

Instituto Federal da Paraíba

Apostila de Python para Iniciantes

Explorando o Universo da Programação com Aventura e Diversão
Aprenda Python de forma leve e divertida!



imagem gerada por IA

Sumário

- I. Apresentação do Cenário e dos Personagens
- II. Introdução à Programação com Python
 - A. O Conceito de Programação
 - B. As Vantagens da Escolha do Python
 - C. Preparando o Ambiente de Desenvolvimento
- III. Conceitos básicos
 - A. Algoritmos e Lógica de Programação
 - B. Tipos de Dados e Dados
 - C. Expressões e Operações
 - D. Variáveis e Atribuições
 - E. Entrada e Saída de Dados
 - F. Conversão de Tipos
- IV. Estruturas de Decisão
 - A. If
 - B. Else
 - C. Elif
- V. Estruturas de Repetição
 - A. For
 - B. While

I - HISTÓRIA E PERSONAGENS

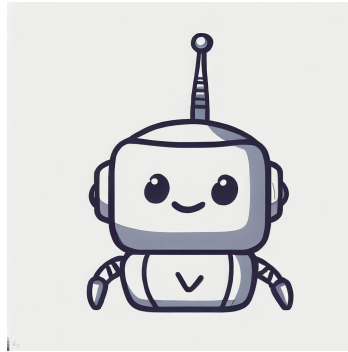
Em uma galáxia distante, um jovem ET alegre e curioso estava explorando o espaço sideral em sua nave espacial. Infelizmente, devido a um mau funcionamento em seu motor, ele acabou perdendo o controle e caiu acidentalmente na Terra.



ET curioso
imagem gerada por IA

Desorientado e com sua nave danificada, o ET sentia-se perdido em um planeta desconhecido. Foi então que ele encontrou Mariazinha, uma garota inteligente e curiosa, que vivia em uma casa próxima ao local do acidente. Mariazinha era conhecida por sua habilidade em lidar com tecnologia e sua mente inventiva.

Ao se deparar com o extraterrestre, Mariazinha ficou surpresa e fascinada. Junto ao ET ela encontrou um pequeno robô com um desenho de cobrinha em seu peito, foi então que ela o apelidou de Python. O robô Python é o responsável por possibilitar a Mariazinha conversar com o ET curioso, já que eles não falam a mesma língua. O robô Python será o tradutor da conversa entre o ET e Mariazinha, mas para ele fazer a tradução Mariazinha deve escrever em uma folha de papel em inglês (que é a língua que o robô entende) e entregar para o robô. O robô vai analisar o que foi escrito e fazer a tradução para o “ETnês”, devolvendo uma outra folha de papel.



Robô Python
imagem gerada por IA

Por sorte, Mariazinha estuda inglês desde seus 5 anos de idade. Com a ajuda do robô, Mariazinha conversou sobre várias coisas com o ET, sobre brincadeiras, artes, ciência e matemática e também sobre a vida fora da Terra. Após muito papo, ela ofereceu sua ajuda para consertar a nave e ajudar o ET a voltar para casa. O ET, emocionado com a generosidade da garota, aceitou a oferta e juntos começaram a trabalhar incansavelmente para restaurar a nave espacial.

Então eles começaram a trabalhar juntos no conserto da nave espacial. Identificaram as peças e ferramentas que iriam precisar para fazer os reparos e foram às compras. Depois de comprar todas as peças, eles identificaram as partes danificadas da nave e fizeram os reparos necessários.

Por fim, a nave estava completamente consertada e o ET curioso agradeceu Mariazinha por sua grande ajuda e se despediu com um belo aperto de mão.

II - INTRODUÇÃO A PYTHON

Bem-vindo à apostila de Python para iniciantes! Neste guia, você aprenderá os fundamentos da linguagem de programação Python de uma forma leve e divertida. Python é uma linguagem poderosa, versátil e fácil de aprender, amplamente utilizada em áreas como desenvolvimento web, análise de dados e automação de tarefas.

Python é uma linguagem de programação criada pelo matemático e programador *Guido van Rossum* em 1991. Embora muita gente faça associação direta com cobras, por causa das serpentes píton ou pitão, a origem do nome se deve ao grupo humorístico britânico Monty Python, que inclusive, em 2019, completou 50 anos de existência.

O objetivo de *Guido* era criar uma linguagem de programação fácil de ler, com uma sintaxe clara e legível. Ele queria que a linguagem fosse eficiente e produtiva, permitindo que os desenvolvedores escrevessem código de forma rápida e intuitiva.

Características importantes do python:

É case-Sensitive;
O que significa que diferenças de maiúsculas e minúsculas são consideradas distintas. Isso significa que palavras escritas com letras maiúsculas e minúsculas são tratadas como entidades diferentes. Por exemplo, as palavras "olá" e "Olá" têm significados diferentes em Python.

```
variavel = 10
VARIABEL = 20

print(variavel) # Saída: 10
print(VARIABEL) # Saída: 20
print(Variavel) # Erro! O nome da variável é diferente de "variavel"
```

Aprendizado gradual;
Python permite que você comece com conceitos básicos e vá progredindo gradualmente para tópicos mais avançados. Isso torna a curva de aprendizado mais suave e permite que você se concentre em construir uma base sólida.

```
print("Olá mundo!")
```

Python é uma linguagem muito famosa e grandes empresas utilizam ela. Aqui vai alguns exemplos de empresas que usam python:

- Netflix
- Spotify
- Google
- Amazon
- Facebook

A - O que é programação?

Vamos relembrar a história do ET curioso. Para Mariazinha conversar com o ET foi necessário que o robô Python fizesse a tradução do inglês para o “ETnês”,

lembra ? A partir disso Mariazinha conseguiu se comunicar com o ET para conversar, trocar informações, pedir ajuda ou ajudar em uma tarefa.

A programação funciona desta forma. Utilizamos uma linguagem que nós conhecemos para enviar instruções para o computador. Podemos dizer que é a ação de escrever um conjunto de instruções para o computador e dizer o que fazer e como fazê-lo. Ao escrever um conjunto de instruções precisas, podemos automatizar tarefas, resolver problemas complexos e criar soluções inovadoras. Existem muitas linguagens de programação, Javascript, PHP, Java e várias outras, mas nesta apostila vamos usar a linguagem Python. Mas porque escolhemos Python ?

B - Por que escolher Python?

Agora que aprendemos que programação é ação de escrever instruções para o computador executar, podemos entender porque o Python é uma boa linguagem que fará a tradução dessas instruções para o computador.

