

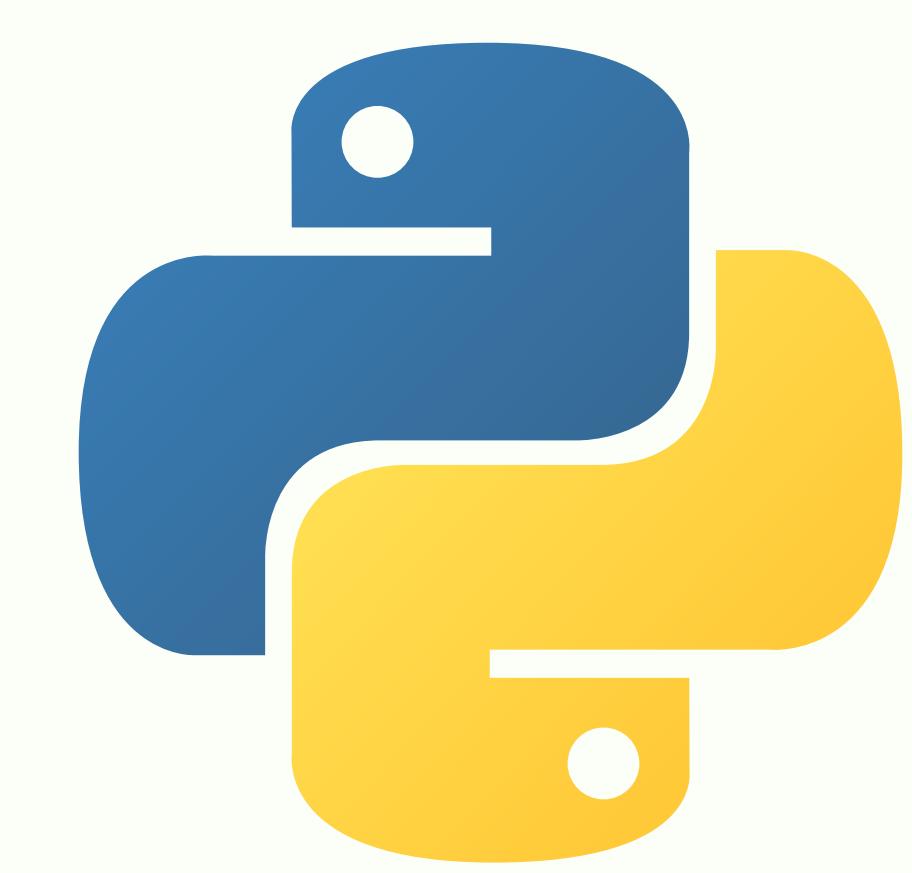
GUIA PYTHON

As aventuras da programação

Durante sua jornada
Domine ela!
jámais corra da serpente



Luiz Hanrry



Sumário

1 Introdução

2 Conceitos Básicos de Programação

3 Tipos de Dados e Variáveis

4 Operadores e Expressões

5 Estruturas Condicionais

6 Estruturas de repetição

7 Funções em Python

8 Trabalhando com Listas, Tuplas e Dicionários

9 Manipulação de Strings

10 Desafios

11 Funções úteis

12 Agradecimentos



introdução

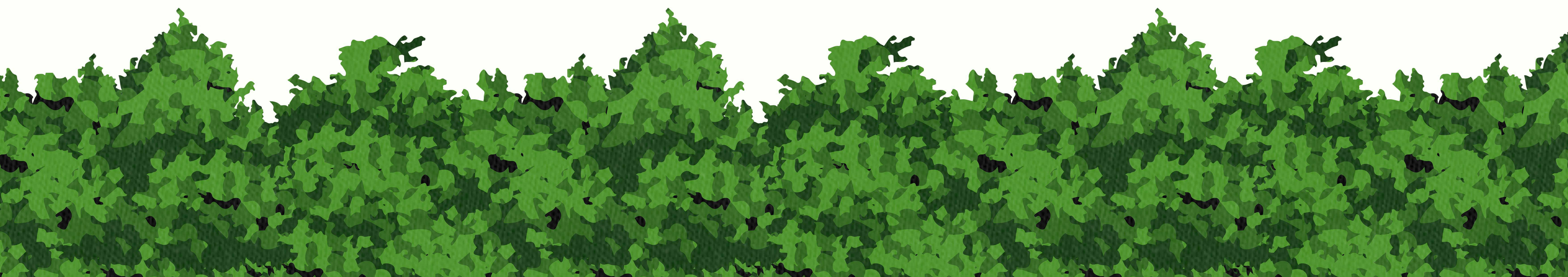
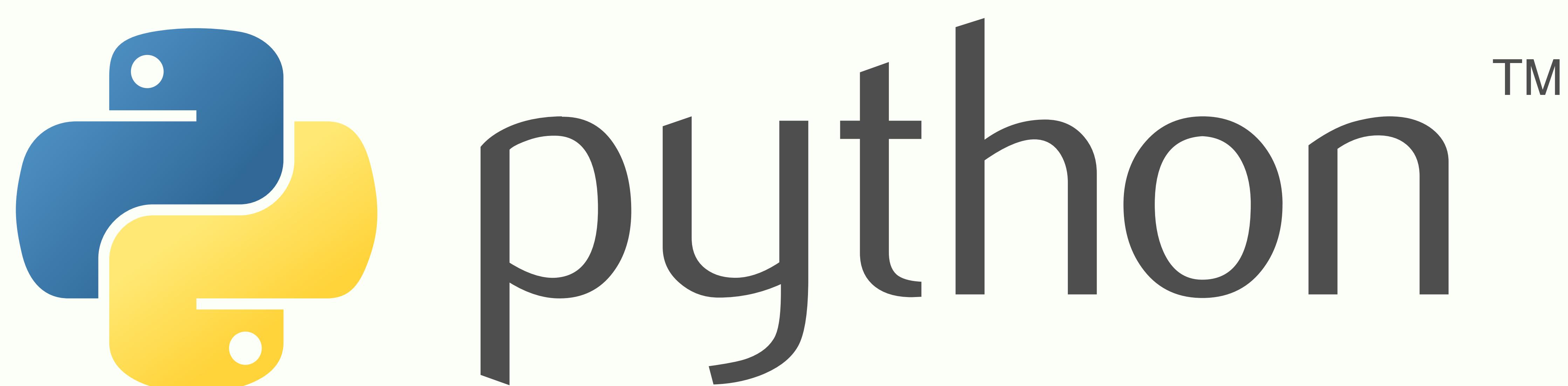


