



Ebook

Primeiras Sintaxes do Python

Dando os primeiros passos na
linguagem mais amigável
da programação

Sumário

Capítulo 01	003
Capítulo 02	008
Capítulo 03	013
Capítulo 04	018
Capítulo 05	022
Capítulo 06	027
Capítulo 07	034

Capítulo 01

Introdução ao Python

- O que é Python?
- Instalação
- Usando o IDLE ou VS Code
- Primeiro programa



O que é Python e por que aprender

Python é uma **linguagem de programação** criada com o objetivo de ser **fácil de entender e usar**. Ela tem uma **sintaxe simples**, o que significa que o código parece quase uma frase escrita em português ou inglês. Por isso, é uma das linguagens mais indicadas para quem está começando no mundo da programação.

Além de ser simples, o Python é muito poderoso. Ele é usado em várias áreas, como:

- **Desenvolvimento de sites e aplicativos** (ex: Instagram, YouTube, Netflix usam Python em parte do seu sistema);
- **Análise de dados e Inteligência Artificial**;
- **Automação de tarefas** (como mover arquivos, enviar e-mails, preencher planilhas automaticamente);
- **Criação de jogos e softwares**;
- **Ciência e pesquisa**, em simulações e cálculos complexos.

Ou seja, ao aprender Python, você abre muitas portas — tanto para o mercado de trabalho quanto para projetos pessoais.

Instalação e configuração (Windows, macOS e Linux)

Antes de começar a programar, você precisa **instalar** o Python no seu computador.

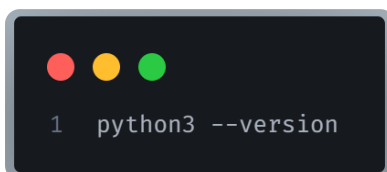
Acesse o site oficial: <https://www.python.org/downloads>

Baixe a **versão recomendada** (geralmente a mais recente).

- **Windows:** Durante a instalação, marque a opção "Add Python to PATH" — isso facilita o uso do Python pelo terminal.

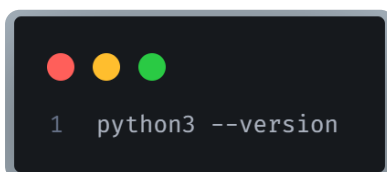
Clique em Install Now e aguarde a conclusão.

- **macOS:** O macOS já vem com uma versão antiga do Python instalada, mas o ideal é baixar a mais nova pelo site. Depois de instalar, você pode abrir o Terminal e digitar: Se aparecer algo como Python 3.x.x (x → representa as versões), tudo está funcionando!



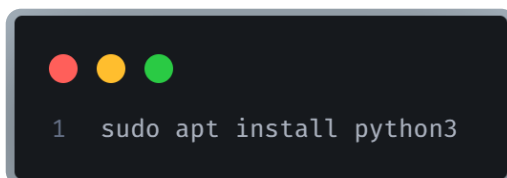
```
1 python3 --version
```

- **Linux:** A maioria das distribuições Linux já tem o Python instalado. Para conferir, abra o terminal e digite:



```
1 python3 --version
```

Caso precise instalar, use:



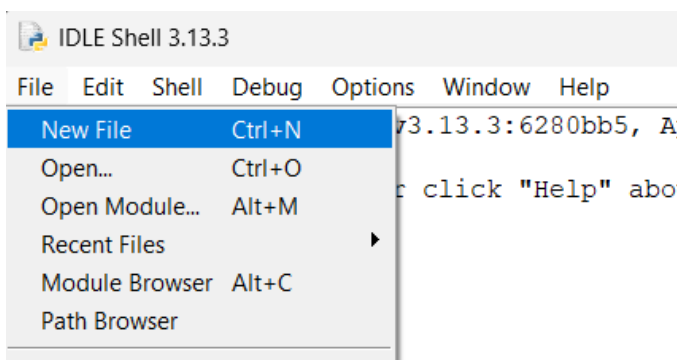
```
1 sudo apt install python3
```

Usando o IDLE ou VS Code

O Python vem com um programinha chamado **IDLE**, que é o ambiente padrão para escrever e testar códigos.

Para abrir o IDLE, basta procurar por "IDLE" no menu do seu computador. Ele possui uma janela chamada **Shell**, onde você pode digitar comandos e ver os resultados imediatamente.