O Python é um ótima linguagem, pois possui uma sintaxe simples e de fácil compreensão, sendo talvez a melhor opção para quem está entrando no mundo da programação. Além disso, Python também é uma linguagem com diversos recursos, que permite você trabalhar nas mais variadas áreas da tecnologia, por exemplo desenvolvimento de jogos.

Python é uma excelente escolha para quem está começando a estudar programação. Na próxima seção vamos aprender a como configurar um ambiente de desenvolvimento para criarmos nossos programas.

C - Configurando o ambiente de desenvolvimento

Agora vamos entender um pouquinho sobre o processo de configuração do ambiente para utilizar o Python no celular e/ou no computador. Configurar corretamente o ambiente é o primeiro passo para começar a desenvolver em Python. A primeira coisa que você deve entender antes de começar a programar é sobre o interpretador Python.

Mas o que seria um interpretador e por que devo utilizá-lo? Bem, vamos voltar a história anterior, Mariazinha não sabe falar o “ETnês”, então ela precisa do robô para fazer a tradução do que ela falou (em inglês) para traduzir para o “ETnês”. A função do interpretador Python é exatamente esta, vamos fornecer instruções

escritas em inglês e no formato que Python exige, para então ele traduzir para a máquina o que ela deve fazer.

Agora que entendemos para que serve o interpretador, vamos precisar de um local para escrever nossos códigos. Você pode utilizar qualquer editor de texto para escrever seus programas, inclusive um simples bloco de notas. Mas mesmo que você possa usar, não significa que seja a melhor ferramenta para trabalhar. Existem programas que facilitam a nossa vida na hora de programar.

Um grande exemplo disso é o Replit, um aplicativo para celular que vai te auxiliar na hora de programar no celular. Ele permite criar e executar projetos diretamente do seu dispositivo móvel. Assim, através de um celular com acesso à internet, você terá a flexibilidade de programar em qualquer lugar e a qualquer momento.

O primeiro passo para conseguir adquiri-lo será acessar a loja de aplicativos disponível no seu celular e fazer a sua instalação. Após a conclusão da instalação, abra-o e crie uma nova conta ou apenas faça o login com uma já existente. Prontinho, agora você já pode programar no seu celular. Lembrando que o Replit não vai instalar um interpretador Python em seu dispositivo, por isso, para conseguir acesso, você precisará estar conectado à internet.

No computador, por outro lado, a depender do sistema operacional você deverá fazer a instalação do Python. Essa primeira etapa só será necessária caso esteja utilizando o sistema do Windows, para Linux esse processo é desnecessário. Assim, para conseguir programar em Python, você deve acessar o site oficial do Python (<https://www.python.org/>) e fazer o download da versão mais recente compatível. Nesse caso, um exemplo de interpretador muito potente seria o VSCode (<https://code.visualstudio.com/>), uma ferramenta extremamente eficiente para auxiliá-lo.

III - CONCEITOS BÁSICOS

Agora que já sabemos que programação é a ação de escrever um conjunto de instruções para o computador e dizer o que fazer e como fazê-lo e nesse caso vamos usar a linguagem Python para isso, mas o que precisamos entender para conseguir escrever essas instruções?

É necessário compreender alguns termos básicos e os conceitos de programação que são comuns a todas as linguagens de programação. Eles incluem

coisas como dados, algoritmos, variáveis, loops e condicionais... Calma! Não se assuste com os nomes, nós vamos explicar o que é cada um, mas primeiro vamos entender melhor sobre os **conceitos básicos da programação**.

A - Algoritmo e lógica de programação

Uma sequência lógica de afazeres são elementos que precisamos utilizar em todos os lugares, como em casa, no quarto e até mesmo na nossa vida. Com a programação não é diferente, o nosso código precisa seguir uma sequência lógica de representações, até porque uma instrução isolada não permite realizar um processo completo. Para seguir uma sequência lógica é necessário um conjunto de instruções colocadas em ordem sequencial lógica.

E, para conseguir completar uma tarefa, precisamos seguir as instruções em uma ordem adequada. Imagine a tarefa de beber um refrigerante em lata, por exemplo: comprar o refrigerante, abrir a latinha e beber.

Lembre-se, quanto mais detalhadas forem as instruções do algoritmo para o computador, mais rápido e fácil ele compreenderá e irá executá-las chegando, assim, ao objetivo final.

Conforme vimos no exemplo anterior (de beber o refrigerante em lata), para chegarmos a um resultado é necessário uma ordem sequencial lógica. A mesma coisa acontece com a lógica de programação.

Quando pensamos em iniciar uma programação para computador, temos de ter em mente que a máquina desconhece totalmente alguns conceitos que para nós são muito óbvios.

Por isso, devemos descrever cada passo de forma detalhada, por mais simples que possa parecer, para que tenha uma sequência lógica na programação e, assim, o computador possa executar todas as instruções necessárias para uma determinada tarefa.

Pronto! Agora que já sabemos o que é um algoritmo e suas formas de representação, chegou a hora de realizar um exercício! Escreva um algoritmo para comprar um refrigerante no supermercado da esquina, seguindo uma representação descritiva da narrativa.

Faça este algoritmo com uma sequência de, no mínimo, 10 passos.

B - Dados e Tipos de Dados

Com o ambiente de desenvolvimento configurado, seja no celular ou no computador, vamos começar a entender os conceitos básicos de programação com Python, e o primeiro deles é sobre dados e tipos de dados.

Em nossas vidas estamos lidando com dados o tempo todo, seja em casa, na escola ou até mesmo nos jogos, e na programação não será diferente. Vamos pensar em duas categorias de dados inicialmente: textos e números.

Me diga o que você acha que podemos fazer com números. Podemos dividir um número por outro número ? Sim. Podemos somar dois números diferentes ? Sim. Podemos multiplicar 3 ou 4 números ? Com certeza. Podemos somar as notas de uma prova e dividir para conseguir a média ? Sim 🤔. Podemos fazer tudo isso usando números, mas você acha que podemos obter as consoantes de um número ? Isso é meio esquisito, não é ? Só faz sentido fazer isso com textos.

Também podemos fazer diversas coisas com textos. Podemos pegar todas as vogais de um nome. Podemos inverter uma frase. Podemos colocar todas as letras de um nome em maiúsculas ou minúsculas. Mas podemos dividir uma frase por 4 ? Podemos somar uma frase com 3 números ? Isso não faz sentido.

Trabalhar com dados e tipos de dados é avaliar o que podemos fazer com uma determinada informação. Por exemplo, não podemos somar 4 ao texto “Olá galera”, mas podemos somar 4 com outro número para obter uma soma.

