

# Linguagem de Programação Python

## Condicionais aninhadas

Professor: Ritomar Torquato

1

# Condicionais aninhadas

Objetivo: Demonstrar o uso de comandos condicionais aninhados para obter o comportamento desejado por um programa.

2

A loja de departamentos **MEGALOJA** paga aos colaboradores um percentual de comissão baseado na venda loja.

Vejamos o plano de pagamento de comissões da empresa MEGALOJA:

Total vendido abaixo de **200 mil** reais:  
comissão de 0,15%

Total vendido de **200 mil a 400 mil** reais:  
comissão de 0,18%

Total vendido **acima de 400 mil** reais:  
comissão de 0,20%

**Vejamos uma função em Python para calcular a comissão:**

3

```
def calcula_comissao(valor_vendido):
    if valor_vendido < 200000:
        comissao = 0.0015
    else:
        if valor_vendido < 400000:
            comissao = 0.0018
        else:
            comissao = 0.0020
    return valor_vendido * comissao
```

Observe o bloco com um comando `if` dentro do `else` da linha anterior. Dizemos que o comando está aninhado entro de `else`. Lembre-se que o alinhamento (**a indentação**) do código é importante

É comum aninhar vários `if` para obter um comportamento.

4

Vejamos, agora, uma situação em que cinco categorias são necessárias:

Categoria	Preço
1	10,00
2	18,00
3	23,00
4	26,00
5	31,00

5


O alinhamento pode se tornar um problema com o vários níveis.

```
def preco_por_categoria_(categoria):
    if categoria == 1:
        preco = 10
    else:
        if categoria == 2:
            preco = 18
        else:
            if categoria == 3:
                preco = 23
            else:
                if categoria == 4:
                    preco = 26
                else:
                    if categoria == 5:
                        preco = 31
                    else:
                        raise ValueError(f'categoria inválida: {categoria}')
    return preco
```

6

Antes que perguntem....

... o que é isso?



```
raise ValueError(f'categoria inválida: {categoria}')
```

A função foi chamada com um valor inválido para `categoria`.  
Programadores inexperientes poderiam pensar em responder com uma mensagem ou `print()`.

A forma profissional de retornar um erro é **lançando uma exceção**. É isso que estamos fazendo:


Retornando um erro do tipo `ValueError` com a mensagem `'Categoria Inválida'`. Veja mais no link:

<https://docs.python.org/pt-br/3/tutorial/errors.html>

7

Antes que perguntem....

... o que é isso?



```
raise ValueError(f'categoria inválida: {categoria}')
```

Por exemplo, se a função `int()` for chamada com um valor inválido ela também retorna um `ValueError`.

```
>>> int('amor')
Traceback (most recent call last):
  File "<pyshell#0>", line 1, in <module>
    int('amor')
ValueError: invalid literal for int() with base 10: 'amor'
```

Retornando um erro do tipo `ValueError` com a mensagem `'invalid literal for...'`. Veja mais no link:

<https://docs.python.org/pt-br/3/tutorial/errors.html>


8

```
def preco_por_categoria(categoria):
    if categoria == 1:
        preco = 10
    elif categoria == 2:
        preco = 18
    elif categoria == 3:
        preco = 23
    elif categoria == 4:
        preco = 26
    elif categoria == 5:
        preco = 31
    else:
        raise ValueError('Categoria Inválida')
    return preco
```

O mesmo problema do preço por categoria usando `elif`

9

## Referência



10