CAPÍTULO 5

Antes de estruturas condicionais precisamos ter conhecimento sobre mais alguns operadores. Como visto no capítulo 1, ao dividirmos um inteiro por um outro inteiro, obtemos como resposta um número inteiro, independente da divisão ser exata ou não. Para obtermos a divisão exata, é necessário que os dois números sejam do tipo float. Porém, em alguns casos pode ser interessante obtermos somente a parte inteira da divisão ou somente o resto, independente de que tipo são o dividendo e o divisor.

O operador floor division, representado por //, retorna a parte inteira da divisão enquanto o operador módulo, representado por %, retorna o resto. Exemplificando.

```
>>>quociente = 11 // 3
>>>print(quociente)
3

>>>resto = 11%3
>>>print(resto)
2
```

Existem também os operadores de comparação:

x == y, verifica se x é igual a y
x != y, verifica se x é diferente de y
x > y, verifica se x é maior que y
x < y, verifica se x é menor que y
x >= y, verifica se x é maior ou igual a y
x <= y, verifica se x é menor ou igual a y

Sendo esses operadores utilizados para dar significado a uma expressão booleana, que é do tipo verdadeiro ou falso, sendo que em python o número 0 é equivalente a falso e qualquer número diferente de 0 é equivalente a verdadeiro.

Além disso, também temos os operadores lógicos and, or e not. Sendo x e y expressões booleanas, temos:

x and y retorna true somente se ambas as expressões forem verdadeiras.
x or y retorna true se pelo menos uma das expressões forem verdadeiras.
not x retorna true se a expressão x for falsa.

Isto posto, podemos analisar as estruturas condicionais, isto é, determinadas linhas do código só serão executadas caso uma condição inicial seja satisfeita.

```
if (condição):
        código
```

Onde o código deve conter no mínimo uma linha, nem que seja o comando pass, que não faz nada. O condicional if tanto pode ser utilizado sozinho, como pode ser acompanhado por um else, que executa o código caso a condição do if não tenha sido satisfeita, ou de um ou mais elif, que executa o código caso a condição do if não tenha sido satisfeita e a condição do elif tenha sido.

```
if (nota_rec >= 7):
        print('Aluno esta aprovado')
elif(nota_rec >= 3):
```

```
        print(‘Aluno vai para prova oral’)
else:
        print(‘Aluno esta reprovado’)
```

Outro importante recurso é a recursão, onde uma função é chamada dentro dela mesma, de modo que ela seja executada várias vezes. É importante ressaltar que devemos garantir que essa recursão chegue ao fim, impedindo que o programa rode um número muito grande de vezes sem chegar em um resultado. Um exemplo clássico é a função fatorial, em python temos:

```
def fat(n):
        if(n==0 or n==1):
                return 1
        else:
                return n*fat(n-1)
```

Neste caso, para qualquer n negativo a função não terminaria, pois o n seria cada vez mais negativo e, portanto, nunca seria 1 ou 0, que são os valores que encerram a função. Logo, devemos garantir que n é um inteiro positivo antes de chamarmos a função fatorial.

Outro recurso da linguagem é a possibilidade de o usuário do programa fornecer valores a certas variáveis, aumentando a utilidade dos programas. A função que realiza essa ação é input(), onde atribuímos uma variável a ela, podendo ou não colocar uma mensagem dentro dos parênteses para mandar uma mensagem ao usuário.

```
>>>nome = input()
Mario, que mário?
>>>nome
Mario, que mário?

>>>nome = input(‘Qual seu nome?’)
Alberto
>>>nome
Alberto
```

Caso seja esperado que o usuário entre com um tipo específico de variável, podemos tentar converter para este tipo.

```
>>>text = ‘Qual a sua idade?’
>>>idade = input(text)
20
>>>int(idade)
20
```