que vimos ao longo de nossa jornada como estudantes, aprendendo álgebra e aritmética na escola. Existem algumas pequenas diferenças, como a divisão (que pode ser a usual ou a divisão inteira). Mas é possível identificar operações que fizemos ao longo de toda nossa vida. A tabela a seguir lista os operadores de expressão aritmética disponíveis em Python. Relembre!

Operação matemática	Símbolo usado	Exemplo	
		Equação	Resultado
Soma	+	2.5 + 1.3	3.8
Subtração	-	2.5 - 1.3	1.2
Multiplicação	*	2.5 * 1.3	3.25
Divisão	/	2.5/1.3	1.923076923076923
Divisão inteira	//	9/2	4
Resto na divisão inteira	%	9%2	1
Valor absoluto	abs(parâmetro)	abs(-2.5)	2.5
Exponenciação	**	2**4	16

Tabela: Operadores matemáticos.
Humberto Henriques de Arruda.

Além das operações algébricas, é possível realizar operações de comparação. Os operadores de comparação têm como resultado um valor booleano (**True** ou **False**). Observe a tabela a seguir.

Símbolo usado	Descrição
<	Menor que
<=	Menor ou igual a
>	Maior que
>=	Maior ou igual a
==	Igual
!=	Não igual

Tabela: Operadores de comparação.
Humberto Henriques de Arruda.

Existe outra lista de operadores que executam operações matemáticas, mas, além disso, atualizam o valor da variável utilizada. Eles são chamados de operadores compostos e serão detalhados no módulo 4. Para mais funções matemáticas, você pode utilizar os módulos matemáticos math e fractions.

Dica

O operador utilizado para comparar se dois valores são iguais é o ==, ou seja, duplo sinal de igual. Tome cuidado para não confundir com o operador de atribuição, que é representado pelo sinal de igual apenas uma vez (=).

Operadores booleanos

As expressões booleanas são aquelas que podem ter como resultado um dos valores booleanos True ou False. É comum utilizarmos os operadores de comparação em expressões booleanas, mas não só eles.

Assim como é possível escrever expressões algébricas complexas concatenando diversas expressões menores, podemos escrever expressões booleanas grandes, com os operadores **and**, **or** e **not**.

Observe o comportamento dos operadores booleanos nas três imagens a seguir.

p	not (p)
True	False
False	True

Operador not

p	q	p and q
True	True	True
True	False	False
False	True	False
False	False	False

Operador and

p	q	p orq
True	True	True
True	False	True
False	True	True
False	False	False

Operador or

Atividade 1

Ao lidar com os tipos de dados numéricos em Python, é importante compreender as distinções entre os tipos int e float. Embora ambos representem números, suas

propriedades diferem sutilmente em expressões algébricas. Nesse sentido, ao executar as expressões print(type(5+3.0+1)) e print(type(102+3+1)), qual será o tipo do resultado de cada expressão?

A float, int

B int, float

C float, float

D int, int

E float, None

Parabéns! A alternativa A está correta.

No primeiro caso, 5+3.0+1 inclui um número em ponto flutuante (3.0), levando o resultado a ser um número em ponto flutuante (float). No segundo caso, 102+3+1 envolve apenas números inteiros, resultando em um número inteiro (int). Compreender

essas distinções é essencial para evitar resultados inesperados em operações numéricas em Python.

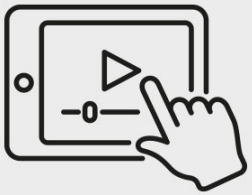
Relação de precedência entre operadores

O entendimento da precedência de operadores em expressões algébricas é um ponto muito importante para programadores iniciantes em Python, pois influencia diretamente no resultado das operações realizadas em seus programas.

Compreender como o Python avalia e executa expressões matemáticas pode evitar erros e garantir que os cálculos sejam feitos de acordo com as expectativas do programador. Apresentaremos agora as relações de precedência entre os operadores, destacando a importância de respeitá-las para obter resultados precisos e coerentes.

Neste vídeo, falaremos sobre a relação de precedência entre operadores em Python, que define a ordem de avaliação das expressões, determinando qual operação é realizada primeiro, seguindo as regras matemáticas convencionais. Acompanhe!

Para assistir a um vídeo sobre o assunto, acesse a versão online deste conteúdo.



Ao escrever uma expressão algébrica, o programador pode utilizar a precedência de operadores existente em Python (implícita) ou explicitar a ordem em que ele deseja que a expressão seja avaliada.

Exemplo

A expressão **3 + 2 * 5** tem como resultado 25 ou 13? Aprendemos no ensino fundamental que as operações de produto e divisão têm precedência sobre as operações de soma e subtração. Ou seja, um produto será realizado antes de uma soma, na mesma expressão. Assim, a expressão acima tem como resultado 13. Isso ocorre sempre que não forem explicitadas outras relações de precedência com o uso de parênteses. Caso o programador quisesse forçar que a soma ocorresse primeiro, ele deveria escrever assim: **(3 + 2) * 5**.

Sempre que o programador quiser forçar a ocorrência de uma operação antes de outras, ele pode utilizar os parênteses para aumentar a prioridade sobre ela. A tabela a seguir traz as relações de precedência entre os operadores, com as linhas mais altas tendo prioridade sobre as linhas mais baixas. Ou seja, elas ocorrem primeiro. Dentro da mesma linha, a precedência é da esquerda para a direita. Observe!

Operador	Descrição
[expressões ...]	Definição de lista
x[ ], x[índice : índice]	Operador de indexação
**	Exponenciação
+x, -x	Sinal de positivo e negativo

Operador	Descrição
*, /, //, %	Produto, divisão, divisão inteira, resto
+, -	Soma, subtração
in, not in, <, <=,>, >=, <>, !=, ==	Comparações, inclusive a ocorrência em listas
not x	Booleano NOT (não)
and	Booleano AND (e)
or	Booleano OR (ou)

Tabela: Operadores sequenciais.
Humberto Henriques de Arruda.

É importante ficar atento ao uso correto dos operadores, respeitando a precedência entre eles, para evitar que algum cálculo seja realizado corretamente, mas com resultado diferente do esperado por você ao programar.

Atividade 2

Considere a expressão a seguir: 2 + 3 – 4 ** 2 + 5 / 2 – 5 // 2

Assinale a opção com o valor correto dessa expressão em Python.

A -10.5

B -1

C 1.5

D 2

E 1

Parabéns! A alternativa A está correta.

Lembre-se de que o operador ** tem precedência maior do que os operadores / e //, os quais, por sua vez, têm precedência sobre + e -. Ou seja, primeiro será efetuada a exponenciação (4**2), depois as divisões, comum (5/2) e inteira (5//2), para posteriormente serem efetuadas as somas e subtrações.

Conversões de tipos de dados

Compreender as conversões implícitas e explícitas de tipos em Python é essencial para qualquer programador, especialmente para aqueles que estão iniciando sua jornada na linguagem. Ao escrever código Python, é comum lidar com diferentes tipos de dados em expressões matemáticas e lógicas. Assim, entender como o Python converte automaticamente os tipos para realizar operações é fundamental para evitar resultados inesperados e garantir a correta manipulação dos dados. Além disso, conhecer a possibilidade de realizar conversões explícitas oferece ao programador maior controle sobre o tipo de dados que está sendo utilizado em suas operações.

No vídeo a seguir, falaremos sobre as conversões de tipos de dados em Python que permitem alterar o tipo de uma variável para outro, facilitando a manipulação e o processamento de informações de acordo com as necessidades do programa. Confira!

Para assistir a um vídeo sobre o assunto, acesse a versão online deste conteúdo.

Quando temos tipos diferentes envolvidos na mesma expressão, o Python converte implicitamente cada operando para o tipo mais abrangente envolvido na expressão. Estamos usando a palavra abrangente, mas poderíamos falar que existem tipos que englobam (ou contêm) outros.