01. Introdução



1.1 o que é linguagem de programação

Uma linguagem de programação é como um conjunto de instruções que você dá a um computador para que ele execute tarefas específicas. Imagine que você está ensinando truques a um animal de estimação inteligente: você precisa usar uma linguagem que ele entenda. Python é como uma caixa de ferramentas mágica para programadores, cheia de feitiços poderosos que transformam linhas de texto em ações incríveis. Ele é amado por sua simplicidade e versatilidade, sendo usado em áreas como desenvolvimento web, ciência de dados e automação.



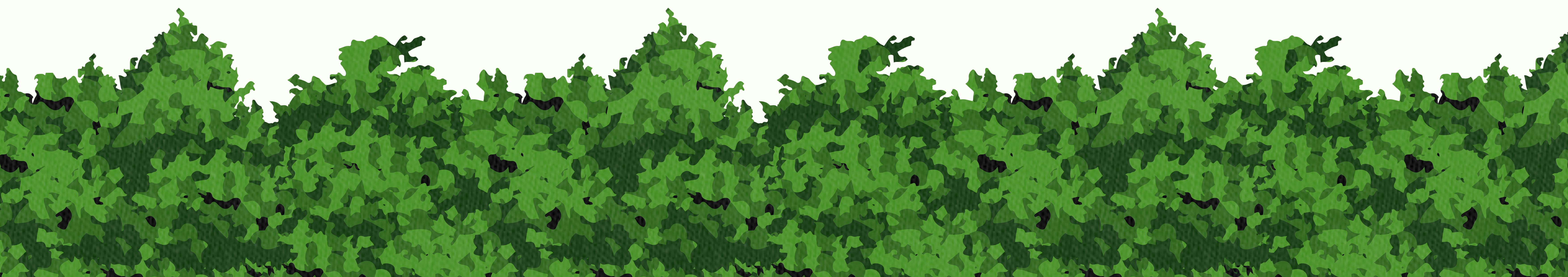


01. Introdução



1.2 Aplicações no nosso dia-a-dia

1. Desenvolvimento Web: Python é usado para criar sites e aplicativos na internet.
2. Análise de Dados: Com Python, as pessoas podem estudar informações para entender melhor o que está acontecendo em diferentes situações.
3. Inteligência Artificial e Aprendizado de Máquina: É quando os computadores aprendem a fazer coisas sozinhos, como reconhecer fotos ou prever o tempo.
4. Automação de Tarefas: Python pode ajudar a fazer com que o computador faça coisas por conta própria, como enviar e-mails ou organizar arquivos.
5. Desenvolvimento de Jogos: Com Python, as pessoas podem fazer seus próprios jogos de computador.
6. Aplicações de Desktop: Python é usado para fazer programas que podem ser usados diretamente em um computador, como editores de texto ou programas de desenho.





Conceitos básicos



02. CONCEITOS BÁSICOS

2.1 O que são variáveis e para que servem

Uma variável em programação é como uma caixa mágica onde você pode armazenar diferentes tipos de dados, como números, textos, listas, entre outros. Ela possui um nome único e permite que você se refira a um valor específico dentro do seu código. Por exemplo, em Python, podemos ter uma variável chamada `idade` que guarda o valor da idade de uma pessoa, ou uma variável chamada `nome` que guarda o nome de alguém. Essas variáveis podem ser modificadas ao longo do programa e são fundamentais para armazenar e manipular informações.

```
... Guia python

#Atribuição de valores a variáveis
nome = "Barry"
idade = 11
```

Quando você coloca `#` no começo da linha ela
não faz nada e se torna um comentário.



02. CONCEITOS BÁSICOS

2.2 Entradas e saídas de dados

Em Python, a entrada de dados é quando você pede ao usuário para digitar alguma informação. Por exemplo, podemos pedir ao usuário que digite seu nome ou idade, `input()` é usado para receber informações do usuário. O texto entre parênteses é exibido como uma mensagem para orientar o usuário sobre o que digitar.