Para criar programas mais complexos, crie um novo arquivo.



Outra opção (e a preferida de muitos programadores) é o **VS Code (Visual Studio Code)**, que é um editor de código moderno e cheio de recursos.

Você pode baixá-lo em <https://code.visualstudio.com>

Depois de instalar, procure e adicione a **extensão "Python"** (ícone azul com uma cobra).

Com isso, o VS Code reconhecerá automaticamente seus arquivos **.py** e ajudará a rodar o código.



Seu primeiro programa: print("Olá, mundo!")

Agora chegou a hora de escrever seu primeiro programa em Python!

Abra o IDLE ou o VS Code. Crie um novo arquivo e salve com o nome primeiro_programa.py. Digite o seguinte código:



```
1 print("Hello World")
```

- OBS: print() → usado para exibir textos / informações.
- Execute o programa.
 - No **IDLE**, clique em *Run* → *Run Module (F5)*.
 - No **VS Code**, clique no botão "Run Python File".
- Você verá a frase:
 - No VS Code:

```
[Running] python -u "c:\Users\rapha\OneDrive\Programas\bootcamp Santander\Introdução IA generativa\primeiro_programa.py"
Hello World
```

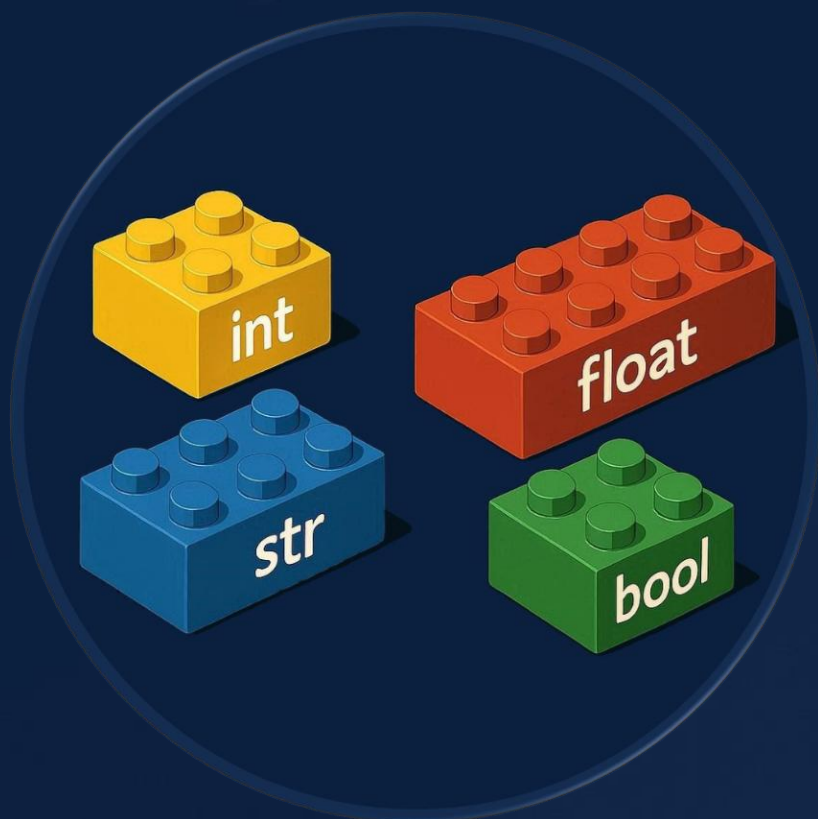
- No IDLE:

```
= RESTART: C:/Users/rapha/OneDrive/Programas/bootcamp Santander/Introdução IA ge
nerativa/primeiro2.0.py
Hello World
|
```

Capítulo 02

Comentários e tipos de dados

- Comentários (# e """ """)
- Tipos básicos: int, float, str, bool
- Função type()
- Conversão de tipos (int(), float(), str())



Comentários no código

Comentários são textos que o Python ignora quando o programa é executado. Eles servem para explicar o que o código faz, tornando-o mais fácil de entender — tanto para você quanto para outras pessoas.

Existem duas formas principais de criar comentários:

- **Comentário de uma linha:** use o símbolo # antes da frase.



```
1 # Este programa mostra uma mensagem na tela
2 print("Olá, mundo!") # Também posso comentar no fim da linha
```

- **Comentário de várias linhas:** Use três aspas duplas (""") no início e no fim.



```
1 """
2 Este é um comentário
3 que ocupa várias linhas.
4 Ótimo para explicações longas!
5 """
6 print("Python é divertido!")
```

- Dessa forma também é possível criar linhas de texto com mais de uma linha.