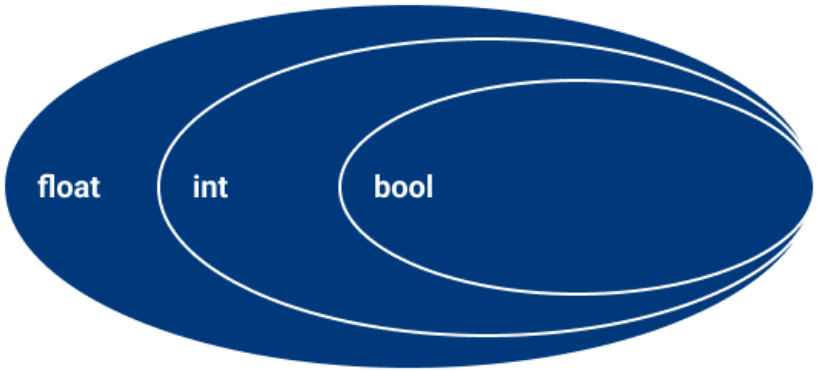
Exemplo

Um número do tipo **int** pode ser visto como um **float** com a parte decimal nula. Porém, o inverso não é verdade. Ou seja, o conjunto dos inteiros (int) é um subconjunto do conjunto dos reais (float). Assim, a expressão **5 + 0.68** – que envolve um **int** e um **float** – tem como resultado **5.68**. O inteiro 5 é convertido pelo Python para o número de ponto flutuante 5.0 antes que a soma (de dois valores **float**) seja realmente efetuada.

Uma conversão implícita não intuitiva é a dos valores booleanos **True** e **False** em inteiros, respectivamente, 1 e 0. Veja o exemplo a seguir.

Prompt

Com isso, podemos perceber a seguinte relação entre os tipos **bool**, **int** e **float** na imagem.



Relação entre os tipos bool, int e float.

Além das conversões implícitas, o programador também pode usar as conversões explícitas, quando ele força que o valor seja tratado como de determinado tipo. Para isso, é necessário usar o construtor do tipo desejado, com o valor passado como parâmetro (entre parênteses). Veja o exemplo!

Prompt

O **int** 2 pode ser tratado naturalmente como o **float** 2.0, basta acrescentar a parte decimal nula. Porém, ao tentar tratar um **float** como **int**, ocorre a remoção da parte decimal.

Fique atento, pois não é uma aproximação para o inteiro mais próximo, e sim o truncamento.

Atividade 3

No contexto da linguagem Python, as operações envolvendo diferentes tipos de dados são tratadas de maneira específica. Quando tipos de dados diferentes estão em uma expressão em Python, o que o Python faz implicitamente? Analise as alternativas e assinale a opção correta.

A Realiza uma conversão explícita de cada operando para o tipo mais abrangente.

B Realiza uma conversão para o tipo menos abrangente envolvido na expressão.

C Realiza uma conversão para o tipo inteiro envolvido na expressão.

D Não realiza nenhuma conversão, resultando em erro.

E Realiza uma conversão implícita para o tipo mais abrangente envolvido na expressão.

Parabéns! A alternativa E está correta.

O Python realiza conversões implícitas para garantir que os diferentes tipos de dados presentes em uma expressão sejam compatíveis. Essas conversões são feitas para o tipo mais abrangente envolvido na expressão, garantindo a coerência dos cálculos realizados. Por exemplo, no caso da expressão 5 + 0.68, o Python converte implicitamente o número inteiro 5 para o número de ponto flutuante 5.0 antes de realizar a soma com 0.68. Isso garante que a expressão produza o resultado esperado de 5.68.

Tipos de dados e expressões na

prática

Explorar as conversões de tipos em Python é essencial para aprimorar suas habilidades de programação. Ao compreender como o Python manipula diferentes tipos de dados e realiza conversões automáticas entre eles, você estará mais preparado para desenvolver programas eficientes e precisos. Com exemplos práticos e exercícios simples, você poderá consolidar seu entendimento sobre como converter entre tipos numéricos, strings e outros tipos de dados, preparando-se para enfrentar desafios mais complexos na programação. Coloque em prática seus conhecimentos e descubra o poder das conversões de tipos em Python.

No vídeo a seguir, praticaremos a implementação de tipos de dados e expressões utilizando a linguagem de programação Python. Acompanhe!

Para assistir a um vídeo sobre o assunto, acesse a versão online deste conteúdo.

Roteiro de prática

Vimos a importância da conversão de tipos em dados em Python. Esse tipo de situação é bastante comum na prática. Agora, vamos criar um exemplo simples que envolva conversões de tipos de dados e expressões em Python.

Passo 1

Vamos criar um programa que calcule a média de três notas fornecidas pelo usuário, mostrando o resultado final com duas casas decimais. Veja o código com o qual vamos trabalhar!

Python

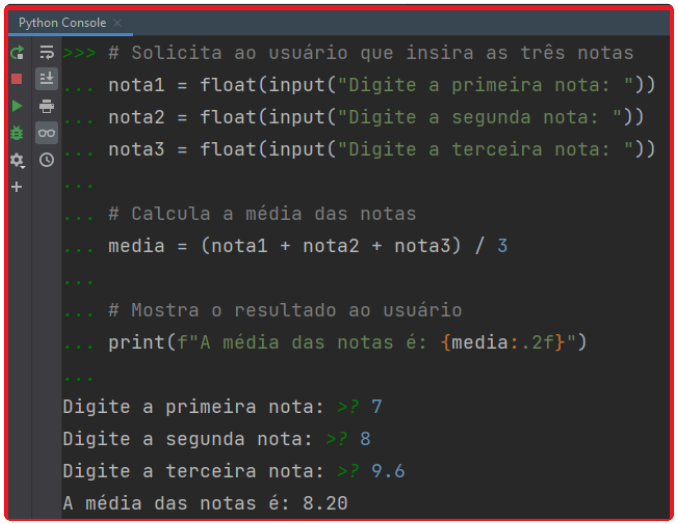
Veja as ações realizadas no exemplo.

1. Solicitamos ao usuário que insira as três notas, utilizando a função input.

2. Convertemos esses valores para o tipo float, pois a entrada padrão do input é uma string.
3. Calculamos a média das notas utilizando a fórmula (nota1 + nota2 + nota3)/3.
4. Exibimos o resultado da média ao usuário com duas casas decimais utilizando a formatação de string :.2f.

Passo 2

Confira a execução do código no Python Console e o respectivo resultado.



Resultado da execução do programa.

Esse exemplo é uma introdução prática às conversões de tipos de dados e expressões em Python, mostrando como podemos manipular diferentes tipos de dados e realizar cálculos simples de forma eficaz.

Agora é a sua vez de praticar. Reveja o que estudou até aqui e pratique o que aprendeu!

