



7 H 37ec7
2 LC18c7
7' wX
2 H 37ec7
7 LC18c7
H wX
3 wX23c7
7 wX
7 H 37ec7
7 LC18c7
7 wX
7 H 37ec7
7 LC18c7
7 wX
7 H 37ec7
7 LC18c7
7 wX
7

7
(S000)
713
(S000)
(S000)
7

O CÓDIGO DO MAPA DO MAROTO

DESBRAVANDO OS CAMINHOS DA PROGRAMAÇÃO

PRISCILA NAYADE

01

Vamos pelo começo...

02

Poções digitais: variáveis e operadores como ingredientes mágicos

03

Feitiços de controles: estruturas de decisão e repetição na ponta da varinha

04

Alquimia digital: funções e modularidade

05

Parabéns, jovem bruxo!

ÍNDICE

01

**“JURO
SOLENEAMENTE
DESVENDAR
ESSES
CÓDIGOS.”**

VAMOS PELO COMEÇO...

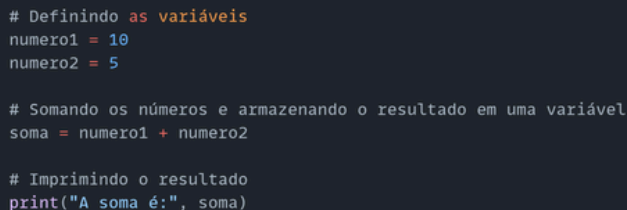
Bem-vindo ao mundo encantado da programação, em que a lógica é a nossa varinha mágica e os códigos são os feitiços que criamos para dar vida às nossas ideias! Neste primeiro capítulo, vamos mergulhar nas águas cristalinas da lógica de programação, desvendando seus mistérios e preparando o terreno para aventuras ainda mais fascinantes.

Por essência, a lógica de programação é como a *poção polissuco* da magia digital: ela nos permite transformar problemas complexos em sequências lógicas simples e compreensíveis. Assim como um mago precisa compreender os fundamentos dos encantamentos para lançar feitiços poderosos, um programador deve dominar os princípios básicos da lógica para escrever códigos eficientes.



Vamos começar com um exemplo simples. Imagine que você é um jovem bruxo aprendendo a fazer poções. Para criar uma poção de cura, você precisa seguir uma receita específica, misturando os ingredientes na ordem correta. Da mesma forma, na programação, temos instruções precisas, chamadas de **algoritmos**, que nos guiam na resolução de problemas.

Vejamos um algoritmo básico em Python para somar dois números (existem várias linguagens, porém usaremos essa para exemplificar melhor, ok?):

A dark-themed terminal window with three colored window control buttons (red, yellow, green) in the top-left corner. The window contains Python code for variable assignment and arithmetic. The background of the entire image is a horizontal gradient from teal on the left to yellow on the right.

```
# Definindo as variáveis
numero1 = 10
numero2 = 5

# Somando os números e armazenando o resultado em uma variável
soma = numero1 + numero2

# Imprimindo o resultado
print("A soma é:", soma)
```

snappify.com

Neste código simples, nós **declaramos** duas variáveis, **numero1** e **numero2**, damos a elas os valores 10 e 5, respectivamente, para aí então realizarmos a operação de soma. Por fim, exibimos o resultado na tela.

É como seguir os passos de uma receita de poção, não é? Primeiro, reunimos os ingredientes (**variáveis**), depois combinamos eles e misturamos no caldeirão para obter o resultado desejado.

A lógica de programação é a base de tudo que faremos neste mundo mágico da computação. Ela nos ensina a **pensar de forma estruturada**, a **decompor problemas** complexos em partes menores e a **encontrar soluções eficientes**. Assim como um bruxo que domina seus feitiços mais básicos, um programador que compreende os fundamentos da lógica estará preparado para enfrentar desafios cada vez maiores.

Prepare-se, jovem aprendiz, pois esta é apenas o primeiro capítulo de uma jornada repleta de descobertas e desafios emocionantes. Nos próximos passos, iremos aprofundar nossos conhecimentos e desvendar vários segredos da programação. Até lá, pratique seus encantamentos básicos e esteja pronto para os desafios que virão pela frente!