Em python os números são representados de duas formas: números inteiros (int) e números reais (float). Com esses dois tipos de dados podemos fazer qualquer

operação com os números, somar, dividir, criar uma fórmula matemática para resolver aquela questão da prova 😬.

Uma observação é que no python números reais não usam vírgula (,) como estamos acostumados (exemplo: 120,45), mas sim um ponto (.) no lugar (exemplo: 120.45).

Também podemos trabalhar com textos em python (chamamos de strings). Com isso podemos fazer qualquer operação relacionada a textos, por exemplo: obter todas as vogais de uma frase, obter todas as consoantes, inverter as palavras de uma frase e etc. As strings em python são representadas por aspas duplas ("eu vou aprender python") ou aspas simples ('eu vou aprender python'). Use o que você achar melhor! 😊

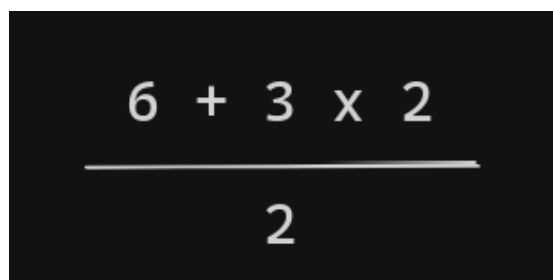
E por fim temos o tipo booleano, que tem somente dois valores: verdadeiro ou falso. É um tipo intuitivo de entender, basta olharmos para uma afirmação e podemos dizer se ela é verdadeira ou falsa. Exemplos:

- 24 é maior que 30 ? Falso
- 5 é menor que 50 ? Verdadeiro
- A frase "oi galera" é igual a frase "ola galera" ? Falso
- 31.5 é igual a 31.5 ? Verdadeiro

Já que falamos bastante sobre números, que tal ver como montamos expressões matemáticas em Python ?

C - Expressões

Lembra que uma das conversas que Mariazinha teve com o ET foi sobre matemática ? Em nossas vidas nós escrevemos expressões matemáticas semelhante a imagem a seguir:


$$\begin{array}{r} 6 + 3 \times 2 \\ \hline 2 \end{array}$$

Porém em Python devemos utilizar símbolos diferentes para realizar uma operação. Na tabela a seguir estão os símbolos usados para diferentes operações.

| | |
|---------------|---|
| Adição | + |
| Subtração | - |
| Multiplicação | * |
| Divisão | / |

Então, seguindo esta tabela a expressão passada para Python ficaria assim:

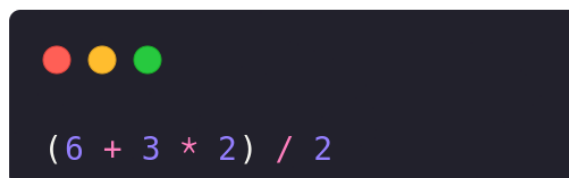


```
6 + 3 * 2 / 2
```

Mas temos um pequeno erro nesta expressão. Na matemática temos a precedência de operadores, isso significa que a multiplicação e a divisão serão feitas antes da soma, o que resultará em 9. Para resolver este problema na matemática podemos usar:

- chaves { }
- colchetes []
- parênteses ()

No entanto, em Python utilizaremos apenas os parênteses. A expressão para Python ficaria assim:



```
(6 + 3 * 2) / 2
```

Agora sim o resultado será **2**, como esperado. Mas quando o robô Python nos fala o resultado estamos fazendo nada com ele, onde poderíamos guardar esse valor para usar depois ?

D - Variáveis

Antes de definirmos o que é uma variável vamos primeiro pensar em um exemplo mais próximo de nós: cozinhar um alimento.

Para uma comida ficar gostosa precisamos colocar temperos nela, e dependendo da combinação de temperos ela pode ficar MUITO gostosa (deu até fome 😋). Mas não podemos simplesmente deixar todos os temperos misturados em um pote, como saberíamos qual tempero escolher? Ficaria impossível.



Panela com alguns temperos misturados
imagem gerada por IA

Para resolver este problema podemos guardar cada tempero em um pequeno pote, e para identificar o tempero do pote podemos colocar uma fita informando o nome do tempero. Fica bem mais fácil identificar os temperos agora, não é mesmo? Mas é importante se atentar ao nome de cada pote, eles devem ser significativos. Imagine que você guardou 3 temperos diferentes com a coloração branca, se você der os nomes A, B e C fica muito difícil saber a diferença de cada tempero.

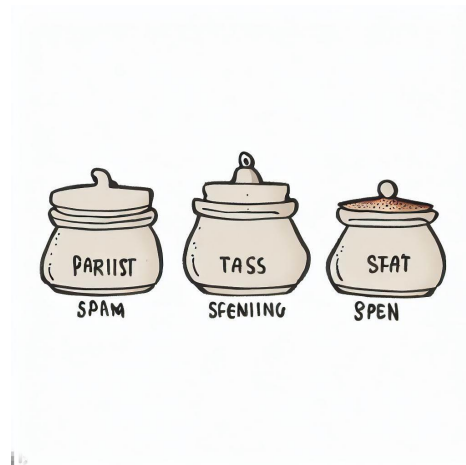
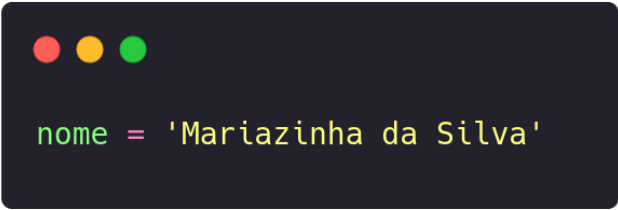


Imagem de 3 potes com diferentes temperos

imagem gerada por IA

Com este exemplo simples fica fácil de entender o que é uma variável. Uma variável em um local para armazenarmos um dado. Precisamos também dar um nome para variável que estamos criando, por exemplo: se eu criar uma variável para guardar o meu nome, faz sentido o nome desta variável ser "nome". Mas como criamos variáveis em Python ? Veja no exemplo abaixo.



```
nome = 'Mariazinha da Silva'
```

Para criarmos uma variável em Python basta escolher o nome da variável. Em seguida usamos o carácter de igual (=) para atribuir um valor para a variável.

Importante!