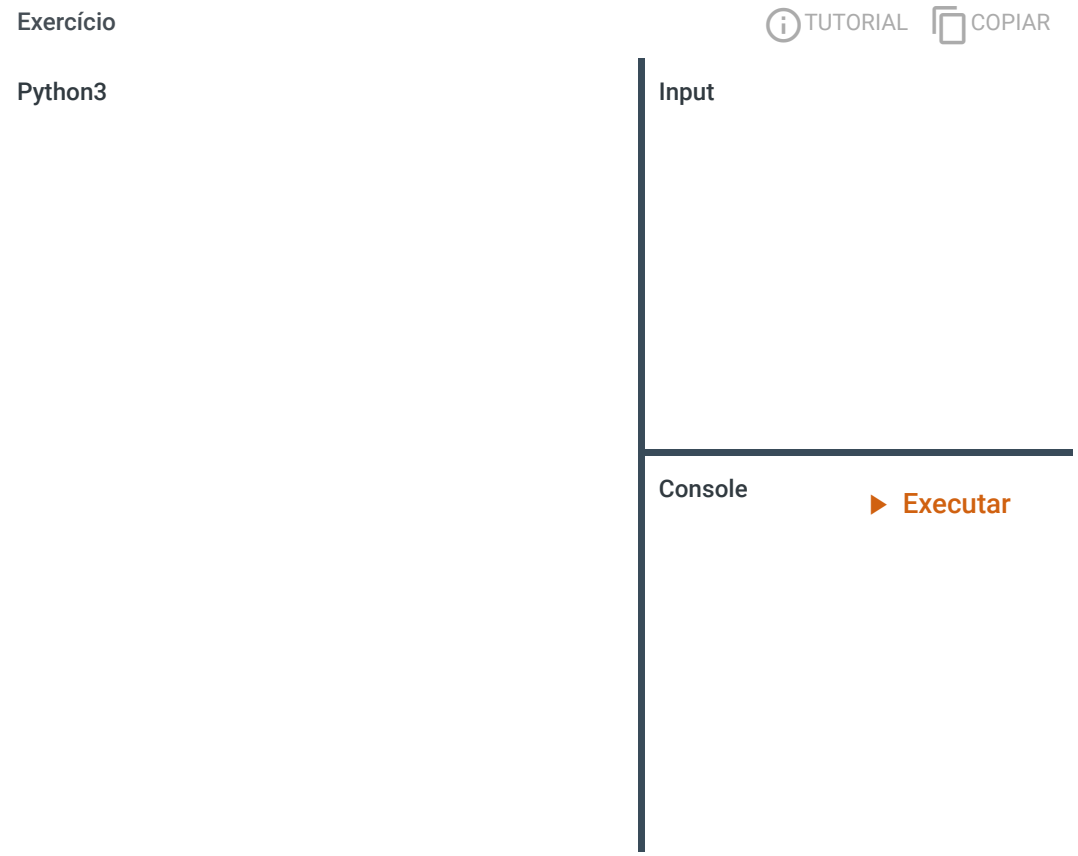
Atividade 4

Você foi designado para desenvolver um programa que solicita ao usuário que insira três números: um número inteiro, um número de ponto flutuante e uma string representando um valor booleano (True ou False). Seu objetivo é converter esses valores para os tipos corretos e exibi-los ao usuário de forma formatada. Você deve seguir as seguintes instruções:

1. Solicite ao usuário que insira um número inteiro.
2. Solicite ao usuário que insira um número de ponto flutuante.
3. Converta os valores fornecidos pelo usuário para os tipos corretos (int, float e bool).
4. Exiba os valores convertidos ao usuário de forma formatada, mostrando o tipo de cada valor.

Exemplo de saída:
```
Digite um número inteiro: 10
Digite um número de ponto flutuante: 3.14
Digite um valor booleano (True ou False): True
```

Valores convertidos:
- Número inteiro: 10 (tipo: int)
- Número de ponto flutuante: 3.14 (tipo: float)
- Valor booleano: True (tipo: bool)
```


```

Digite sua resposta aqui

Chave de resposta

Vamos fazer um passo a passo da solução. Acompanhe!

Passo 1: Solicite ao usuário que insira um número inteiro:

Python

Passo 2: Solicite ao usuário que insira um número de ponto flutuante:

Python

Passo 3: Solicite ao usuário que insira o valor booleano True ou False:

Python

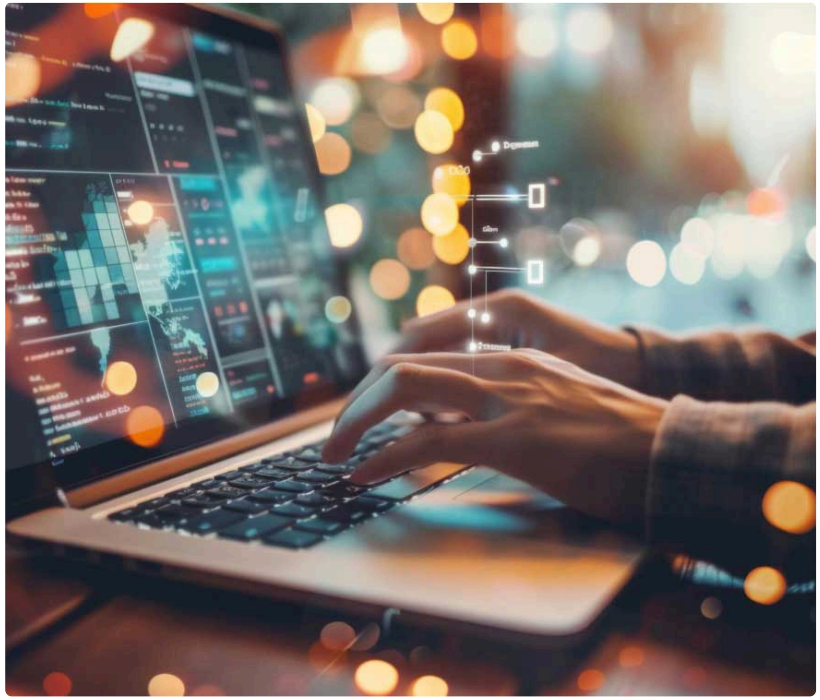
Passo 4: Converta os valores fornecidos pelo usuário para os tipos corretos (int, float e bool):

Python

Passo 5: Exiba os valores convertidos ao usuário de forma formatada, mostrando o tipo de cada valor:

Python

Esse é o gabarito passo a passo para a atividade prática de conversão de tipos em Python. Ele vai ajudar você na implementação do programa e na compreensão de como converter e exibir os diferentes tipos de dados fornecidos pelo usuário.



4 - Atribuição, entrada e saída de dados em Python

Ao final deste módulo, você será capaz de identificar na linguagem Python as formas de atribuição, de entrada e saída de dados.

Variáveis: formas de atribuição

Entender os conceitos de atribuição, entrada e saída de dados em Python é fundamental para qualquer programador, independentemente do nível de experiência.

Atribuição é o processo de associar um valor a uma variável, permitindo que os dados sejam armazenados e manipulados durante a execução do programa. Por outro lado, a entrada de dados refere-se à capacidade de receber informações do usuário, seja por meio do teclado, de arquivos ou de outros dispositivos externos. Já a saída de dados envolve a exibição de resultados ou mensagens ao usuário, seja no console, em interfaces gráficas ou em arquivos.

Dominar esses conceitos é essencial para criar programas funcionais e interativos em Python, pois permite que os desenvolvedores criem aplicações que recebam, processem e exibam informações de forma eficiente e intuitiva. Já vimos, basicamente, como podemos atribuir valor a uma variável. Vamos agora conhecer outras formas de atribuição.

Neste vídeo, falaremos sobre as variáveis em Python, que podem ser atribuídas de várias maneiras, oferecendo flexibilidade na manipulação de dados. Confira!

Para assistir a um vídeo sobre o assunto, acesse a versão online deste conteúdo.

Sentenças de atribuição

Atribuição simples

Chamamos de atribuição simples a forma que já utilizamos neste conteúdo, com uma expressão parecida com **x = 10**. Nessa atribuição, a variável x recebe o valor 10.

Atribuição múltipla

Python também permite a atribuição múltipla, ou seja, mais de uma variável receber atribuição na mesma linha. A título de exemplo, clique em Executar no emulador.

Exercício

TUTORIAL COPIAR

Python3

Input

Console

Executar

Observe que as variáveis **x** e **y** receberam atribuição na mesma instrução, com a variável x armazenando o valor 2, e a variável **y** armazenando o valor 5.

Vamos avançar!

Operadores de atribuição compostos

Os operadores de atribuição compostos executam operações matemáticas e atualizam o valor da variável utilizada. Clique em Executar o código a seguir e observe!