Tipos básicos de dados

Em Python, cada valor tem um tipo, que indica o que ele representa. Os principais são:

Tipo	Exemplo	Significado
int	10	Número inteiro
float	3.14	Número decimal
str	"Olá"	Texto (string)
bool	True / False	Valor lógico (verdadeiro ou falso)

Código:

```
1  idade = 25          # int
2  altura = 1.75       # float
3  nome = "Ana"        # str
4  estudando = True    # bool
5
6  print(idade, altura, nome, estudando)
```

Saída:

```
[Running] python -u "c:\Users\rapha\OneDrive
25 1.75 Ana True
```

função type()

A função type() mostra qual é o tipo de um valor ou variável.



```
1  numero = 10
2  print(type(numero))  # <class 'int'>
3
4  texto = "Python"
5  print(type(texto))   # <class 'str'>
```

Isso é útil para entender o que o Python está interpretando.

- OBS: O comentário no código é o que será exibido após executar o código.

Conversão de tipos

Às vezes, você precisa transformar um tipo de dado em outro. Por exemplo, quando o usuário digita algo com `input()`, o valor vem sempre como texto (`str`), mesmo que pareça um número.

- Exemplo simples:

```
1 numero_texto = "10"
2 numero = int(numero_texto) # converte de texto para número
3 print(numero + 5)          # agora funciona como número!
```

- Outras conversões:

```
1 # Número para texto
2 idade = 20
3 texto = str(idade)
4 print("Tenho " + texto + " anos.")
5
6 # Float para inteiro
7 altura = 1.73
8 print(int(altura)) # 1
9
10 # Inteiro para float
11 x = 7
12 print(float(x))   # 7.0
```

Essas conversões são chamadas de casting e ajudam muito na hora de formatar ou calcular valores.

Capítulo 03

Variáveis e boas práticas de nomeação

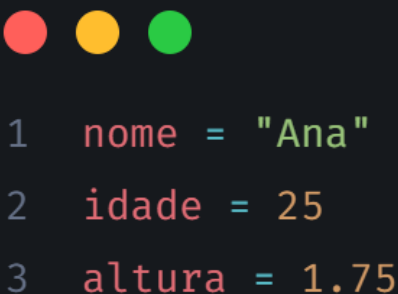
- O que são variáveis
- Criando e atribuindo valores
- Regras para nomes válidos
- Convenção: Snake case e constantes



```
1 nome = "Andedigmon"
2 soma = 0
3 numero = float(input("Digite um número: "))
4 soma += numero
```

O que são variáveis

Variáveis são espaços na memória onde o Python guarda informações. Pense nelas como caixinhas que armazenam dados para serem usados depois.



```
1 nome = "Ana"
2 idade = 25
3 altura = 1.75
```

Essas variáveis podem ser usadas em cálculos ou mensagens:



```
1 print(f"{nome} tem {idade} anos e mede {altura}m.")
```


Saída:



```
[Running] python -u "c:\Users\rapha\O...
Ana tem 25 anos e mede 1.75m.
```

Criando e atribuindo valores

O símbolo = atribui um valor à variável. Você também pode atribuir várias de uma vez:



```
1  a, b, c = 10, 20, 30
```

Ou usar uma variável em outra:




```
1  total = a + b + c
```

Regras para nomes válidos


- 1 - Devem começar com uma letra ou _ (underline).
- 2 - Não podem começar com número.
- 3 - Sem espaços nem acentos.
- 4 - Diferenciam maiúsculas e minúsculas (idade ≠ Idade).

Forma **correta**:



```
1 nome_completo = "Ana Maria"  
2 _idade = 22
```

Forma **errada**:



```
1 2ano = 2025  
2 nome completo = "João"
```


Convenção: Snake case

Em Python, usamos nomes com letras minúsculas e _ (underline) para separar palavras. Isso é chamado de snake_case.



```
1 nota_final = 9.5
2 preco_produto = 29.99
```

Constantes

Python **não possui constantes reais**, mas usamos **nomes em maiúsculas** para representar valores que não devem mudar.