02

“UMA POÇÃO
PREPARADA
CORRETAMENTE
TRARÁ UM
CÓDIGO SEM
BUG”

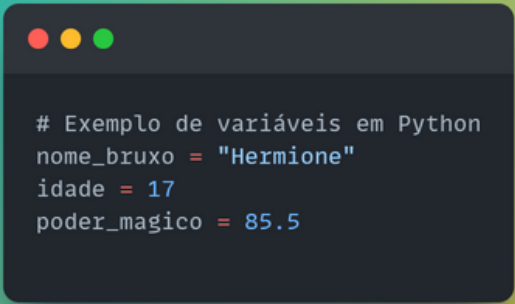
POÇÕES DIGITAIS

Variáveis e operadores como
ingredientes mágicos

E aí, jovem bruxo! Neste capítulo, vamos mergulhar no mundo mágico das variáveis e operadores em Python, explorando como eles são os ingredientes fundamentais para criar feitiços poderosos e manipular dados em nossos programas. Prepare-se para dominar os mistérios das poções digitais e lançar seus próprios encantamentos!

Mas atenção: estamos usando o Python como linguagem apenas para exemplificar. Tenha em mente que existem várias outras que executam as mesmas tarefas, porém precisam de ingredientes e comandos diferentes!

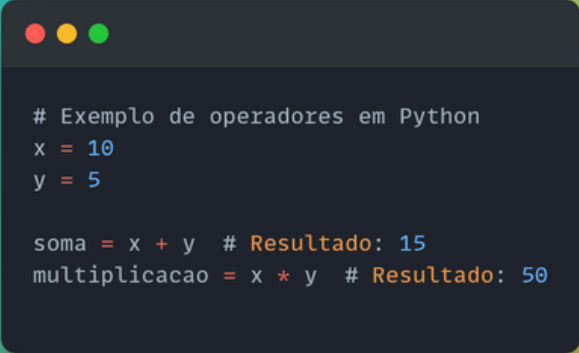
No universo da programação, as **variáveis** são como os ingredientes mágicos que guardam valores importantes. Elas nos permitem armazenar e manipular informações, como números, textos e valores booleanos (valores que denominamos como "True" ou "False" em nosso código. Assim como em uma poção de cura que guarda os elementos essenciais para restaurar a saúde de um bruxo, as variáveis nos permitem guardar e acessar dados importantes em nossos programas.



```
# Exemplo de variáveis em Python
nome_bruxo = "Hermione"
idade = 17
poder_magico = 85.5
```

Como você pode ver no exemplo, as variáveis são esses nomes que antecedem o sinal de =. Isso indica para o programa que, a partir daquele momento, o termo que você escolheu terá aquele valor que está após desse símbolo. No caso, "nome_bruxo" terá o valor de "Hermione", porque você está determinando isso para o computador. É claro que isso vale para todos aqueles atributos que comentamos no início desse capítulo.

Agora, os **operadores** em Python são como os feitiços que lançamos para manipular os valores das nossas variáveis. Eles nos permitem realizar operações matemáticas, comparações e outras manipulações de dados. Da mesma forma que um feitiço de transfiguração transforma um objeto em outro, os operadores podem modificar valores e realizar diversas ações em nossos programas. No mundo da programação, os nossos operadores realizam adição (+), subtração (-), multiplicação (*) e divisão (/).



```
# Exemplo de operadores em Python
x = 10
y = 5

soma = x + y # Resultado: 15
multiplicacao = x * y # Resultado: 50
```

03

“SÃO NOSSAS
ESTRUTURAS
DE DECISÃO E
LAÇOS DE
REPETIÇÃO QUE
REVELAM UM CÓDIGO
QUE FUNCIONA”

FEITIÇOS DE CONTROLE

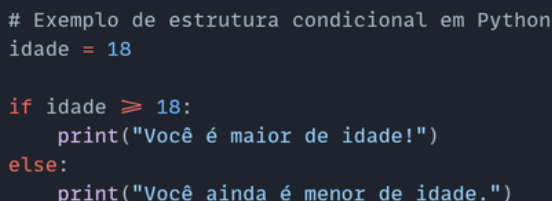
Estruturas de decisão e
de repetição na ponta da varinha

Neste capítulo, vamos explorar os feitiços de controle e repetição em Python, que nos permitem tomar decisões e repetir ações em nossos programas. Prepare-se para desvendar os segredos das estruturas condicionais e laços de repetição, e começar a ser um verdadeiro Auror de bugs no código!