Exercício

TUTORIAL COPIAR

Python3

Input

Console

Executar

A variável **x**, inicialmente, recebeu o valor 10. Em seguida, a instrução **x = x + 1**, que causa estranheza quando lembramos da matemática aprendida ao longo da vida, é muito comum quando estamos programando. Essa instrução significa "acrescente uma unidade ao valor de x e guarde este resultado na própria variável **x**". Como **x** valia 10, o resultado do lado direito do operador (=) é 11. Esse resultado é, então, armazenado na própria variável **x**.

Dica

A operação de acrescentar determinado valor a uma variável e armazenar o resultado na própria variável poderia ser feita com o operador += (mais igual). Altere a 2ª linha no código do emulador anterior para **x += 1**, clique em Executar e veja o resultado.

Na tabela a seguir, estão os operadores compostos disponíveis em Python. Considere a variável x, com o valor inicial 10, para verificar os resultados.

Nome	Símbolo usado	Exemplo	
		Instrução	Resultado
Mais igual	+=	x += 2	x passa a valer 12
Menos igual	-=	x -= 2	x passa a valer 8
Vezes igual	*=	x *= 2	x passa a valer 20

Nome	Símbolo usado	Exemplo	
		Instrução	Resultado
Dividido igual	/=	x /= 2	x passa a valer 5
Módulo igual	%=	x %= 3	x passa a valer 1

Tabela: Operadores compostos.
Humberto Henriques de Arruda.

Diferentemente de C, em Python não é possível incrementar ou decrementar uma variável com um operador unário, como o ++ ou --.

Troca de variáveis

Um dos problemas iniciais que envolvem atribuição de valores a variáveis é a troca entre duas delas. Suponha que as variáveis a e b armazenem, respectivamente, os valores 1 e 2. Caso quiséssemos inverter os valores em linguagens como C ou Java, seria necessário usar uma variável auxiliar. Em Python, é possível fazer essa troca de

Código em tela de computador.

uma maneira muito mais fácil, com o uso
da atribuição múltipla.

Veja no código a seguir (clique em Executar) as duas maneiras de se trocar valores entre 2 variáveis, através do uso de variável auxiliar e através de atribuição múltipla. Veja!

Exercício

TUTORIAL COPIAR

Python3

Input

Console

Executar

Atividade 1

No contexto da programação, a troca de valores entre variáveis é uma tarefa comum e pode ser desafiadora em algumas linguagens de programação, como C ou Java, em que é necessário o uso de uma variável auxiliar. No entanto, em Python, essa tarefa é simplificada pelo conceito de atribuição múltipla. Já observamos duas maneiras de realizar a troca de valores entre as variáveis a e b: a primeira usando uma variável auxiliar chamada temp, e a segunda utilizando atribuição múltipla.

A atribuição múltipla em Python permite que valores sejam trocados entre variáveis de forma direta e elegante, sem a necessidade de variáveis auxiliares. Nesse sentido, selecione a opção correta que apresenta a principal diferença entre as duas maneiras de trocar valores entre as variáveis a e b do mesmo tipo.

A primeira maneira usa uma variável auxiliar, enquanto a segunda usa uma função de troca embutida em Python.

B

A primeira maneira usa uma variável auxiliar, enquanto a segunda usa atribuição múltipla.

C A primeira maneira usa uma variável auxiliar, enquanto a segunda usa uma técnica de troca de ponteiros.

D A primeira maneira usa uma função de troca embutida em Python, enquanto a segunda usa uma variável auxiliar.

E A primeira maneira usa uma técnica de troca de ponteiros, enquanto a segunda usa atribuição múltipla.

Parabéns! A alternativa B está correta.

Na primeira maneira de trocar valores entre as variáveis a e b, uma variável auxiliar chamada temp é usada para armazenar temporariamente um dos valores enquanto os valores são trocados. Por outro lado, na segunda maneira, conhecida como atribuição múltipla, os valores são trocados diretamente utilizando a sintaxe a, b = b, a, eliminando a necessidade de uma variável auxiliar. Isso demonstra como a atribuição múltipla em Python simplifica tarefas comuns de manipulação de variáveis.

Programação em Python

Aprender programação é essencial nos dias de hoje, e Python se destaca como uma escolha popular para iniciantes devido à sua sintaxe clara e legível, semelhante ao inglês. Python é versátil, usado em diversas áreas como desenvolvimento web, científico, automação e inteligência artificial. Suas bibliotecas prontas para uso facilitam a criação de projetos funcionais desde o início. Além disso, a comunidade oferece amplo suporte e recursos de aprendizado, tornando o processo de aprendizado mais acessível e estimulante para os novos programadores.

Neste vídeo, abordaremos a versatilidade da linguagem de programação de alto nível Python, que é fácil de aprender e amplamente utilizada em diversas áreas devido à sua sintaxe simples e vasta biblioteca padrão. Confira!

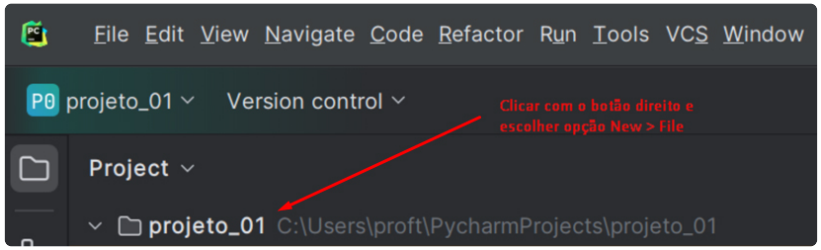
Para assistir a um vídeo sobre o assunto, acesse a versão online deste conteúdo.

O primeiro programa em Python

Para escrever um programa em Python, será essencial utilizar as formas de **saída de dados** para exibir ao usuário mensagens e resultados de operações. Caso você deseje que o usuário informe algum dado para que seu programa processe, será necessário utilizar as formas de **entrada de dados**.

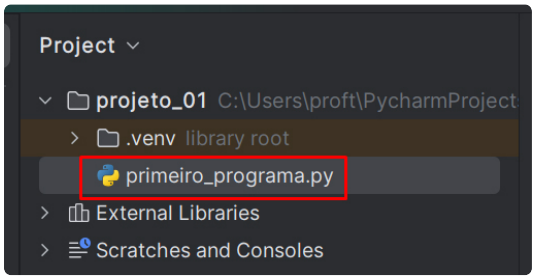
Para criar seu primeiro programa, utilize o PyCharm. Nele, clique com o botão direito do mouse no nome do projeto, na guia de navegação do lado esquerdo. Em seguida,

escolha a opção New > File, como na imagem a seguir.



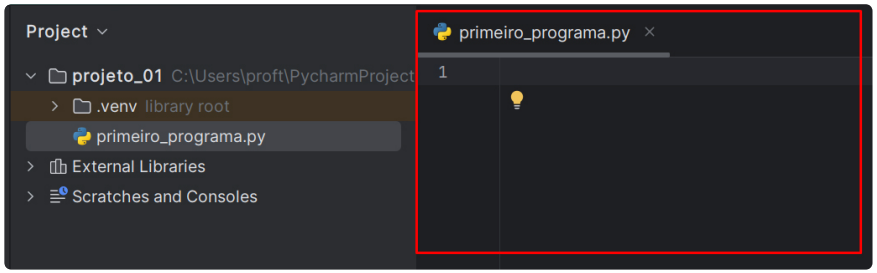
Novo arquivo no PyCharm.

Agora, atribua um nome a seu arquivo. Neste primeiro exemplo, vamos chamar de primeiro_programa.py. Veja!



Primeiro_programa.py.

Ao nomear o arquivo, será aberta uma nova aba do lado direito, com o espaço para que você efetivamente digite as instruções.



Aba de codificação.

Saída de dados com a função print()

A função **print()** em Python atua de forma semelhante à **printf()** em C. Confira agora as maiores diferenças entre elas para um programador iniciante.