A saída de dados é quando você mostra informações para o usuário. Por exemplo, podemos mostrar uma mensagem na tela ou o resultado de algum cálculo.`.print()` é usado para exibir uma mensagem na tela. Podemos combinar texto e variáveis separando-os por vírgulas para exibi-los juntos na mesma linha.

```
••• Guia python

#Entrada de dados
nome = input("Digite seu nome: ")

#Saída de dados
print("Olá,", nome, ", bom dia!")
```



Tipos de dados

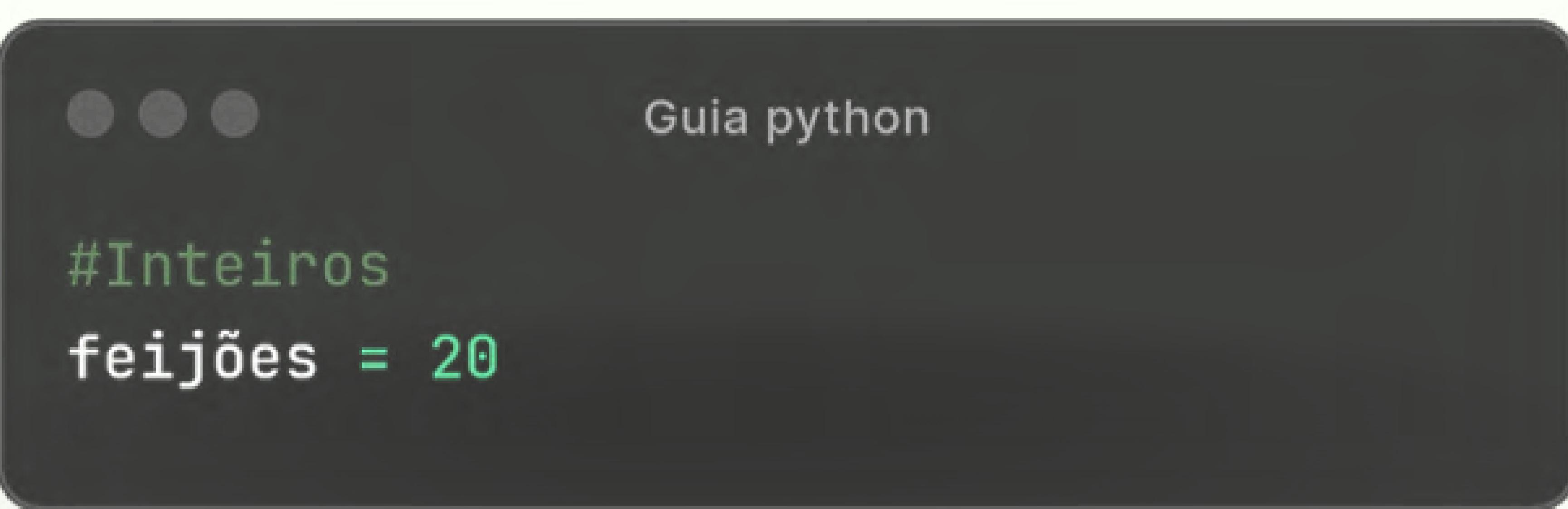


03. Tipos de dados



3.1 Inteiros (Int)

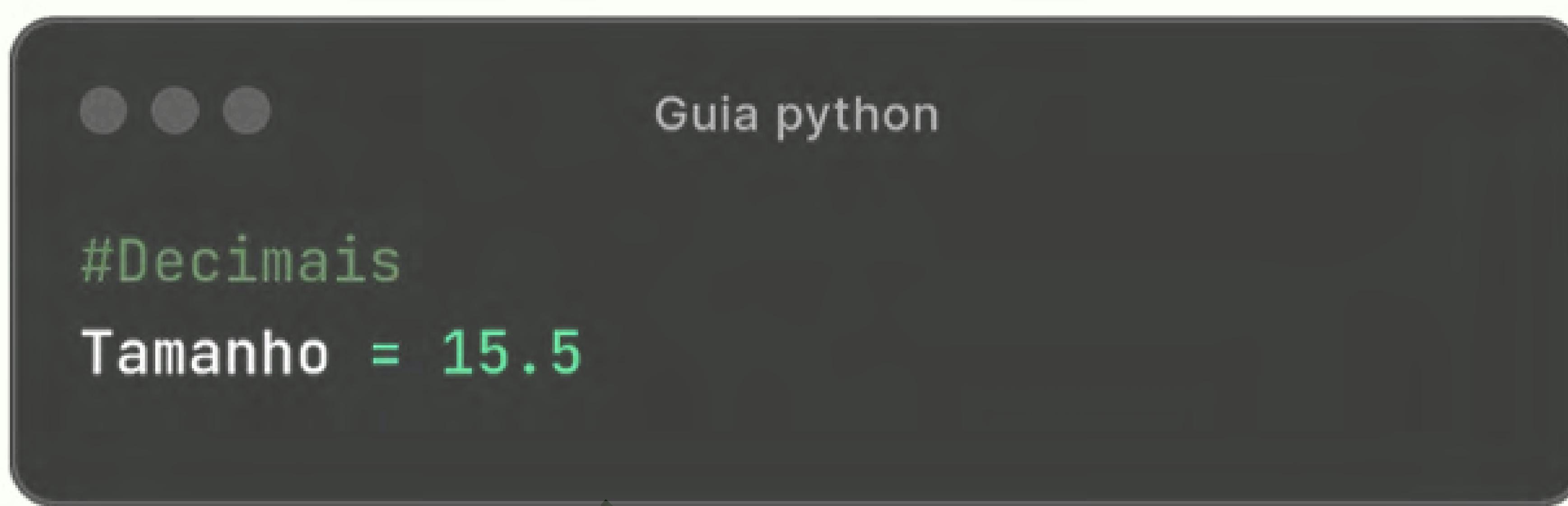
Inteiros (int): São como os tijolos de uma casa, números inteiros e sólidos. Eles representam números sem partes fracionárias. Por exemplo, você pode contar o número de feijões em uma vasilha usando inteiros.



