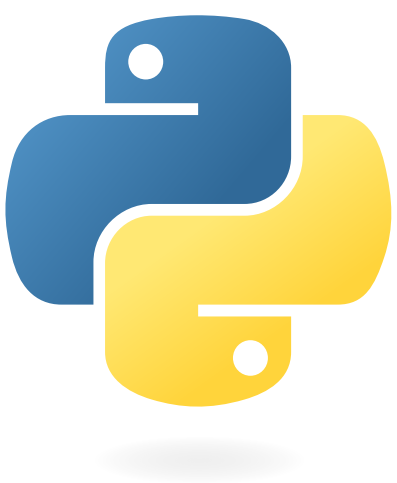
Introdução à Programação em



python™

https://evandro-crr.github.io/intro-python

Evandro Chagas Ribeiro da Rosa

Operadores de Comparação

- `==` : Igual a
- `!=` : Diferente de
- `>` : Maior que
- `>=` : Maior ou igual a
- `<` : Menor que
- `<=` : Menor ou igual a

O resultado de uma comparação é sempre `True` ou `False`.

```
>>> a = 10
>>> b = 12
>>> print(a == b)
False
>>> print(a != b)
True
>>> print(a > b)
False
>>> print(a <= b)
True
```

Variáveis Booleanas

Variáveis que armazenam valores `True` ou `False`.

```
>>> print(type(a <= b))
<class 'bool'>
>>>
>>>
```

```
>>> numero = 15
>>> valor = numero >= 10 and numero <= 20
>>> print(valor)
True
```

Operadores Lógicos

- `and` – Retorna `True` se **ambos** forem `True`.
- `or` – Retorna `True` se **ao menos um** for `True`.
- XOR `^` – Retorna `True` se **apenas um** for `True`.
- `not` – Inverte o valor lógico.

⚙ Exercício

Monte a tabela verdade dos operadores lógicos.

Exemplo: Tabela do operador `and`

a	b	a · b
False	False	False
False	True	False
True	False	False
True	True	True

Operadores Bitwise

Operadores que trabalham diretamente sobre os bits de números inteiros.

- `&` : AND bit a bit
- `|` : OR bit a bit
- `^` : XOR bit a bit

Conversão de Binário

```
>>> int('10111', 2) # str para int
23
>>> bin(23)         # int para str
'0b10111'
```

```
>>> a = 0b1010  # 10 em decimal
>>> b = 0b1100  # 12 em decimal
>>> print(a & b)  # 0b1000
8
>>> print(a | b)  # 0b1110
14
>>> print(a ^ b)  # 0b0110
6
```

Funções `any` e `all`

- `any(iterável)`
  Retorna `True` se **qualquer** elemento for `True`.
- `all(iterável)`
  Retorna `True` se **todos** os elementos forem `True`.

```
>>> lista = [True, False, True]
>>> print(any(lista))
True
>>> print(all(lista))
False
```

```
>>> pares = list(range(0, 10, 2))
>>> todos_pares = all(i % 2 == 0 for i in pares)
>>> print(todos_pares)
True
```

Desvio Condicional

Permite executar diferentes blocos de código dependendo de condições booleanas.

Instrução `if`

```
idade = 18
if idade >= 18:
    print("Maior de idade")
```

O bloco `if` é executado apenas se a condição for verdadeira.

Instrução `if-else`

```
idade = 16
if idade >= 18:
    print("Maior de idade")
else:
    print("Menor de idade")
```

O `else` é executado quando o `if` não é satisfeito.

Instrução `if-elif-else`

```
nota = 75
if nota >= 90:
    print("A")
elif nota >= 80:
    print("B")
elif nota >= 70:
    print("C")
else:
    print("D")
```

`elif` permite testar múltiplas condições em sequência, até que uma seja verdadeira.

Operador Ternário

Uma forma compacta de escrever um `if-else`.

```
>>> idade = 18
>>> status = "Maior de idade" if idade >= 18 else "Menor de idade"
>>> print(status)
Maior de idade
```

```
>>> notas = [5.5, 8.0, 6.5, 9.0, 4.0]
>>> status = ["Aprovado" if nota >= 7 else "Reprovado" for nota in notas]
>>> print(status)
['Reprovado', 'Aprovado', 'Reprovado', 'Aprovado', 'Reprovado']
```

⚙ Exercício

Objetivo: Simular o gerenciamento de notas e gerar um relatório.

1. Crie duas listas: `nomes` e `notas`. Use `input()` para adicionar pelo menos 3 alunos.
2. Crie uma nova lista que classifique cada aluno como:
   - **Aprovado:** Nota ≥ 7
   - **Recuperação:** 3 ≤ Nota < 7
   - **Reprovado:** Nota < 3
3. Exiba a situação dos alunos. Resuma se: todos estão aprovados; há alunos em recuperação.

Exemplo de Saída:

```
Nome: João, Situação: Aprovado
Nome: Maria, Situação: Recuperação
Nome: Pedro, Situação: Reprovado

Resumo da Turma:
Todos estão aprovados? Não
Há pelo menos um aluno em recuperação? Sim
```

Laço `while`

Um laço que continua a executar enquanto uma condição for verdadeira.

```
i = 0
while i < 5:
    print(i)
    i += 1
```

O `while` repete a execução enquanto `i` for menor que 5.

Instrução `break` e `continue`

Podem ser usadas dentro de um `for` ou `while`.

`break`

Interrompe o laço imediatamente.

```
while True:
    mensagem = input()
    if mensagem == "sair":
      break
    print(mensagem)
```

`continue`

Pula para a próxima iteração do laço.

```
i = 0
while i < 10:
    i += 1
    if i % 2 == 0:
        continue
    print(i)
```

⚙️ Exercício 1/2

- Crie um programa que solicita ao usuário que adivinhe um número secreto entre 1 e 20.
- O programa deve dar dicas se o número é maior ou menor que o número secreto.
- O jogo termina quando o usuário acerta o número ou escolhe desistir.

Dicas:

- Use um loop `while` para permitir múltiplas tentativas.
- Use um `if-elif-else` para fornecer feedback ao usuário.

⚙ Exercício 2/2

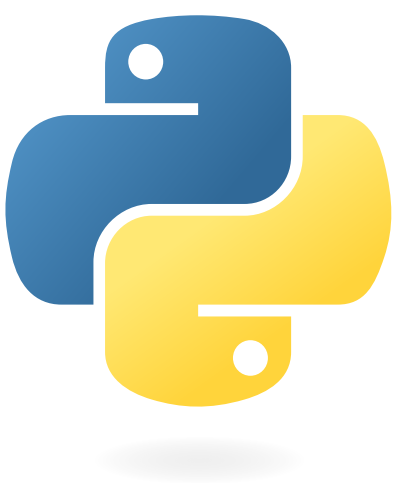
- Crie um programa que verifica se uma senha atende aos seguintes critérios :
  - Pelo menos 8 caracteres.
  - Contém pelo menos uma letra maiúscula. `str.isupper()`
  - Contém pelo menos um número. `str.isdigit()`
  - Contém pelo menos um caractere especial. `str.isalnum()`
- O usuário pode tentar novamente até inserir uma senha válida.

Dicas:

- Utilize as funções `any` para validar a senha.
- É possível separar os caracteres de uma string em uma lista:

```
>>> list("minha_senha")
['m', 'i', 'n', 'h', 'a', '_', 's', 'e', 'n', 'h', 'a']
```

Introdução à Programação em



python™

https://evandro-crr.github.io/intro-python

Evandro Chagas Ribeiro da Rosa