Lançamento de Exceções

Alessandro Botelho Bovo

Função `preco_unitario` — Usando o `print`

```
def preco_unitario(total, qtd):
    if total < 0 or qtd < 0:
        print('Erro: valores negativos não são permitidos.')
        return None
    if qtd == 0:
        print('Erro: a quantidade não pode ser zero.')
        return None
    return total / qtd

try:
    total = float(input('Valor total (R$): '))
    qtd   = int(input('Quantidade (itens): '))
    pu = preco_unitario(total, qtd)
    if pu is not None:
        print(f'Preço unitário: R$ {pu:.2f}')
except ValueError:
    print('Erro: entradas numéricas inválidas.')
```

Função `preco_unitario` — Lançando `ValueError`

```
def preco_unitario(valor_total, quantidade):
    if valor_total < 0 or quantidade < 0:
        raise ValueError('Valores negativos não são permitidos.')
    if quantidade == 0:
        raise ZeroDivisionError('A quantidade não pode ser zero.')
    return valor_total / quantidade

try:
    total = float(input('Valor total (R$): '))
    qtd   = int(input('Quantidade (itens): '))
    pu = preco_unitario(total, qtd)
    print(f'Preço unitário: R$ {pu:.2f}')
except ValueError as e:          # negativos ou conversão de entrada
    print(f'Erro: {e}')
except ZeroDivisionError as e:  # divisor zero
    print(f'Erro: {e}')
```

Comparando as duas versões de `preco_unitario`

```python
def preco_unitario(total, qtd):
    if total < 0 or qtd < 0:
        print('Erro: valores negativos não são permitidos.')
        return None
    if qtd == 0:
        print('Erro: a quantidade não pode ser zero.')
        return None
    return total / qtd
```

```python
def preco_unitario(valor_total, quantidade):
    if valor_total < 0 or quantidade < 0:
        raise ValueError('Valores negativos não são permitidos.')
    if quantidade == 0:
        raise ZeroDivisionError('A quantidade não pode ser zero.')
    return valor_total / quantidade
```

Função `preco_unitario` — Exceção personalizada

```
class ValoresNegativosError(ValueError):
    '''Erro de domínio: números negativos não são aceitos.'''
    pass

def preco_unitario(valor_total, quantidade):
    if valor_total < 0 or quantidade < 0:
        raise ValoresNegativosError('Valores negativos não são permitidos.')
    if quantidade == 0:
        raise ZeroDivisionError('A quantidade não pode ser zero.')
    return valor_total / quantidade


# A ordem importa: ValoresNegativosError deve vir antes do ValueError genérico
try:
    total = float(input('Valor total (R$): '))
    qtd   = int(input('Quantidade (itens): '))
    pu = preco_unitario(total, qtd)
    print(f'Preço unitário: R$ {pu:.2f}')
except ValoresNegativosError as e:
    print(f'Domínio inválido: {e}')
except ZeroDivisionError as e:
    print(f'Erro matemático: {e}')
except ValueError:
    print('Erro: entradas numéricas inválidas.')
```

Comparando as três versões de `preco_unitario`

```
def preco_unitario(total, qtd):
    if total < 0 or qtd < 0:
        print('Erro: valores negativos não são permitidos.')
        return None
    if qtd == 0:
        print('Erro: a quantidade não pode ser zero.')
        return None
    return total / qtd
```

```
def preco_unitario(valor_total, quantidade):
    if valor_total < 0 or quantidade < 0:
        raise ValueError('Valores negativos não são permitidos.')
    if quantidade == 0:
        raise ZeroDivisionError('A quantidade não pode ser zero.')
    return valor_total / quantidade
```

```
def preco_unitario(valor_total, quantidade):
    if valor_total < 0 or quantidade < 0:
        raise ValoresNegativosError('Valores negativos não são permitidos.')
    if quantidade == 0:
        raise ZeroDivisionError('A quantidade não pode ser zero.')
    return valor_total / quantidade
```

Contato

Alessandro Botelho Bovo

alessandrobovo@utfpr.edu.br