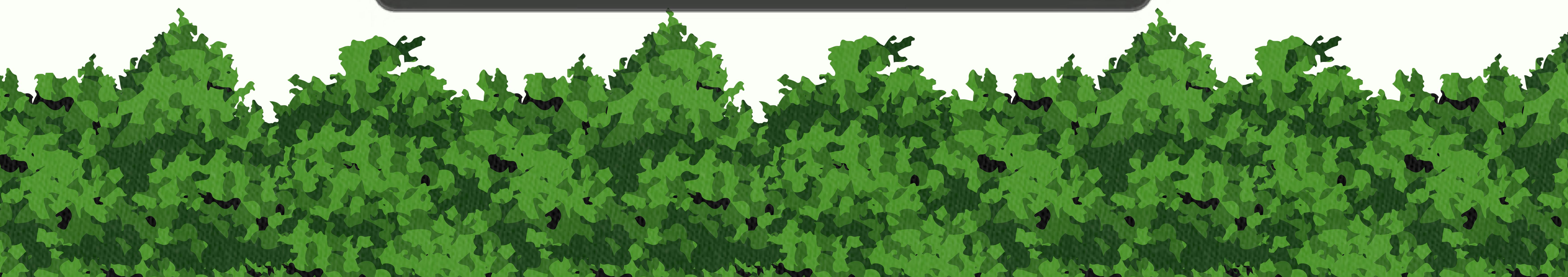
```
••• Guia python  
#Inteiros  
feijões = 20
```

3.2 Decimais (Float)

Decimais (float): São como os ingredientes de uma receita, podem ser divididos em partes menores. Os números decimais têm partes fracionárias. Por exemplo, se você medir o tamanho de um objeto , pode obter um número decimal.



```
••• Guia python  
#Decimais  
Tamanho = 15.5
```





03. Tipos de dados



3.1 Texto (Strings)

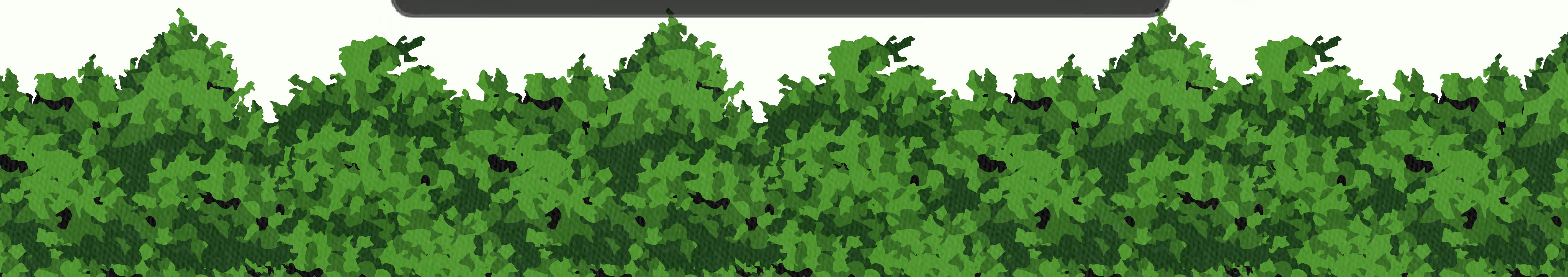
Texto (string): São como as letras em um livro, formando palavras e frases. As strings são usadas para representar texto e ela precisa estar entre aspas (""). Por exemplo, quando você escreve seu nome em um formulário, está usando uma string.

```
... Guia python  
#String  
nome = "João"
```

3.2 Booleanos (Bool)

Booleanos (bool): São como interruptores de luz, podem estar ligados (verdadeiro) ou desligados (falso). Os booleanos representam apenas dois valores: verdadeiro ou falso. Por exemplo, se estiver chovendo, o valor pode ser verdadeiro, se não, falso.

```
... Guia python  
#Booleanos  
esta_chovendo = True
```





Operadores e expressões



04. Operadores

4.1 Operadores Aritméticos

Operadores Aritméticos: São como ferramentas matemáticas em uma caixa de ferramentas. Eles são usados para realizar operações matemáticas básicas, como adição, subtração, multiplicação e divisão. Por exemplo, se você quiser calcular a soma de dois números, usaria o operador de adição.

... Guia python
#Aritmeticos
soma = 10 + 5

4.2 Operadores de Comparaçāo

Operadores de Comparaçāo: São como balanças que comparam dois valores. Eles são usados para comparar valores e produzir um resultado booleano (verdadeiro ou falso). Por exemplo, se você quiser verificar se dois números são iguais, usaria o operador de igualdade.

... Guia python
igualdade = (10 == 5)
#igualdade = False



04. Operadores

4.3 Operadores lógicos

Operadores Lógicos: São como portões em um labirinto, controlando o fluxo com base em condições. Eles são usados para combinar expressões booleanas e produzir um novo resultado booleano. Por exemplo, se você quiser verificar se duas condições são verdadeiras, usaria o operador "e".

```
● ● ●           Guia python  
condicao = (idade > 10) and (idade < 20)  
#passam apenas idades entre 10 e 20
```

4.4 Operadores de atribuição

Operadores de Atribuição: São como etiquetas em potes de tempero, atribuindo valores a variáveis. Eles são usados para atribuir valores a variáveis. Por exemplo, se você quiser atribuir o valor 10 a uma variável chamada "idade", usaria o operador de atribuição.

```
● ● ●           Guia python  
#Atribuindo valor a variável idade  
idade = 10
```



04. Operadores

4.5 Operadores de pertencimento

Operadores de Pertencimento: São como chaves em um chaveiro, verificando se um valor está contido em outro. Eles são usados para verificar se um valor está presente em uma sequência, como uma lista ou uma string. Por exemplo, se você quiser verificar se um elemento está em uma lista, usaria o operador "in".

```
•••          Guia python

#verifica se o número 3 esta na lista
lista = [1, 2, 3, 4, 5]
pertence = (3 in lista)
```