As estruturas de decisão são como bifurcações em um caminho, onde o código pode seguir diferentes direções dependendo de certas condições. Imagine que você é um bruxo enfrentando uma encruzilhada e precisa decidir qual feitiço lançar com base em determinadas situações.

No código, usamos estruturas de decisão para tomar decisões semelhantes. Se uma condição específica for verdadeira, o código seguirá um caminho; caso contrário, seguirá outro. É como se lançássemos um feitiço de proteção se detectarmos perigo ou um feitiço de cura se encontrarmos um amigo ferido.

Em resumo, as **estruturas de decisão** nos permitem controlar o fluxo do nosso programa, tornando-o mais dinâmico e capaz de lidar com diferentes situações. Elas são essenciais para escrever código que possa tomar decisões inteligentes e adaptativas, assim como um bruxo habilidoso que escolhe o feitiço certo para cada desafio que enfrenta.



```
# Exemplo de estrutura condicional em Python
idade = 18

if idade >= 18:
    print("Você é maior de idade!")
else:
    print("Você ainda é menor de idade.")
```

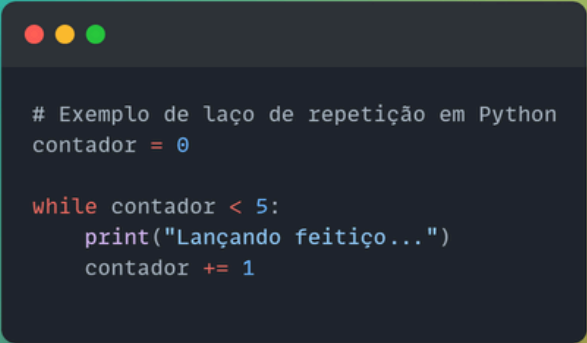
Você já deve ter percebido que, diferentemente dos feitiços que estamos acostumados em Latim, os termos usados na programação são, em sua maioria, em Inglês. Vou te dar uma ajuda para você entender o exemplo!

Esse IF significa "SE" em português, enquanto ELSE significa "OUTRO". Ou seja, nós estamos falando para o programa: Se a idade que for determinada for maior ou igual a 18, o código vai mostrar a frase "Você é maior de idade". Se essa condição não for atendida, iremos para o outro caminho que, por sua vez, mostrará a frase "Você ainda é menor de idade".

Fica fácil se olharmos assim, não é? Mas agora precisamos que nossos códigos tenham sustentação e funcionem dentro de uma espécie de símbolo do infinito, para que possam ser autônomos nas suas tarefas.

Os **laços de repetição** são como os encantamentos que nos permitem realizar uma ação repetidamente enquanto uma condição for verdadeira. Assim como um bruxo lança um feitiço de iluminação várias vezes para iluminar um corredor escuro, os laços de repetição nos permitem executar um bloco de código várias vezes até que uma condição seja alcançada.

Vamos visualizar isso dentro de um exemplo, como estamos fazendo desde o início da nossa jornada?



```
# Exemplo de laço de repetição em Python
contador = 0

while contador < 5:
    print("Lançando feitiço...")
    contador += 1
```

Vou te explicar agora o que está acontecendo naqueles feitiços ali e o que eles acarretam no final do processo.

Primeiro, criamos uma variável chamada "contador" e atribuímos o valor 0 a ela. Imagine o "contador" como um contador mágico que usamos para acompanhar quantas vezes lançamos um feitiço.

Em seguida, vemos uma linha que começa com "while". Isso significa que vamos repetir uma parte do código "enquanto" uma condição for verdadeira. No nosso caso, a condição é "contador < 5", ou seja, enquanto o contador for menor que 5.

Vamos continuar?

Agora, olhe para o que aparece abaixo do "while". Isso é o que será repetido várias vezes. No nosso caso, estamos apenas fazendo com que apareça a mensagem "Lançando feitiço...".