Duas chamadas da print() em Python

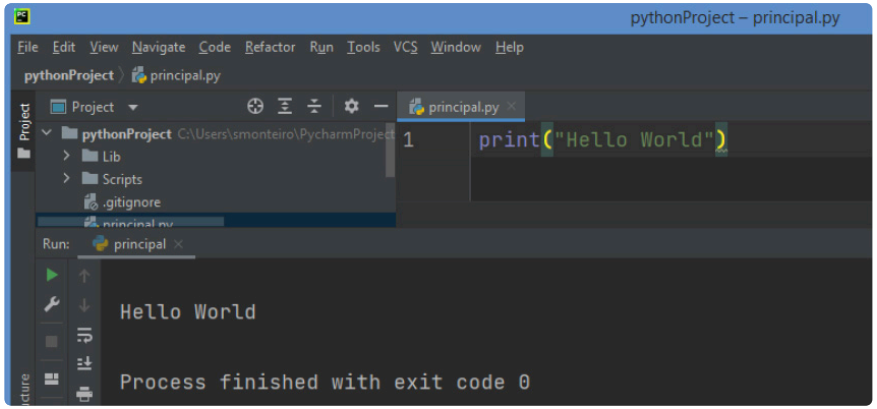
São impressas na tela em linhas diferentes, sem a necessidade do uso do caractere ‘\n’ para pular a linha, como ocorre na printf() em C.

×

Uma chamada da print() em Python

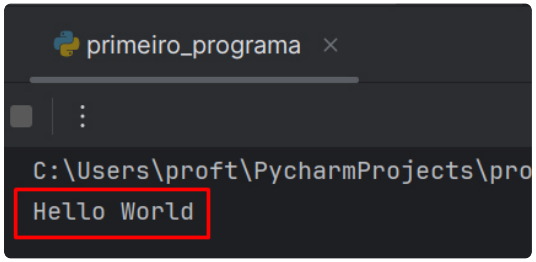
Permite a impressão de valores de variáveis sem a indicação do formato, como ocorre na printf() em C, quando precisamos escrever %c, %d ou %f, por exemplo.

Para escrever seu Hello World em Python, digite a seguinte linha, exatamente como está escrita.



Execução do Hello World em Python.

Em seguida, clique com o botão direito do mouse sobre o nome do programa e escolha a opção Run 'primeiro_programa'. Também é possível executar com a combinação de teclas CTRL+Shift+ F10. Após executar, observe o console na imagem.

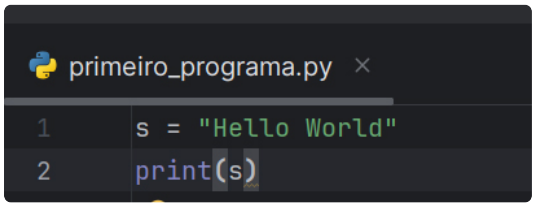


Execução do Hello World em Python.

Veja que foi impresso no console exatamente o que colocamos entre aspas, ao chamar a função **print()**. Essa é a primeira forma de saída de dados, usar a função **print()** com uma string sendo passada como parâmetro (entre os parênteses).

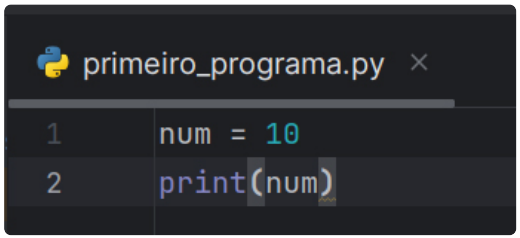
> A função print(), além de imprimir a string, também salta o cursor para a próxima linha.

Como você deve ter percebido, o que a função **print()** recebeu entre parênteses foi uma string. Ou seja, poderíamos ter passado para ela uma string já definida. Veja no exemplo!

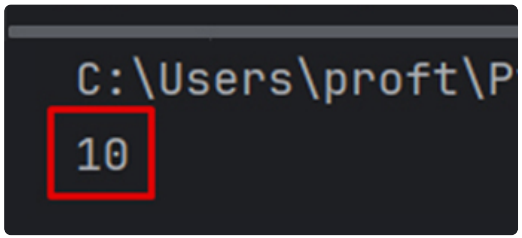


Hello World com string.

Também poderíamos ter passado como parâmetro uma variável já definida. A função print() vai trabalhar com o valor dessa variável. Observe as imagens.



Print de variável.

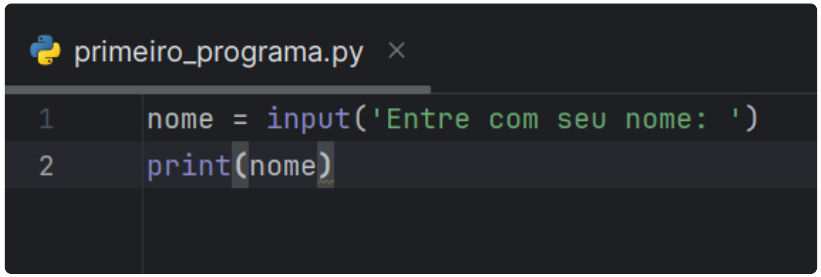


Execução do print com variável.

Entrada de dados com a função Input()

Quando o programador quiser que o usuário entre com algum valor, ele deverá exibir na tela o seu pedido. Em C, é necessário utilizar a função printf() para escrever a

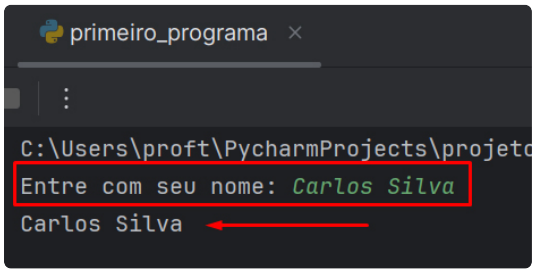
solicitação ao usuário e a função scanf() para receber a entrada e armazenar em uma variável. Em Python, é possível utilizar a função input(). Ela tanto exibe na tela o pedido, como permite que o valor informado pelo usuário seja armazenado em uma variável do seu programa. Analise a imagem.



A função input().

A linha 1 fará com que a frase **Entre com seu nome**: seja exibida no console, mas a execução do programa fica travada até que o usuário aperte [ENTER] no teclado. Tudo o que foi digitado até o [ENTER] vai ser armazenado na variável **nome**. A linha 2 fará a exibição do conteúdo da variável **nome**.

Veja o resultado no console, com o usuário tendo digitado **Carlos Silva**.



Execução da entrada de dados.

Perceba que a função input() trata tudo o que for digitado pelo usuário como uma string, armazenando na variável designada pelo programador para isso. Mesmo que o

usuário entre com apenas uma letra ou um número, isso será armazenado como uma string na variável.

Vamos, então, a mais um exemplo. Observe!

primeiro_programa.py

```
1  numero = input('Entre com um inteiro: ')
2  numero = numero + 2
```

A função input() e operação numérica.

Veja o console quando o programa é executado.

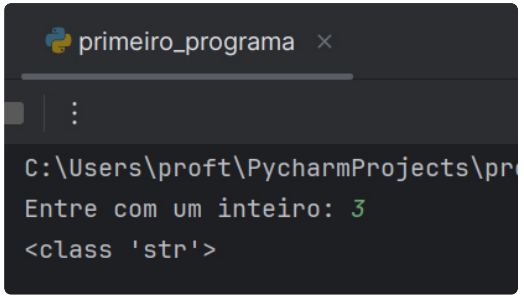
primeiro_programa

```
C:\Users\proft\PycharmProjects\projeto_01\.venv\Scripts\python.exe C:\Users\proft\PycharmProj
Entre com um inteiro: 3
Traceback (most recent call last):
  File "C:\Users\proft\PycharmProjects\projeto_01\primeiro_programa.py", line 2, in <module>
    numero = numero + 2
TypeError: can only concatenate str (not "int") to str
```

Erro com a função input().

O usuário digitou 3 e [ENTER]. Mesmo sendo um valor, a variável **numero** trata como a string '3'. Isso impede que seja realizada a operação de soma com o inteiro 2, por exemplo. Poderíamos também usar a instrução **print(type(numero))** na linha 2 para confirmar. Veja!