Apenas um "=" significa a atribuição de um valor para a variável.
Dois "=" (==) representam o símbolo de igual

Mas preste bastante atenção, ao criar uma variável devemos ficar atentos a algumas regras que seus nomes devem seguir, aqui está uma lista das regras:

- Não podemos usar espaço para separar nome de variáveis, exemplo:
 - meu nome = 'Mariazinha'
- Para separar palavras você pode utilizar um underline (_) ou separar as palavras com letras maiúscula:
 - meu_nome = 'Mariazinha'
 - meuNome = 'Mariazinha'
- Não podemos iniciar nome de variáveis com números
- Não podemos utilizar acentuação
- Não devemos usar palavras que o já Python (chamamos de "palavras reservadas"), como int, bool, float

Também preste atenção ao nome que você estará dando a sua variável. Imagine que nós vamos criar uma expressão para calcular nossa média, faz sentido

colocar o nome da variável de **gabigol**? Acho que não (apesar de ser um baita jogador 🤔).

Já que falamos em média, o que acha de criarmos esta expressão e armazenar o valor da média em uma variável ? Vamos lá, observe o código abaixo:

```
media = (10 + 10 + 8) / 3
```

Que notão hein? Mandou bem. Mas podemos melhorar este código ainda mais, você tem alguma sugestão de melhoria?

Vou falar uma sugestão: podemos armazenar cada uma das notas em uma variável separada, e utilizar essas variáveis na expressão 🤖.

```
nota1 = 10
nota2 = 10
nota3 = 8

media = (nota1 + nota2 + nota3) / 3
```

Agora ficou bem legal, assim conseguimos mudar os valores das notas e elas vão impactar automaticamente no resultado da média.

Mas toda vez que quiséssemos calcular nossa média teríamos que ficar alterando os valores de cada variável, meio chato isso né ? Como podemos resolver este problema?

E - Entrada e Saída

Nós montamos uma expressão matemática para calcular uma média, mas não estamos fazendo nada com esse resultado. Na história que contamos anteriormente, Mariazinha mandava as instruções para o robô através de um papel, e após isto o robô devolvia uma resposta para ela.

Podemos pensar nessa mesma ideia para melhorar nosso código anterior. Vamos pedir como **entrada** as notas para calcular a média, e depois de calcular vamos mostrar uma **saída** para o usuário, informando o valor da média.

Podemos pensar em saída como uma resposta, semelhante a que o robô faz quando traduz o “ETnês” e envia um papel para Mariazinha. Para exibir mensagens para o usuário em Python (ou seja, exibir uma saída), vamos utilizar a função chamada **print**.

Vamos olhar um pequeno exemplo de código para ficar mais fácil de entender:

A terminal window with a dark background and three colored window control buttons (red, yellow, green) in the top left corner. The text `print('Olá galera!')` is displayed in a light-colored monospace font.

```
print('Olá galera!')
```

Com o `print` podemos exibir uma informação para o usuário, seja um número, uma frase ou o que você desejar.

Agora que sabemos usar o **print** vamos modificar nosso código anterior para o robô Python ser capaz de mostrar o resultado da média para Mariazinha.

```
nota1 = 10
nota2 = 10
nota3 = 8

media = (nota1 + nota2 + nota3) / 3
print(media)
```

Agora sim, estamos exibindo o valor calculado! Mas perceba que poderíamos exibir esse valor de uma forma melhor de entender, o que acha? Poderíamos colocar uma frase, exemplo: "O valor da média é X". Fica bem melhor de entender o que é esse resultado não é mesmo ? 😊

Vamos olhar como fazemos isso em Python:

```
nota1 = 10
nota2 = 10
nota3 = 8

media = (nota1 + nota2 + nota3) / 3
print('O resultado da média é:', media)
```

Perceba que para exibir de uma forma mais legível colocamos uma frase e o valor da variável separados por uma vírgula. Dessa forma o resultado vai ser: "O resultado da média é: 9.333333".

Agora imagine que você quer exibir junto do resultado da média as notas do aluno. O código ficaria um pouco confuso:


```

nota1 = 10
nota2 = 10
nota3 = 8

media = (nota1 + nota2 + nota3) / 3
print('(', nota1, '+', nota2, '+', nota3, ') / 3:', media)

```

Não sei você, mas eu já fiquei perdidinho com tanta vírgula 😓. Mas Python tem uma forma mais simples de montarmos essa grande string de saída, vamos dar uma olhada!

```

nota1 = 10
nota2 = 10
nota3 = 8

media = (nota1 + nota2 + nota3) / 3
print(f'({nota1} + {nota2} + {nota3}) / 3: {media}')

```

Bem mais simples não é ? Estamos formatando uma string, com isso podemos adicionar variáveis dentro de chaves dentro da string e o resultado delas será exibido. Note que para usar essa formatação precisamos colocar a letra **f** antes de começar a string.

Quando executamos o código acima conseguimos ver o resultado na tela 😎. Mas e se quisermos informar o valor de cada nota ? Como faríamos isso ?

Devemos então informar uma **entrada** para o Python. Para isso vamos utilizar a função **input** para que o usuário informe um dado. Olha só este exemplo simples:

```
nome = input('Informe seu nome: ')
print(f'Seu nome é: {nome}')
```

Estamos usando a função **input** para obter uma entrada de dados. Podemos informar uma mensagem descritiva para ajudar o usuário a entender o que precisa ser informado. Quando o usuário digitar o valor e der enter, o dado informado será armazenado na variável **nome**. Um detalhe importante é que o **input** sempre vai retornar o dado no formato **string**, mesmo se você informar um número ele será uma string. Em seguida estamos exibindo o que está armazenado em **nome**, olha que maneiro 😊.

Agora que sabemos como pedir entradas de dados para o usuário, Mariazinha vai ser capaz de informar os valores das 3 notas para o robô Python calcular! Vamos olhar como ficará o código agora:

```
nota1 = input('Informe a 1º nota: ')
nota2 = input('Informe a 2º nota: ')
nota3 = input('Informe a 3º nota: ')

media = (nota1 + nota2 + nota3) / 3
print(f'({nota1} + {nota2} + {nota3}) / 3: {media}')
```

Nosso programa está ficando cada vez mais completo, agora conseguimos pedir as 3 notas para calcular o valor da média ! Mas temos um problema, ao executar o código ele deu um terrível erro 🤔. Por que isto aconteceu? Qual problema você imagina que tenha ocorrido? Me responde aí.

Lembre-se que a função **input** retorna os valores em formato de **strings**, e como vimos anteriormente, não podemos fazer uma operação de divisão em uma string. O que devemos fazer então ? Devemos realizar uma **conversão de tipos**.

F - Conversão de Tipos

Uma analogia que pode nos ajudar a entender melhor as conversões de tipos é: fazer gelo. Fazer gelo nada mais é do que pegarmos a água e resfriá-la até ficar sólida. Fizemos então uma espécie de “conversão de tipo”, onde trocamos o “tipo” (o estado) da água de líquido para sólido.