4.6 Operadores de identidade

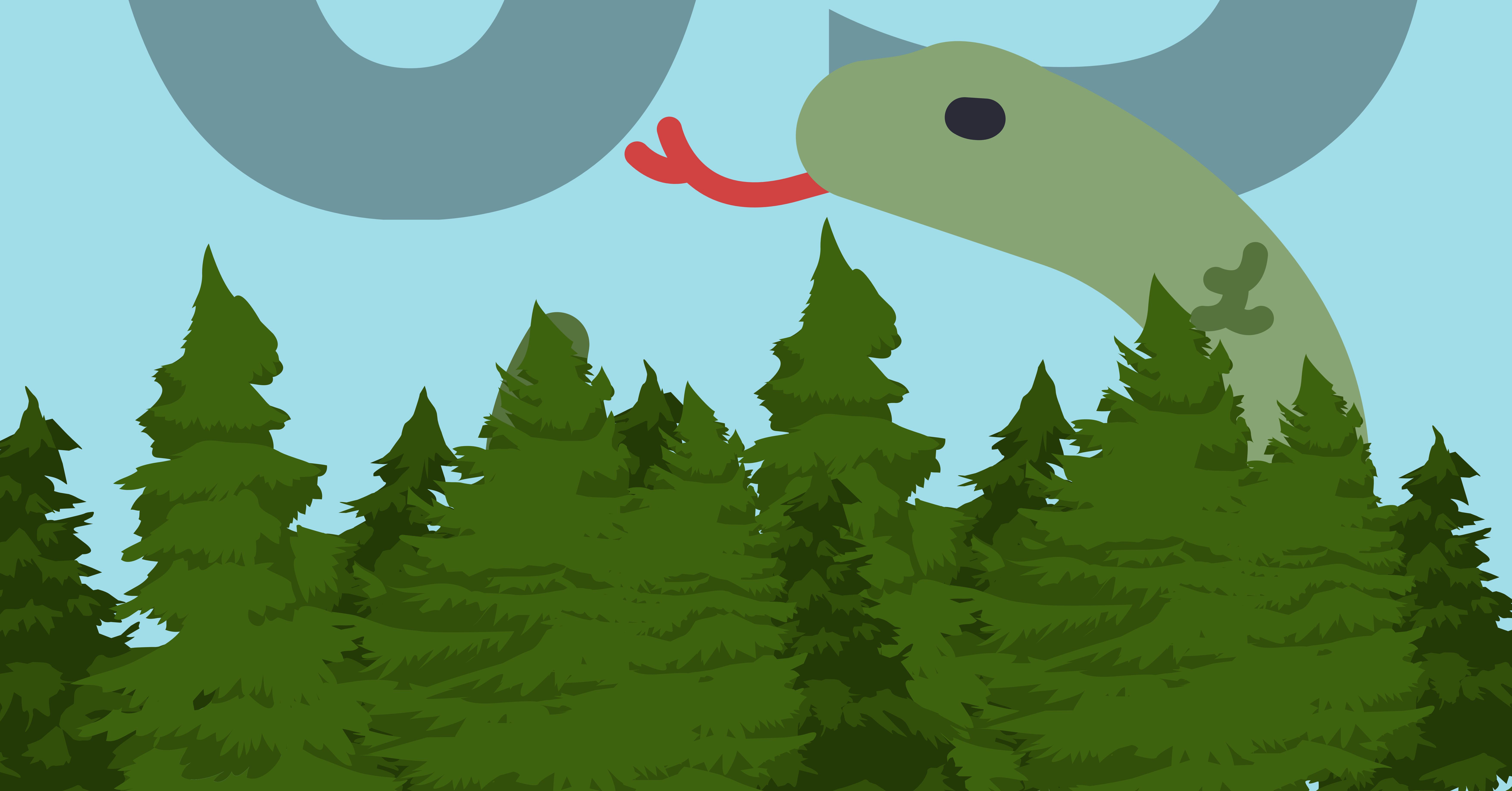
Operadores de Identidade: São como impressões digitais, verificando se dois objetos são os mesmos. Eles são usados para verificar se dois objetos têm a mesma identidade. Por exemplo, se você quiser verificar se duas variáveis apontam para o mesmo objeto, usaria o operador "is".

```
•••          Guia python

#verifica se x e y são os mesmos
x = [1, 2, 3]
y = [1, 2, 3]
mesma_identidade = (x is y)
```



Estruturas Condicionais





05. Condicionais



5.1 O que são estruturas condicionais

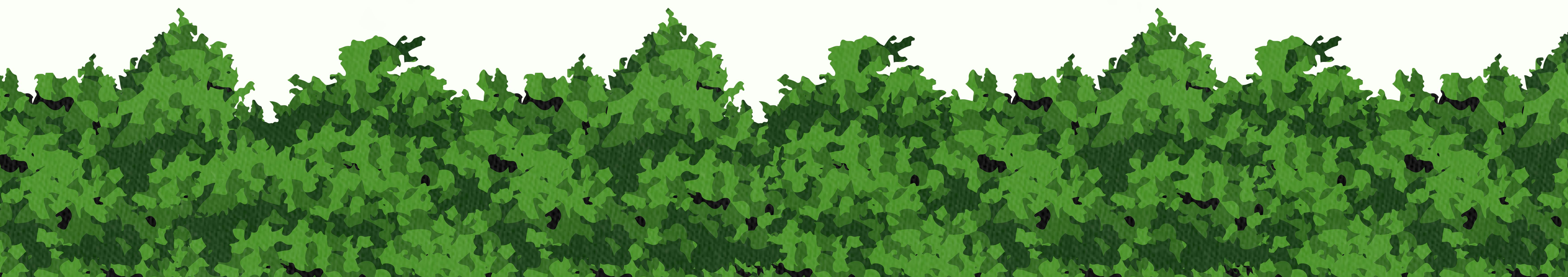
Estruturas Condicionais: São como bifurcações em um caminho, onde você decide qual direção tomar com base em uma condição. Elas permitem que um programa execute diferentes blocos de código com base em se uma condição é verdadeira ou falsa. Por exemplo, se estiver chovendo, você pode decidir levar um guarda-chuva; se não estiver, pode deixá-lo em casa.

