Estruturas de Controle

Nem sempre todas as linhas dos programas serão executadas. Muitas vezes, será mais interessante decidir que partes do programa devem ser executadas com base no resultado de uma condição. A base dessas decisões consistirá em expressões lógicas que permitam representar escolhas em programas.

Instrução if

Em Python, a estrutura de decisão é o `if`. Seu formato é apresentado a seguir:

```
if <condição>:
    bloco verdadeiro
```

Exemplo 1

```
# Lê dois valores e imprime qual é o valor maior
a = int(input("Primeiro valor: "))
b = int(input("Segundo valor: "))
```

```
if a>b:
    print("O primeiro valor é o maior")
if b>a:
    print("O segundo valor é o maior")
```

💡 **Python** é uma das poucas linguagens de programação que utiliza o deslocamento do texto à direita (**recuo**) para marcar o início e o fim de um bloco. Outras linguagens contam com palavras especiais para isso, como `BEGIN` e `END`, em Pascal; ou as famosas chaves (`{` e `}`), em C e Java.

Exemplo 2

Um problema comum é quando temos de pagar Imposto de Renda. Normalmente, pagamos o Imposto de Renda por faixa de salário. Imagine que, para salários menores que **R$ 1.000,00**, não teríamos imposto a pagar, ou seja, alíquota 0%. Para salários entre **R$ 1.000,00 e R$ 3.000,00**, pagaríamos 20%. **Acima** desses valores, a alíquota seria de 35%. Quem ganha R$ 4.000,00 tem os primeiros R$ 1.000,00 isentos de imposto; com o montante entre R$1.000,00 e R$ 3.000,00 pagando 20%, e o restante pagando os 35%.

```
#Cálculo do Imposto de Renda
salario = float(input("Digite o salário para cálculo do impos
base = salario
imposto = 0
if base > 3000:
    imposto = imposto + ((base-3000)*0.35)
    base = 3000
if base>1000:
    imposto = imposto + ((base-1000)*0.20)
print(f"Salário: R${salario:6.2f} Imposto a pagar: R${imposto
```

```
print("Salário: R$%6.2f Imposto a pagar: R$%6.2f" %(salario,
print("Salário: R${} Imposto a pagar: R${}".format(salario, i
```

Exemplo 3

```
#Categoriaxpreço
categoria = int(input("Digite a categoria do produto: "))
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:
                    print("Categoria inválida, digite um valo
                    preco=0
print(f"O preço do produto é: R${preco:6.2f}")
```

Usando a solução `elif` :

```
#Categoriaxpreço
categoria = int(input("Digite a categoria do produto: "))
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:
    print("Categoria inválida, digite um valor entre 1 e 5!")
    preco=0
print(f"O preço do produto é: R${preco:6.2f}")
```

Instrução Match

No Python 3.10 e posteriores, foi introduzido um novo recurso chamado `match`. Ele é uma nova estrutura de controle que foi adicionada para simplificar o código quando você precisa fazer várias comparações em uma expressão.

```
def operacao(a, b, operador):
    match operador:
        case '+':
            resultado = a + b
            print(f'A soma de {a} e {b} é {resultado}')
        case '-':
            resultado = a - b
            print(f'A subtração de {a} e {b} é {resultado}')
        case '*':
            resultado = a * b
            print(f'O produto de {a} e {b} é {resultado}')
        case '/':
            resultado = a / b
            print(f'A divisão de {a} por {b} é {resultado}')
        case _:
            print('Operador inválido')

operacao(5, 3, '+')
operacao(5, 3, '*')
```

```
operacao(5, 3, '/')
operacao(5, 3, '%')
```

Instrução While

Uma das estruturas de repetição do Python é o `while`, que repete um bloco enquanto a condição for verdadeira.

```
while<condição>:
    bloco
```

Exemplo 4

```
#contagem regressiva
int i=10
while(i>=1):
    print(i)
    i = i-1
print("Fogo!")
```

💡 A instrução `break` é utilizada para interromper a execução de `while` independentemente do valor atual de sua condição.

Operadores de atribuição especiais

Expressão	Forma Compacta
x = x + y	x += y
x =x - y	x -=y
x = x * y	x *=y
x = x / y	x /=y
x = x%y	x %=y

Instrução FOR

Existe um outro *loop* que simplifica essa ideia de começar com um valor e incrementá-lo até chegar em um valor final: o *loop* **for**.

```
for rodada in range(1,10):
    print(rodada)

for rodada in range(1,10,2):
    print(rodada)

for rodada in [1,2,3,4,5]:
    print(rodada)
```

A função `range(m, n, p)`, é muito útil em laços, pois retorna uma lista de inteiros, começando em *m* e menores que *n*, em passos de comprimento *p*, que podem ser usados como sequência para o laço. `range` não é exatamente uma lista.