Agora que entendemos o que é a conversão de tipos, vamos entender como usamos este recurso para corrigir nosso código anterior. Lembre que a função **input** sempre devolverá os dados no formato **string**, mas não podemos realizar cálculos com strings, devemos então converter o dado que recebemos para número.

Python tem duas funções que realizam uma conversão para números: **int** para números inteiros e **float** para números com casas decimais. Como uma nota pode ter valores como 9.5, 8.7, 4.2 então iremos utilizar a função **float**.

```
notas1 = float(input('Informe a 1ª nota: '))
notas2 = float(input('Informe a 2ª nota: '))
notas3 = float(input('Informe a 3ª nota: '))

media = (notas1 + notas2 + notas3) / 3

print(f'({notas1} + {notas2} + {notas3}) / 3: {media}')
```

Observe que usamos a função **float** por volta de todo o **input**, basicamente o que está acontecendo é: quando a função **input** é chamada e ela retorna o dado que o usuário enviou esse dado é passado para a função **float**, que fará a conversão da string e número.

Mas atenção, nem sempre essa conversão irá funcionar. Não é possível converter um nome ou uma frase em números, se tentarmos fazer isso receberemos um erro de Python dizendo que não é possível fazer essa operação.

Agora sim, quando executamos o código podemos observar a nossa média aritmética correta sendo exibida 🧐. Podemos deixar este exemplo ainda mais completo, onde vamos exibir se o aluno foi aprovado ou não de acordo com sua média. Para isso vamos entrar na próxima sessão onde vamos tomar **decisões**.

IV - CONDIÇÕES - ESTRUTURAS DE DECISÃO

Em nossas vidas constantemente estamos tomando decisões. Mariazinha, por exemplo, para ajudar no conserto da nave do ET teve que escolher quais eram os parafusos corretos para usar, as ferramentas necessárias e até onde cada fio podia se ligar.

A tomada de decisão num código permite que o programa tome decisões com base em certas condições. Em outras palavras, o programador pode fazer uma pergunta e, dependendo da resposta, tomar diferentes caminhos, exatamente igual Mariazinha, já que ela precisa observar, comparar e decidir se o parafuso será útil ou não para o conserto da nave de seu amigo ET.

Agora que você entendeu, que tal treinar um pouco o que são as decisões no nosso dia a dia. Imagine que você é o gerente de uma loja e deseja criar um programa de descontos a ser dado aos clientes com base no valor total de suas compras.

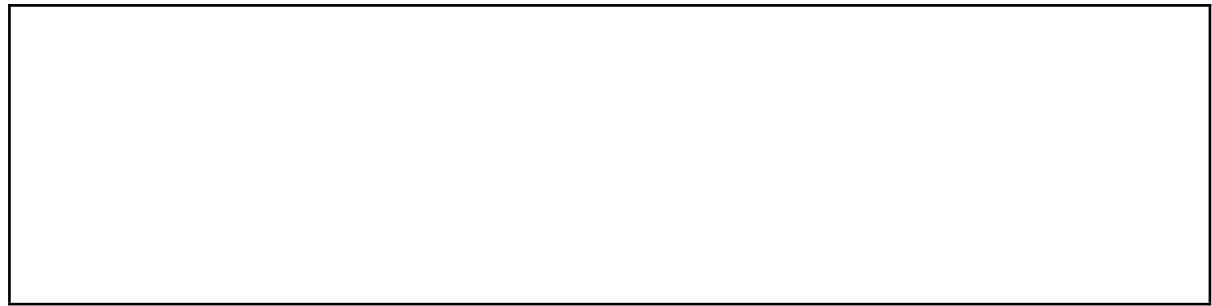


15

*Supermercado
imagem gerada por IA*

Os clientes que comprarem R\$500 ou mais, ganham 10% de desconto. Com base nessa regra, crie um exercício de decisão em que você possa determinar o desconto a ser dado a um cliente com base no valor total de suas compras.

Utilize a estrutura “se” para essa atividade. Por exemplo, **se** estiver ensolarado lá fora, você decide sair para aproveitar o dia e fazer atividades ao ar livre.



Veja o quadro abaixo, com um exemplo do dia a dia, para entender melhor sobre estruturas de decisão:

Imagine que você está decidindo se vai sair de casa ou não, com base nas condições climáticas. As estruturas de decisão podem te ajudar nessa escolha.

- **Se** estiver ensolarado lá fora, você decide sair para aproveitar o dia e fazer atividades ao ar livre.
- **Se** estiver chovendo, você decide ficar em casa e assistir a um filme ou fazer algo que você goste dentro de casa.

Nesse exemplo, você está usando as condições climáticas para tomar decisões sobre o que fazer. Essa é a ideia por trás das estruturas de decisão em programação: você avalia uma condição e toma uma ação com base nessa avaliação.

Importante

Agora que vamos aprender sobre estruturas de decisão na prática é importante dizer que Python usa indentação para marcação de blocos; *Indentação são blocos de códigos que recebem um recuo para definir a hierarquia em que o código será executado. O recuo é como se fosse um espaço que você dá e faz um alinhamento do código dentro de outro.*

Vamos entrar a fundo agora em como podemos tomar decisões em Python, a primeira forma que veremos é usando o **if**.

A - If

Em Python, usamos o "if" para tomar decisões. O "if" é semelhante ao "se" em português. Podemos entendê-lo como: se a condição for verdadeira, faça algo ou execute a instrução seguinte.

A sintaxe básica do "if" em Python é a seguinte:

```
if condição:
    # código a ser executado se a condição for verdadeira
```

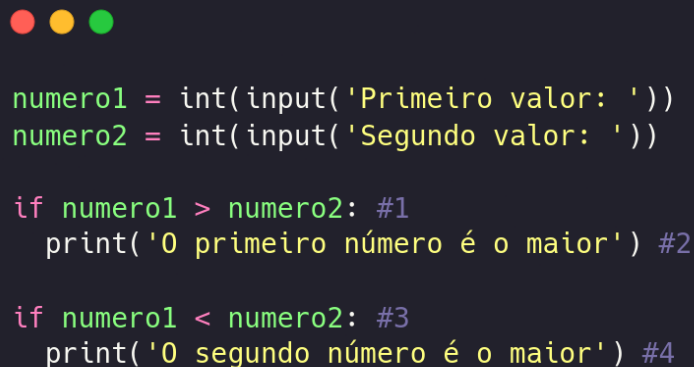
O "if" avalia a expressão "condição" e, se ela for avaliada como verdadeira (True), o bloco de código indentado logo abaixo dele será executado. Caso a condição seja falsa (False), o bloco será ignorado, e a execução do programa continuará com as instruções seguintes ao bloco do "if".