5.2 Estrutura IF

Estrutura If: É como uma porta que se abre apenas se uma condição for atendida. Ela avalia uma expressão e executa um bloco de código se essa expressão for verdadeira. Por exemplo, se a sua nota em um teste for maior que 70, você passa de ano.

Guia python

```
#se a nota for maior que 70 execute
nota = 75
if nota > 70:
    passar_de_ano()
```





05. Condicionais

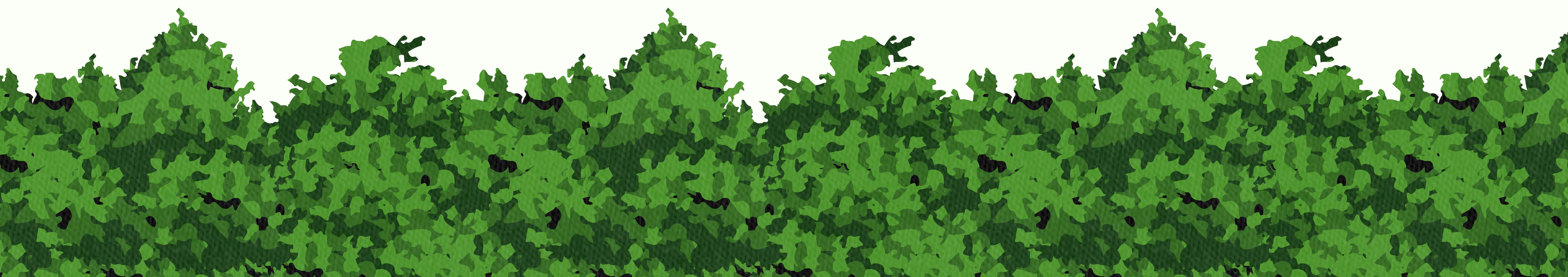


5.3 Estrutura If-Else

Estrutura If-Else: É como ter duas opções definidas: uma para quando a condição é verdadeira e outra para quando é falsa. Se a condição for verdadeira, o bloco de código dentro do "if" é executado; caso contrário, o bloco de código dentro do "else" é executado. Por exemplo, se estiver chovendo, você leva um guarda-chuva; se não, leva óculos de sol.

The screenshot shows a dark-themed code editor window titled "Guia python". In the code area, there is a single-line triple-dot menu icon on the left and the following Python code in the center:

```
if esta_chovendo:  
    levar_guarda_chuva()  
else:  
    deixar_em_casa()
```





05. Condicionais

5.4 Estrutura If-Elif-Else

Estrutura If-Elif-Else: É como ter várias portas, cada uma com uma condição diferente. O programa verifica cada condição em ordem e executa o bloco de código correspondente à primeira condição verdadeira encontrada. Se nenhuma condição for verdadeira, o bloco de código dentro do "else" é executado. Por exemplo, se a sua nota for maior que 90, você tem uma A; se for maior que 80, tem uma B; senão, tem uma C.

```
••••• Guia python

nota = 85
if nota > 90:
    nota = 'A'
elif nota > 80:
    nota = 'B'
else:
    nota = 'C'
```



Estruturas de repetição



06. Estruturas de repetição

6.1 Estruturas de repetição

Estruturas de Repetição: São como loops em uma pista de corrida, onde você executa uma mesma ação várias vezes até atingir um objetivo. Elas permitem que um bloco de código seja repetido múltiplas vezes, economizando tempo e tornando o código mais eficiente.

6.2 Estrutura While

Estrutura While: É como uma montanha-russa que continua em movimento enquanto houver pessoas na fila. Você define uma condição e o código continua sendo executado enquanto essa condição for verdadeira. É útil quando não se sabe exatamente quantas vezes o loop deve ser executado. Por exemplo, enquanto você tiver balas na sua sacola, continua comendo uma por uma.

```
● ● ●           Guia python

balas_na_sacola = 10
while balas_na_sacola > 0:
    comer_bala()
    balas_na_sacola -= 1
```



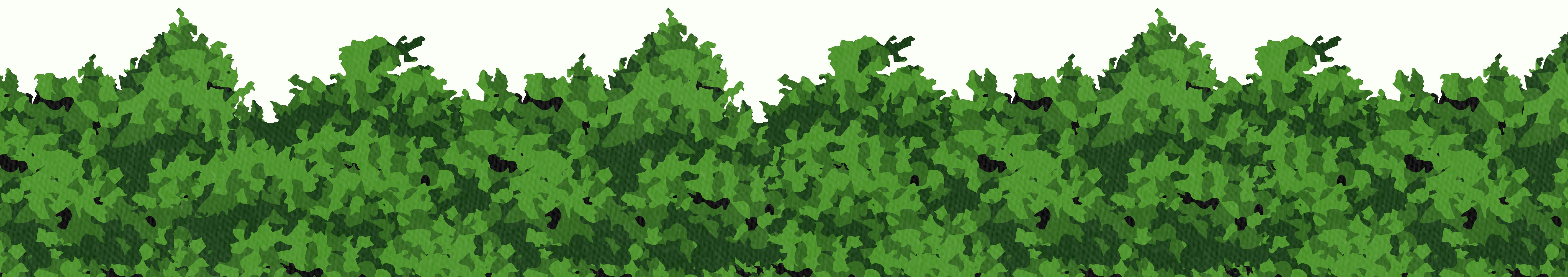
06. Estruturas de repetição

6.3 Estrutura For

É como contar as voltas em uma corrida de carro. Você especifica um intervalo ou uma coleção de itens e o código é executado para cada item desse intervalo ou coleção. É útil quando se sabe quantas vezes o loop deve ser executado ou quando se quer percorrer todos os elementos de uma sequência. Por exemplo, para cada número de 1 a 10, você imprime o número.