Depois da aparição dessa mensagem, vemos "contador += 1". Isso é como um feitiço que aumenta o valor do contador em 1 unidade a cada repetição. Ou seja, estamos contando quantas vezes já lançamos o feitiço.

Quando o contador chegar a 5, a condição "contador < 5" não será mais verdadeira. Então, o laço de repetição terminará e o programa continuará executando o código que vem depois dele. Em resumo, esse código está simulando um lançamento de feitiço 5 vezes!

Compreender o funcionamento das estruturas de controle é fundamental para se tornar um verdadeiro mestre da programação. Assim como um bruxo que domina seus feitiços mais poderosos, você, ao entender e praticar as estruturas de controle, estará preparado para enfrentar desafios cada vez maiores neste mundo mágico! Continue praticando, explorando e desvendando os segredos desse universo fascinante, pois é através da prática e da exploração que você se tornará um verdadeiro expert da área!

04

**“ONDE SERÁ QUE
NICOLAU FLAMEL
ESCONDERIA
A PEDRA FILOSOFAL
FUNCIONAL
MODULAR?”**

ALQUIMIA DIGITAL

Funções e modularidade

Já imaginou que loucura seria o Nicolau Flamel um programador a frente do seu tempo? Neste capítulo, vamos mergulhar na arte da alquimia digital, explorando o poder das funções e a modularidade na programação. Prepare-se para desbravar os mistérios das funções e organizar seus feitiços em módulos, dando assim mais um passo nessa arte da programar em Python.

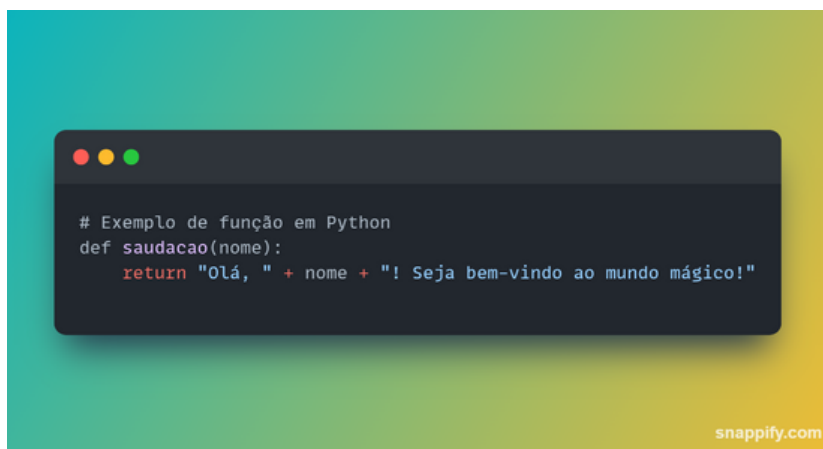
As **funções** na programação são blocos de código que realizam uma tarefa específica e podem ser reutilizados em diferentes partes do programa. Imagine que cada função é como um feitiço especializado que um bruxo domina para realizar uma tarefa específica de forma eficiente e precisa.

Quando escrevemos uma função, estamos encapsulando um conjunto de instruções que realizam uma determinada operação. Por exemplo, podemos ter uma função que calcula a média de uma lista de números, uma função que verifica se um número é primo, ou uma função que formata uma *string* (valor em forma de palavra) de acordo com um determinado padrão.

Assim como um bruxo escolhe o feitiço certo para uma tarefa específica, podemos chamar uma função sempre que precisarmos realizar uma operação específica em nosso programa. Isso torna nosso código mais organizado, modular e fácil de manter.

Além disso, as funções nos permitem evitar a repetição de código. Em vez de escrever o mesmo conjunto de instruções várias vezes em diferentes partes do programa, podemos simplesmente chamar a função sempre que precisarmos realizar aquela operação.

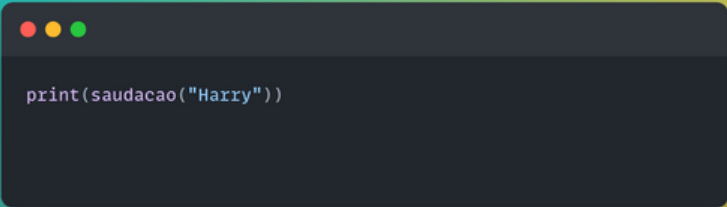
Vou te dar um exemplo na prática para você conseguir entender melhor:



Neste caso, a função "saudacao" recebe um parâmetro chamado "nome", que é o nome da pessoa que queremos saudar. Sabemos disso porque foi inserido o termo "DEF", que nos ajuda a traduzir essa nossa ideia para o computador.