Vamos testar algumas condições para entender melhor como "if" funciona:

```
numero1 = int(input('Primeiro valor: '))
numero2 = int(input('Segundo valor: '))

if numero1 > : #1
    print('O primeiro número é o maior') #2
```

Note que no ponto #1 a condição que montamos no if, primeira variável que estamos comparando, um **operador lógico** (um operador que vai comparar duas informações e dizer se é verdadeiro ou falso) que é o **maior que** (>) e por fim a segunda variável. Assim, temos a condição numero1 > numero2 (a maior que b). Essa expressão será avaliada e, se for verdadeira, a linha #2 será executada. Caso seja falsa, a linha #2 será ignorada. Mas agora se quisermos verificar se o numero2 é o maior ? Só adicionar outro if.



```

numero1 = int(input('Primeiro valor: '))
numero2 = int(input('Segundo valor: '))

if numero1 > numero2: #1
    print('O primeiro número é o maior') #2

if numero1 < numero2: #3
    print('O segundo número é o maior') #4

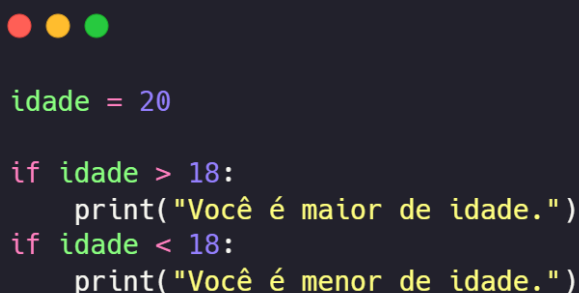
```

O mesmo comportamento ocorre com a condição `numero1 < numero2` (agora usamos o operador lógico **menor que**) na linha **#3**. Se essa condição for verdadeira, a linha **#4** será executada; caso contrário, será ignorada.

Quando o primeiro valor é maior do que o segundo, as seguintes linhas são executadas: **#1, #2, #3**. Quando o primeiro valor é menor do que o segundo, temos uma sequência diferente: **#1, #3, #4**. É importante ressaltar que a linha com a condição em si é executada mesmo se o resultado da expressão for falso.

Vamos praticar o que aprendemos?

Dado o exemplo abaixo marque a alternativa correta:



```

idade = 20

if idade > 18:
    print("Você é maior de idade.")
if idade < 18:
    print("Você é menor de idade.")

```

O que será impresso na tela?

- ☐ Você é menor de idade.
- ☐ Você é maior de idade.
- ☐ Idade > que 18:
- ☐ Nada

Desta forma podemos concluir que, o "if" é uma ferramenta importante na programação que nos permite tomar decisões com base em condições. Com ele, podemos executar diferentes partes do código dependendo da situação. Isso nos ajuda a criar programas flexíveis e eficientes, adaptados às necessidades específicas. O "if" nos dá o poder de controlar o fluxo do programa e fazer com que ele tome ações específicas quando certas condições são atendidas.

B - Else

Em Python, além do "if", temos a poderosa estrutura condicional "**else**" (senão), que trabalha em conjunto com o "if" para proporcionar múltiplas possibilidades de decisões no fluxo do programa. O "else" permite especificar um bloco de código a ser executado quando a condição do "if" é falsa. Pensando na história do ET, Mariazinha verificava se o parafuso que ela estava analisando era o ideal, senão ela podia guardar em uma caixa para usá-los em outras ocasiões.

A sintaxe básica do "else" é a seguinte:

```
if condição:
    # código a ser executado se a condição for verdadeira
else:
    # código a ser executado se a condição for falsa
```

O "else" oferece uma alternativa quando a condição do "if" é avaliada como falsa (False). Isso significa que se a expressão do "if" não for satisfeita, o bloco de código indentado sob o "else" será executado, garantindo que o programa tenha uma ação pré-determinada caso a condição inicial não seja atendida.

Vejamos um exemplo simples já usado anteriormente:

```
idade = 20

if idade >= 18: # if = se
    print("Você é maior de idade.")

else: #else = senão
    print("Você é menor de idade.")
```

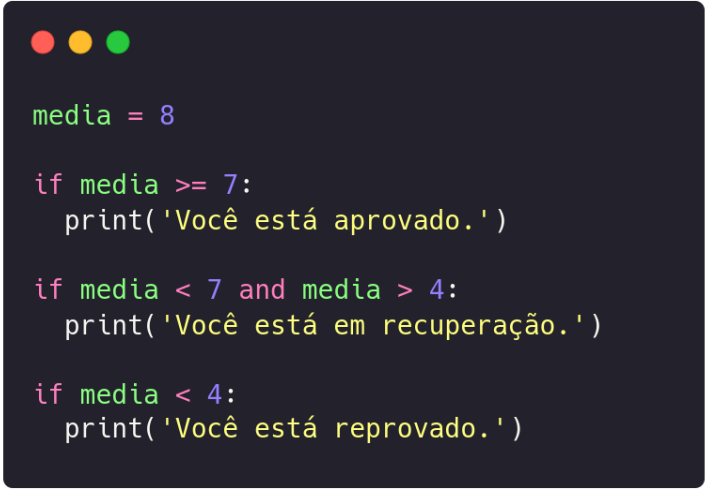
Neste caso, se a variável "idade" for maior ou igual a 18, o bloco de código associado ao "if" será executado, exibindo a mensagem "Você é maior de idade.". No entanto, caso a condição do "if" não seja verdadeira (ou seja, a idade seja menor que 18), o bloco do "else" será executado, mostrando a mensagem "Você é menor de idade.".

C - Elif

Nem sempre temos dois caminhos para tomar decisões. De acordo com uma situação podemos tomar vários caminhos, por exemplo, de acordo com o resultado do cálculo da média você pode ser aprovado, ficar em recuperação ou reprovado. No exemplo dos parafusos de Mariazinha, ela pode usar o parafuso, guardar na caixa de ferramentas ou ser descartado.

Na programação não é diferente, podemos ter vários caminhos para seguir de acordo com uma decisão. Vamos usar o exemplo da média escolar, onde você pode ser aprovado se a nota for maior ou igual a 7, recuperação se a nota foi abaixo de 7 mas foi acima de 4, e se for menor que 4 está reprovado. Como faria isso em código ?