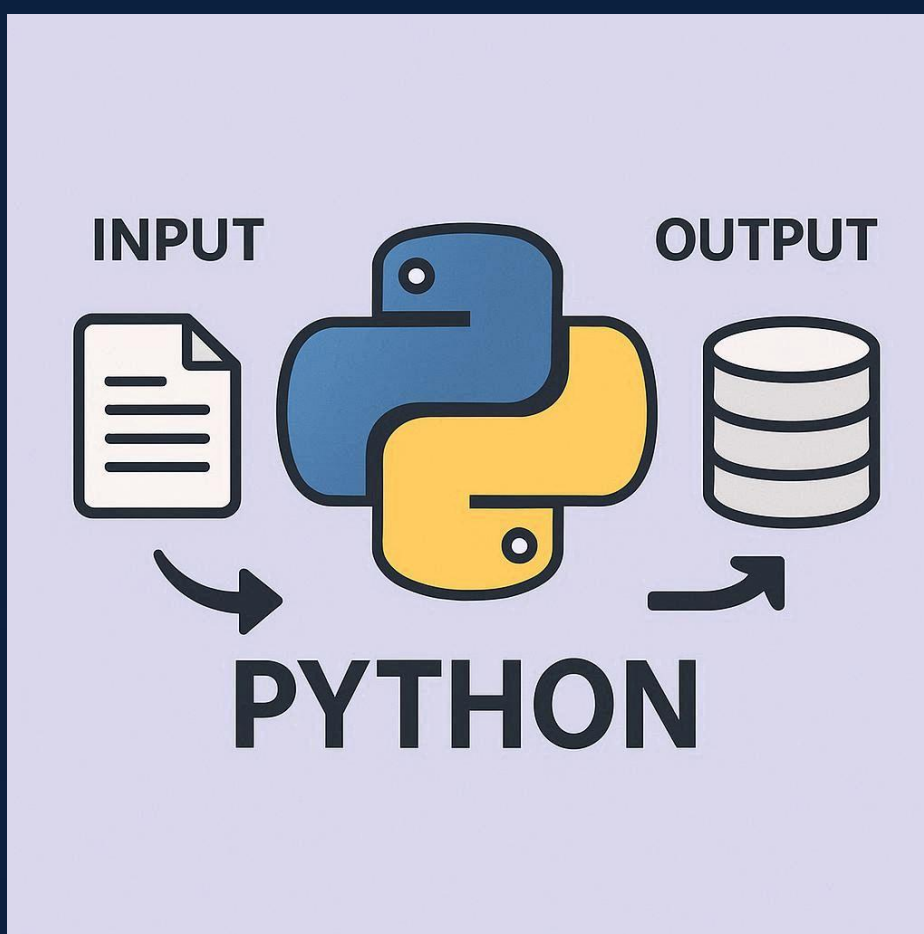


```
1 PI = 3.14159
2 TAXA_JUROS = 0.05
```

Capítulo 04

Entrada e saída de dados

- Capturando dados com `input()`
- Mostrando informações com `print()`
- Formatando números e texto



Capturando dados com input()

A função `input()` permite interagir com o usuário, pedindo que ele digite algo.



```
1 nome = input("Digite seu nome: ")
2 print(f"Olá, {nome}!")
```

⚠️ Tudo que vem do `input()` é do tipo `string` (texto), mesmo que pareça número.



```
1 idade = int(input("Digite sua idade: "))
2 print(f"Daqui a 10 anos você terá {idade + 10}.")
```

Mostrando informações com print()

O `print()` serve para exibir mensagens na tela. Você pode juntar valores de várias formas:

- Concatenação:



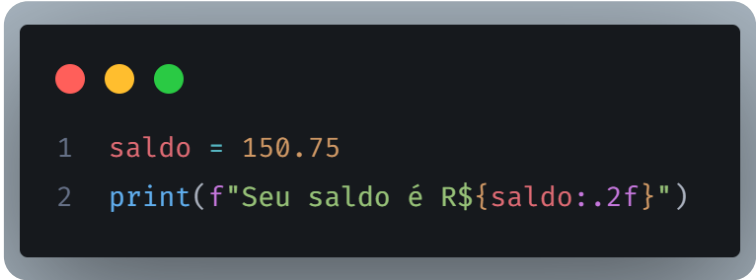
```
1 nome = "Lucas"
2 print("Olá, " + nome + "!")
```

