



escola
britânica de
artes criativas
& tecnologia

Desenvolvedor Full Stack Python

Fluxo Condicional e Repetição

Módulo | Python: Fluxo Condicional & Repetição

Caderno de Aula

Professor [André Perez](#)

Tópicos

1. Estrutura condicional if / else / elif;
 2. Estrutura condicional try / catch / finally;
 3. Estrutura de repetição for / in.
-

Aulas

1. Estrutura condicional if / else / elif

1.1. if / else

Estrutura de alteração de fluxo lógico do código, avalia um valor booleano ou uma comparação lógica. **Note** a indentação do código.

```
if <booleano / comparação lógica> == True:
    <execute este código>
else:
    <senão execute este código>
```

```
In [ ]: if True:
        print("Verdadeiro")
        else:
        print("Falso")
```

Exemplo: Código de segurança de um cartão de crédito

```
In [ ]: codigo_de_seguranca = '291'
        codigo_de_seguranca_cadastro = '010'
```

```
pode_efetuar_pagamento = codigo_de_seguranca == codigo_de_seguranca_cadastro
print(pode_efetuar_pagamento)
```

```
In [ ]: if pode_efetuar_pagamento:
        print("Pagamento efetuado")
        else:
        print("Erro: Código de segurança inválido")
```

```
In [ ]: if codigo_de_seguranca == codigo_de_seguranca_cadastro:
        print("Pagamento efetuado")
        else:
        print("Erro: Código de segurança inválido")
```

Exemplo: Código e senha de segurança de um cartão de crédito

```
In [ ]: codigo_de_seguranca = '852'
        codigo_de_seguranca_cadastro = '852'

        senha = '7783'
        senha_cadastro = '7783'
```

Revisitando a tabela da verdade:

CÓDIGO	SENHA	CÓDIGO OR SENHA	CÓDIGO AND SENHA	NOT CÓDIGO
TRUE	TRUE	TRUE	TRUE	FALSE
TRUE	FALSE	TRUE	FALSE	FALSE
FALSE	FALSE	FALSE	FALSE	TRUE
FALSE	TRUE	TRUE	FALSE	TRUE

```
In [ ]: if (codigo_de_seguranca == codigo_de_seguranca_cadastro) & \
        (senha == senha_cadastro):
        print("Pagamento efetuado")
        else:
        print("Erro: Pagamento não efetuado")
```

```
In [ ]: if (codigo_de_seguranca != codigo_de_seguranca_cadastro) | \
        (senha != senha_cadastro):
        print("Erro: Pagamento não efetuado")
        else:
        print("Pagamento efetuado")
```

1.2. if / elif / else

Podemos também avaliar múltipla condições.

```
if <1º booleano / 1ª comparação lógica> == True:
    <execute este código se a primeira condição for verdade>
elif <2º booleano / 2ª comparação lógica> == True:
    <execute este código se a segunda condição for verdade>
else:
    <senão execute este código>
```

```
In [ ]: codigo_de_seguranca = '802'
        codigo_de_seguranca_cadastro = '852'
```

```
senha = '7703'
senha_cadastro = '7783'
```

CÓDIGO	SENHA	CÓDIGO AND SENHA	MENSAGEM
TRUE	TRUE	TRUE	Pagamento efetuado
TRUE	FALSE	FALSE	Erro: Senha inválida
FALSE	FALSE	FALSE	Erro: Código de segurança e senha inválidos
FALSE	TRUE	FALSE	Erro: Código de segurança inválido

```
In [ ]: if (codigo_de_seguranca == codigo_de_seguranca_cadastro) & \
        (senha == senha_cadastro):
        print("Pagamento efetuado")

        elif (codigo_de_seguranca != codigo_de_seguranca_cadastro) & \
        (senha == senha_cadastro):
        print("Erro: Código de segurança inválido")

        elif (codigo_de_seguranca == codigo_de_seguranca_cadastro) & \
        (senha != senha_cadastro):
        print("Erro: Senha inválida inválida")

        else:
        print("Erro: Código de segurança e senha inválidos")
```

2. Estrutura condicional try / except / finally

2.1. Exceção

Exceções são erros que podem acontecer durante a execução do nosso código.

Exemplo: Erro de operações numéricas impossíveis

```
In [ ]: preco = 132.85
        pessoas = 0
```

```
In [ ]: valor_por_pessoa = preco / pessoas
```

Exemplo: Erro por combinações de tipos diferentes

```
In [ ]: nome = 'Andre Perez'
        idade = True
```

```
In [ ]: apresentacao = 'Fala pessoal, meu nome é ' + nome + \
                      ' e eu tenho ' + idade + ' anos'
```

Exemplo: Erro de indexação de estrutura de dados

```
In [ ]: anos = [2019, 2020, 2021]
```

```
In [ ]: ano_atual = anos[3]
```

```
In [ ]: cursos = {
        'python': {
            'nome': 'Python para Análise de Dados', 'duracao': 2.5
        },
        'sql': {
            'nome': 'SQL para Análise de Dados', 'duracao': 2
        }
    }
```

```
In [ ]: curso_atual = cursos['analista']
```