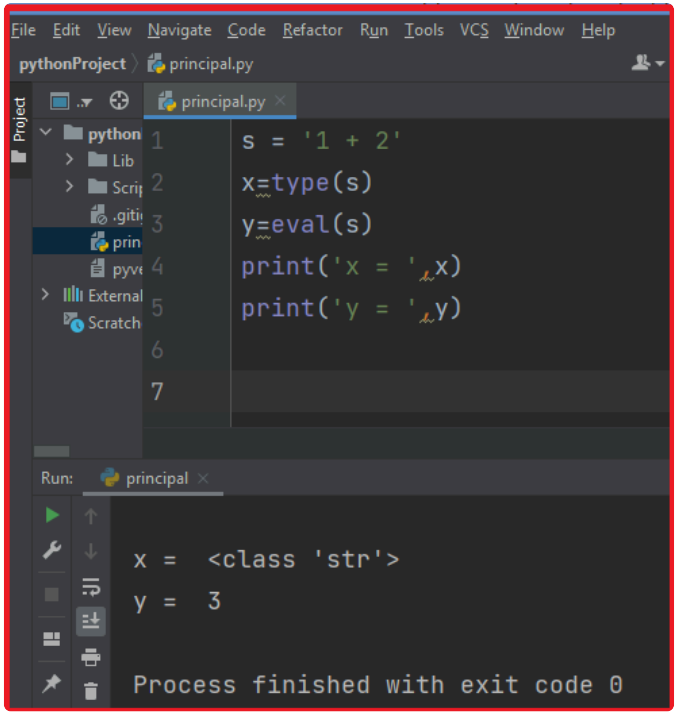
Função print() com função type().



Resultado de print() com type().

A função eval()

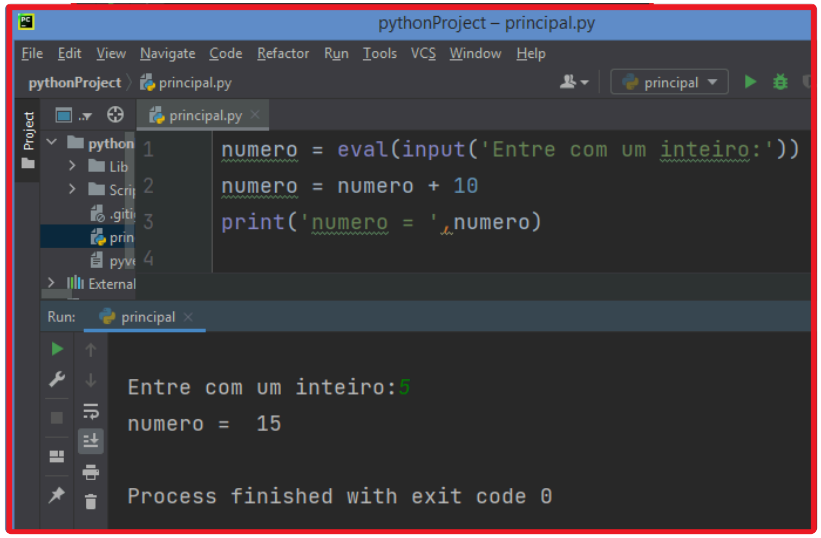
A função **eval()** recebe uma string, mas trata como um valor numérico. Veja o exemplo a seguir.



Função eval().

Mesmo tendo recebido a string ‘1+2’ como parâmetro, a função **eval()** efetuou a soma de 1 com 2. Observe que confirmamos que s é uma string com a instrução type(s).

Para tratar a entrada do usuário como um número e, com isso, realizar operações algébricas, por exemplo, é necessário utilizar a função **eval()** em conjunto com a **input()**. Veja!



Função eval() com função input() e resultado do uso de eval() e input().

Atividade 2

Em um projeto de desenvolvimento de software, um programador precisa solicitar informações ao usuário para interação com o sistema. Em Python, a função input() desempenha esse papel. Ela não apenas exibe a mensagem de solicitação ao usuário, mas também armazena a entrada fornecida em uma variável do programa.

É importante ressaltar que, ao utilizar a função input() em Python, qualquer entrada fornecida pelo usuário será tratada como uma string, independentemente de ser um caractere, um número ou uma sequência de caracteres. Portanto, o programador deve estar ciente desse comportamento ao manipular os dados recebidos. Nesse sentido, analise as alternativas e assinale a opção correta:

A Em Python, a função input() exibe uma mensagem solicitando uma entrada do usuário e armazena automaticamente o valor digitado em uma variável do programa.

B Ao utilizar a função input() em Python, todos os valores digitados pelo usuário são automaticamente convertidos para números inteiros.

C Em Python, a função print() é utilizada para receber entradas do usuário.

D Ao utilizar a função input() em Python, é necessário converter manualmente os valores digitados pelo usuário para outros tipos de dados, se necessário.

E A função input() em Python é exclusivamente utilizada para exibir mensagens na tela sem interação com o usuário.

Parabéns! A alternativa D está correta.

Em Python, a função input() é utilizada para receber entradas do usuário. Porém, por padrão, ela trata todos os valores digitados como strings. Isso significa que se o

usuário fornecer um número, por exemplo, ele será armazenado como uma string na variável. Para lidar com diferentes tipos de dados, como inteiros ou floats, é necessário realizar a conversão manualmente usando as funções int() ou float(). Portanto, ao utilizar a função input() em Python, o programador deve estar ciente desse comportamento e realizar as conversões apropriadas conforme necessário.

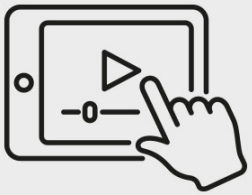
Formatação de dados de saída

Na jornada de aprendizado da programação em Python, compreender a formatação de dados de saída é um passo essencial para os iniciantes. A habilidade de apresentar informações de maneira clara e organizada não apenas torna o código mais legível, mas também facilita a comunicação de resultados e informações para o usuário final. Com a formatação adequada, os iniciantes podem criar saídas visualmente atraentes e compreensíveis, utilizando recursos como strings formatadas e métodos de formatação.

Esses conhecimentos melhoram a experiência do usuário e proporcionam uma base sólida para o desenvolvimento de habilidades mais avançadas na linguagem Python. Assim, a compreensão da formatação de dados de saída é um aspecto fundamental para os iniciantes que buscam construir aplicações funcionais e profissionais em Python.

Neste vídeo, falaremos sobre a formatação de dados de saída, que permite apresentar informações de maneira organizada e legível e também sobre impressão de sequências em Python. Confira!

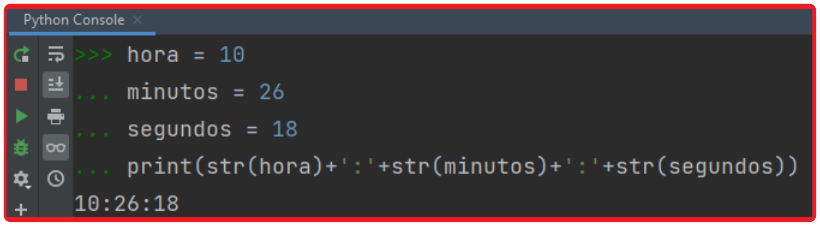
Para assistir a um vídeo sobre o assunto, acesse a versão online deste conteúdo.



Quando desejamos que a saída siga determinado padrão – por exemplo, de hora ou de data – existem algumas possibilidades para usar a função **print()**. É sempre possível utilizar a concatenação de strings, com o operador +, para montar a frase como quisermos. Suponha que tenhamos as variáveis a seguir.