Guia python

```
for numero in range(1, 11):
    print(numero)
```





Funções



06. Funções

6 funções

Funções são como receitas de bolo: você define uma vez e pode usá-las várias vezes para obter o mesmo resultado delicioso. Elas aceitam ingredientes (parâmetros) e podem retornar um bolo pronto (valor de retorno).

Primeiramente você cria a função, neste exemplo é uma função para dar saudações a alguém :

```
● ● ●           Guia python

def saudacao(nome):
    # Esta função cumprimenta uma pessoa pelo nome
    print("Olá,", nome, "! Como vai?")
```

Logo depois podemos chamar a função enumeras vezes para termos o mesmo resultado, mudando apenas o parâmetro que estamos enviando para ela, neste exemplo o nome de uma pessoa :

```
● ● ●           Guia python

saudacao("Maria")
# Saída: Olá, Maria! Como vai?
saudacao("Pedro")
# Saída: Olá, Pedro! Como vai?
```



**listas, tuplas e
dicionários**



08. Listas, tuplas e dicionários

8.1 Listas

Listas: São como sacolas de compras, onde você pode armazenar diferentes itens. Elas são coleções ordenadas de elementos que podem ser modificados. Você pode adicionar, remover e modificar os itens de uma lista conforme necessário. São vetores que possuem valores em cada posição, onde o 1º tem valor de posição igual a 0. ([0,1,2...])

• • • Guia python
`#Lista de compras com várias frutas
compras = ["maçã", "banana", "laranja", "uva"]`

8.2 Tuplas

Tuplas: São como presentes embrulhados, onde o conteúdo não pode ser alterado após ser definido. Elas são coleções ordenadas de elementos imutáveis, o que significa que você não pode adicionar, remover ou modificar os itens após a criação da tupla.

• • • Guia python
`#coordenadas com tuplas
coordenadas = (10, 20)`



08. Listas, tuplas e dicionários

8.3 Dicionários

Dicionários: São como agendas telefônicas, onde você pode procurar informações usando um nome. Eles são coleções de pares chave-valor, onde cada valor é associado a uma chave única. Você pode recuperar rapidamente um valor, fornecendo a chave correspondente.

Por exemplo uma agenda onde se tem informações como nome e número das pessoas.

```
...  
Guia python  
agenda = {"Ana": 123456789, "João": 987654321, "Maria": 456789123}
```



Manipulação de Strings



09. Manipulação

de Strings

9.1 Concatenação

Concatenação de Strings: É como juntar pedaços de papel para formar uma mensagem maior. Você pode combinar duas ou mais strings para criar uma nova string.

```
...          Guia python

nome = "João"
sobrenome = "Silva"
nome_completo = nome + " " + sobrenome
print(nome_completo)
# Saída: João Silva
```

9.2 Formatação

Métodos de Formatação de Strings: São como moldes que você pode usar para organizar e formatar o texto de maneira específica. Eles permitem inserir valores em uma string de forma mais legível e organizada.

```
...          Guia python

nome = "Ana"
idade = 25
mensagem = "Olá, meu nome é {} e tenho {} anos.".format(nome, idade)
print(mensagem)
# Saída: Olá, meu nome é Ana e tenho 25 anos.
```



09. Manipulação de Strings

9.3 Manipulação

Métodos de Manipulação de Strings: São como ferramentas que você pode usar para cortar, dividir, substituir ou modificar partes de uma string.

```
● ● ●           Guia python

texto = "Python é uma linguagem de programação poderosa."

# Cortar uma parte do texto
print(texto[0:6]) # Saída: Python

# Dividir o texto em palavras
palavras = texto.split()
print(palavras) # Saída: ['Python', 'é', 'uma', 'linguagem', 'de',
'programação', 'poderosa.']

# Substituir uma palavra por outra
novo_texto = texto.replace("poderosa", "incrível")
print(novo_texto) # Saída: Python é uma linguagem de programação incrível.
```



Desafios



10. Desafios



Desafio de Programação: Calculadora Simples Crie uma calculadora que some dois números fornecidos pelo usuário e exiba o resultado.

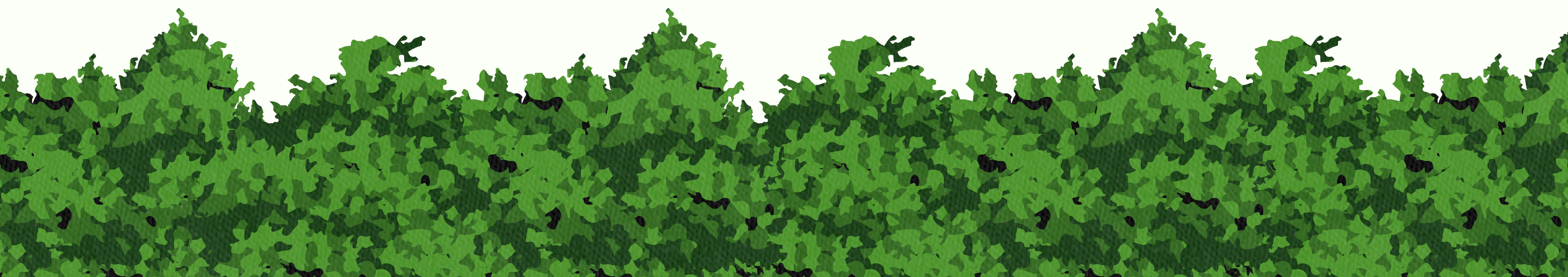
Desafio de Lógica: Verificador de Par ou Ímpar Escreva um programa que solicite um número ao usuário e determine se é par ou ímpar. Exiba uma mensagem correspondente.