2.2. try / except

Estrutura para tratar exceções:

```
In [ ]: preco = 132.85
        pessoas = 2

        try:
            valor_por_pessoa = preco / pessoas
            print(valor_por_pessoa)
        except ZeroDivisionError:
            print('Número de pessoas inválido. ' + \
                  'Espera-se um valor maior que 0 e obteve-se um valor igual a ' + \
                  str(pessoas)
            )
```

```
In [ ]: anos = [2019, 2020, 2021]

        try:
            ano_atual = anos[3]
            print(ano_atual)
        except Exception:
            print('Lista de anos é menor que o valor escolhido. ' + \
                  'Espera-se um valor entre 0 e ' + \
                  str(len(anos) - 1)
            )
```

```
In [ ]: anos = [2019, 2020, 2021]

        try:
            ano_atual = anos[3]
            print(ano_atual)
        except Exception as exc:
            print('Descrição da exceção: ' + str(exc))
            print('Tipo da exceção: ' + str(type(exc)))
            print('Lista de anos é menor que o valor escolhido. ' + \
                  'Espera-se um valor entre 0 e ' + \
                  str(len(anos) - 1)
            )
```

```
In [ ]: anos = [2019, 2020, 2021]

        try:
            ano_atual = anos[3]
            print(ano_atual)
        except IndexError:
```

```

    print('Lista de anos é menor que o valor escolhido. ' + \
          'Espera-se um valor entre 0 e ' + \
          str(len(anos) - 1)
        )
except Exception as exc:
    print(exc)
    print('Erro genérico')

```

2.3. try / except / finally

```

In [ ]: nome = 'Andre Perez'
        idade = 19

        try:
            apresentacao = 'Fala pessoal, meu nome é ' + nome + \
                            ' e eu tenho ' + idade + ' anos'
            print(apresentacao)
        except TypeError:
            idade = str(idade)
        finally:
            print('Segunda chance')
            apresentacao = 'Fala pessoal, meu nome é ' + nome + \
                            ' e eu tenho ' + idade + ' anos'
            print(apresentacao)

```

3. Estrutura repetição for / in

3.1. for / in

Estrutura que permite a execução repetida de um bloco de código repetidas vezes.

```

for variavel_temporaria in coleção:
    <execute este código>

```

3.2. for / in / range

Estrutura que permite a execução repetida de um bloco de código **n** vezes.

```

In [ ]: for valor in range(6):
        print(valor)

```

```

In [ ]: soma = 0

        for valor in range(0, 1000000):
            soma = soma + valor
            # print(soma)

        print(soma)

```

```

In [ ]: for multiplo_dois in range(2, 10, 3):
        print(multiplo_dois)

```

3.3. for / in / list

Estrutura que permite a execução de um bloco de código para todos os elementos de uma lista.

```
In [ ]: frutas = ['maca', 'banana', 'laranja', 'uva', 'pera']

for fruta in frutas:
    print(fruta)
```

```
In [ ]: frase = 'Fala pessoal, meu nome é André Perez.'

for caracter in frase:
    if (caracter == 'A') | (caracter == 'z'):
        print(f"A letra '{caracter}' está presente na frase.")
```

3.4. for / in / dict

Estrutura que permite a execução de um bloco de código para todos os elementos de um dicionário.

```
In [ ]: credito = {'123': 750, '456': 812, '789': 980}
```

```
In [ ]: for chave, valor in credito.items():
    print(f'Para o documento {chave}, ' + \
          'o valor do escore de crédito é {valor}.')
    print('\n')
```

```
In [ ]: for chave in credito.keys():
    print(chave)
    print(credito[chave])
    print(f'Para o documento {chave}, ' + \
          'o valor do escore de crédito é {credito[chave]}.'.')
    print('\n')
```

```
In [ ]: for valor in credito.values():
    print(valor)
    print(f'O valor do escore de crédito é {valor}, ' + \
          'mas não temos mais as chaves :(.')
    print('\n')
```

3.5. break / continue

Estrutura que permite a quebra ou o avanço de um laço de repetição.

```
In [ ]: for i in range(0, 10*10*10*10*10*10):
    print(i)
    if i == 10:
        break
```

```
In [ ]: numero = 3

if numero % 2 == 0:
    print(f'O numero {numero} é par')
else:
    print(f'O numero {numero} é impar')
```

In []:

```
numeros = [361, 553, 194, 13, 510, 33, 135]

for numero in numeros:

    if numero % 2 == 0:
        print(f'O numero {numero} é par')
        break
    else:
        print(f'O numero {numero} é impar')
```

In []:

```
numeros = [361, 553, 194, 13, 510, 33, 135]

for numero in numeros:

    if numero % 2 == 0:
        print(f'O numero {numero} é par')
        break
    else:
        continue
        print(f'O numero {numero} é impar')
```