Dentro da função, há uma instrução que retorna uma mensagem de saudação personalizada. A mensagem contém o nome da pessoa concatenado com a string "Olá, " e "Seja bem-vindo ao mundo mágico!".

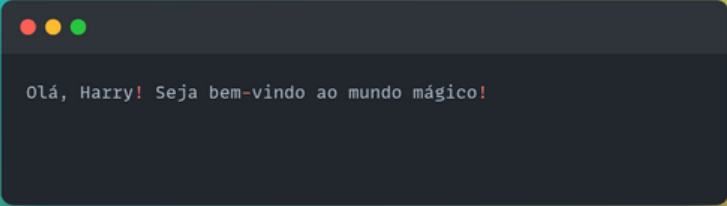
Então, se chamarmos essa função e passarmos um nome como argumento, ela retornará uma saudação personalizada com o nome que fornecemos. Por exemplo:

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

```
print(saudacao("Harry"))
```

snappify.com

Esta linha de código nos revelará:

A terminal window with a dark background and three colored window control buttons (red, yellow, green) in the top-left corner. The text `Olá, Harry! Seja bem-vindo ao mundo mágico!` is displayed in a light-colored monospace font.

```
Olá, Harry! Seja bem-vindo ao mundo mágico!
```

snappify.com

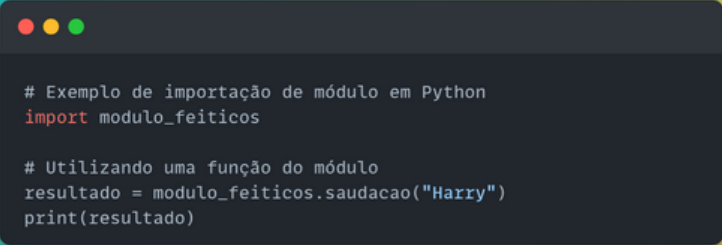
Mas e os **módulos**? Como podem ajudar a gente nessa jornada? Pense nos módulos como grimórios mágicos que guardam uma coleção de feitiços relacionados. Cada módulo pode conter uma variedade de funções, variáveis e até mesmo outras definições mais complexas.

Ao organizar nossos feitiços em módulos, estamos dividindo nosso código em partes mais gerenciáveis e reutilizáveis. Isso significa que podemos agrupar funcionalidades relacionadas em um único lugar, tornando nosso código mais organizado e fácil de entender.

Além disso, a modularidade nos permite reutilizar código de forma eficiente. Se tivermos uma função ou conjunto de funções que são úteis em diferentes partes de nosso programa (ou até mesmo em diferentes programas), podemos simplesmente importar o módulo correspondente e usar essas funcionalidades conforme necessário. Por mais que pareça complexo, é um conceito muito útil e facilmente acessado dentro do código.

Por exemplo, se tivermos um módulo chamado "utilidades.py" que contém algumas funções úteis para manipular *strings*, podemos importar esse módulo em nosso programa principal e usar essas funções sem precisar reescrevê-las.

Vamos ver um exemplo na prática de como isso funciona?



```
# Exemplo de importação de módulo em Python
import modulo_feiticos

# Utilizando uma função do módulo
resultado = modulo_feiticos.saudacao("Harry")
print(resultado)
```

snappify.com

Vamos com calma, ok? Neste código, estamos importando um módulo chamado "modulo_feiticos". Lembra que um módulo é como um livro de feitiços que contém diferentes encantamentos relacionados? Então, vamos aplicar esse conceito.

Depois de importar o módulo, estamos usando uma função específica que está dentro dele. Essa função é chamada de "saudacao". Ela é como um feitiço que cria uma saudação para uma pessoa específica, como estávamos vendo na parte anterior de funções!

A linha

resultado =

modulo_feiticos.saudacao("Harry")

significa que estamos chamando a função "saudacao" do módulo "modulo_feiticos" e passando o nome "Harry" como entrada.