Desafio de Strings: Cumprimentando o Usuário Peça ao usuário para inserir seu nome e exiba uma mensagem de boas-vindas personalizada, utilizando o nome fornecido.

Desafio de Listas: Maior Elemento Crie um programa que solicite ao usuário uma lista de números e exiba o maior número presente na lista.

Desafio de Dicionários: Agenda Telefônica Simples Desenvolva um programa que permita ao usuário armazenar e recuperar números de telefone. Use um dicionário para armazenar pares de nomes e números.

Desafio de Funções: Verificador de Número Primo Escreva uma função que verifique se um número fornecido pelo usuário é primo (divisível apenas por 1 e por ele mesmo). Retorna True se for primo e False caso contrário.





Funções úteis



11 Funções Úteis

`print()`: Função para exibir mensagens ou valores na tela.
`input()`: Função para solicitar entrada do usuário via teclado.
`len()`: Função para obter o tamanho (número de elementos) de uma sequência, como uma lista ou uma string.
`range()`: Função para criar uma sequência de números em um intervalo especificado.
`str()`: Função para converter um valor em uma string.
`int()`: Função para converter um valor em um número inteiro.
`float()`: Função para converter um valor em um número decimal.
`max()`: Função para encontrar o valor máximo em uma sequência.
`min()`: Função para encontrar o valor mínimo em uma sequência.
`sum()`: Função para calcular a soma de todos os elementos em uma sequência.
`sorted()`: Função para ordenar uma sequência.
`append()`: Método para adicionar um elemento ao final de uma lista.
`remove()`: Método para remover o primeiro elemento com um valor específico de uma lista.
`split()`: Método para dividir uma string em uma lista de substrings.
`join()`: Método para concatenar uma lista de strings em uma única string.



11 Funções Úteis



`keys()`: Método para retornar uma lista contendo todas as chaves do dicionário.

`values()`: Método para retornar uma lista contendo todos os valores do dicionário.

`items()`: Método para retornar uma lista contendo todos os pares chave-valor do dicionário, como tuplas.

`get()`: Método para acessar o valor associado a uma chave específica. Se a chave não existir, retorna um valor padrão especificado.

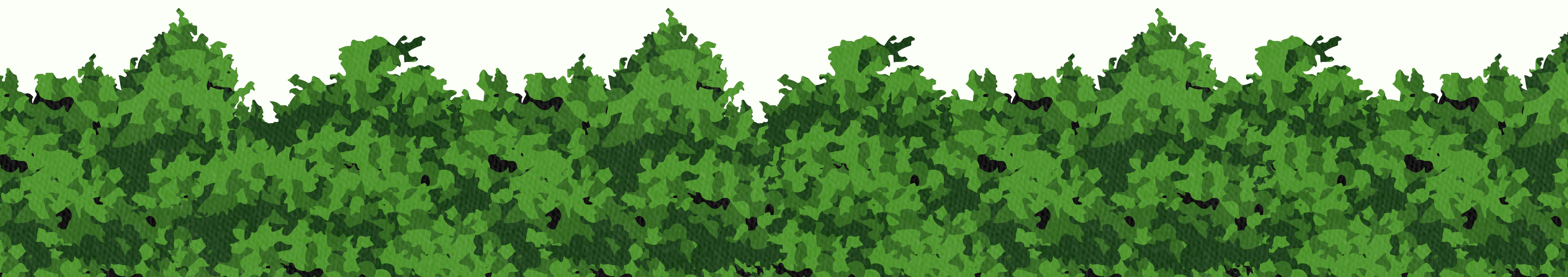
`pop()`: Método para remover um item do dicionário com base na chave especificada e retornar o valor associado a essa chave.

`popitem()`: Método para remover e retornar o último par chave-valor inserido no dicionário como uma tupla.

`clear()`: Método para remover todos os itens do dicionário, deixando-o vazio.

`update()`: Método para atualizar um dicionário com os pares chave-valor de outro dicionário ou de uma sequência de pares chave-valor.

`copy()`: Método para criar uma cópia superficial (shallow copy) do dicionári





Agradecimentos



12 Agradecimentos

Gostaria de expressar minha profunda gratidão a todos que tornaram este livro possível. A jornada para escrever um e-book que ensina Python básico foi desafiadora, mas extremamente gratificante.

Gostaria de agradecer a todos os leitores deste livro. Espero que este livro possa ser um recurso útil em sua jornada de aprendizado de Python. Seu feedback e apoio contínuos são a razão pela qual este livro existe.

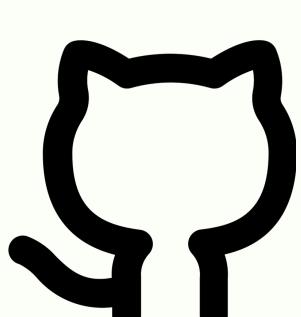
Por fim, gostaria de mencionar que este livro faz parte de um desafio do curso de Inteligência Artificial da DIO. Foi uma experiência incrível e estou grato pela oportunidade de contribuir para a comunidade de IA.

Obrigado,

Luiz Hanrry



Luiz Hanrry Nunes Araújo



Hanrryvis

Para entrar em contato clique nos ícones acima ou pesquise meu nome