- hora = 10
- minutos = 26
- segundos = 18

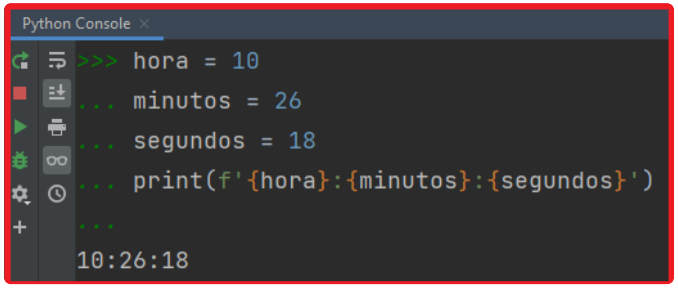
Poderíamos chamar a função **print()** com o separador da seguinte forma:



Saída de dados com separador.

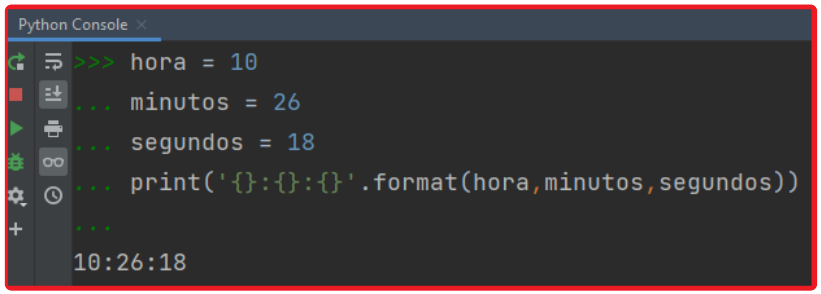
Porém, existe outra possibilidade, usando o método **format()**. Ele permite que a chamada à função **print()** fique muito parecida com as chamadas à função **printf()** em C, com passagem de parâmetros a serem colocados em ordem na string. Com o método **format()**, podemos montar a string com as chaves {} indicando onde entrarão

valores, passados como parâmetros separados por vírgulas, como mostrado na imagem.



Saída de dados com format().

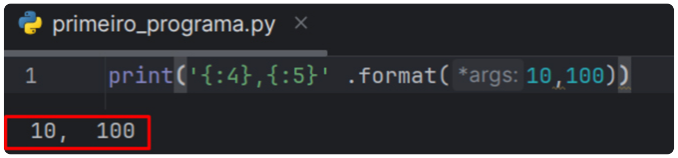
É importante observar que a quantidade de chaves precisa ser igual à quantidade de variáveis passadas como parâmetros no método **format()**. Seria muito bom se não precisássemos nos preocupar com essa correspondência para evitar erros bobos. E isso é possível! Para tornar a saída formatada ainda mais intuitiva, basta utilizar a letra 'f' no início dos parâmetros da função **print()** e colocar cada variável dentro das chaves na posição em que deve ser impressa. Veja como fica ainda mais fácil entender!



Saída de dados com print() e f'.

Também é possível especificar a largura de campo para exibir um inteiro. Se a largura não for especificada, ela será determinada pela quantidade de dígitos do valor a ser

impresso. Confira na imagem!

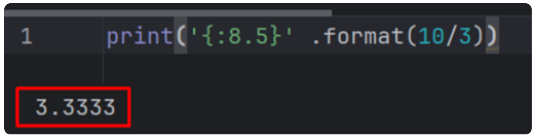


Largura de campo e função format().

Observe que os valores 10 e 100 foram impressos com espaços em branco à esquerda. Isso ocorreu porque definimos que a primeira variável deveria ser impressa com 4 espaços com {:4} (2 foram ocupados e 2 ficaram em branco), e que a segunda variável deveria ser impressa com 5 espaços com {:5} (3 foram ocupados e 2 ficaram em branco).

> O padrão de alinhamento dos valores é à direita do espaço reservado para a impressão da variável.

O método **format()** também pode ser usado para imprimir valores de ponto flutuante com a precisão definida. Vamos a mais um exemplo.

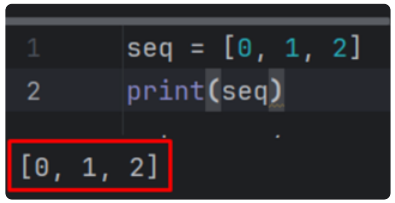


Função format() com ponto flutuante.

Ao usar **{:8.5}**, estamos determinando que a impressão será com 8 espaços, mas apenas 5 serão utilizados.

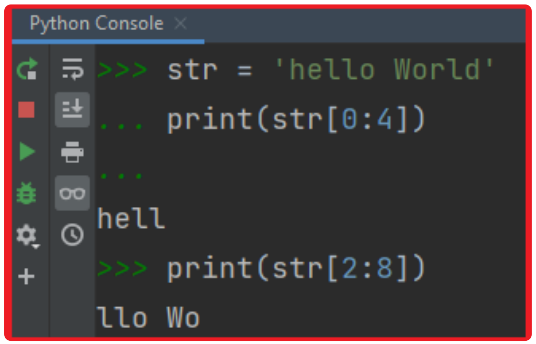
Impressão de sequências

Python também permite a impressão de sequências com mais possibilidades que C, incluindo as strings. Para imprimir um vetor em C, por exemplo, precisamos chamar a **printf()** item a item. Em Python, basta chamar a função **print()** passando como parâmetro a sequência. Veja!



Função print() com sequência.

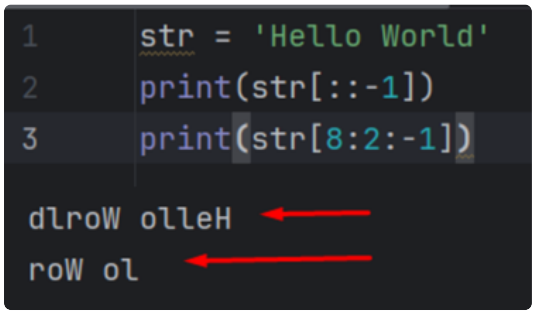
Para imprimir uma substring, por exemplo, basta utilizar os colchetes para indicar o intervalo de índices que deve ser impresso. Vale lembrar que o primeiro caractere da string é indexado com 0. Observe!



Função print() com substring.

Usar **[0:4]** provoca a impressão dos índices 0, 1, 2 e 3, mas não do índice 4. Analogamente, usar **[2:8]** provoca a impressão dos índices de 2 a 7, mas não do 8.

Também é possível imprimir a string como lida da direita para a esquerda. Para isso, deve-se utilizar **[: : -1]**. Esse valor -1 indica que a leitura dos caracteres será feita no sentido oposto ao tradicional. Observe!



Função print() – string em ordem reversa.

Fique atento quando utilizar o intervalo na impressão no sentido inverso, pois os limites do intervalo devem respeitar esse sentido.

Atividade 3

A formatação da saída de dados tem um papel fundamental na apresentação clara e organizada das informações em programas Python. Ao buscar um padrão específico, como exibir horas ou datas, os programadores têm diversas opções. Nesse sentido, analise as alternativas e assinale a opção correta:

A O método format() em Python permite que os parâmetros sejam passados como uma lista.

B A formatação de dados de saída em Python só pode ser realizada utilizando o operador + para concatenar strings.

C Ao utilizar o método format() em Python, não é necessário se preocupar com a correspondência entre as chaves e as variáveis passadas como parâmetros.

D A letra f no início dos parâmetros da função print() em Python indica que as variáveis serão passadas como strings.

E Especificar a largura de campo ao utilizar o método format() em Python garante que os valores serão impressos sem espaços em branco.

Parabéns! A alternativa C está correta.

No método format() em Python, as chaves {} indicam onde os valores serão inseridos e os parâmetros são passados como argumentos separados por vírgulas. Não é

necessário que a quantidade de chaves seja igual à quantidade de variáveis passadas, tornando o processo mais flexível e menos propenso a erros de correspondência.