Naturalmente podemos pensar em usar vários **if** para fazer cada uma dessas verificações, então o código pode ficar desse jeito:

A dark-themed code editor window with three colored window control buttons (red, yellow, green) in the top-left corner. The code is written in Python and uses syntax highlighting: keywords like 'if' and 'print' are in purple, variables like 'media' are in green, and string literals are in yellow. The code defines a variable 'media' with the value 8 and then uses three 'if' statements to check the student's status based on their average score.

```
media = 8

if media >= 7:
    print('Você está aprovado.')

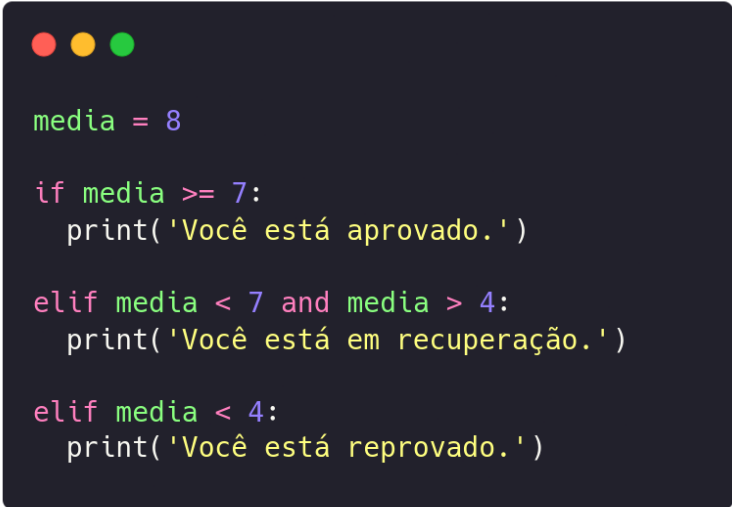
if media < 7 and media > 4:
    print('Você está em recuperação.')

if media < 4:
    print('Você está reprovado.')
```

Cada **if** fará a verificação para saber a situação do aluno, então podemos ter 3 caminhos diferentes. Não se assuste com o segundo **if**, nele estamos usando um operador lógico chamado **and** (também temos o **or** e **not**). Seu entendimento é simples, ele irá comparar duas coisas, se ambas forem verdadeiras o resultado será verdadeiro. Imaginando que a média atual é 5, ela é menor que 7 e também maior que 4, então o aluno está em recuperação. Agora se a média for menor que 4 o aluno está reprovado.

Mas perceba um detalhe, mesmo que o aluno tenha a nota 8 e ele esteja aprovado todas as verificações serão feitas mesmo assim. Para esses casos devemos usar a instrução **elif**.

Podemos ler o **elif** como um **senão se**. Exemplo: se a nota for maior que 7 você está aprovado, senão se a média for menor que 7 e maior que 4 você está em recuperação. Modificando o código ele pode ficar da seguinte forma:



```
media = 8

if media >= 7:
    print('Você está aprovado.')

elif media < 7 and media > 4:
    print('Você está em recuperação.')

elif media < 4:
    print('Você está reprovado.')
```

Agora sim o código tá bem legal. Mas note que esse código só calcula e avalia a média de um aluno por vez, todas as vezes que quisermos avaliar a nota novamente temos que executar o código e alterar o valor da média ou pedir um valor para o usuário como vimos antes. O que podemos fazer para executar esse mesmo código múltiplas vezes com diferentes valores de média ? É o que veremos no próximo capítulo falando sobre **repetições**.

V - ESTRUTURAS DE REPETIÇÃO

Chegamos na última sessão da nossa apostila onde vamos falar sobre estruturas de repetição. O conceito de repetição já estamos familiarizados, é basicamente fazer novamente uma ação. Em nossa história do ET, Mariazinha teve que parafusar diversos parafusos para consertar a nave, ela repetiu diversas vezes o processo de pegar o parafuso, analisar se era o ideal e se sim, parafusar corretamente. Na programação em certas ocasiões também precisamos de repetições em certos pedaços de código.

No Python temos duas estruturas de repetições, são elas: **for** e **while**. Ambas podem ser usadas para repetir trechos de código, mas cada uma tem sua particularidade, e primeiramente veremos a estrutura **for**.

A - For

A estrutura **for** é muito útil quando sabemos exatamente o número de vezes que precisamos repetir algo. Por exemplo, se soubermos o número de alunos que

têm uma sala de aula podemos criar um código para obter nome e notas de todos esses alunos. Vamos para um exemplo:

```
qtd_alunos = 35

for i in range(qtd_alunos):
    nome = input('Informe seu nome: ')
    print(f'Aluno: {nome}')
```

Basicamente o que este código faz é obter somente o nome de todos os alunos da uma sala e exibi-los, simples. Veja que usamos a estrutura **for** pois sabemos exatamente o número de alunos da sala. Talvez você esteja um pouco confuso com este código, mas vamos analisá-lo com calma.

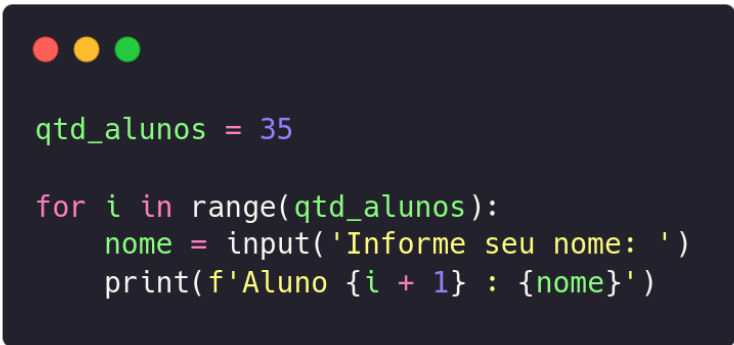
Temos nossa estrutura **for** para executar um trecho de código X vezes. Informaremos essa quantidade de vezes para a função **range**, neste exemplo, o trecho de código será executado 35 vezes. Aquela variável **i** que vem depois do **for** é uma variável de controle que o Python utiliza para não se perder no meio das repetições. Você também pode utilizar essa variável para saber em qual repetição você está, se é a primeira, segunda, terceira e assim por diante. Vamos ver um exemplo:

```
qtd_alunos = 35

for i in range(qtd_alunos):
    nome = input('Informe seu nome: ')
    print(f'Aluno {i} : {nome}')
```

Com isso podemos saber em qual repetição estamos, sendo exibido algo como “Aluno 1...Aluno 2...Aluno 3” e assim por diante. Mas note que ao executar o código o valor de *i* não começou com 1, mas sim com 0.