- Separação por vírgulas:



```
1 idade = 20
2 print("Você tem", idade, "anos.")
```

- f-string (mais moderna e prática):



```
1 saldo = 150.75
2 print(f"Seu saldo é R${saldo:.2f}")
```

Formatando números e texto

Você pode **controlar casas decimais** e **formatar strings**:



```
1 pi = 3.141592
2 print(f"O valor de pi é {pi:.3f}") # 3 casas decimais
```

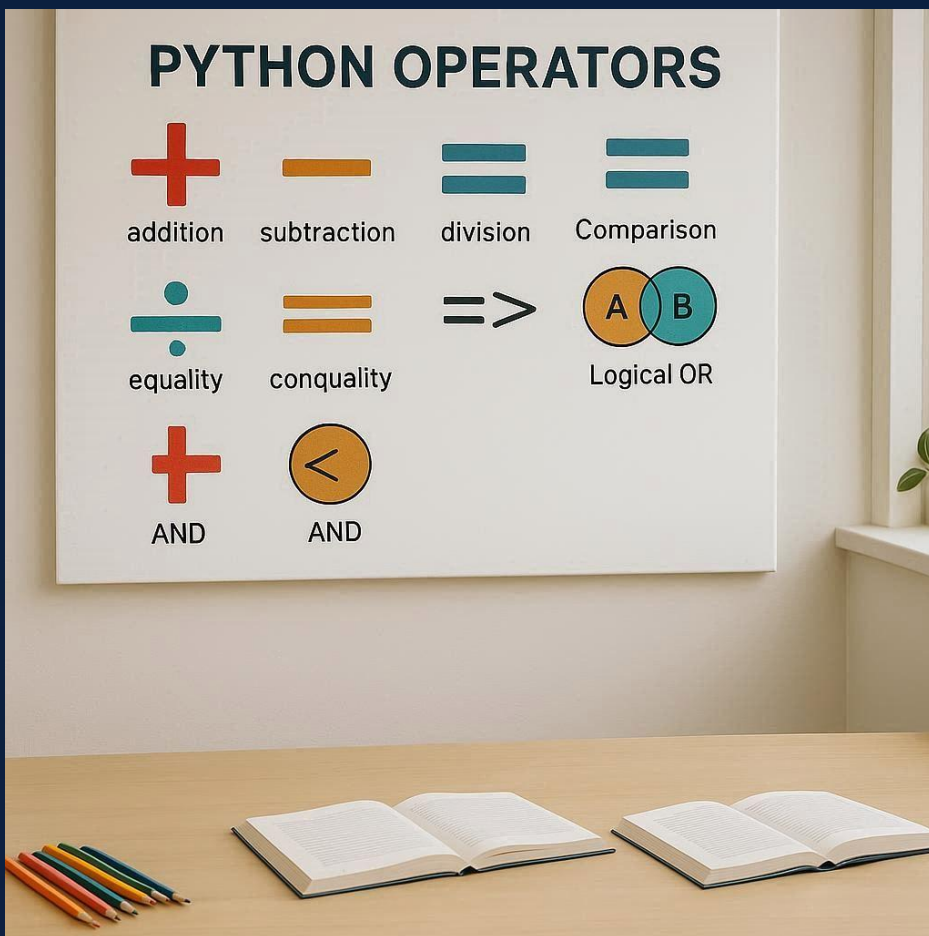
Saída:

```
[Running] python -u "c:\Users\rapha
O valor de pi e 3.142
```

Capítulo 05

Operadores em Python

- Operadores aritméticos
- Operadores relacionais
- Operadores lógicos
- Ordem de precedência



Operadores Aritméticos

Usados para fazer contas.

Operador	Significado	Exemplo	Resultado
+	Soma	5 + 3	8
-	Subtração	5 - 3	2
*	Multiplicação	5 * 3	15
/	Divisão comum	5 / 2	2.5
//	Divisão inteira	5 // 2	2
%	Resto da divisão	5 % 2	1
**	Exponenciação	2 ** 3	8



```
1 a = 10
2 b = 3
3
4 print("Soma:", a + b)
5 print("Divisão inteira:", a // b)
6 print("Potência:", a ** b)
```

Operadores Relacionais

Usados para comparar valores. O resultado sempre será **True** (verdadeiro) ou **False** (falso).