A função "saudacao" faz seu trabalho, cria uma saudação personalizada para "Harry", e retorna essa saudação. Em seguida, armazenamos essa saudação na variável "resultado".

Depois de importar o módulo, estamos usando uma função específica que está dentro dele. Essa função é chamada de "saudacao". Ela é como um feitiço que cria uma saudação para uma pessoa específica, como estávamos vendo na parte anterior de funções!

A linha

resultado =

modulo_feiticos.saudacao("Harry")

significa que estamos chamando a função "saudacao" do módulo "modulo_feiticos" e passando o nome "Harry" como entrada.

A função "saudacao" faz seu trabalho, cria uma saudação personalizada para "Harry", e retorna essa saudação. Em seguida, armazenamos essa saudação na variável "resultado".

Finalmente, imprimimos o resultado usando `print(resultado)`, e o resultado é a saudação personalizada para "Harry".

Basicamente, estamos usando um feitiço (função) de um livro de feitiços (módulo) para criar uma saudação mágica para uma pessoa específica, neste caso, "Harry".

Se pudermos resumir então, as funções são como os feitiços especializados que nos permitem realizar tarefas específicas de forma eficiente e reutilizável em nossos programas, tornando nossa magia da programação mais poderosa e versátil. E organizar nossos feitiços em módulos nos permite dividir nosso código em partes mais gerenciáveis, reutilizáveis e fáceis de entender, tornando nossa prática de programação mais eficiente e poderosa.

05

“QUE MEU
CÓDIGO
NUNCA
SEJA MALFEITO”

PARABÉNS, JOVEM BRUXO!

Chegamos ao final da sua jornada

Ei, queremos parabenizá-lo por ter chegado até aqui em sua jornada no mundo da programação! Assim como um jovem bruxo que avança em seus estudos de magia, você demonstrou determinação, curiosidade e perseverança ao aprender os segredos da programação. Agora, vamos refletir sobre o caminho percorrido e as grandes possibilidades que se abrem diante de você.

UMA JORNADA DE DESCOBERTAS

Ao longo deste e-book inicial, você explorou os fundamentos da programação, desvendando os mistérios das estruturas de controle, funções, módulos e muito mais. Cada conceito aprendido foi como desvendar um novo feitiço, ampliando seu conhecimento e habilidades no mundo mágico da alquimia digital.

SUPERANDO DESAFIOS

No caminho, você pode ter enfrentado obstáculos e desafios, assim como os bruxos enfrentam criaturas mágicas e enigmas durante suas jornadas. Lembre-se de que cada desafio superado é uma oportunidade de crescimento e aprendizado, preparando-o para os desafios ainda maiores que encontrará no futuro. Continue estudando, buscando desafios que você vai longe!

O PODER DA PERSISTÊNCIA

Como em qualquer jornada, o segredo do sucesso na programação é a persistência. Não desanime diante dos obstáculos, mas continue praticando, experimentando e buscando conhecimento. Lembre-se sempre de que até mesmo os mais poderosos bruxos começaram como aprendizes, e é através da persistência que se tornaram mestres.

UM MUNDO DE POSSIBILIDADES

À medida que você irá avançar ainda em sua jornada de aprendizado, lembre-se das infinitas possibilidades que se abrem diante de você. Assim como um bruxo domina uma variedade de feitiços para enfrentar diferentes desafios, você pode usar suas habilidades de programação para criar soluções inovadoras, resolver problemas complexos e deixar sua marca no mundo.

Por mais que esse seja apenas um pequeno início nesse universo tão grande que é a programação, obrigada por ter se disposto e enfrentado!

Continue explorando, aprendendo e aprimorando suas habilidades, pois o mundo da programação é vasto e cheio de oportunidades esperando por você. Que sua jornada seja repleta de descobertas, conquistas e sucesso, e que você sempre mantenha viva a chama da curiosidade e da determinação em seu coração. Que seus próximos passos sejam guiados pelo poder da magia da programação, rumo a um futuro brilhante e cheio de possibilidades!

**PROJETO CRIADO PARA O BOOTCAMP EM
INTELIGÊNCIA ARTIFICIAL -
SANTANDER/2024**