Atribuição, entrada e saída de dados na prática

Para desenvolver habilidades sólidas em Python, é importante praticar conceitos fundamentais como atribuição, entrada e saída de dados. Atribuição permite que os programadores armazenem e manipulem valores em variáveis, enquanto entrada e saída de dados são essenciais para interação com o usuário e processamento de informações.

Logo Python em tela de computador.

Praticar esses conceitos fortalece a compreensão da linguagem e desenvolve habilidades práticas para resolver problemas do mundo real. Ao escrever e executar programas que envolvem atribuição de variáveis, solicitação de entrada do usuário e exibição de resultados, você ganha confiança em sua capacidade de criar soluções funcionais e eficientes em Python. Essa prática é essencial para construir uma base sólida e se preparar para desafios mais avançados na programação.

Neste vídeo, praticaremos a implementação de atribuição, entrada e saída de dados utilizando a linguagem de programação Python. Vamos lá!

Para assistir a um vídeo sobre o assunto, acesse a versão online deste conteúdo.

Roteiro de prática

Vamos criar agora um programa simples em Python que utiliza os conceitos de atribuição, entrada e saída de dados. O programa calculará o preço total de um pedido em um restaurante, considerando o valor dos itens escolhidos pelo usuário.

Passo 1

Atribuição de variáveis

Defina as variáveis para representar o preço de cada item do menu do restaurante. Veja no código.

Python

Passo 2

Entrada de dados do usuário

Solicite ao usuário que insira a quantidade de cada item que deseja pedir.

Python

Passo 3

Cálculo do preço total

Calcule o preço total do pedido, multiplicando o preço de cada item pela quantidade escolhida pelo usuário. Veja!

Python

Passo 4

Saída de dados

Exiba o preço total do pedido para o usuário.

Python

Passo 5

Execução do programa

Execute o programa e teste com diferentes quantidades de itens para verificar se o cálculo está correto. Veja um exemplo de saída.

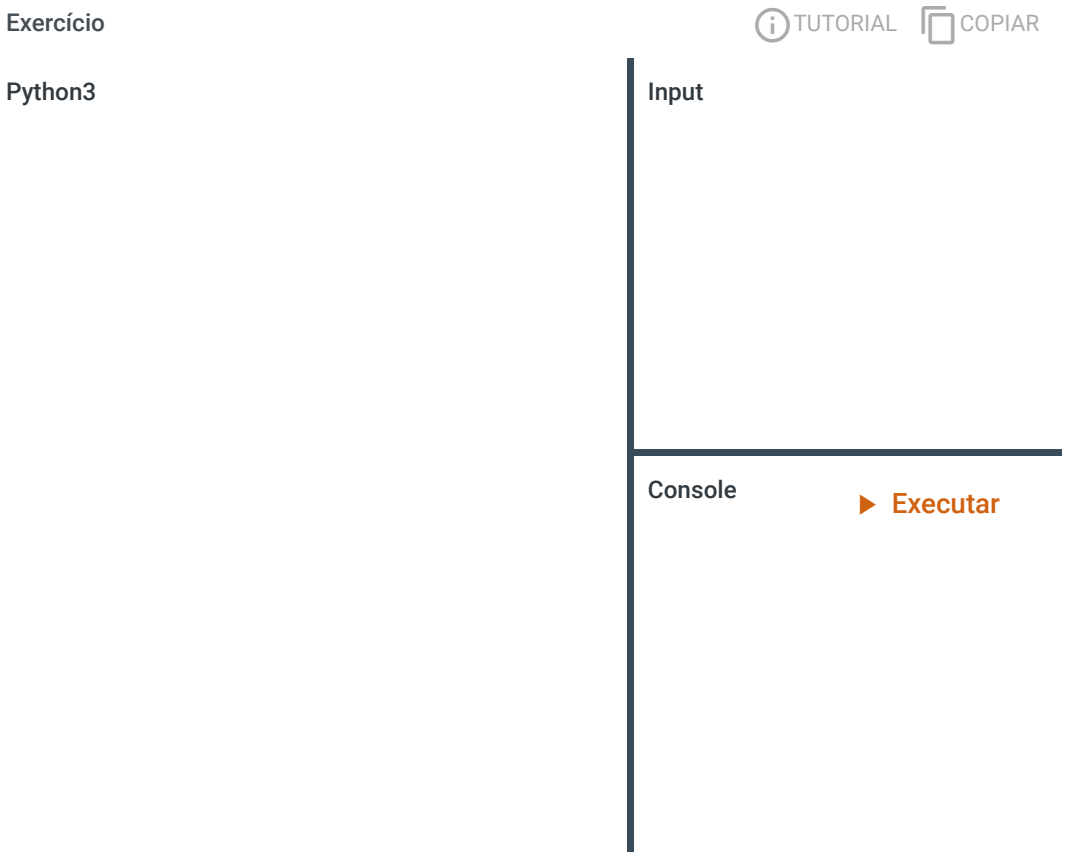
Python

Este roteiro fornece um exemplo simples e prático de como utilizar os conceitos de atribuição, entrada e saída de dados em Python para resolver um problema do mundo real.

Agora, é a sua vez de praticar. Reveja o que estudou até aqui e pratique o que aprendeu.

Atividade 4

Você foi designado para criar um programa em Python para um caixa de supermercado. O programa deve solicitar ao cliente a quantidade de cada produto adquirido e calcular o total da compra com base nos preços unitários dos itens. Utilize os conceitos de atribuição, entrada e saída de dados para implementar o programa.



Digite sua resposta aqui

Chave de resposta

Para calcular o total de uma compra em um supermercado, você deve declarar variáveis para os preços unitários dos produtos, solicitar ao usuário a quantidade de cada produto adquirido, calcular o total da compra e exibir o resultado para o usuário. O uso correto de variáveis, entrada de dados e operações matemáticas garantirá que o programa funcione corretamente, fornecendo o total da compra de acordo com as quantidades inseridas pelo usuário.

Veja a solução da atividade.

Passo 1: Declare variáveis para representar os preços unitários dos produtos no supermercado:

Python

Passo 2: Solicite ao cliente que insira a quantidade de cada produto adquirido:

Python

Passo 3: Calcule o total da compra, multiplicando o preço unitário de cada produto pela quantidade adquirida:

Python

Passo 4: Exiba o total da compra para o cliente:

Python

O que você aprendeu neste conteúdo?

- Linguagem Python e suas características.

- Uso de variáveis em Python.
- Os tipos de dados em Python.
- As expressões em Python.
- As formas de atribuição em Python.
- Entrada e saída de dados em Python.

Podcast

Ouça agora um resumo dos principais tópicos aqui abordados sobre a linguagem Python.

Para ouvir o *áudio*, acesse a versão online deste conteúdo.

Explore +

Confira agora o que separamos especialmente para você!

Se você quiser ter mais exercícios para treinar e desafios mais complexos, recomendamos a visita ao site da Python Brasil.

Como vimos, dentre as PEPs, destaca-se a **PEP8**, que estabelece um guia de estilo de programação. Sugerimos que você pesquise mais sobre isso.

Para mais funções matemáticas, você pode utilizar os **módulos matemáticos math e fractions**.

Referências

BELANI, G. **Programming Languages You Should Learn in 2020**. Consultado na internet em: 26 mai. 2020.

PERKOVIC, L. **Introdução à computação usando Python**: um foco no desenvolvimento de aplicações. Rio de Janeiro: LTC, 2016.

PYCHARM. **The Python IDE for Professional Developers**. Consultado na internet em: 15 jun. 2020.

PYTHON. **PEP 0 – Index of Python Enhancement Proposals (PEPs)**. Consultado na internet em: 2 jun. 2020.

PYTHON. **Fractions – Rational Number**. Consultado na internet em: 16 jun. 2020.

PYTHON. **Math – Mathematical Functions**. Consultado na internet em: 16 jun. 2020.

Material para download

Clique no botão abaixo para fazer o download do conteúdo completo em formato PDF.

Download material

O que você achou do conteúdo?

Relatar problema