Operador	Significado	Exemplo	Resultado
==	Igual a	5 == 5	True
!=	Diferente de	5 != 3	True
>	Maior que	7 > 3	True
<	Menor que	3 < 2	False
>=	Maior ou igual	6 >= 6	True
<=	Menor ou igual	2 <= 3	True



```
1  idade = 18
2  print(idade >= 18) # True
```


Operadores Lógicos

Usados para combinar condições.

Operador	Significado	Exemplo	Resultado
and	Verdadeiro se todas forem verdadeiras	True and False	False
or	Verdadeiro se pelo menos uma for verdadeira	True or False	True
not	Inverte o valor lógico	not True	False



```
1  idade = 20
2  tem_carteira = True
3
4  if idade >= 18 and tem_carteira:
5      print("Pode dirigir!")
```

Ordem de Precedência

A ordem que o Python segue ao calcular expressões:

1. ()
2. **
3. *, /, //, %+, -
4. Operadores relacionais
5. not, and, or

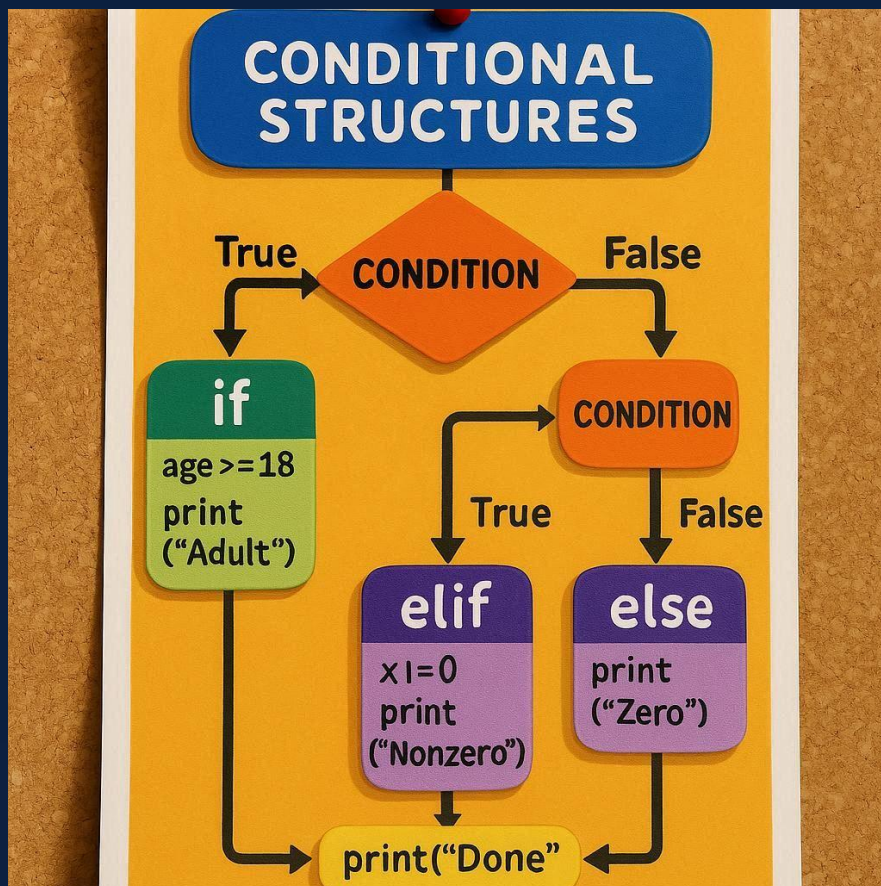


```
1 resultado = 2 + 3 * 4 # Multiplica antes de somar → 14
```

Capítulo 06

Estruturas Condicionais

- if, elif e else
- Condições aninhadas
- Expressões curtas (ternário)
- Exemplos práticos



if, elif e else

As **condições** permitem que o programa tome decisões com base em valores.



```
1  if idade < 18:
2      print("Menor de idade")
3  elif idade == 18:
4      print("Tem 18 anos!")
5  else:
6      print("Maior de idade")
```

Condições aninhadas

Você pode colocar uma condição dentro de outra:



```
1  nota = 8
2
3  if nota >= 7:
4      if nota == 10:
5          print("Nota perfeita!")
6      else:
7          print("Aprovado!")
8  else:
9      print("Reprovado!")
```

Expressões curtas (ternário)

Forma resumida de um if simples:



```
1  idade = 20
2  status = "Maior" if idade >= 18 else "Menor"
3  print(status)
```

Exercícios

1 – Crie um programa que receba a idade do usuário e retorne a mensagem:

"Você é maior de idade.", caso o usuário seja maior de idade;

caso contrário, retorne: "Você é menor de idade."