Isso acontece pois a função **range** gera uma sequência de números de acordo com o número que passamos para ela, mas ela não gera de 1 até 35. Em vez disso, ela começa a partir do 0 e vai até 34, que também dá 35 repetições, então fique bem atento a esse detalhe. Mas para não mostrar os alunos de 0 até 34 basta fazermos uma soma simples no momento de exibir o aluno:



```
qtd_alunos = 35

for i in range(qtd_alunos):
    nome = input('Informe seu nome: ')
    print(f'Aluno {i + 1} : {nome}')
```

Agora sim temos um exemplo top top. Então quando você souber quantas vezes quer repetir uma ação, basta usar a estrutura **for** para isso. Mas e quando não sabemos quantas vezes precisaremos repetir algo ? Por exemplo, imagine que você quer saber a média de um bairro inteiro, isso é muita gente, não é mesmo ?

Para facilitar a vida você pode continuar pedindo a média de cada pessoa enquanto ela for maior do que zero. Ou seja, você vai repetir um trecho de código de acordo com uma condição, e para isso você vai precisar do **while**.

B - While

O comando **while** faz com que um conjunto de instruções seja executado enquanto uma condição é verdadeira. Quando o resultado dessa condição passa a ser falso, a execução do loop é interrompida. Vamos a um exemplo:

Imagine que você esteja em um ônibus e sua parada seja a que se chama Naruto. O ônibus vai passando por cada uma das paradas e você vai comparando. É a parada Naruto? Se for, você desce do ônibus. Se não for, você permanece no ônibus. E vai comparando parada por parada até chegar na parada naruto.

```
parada_atual = "" # Inicializamos a parada atual como uma string vazia

# Loop enquanto a parada atual for diferente de "Naruto"
while parada_atual != "Naruto":
    parada_atual = input("Próxima parada: ") # Solicitamos a entrada da
    próxima parada
    if parada_atual == "Naruto":
        print("Chegamos à parada Naruto! Você pode descer do ônibus.")
    else:
        print("Ainda não chegamos à parada Naruto. Continue no ônibus.")

print("Você desceu do ônibus na parada Naruto.")
```

Neste exemplo, o loop **while** continua executando até que a entrada do usuário coincida com a string "Naruto". Enquanto a entrada for diferente de "Naruto", o programa solicitará a próxima parada e informará se ainda não é a parada desejada. Assim que o usuário inserir "Naruto", o loop é interrompido e uma mensagem é exibida indicando que chegou à parada correta.

Lembre-se de que o uso de `input()` permite que o usuário insira dados no programa durante a execução, e o programa compara essa entrada com a condição definida no loop `while`.

O comando `while` é uma ferramenta poderosa em programação que permite a execução repetida de um conjunto de instruções enquanto uma determinada condição é verdadeira. Quando essa condição se torna falsa, o loop é interrompido, e o programa continua sua execução normal.

A HISTÓRIA NA QUAL VAMOS NOS BASEAR:

Há muito tempo, em uma galáxia distante, um ET chamado Python estava explorando o espaço sideral em sua nave espacial. Infelizmente, devido a um mau funcionamento em seu motor, ele acabou perdendo o controle e caiu acidentalmente na Terra.

Desorientado e com sua nave danificada, Python sentia-se perdido em um planeta desconhecido. Foi então que ele encontrou Joãozinho, um garoto inteligente e curioso, que vivia em uma pequena cidade próxima ao local do acidente. Joãozinho era conhecido por sua habilidade em lidar com tecnologia e sua mente inventiva.

Ao se deparar com o extraterrestre, Joãozinho ficou surpreso e fascinado. Rapidamente, ele ofereceu sua ajuda para consertar a nave e ajudar Python a voltar para casa. O ET, emocionado com a generosidade do garoto, aceitou a oferta e juntos começaram a trabalhar incansavelmente para restaurar a nave espacial.

Enquanto Python e Joãozinho se dedicavam ao conserto da nave, um homem mal-intencionado chamado Sr. Malvado, dono de um circo decadente, ficou sabendo da existência do ET e viu nele uma oportunidade de ganhar muito dinheiro. Sr. Malvado era conhecido por sua natureza gananciosa e sem escrúpulos, e estava determinado a sequestrar Python para usá-lo como atração principal de seu circo.

Quando o vilão descobriu o local onde Python e Joãozinho estavam trabalhando, ele enviou seus capangas para capturá-los. No entanto, a dupla esperta e corajosa conseguiu escapar por pouco. Agora eles precisavam se apressar para terminar o conserto da nave antes que Sr. Malvado os encontrasse novamente.

Com o tempo correndo contra eles, Python e Joãozinho redobram seus esforços. O garoto utilizou seus conhecimentos em eletrônica e robótica para consertar os circuitos danificados, enquanto Python compartilhava seu conhecimento avançado em tecnologia alienígena para resolver problemas complexos. Juntos, eles formavam uma equipe imbatível.

Enquanto trabalhavam, eles também descobriram uma conexão especial e forjaram uma amizade única. Compartilhando histórias sobre seus mundos e trocando experiências, eles perceberam que a amizade e a colaboração eram poderosas ferramentas para superar qualquer obstáculo.

Finalmente, após muitas noites de trabalho árduo, a nave de Python estava completamente restaurada. Chegou o momento de se despedir de Joãozinho e voltar para casa. Mas antes que pudessem partir, Sr. Malvado reapareceu com seus capangas, determinado a capturar Python.

Com uma combinação de astúcia, engenhosidade e coragem, Python e Joãozinho enfrentaram os capangas de Sr. Malvado. Usando truques e estratégias inteligentes, eles conseguiram neutralizar os inimigos um por um. Com o caminho livre, eles correram para a nave e entraram rapidamente, deixando o vilão enfurecido e impotente para detê-los.

Enquanto a nave decolava, Joãozinho e Python olhavam para baixo, vendo a Terra se afastar cada vez mais. O garoto sentiu-se triste em se despedir do seu novo amigo, mas sabia que tinha feito a coisa certa ao ajudá-lo. Eles prometeram manter contato e sempre se lembrar dos momentos incríveis que compartilharam.

De volta ao seu planeta natal, Python reuniu-se com sua família e amigos, contando as histórias emocionantes sobre a Terra e sobre o garoto corajoso chamado Joãozinho. Eles celebraram sua chegada e agradeceram a amizade que ele havia encontrado em um planeta distante.

Enquanto isso, Joãozinho voltou para casa com um sorriso no rosto, sabendo que tinha feito a diferença na vida de Python. Ele continuou a explorar o mundo da tecnologia e da ciência, inspirado pelas aventuras que viveu ao lado do ET.

E assim, a história de Python e Joãozinho tornou-se uma lenda, uma prova de que a amizade e a determinação podem superar qualquer obstáculo, mesmo quando se trata de um ET perdido em um planeta estranho.