2 – Crie um programa que receba duas notas de um aluno. O programa deverá calcular a média do aluno e retornar uma das seguintes mensagens:

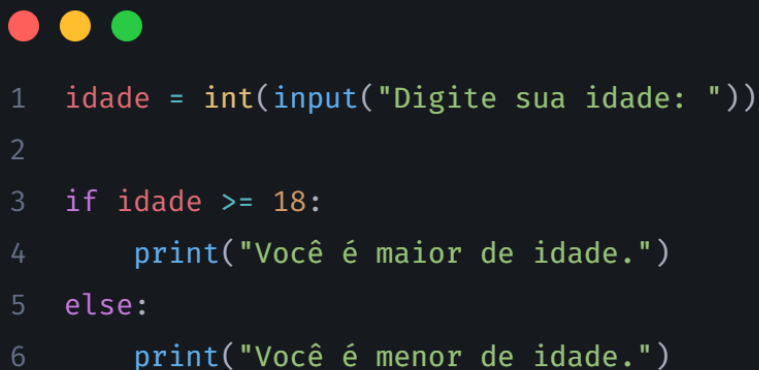
"Aprovado!", caso a média seja maior que 7;

"Recuperação!", caso a média seja menor que 7 e maior ou igual a 5;

"Reprovado!", caso a média seja menor que 5.

Resolução exercício 1

1 –

A terminal window with a dark background and three colored window control buttons (red, yellow, green) at the top left. It contains a Python script with six lines of code, each numbered from 1 to 6 on the left. The code uses syntax highlighting: keywords like 'int', 'input', 'if', 'else', and 'print' are in blue, strings are in green, and the variable 'idade' is in red. The code prompts the user for their age and prints a message based on whether they are 18 or older.

```
1  idade = int(input("Digite sua idade: "))
2
3  if idade >= 18:
4      print("Você é maior de idade.")
5  else:
6      print("Você é menor de idade.")
```

- `idade = int(input("Digite sua idade: "))`:
 - `input("Digite sua idade: ")` → exibe a mensagem pedindo que o usuário digite sua idade e lê o valor digitado. Por padrão, o `input()` retorna uma string, então usamos `int()` para converter o valor para número inteiro. O resultado é guardado na variável "idade".
- `if idade >= 18`:
 - O comando "if" verifica uma condição. Aqui, ele testa se a variável `idade` é maior ou igual a 18. Se for verdadeiro, o bloco logo abaixo (indentado) será executado.
- `print("Você é maior de idade.")`
 - Esse comando será executado somente se a condição do "if" for verdadeira. Ele mostra na tela a mensagem "Você é maior de idade."
- `else`:
 - O "else" é executado quando a condição do "if" não é satisfeita, ou seja, quando a idade é menor que 18.
- `print("Você é menor de idade.")`
 - Exibe na tela a mensagem "Você é menor de idade.", caso o "if" não tenha sido verdadeiro.

Resolução exercício 2

2 –

```
1  nota1 = float(input("Primeira nota: "))
2  nota2 = float(input("Segunda nota: "))
3  media = (nota1 + nota2) / 2
4
5  if media >= 7:
6      print("Aprovado!")
7  elif media >= 5:
8      print("Recuperação!")
9  else:
10     print("Reprovado!")
```

- `nota1 = float(input("Primeira nota: "))`
 - O programa pede que o usuário digite a primeira nota. `input()` lê o que foi digitado como texto. `float()` converte esse texto para um número decimal, permitindo cálculos. O valor é armazenado na variável `nota1`.
- `nota2 = float(input("Segunda nota: "))`
 - Faz a mesma coisa da linha anterior, mas agora para a segunda nota, que é guardada em `nota2`.
- `media = (nota1 + nota2) / 2`
 - Calcula a média aritmética das duas notas. Soma as duas (`nota1 + nota2`) e divide por 2. O resultado é armazenado na variável `media`.
- `if media >= 7:`
 - `print("Aprovado!")`
 - O `if` verifica se a média é maior ou igual a 7. Se for verdade, o programa mostra "Aprovado!". Nesse caso, as outras condições não são verificadas (o programa sai do bloco condicional).
- `elif media >= 5:`
 - `print("Recuperação!")`
 - O `elif` (abreviação de "else if") é verificado somente se o `if` for falso. Aqui ele testa se a média é maior ou igual a 5. Se for verdade, o programa mostra "Recuperação!".
- `else:`
 - `print("Reprovado!")`
 - O `else` é o caso final — ele será executado se nenhuma das condições anteriores for verdadeira, ou seja, quando a média for menor que 5.

Nesse caso, o programa mostra "Reprovado!".

Capítulo 07

Estruturas de Repetição


- While
- For
- Interrompendo laços
- Exemplos práticos

REPEATING STRUCTURES

for	while
variable	condition
in iterable:	:
:	statement
statement	

While

Repete enquanto a condição for verdadeira:

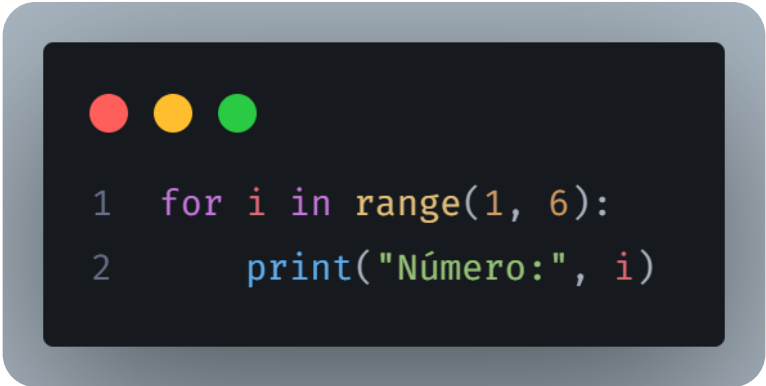
A screenshot of a terminal window with a dark background and light blue text. The code is as follows:

```
1 contador = 1
2 while contador <= 5:
3     print("Contando:", contador)
4     contador += 1
```

- `contador = 1`
 - Aqui é criada a variável `contador` e ela começa com o valor 1. Essa variável servirá para controlar o loop, ou seja, contar quantas vezes o programa repetirá o bloco.
- `while contador <= 5:`
 - O comando `while` cria um laço de repetição (loop). Ele diz: "Repita tudo que está dentro enquanto `contador` for menor ou igual a 5." Assim, o bloco dentro do `while` continuará rodando até que essa condição deixe de ser verdadeira.
- `print("Contando:", contador)`
 - Essa linha exibe na tela a mensagem "Contando:" seguida do valor atual da variável `contador`. Por exemplo, na primeira vez mostrará: Contando: 1, depois Contando: 2, e assim por diante.
- `contador += 1`
 - Essa linha soma 1 ao valor de `contador` a cada repetição. É o mesmo que escrever `contador = contador + 1`. Isso faz o loop avançar e impede que ele fique infinito.

For

Usado para percorrer um intervalo ou uma sequência:

A terminal window with a dark background and a light gray border. At the top left, there are three colored circles: red, yellow, and green. Below them, the following Python code is displayed:

```
1  for i in range(1, 6):  
2      print("Número:", i)
```

A função `range(início, fim)` gera uma sequência de números.

Exemplo: `range(1, 6)` → 1, 2, 3, 4, 5

Interrompendo laços

- break: sai do laço
- continue: pula para a próxima repetição



```
1 for i in range(1, 10):  
2     if i == 5:  
3         break  
4     print(i)
```

Quando "i" for igual a 5, o laço irá quebrar e parará de ser executado.




```
1 for i in range(1, 6):  
2     if i == 3:  
3         continue  
4     print(i)
```

Quando "i" for igual a 3, o laço irá pular e a variável "i" não será exibida.

Exemplos práticos

1. Contador simples:



```
1 for i in range(1, 6):  
2     print("Contando:", i)
```

2. Tabuada:



```
1 num = int(input("Digite um número: "))  
2 for i in range(1, 11):  
3     print(f"{num} x {i} = {num * i}")
```

3. Somador:



```
1 soma = 0  
2 for i in range(3):  
3     numero = float(input("Digite um número: "))  
4     soma += numero  
5 print("Soma total:", soma)
```