CAPÍTULO 3

Números inteiros e flutuantes

3.1 Inteiros

Em Python, números inteiros são uma forma de dados numéricos que representam valores inteiros sem parte fracionária. Eles são usados para realizar operações matemáticas simples e complexas, além de serem armazenados em variáveis para posterior utilização no código.

Os números inteiros em Python podem ser positivos, negativos ou zero. Aqui estão alguns exemplos de números inteiros:

Por exemplo:

```
1 >>> x = 1
2 >>> y = -1
3 >>> z = 0
```

Neste exemplo, **x** é um número inteiro positivo, **y** é um número inteiro negativo e **z** é o número inteiro zero.

Os números inteiros em Python têm tamanho ilimitado, o que significa que você pode trabalhar com números inteiros muito grandes sem se preocupar com limitações de tamanho. Isso é diferente de algumas outras linguagens de programação que têm um limite definido para o tamanho dos números inteiros.

3.1.1 Função int()

Em Python, a função int() é uma função embutida que converte um valor para o tipo de dado inteiro. Ela pode ser usada para converter diferentes tipos de dados em números inteiros.

A sintaxe básica da função int() é a seguinte:

```
1 >>> int(x)
```

Neste exemplo, **x** representa o valor que você deseja converter em um número inteiro, que pode ser um número ou uma string.

A função int() pode ser usada de diferentes maneiras, dependendo do tipo de dado que você está convertendo. Aqui estão alguns exemplos:

Converter uma string em um número inteiro:

```
1 >>> numero_reads = int("1000")
2 >>> print(numero_reads) # Saída:  1000
```

Nesse exemplo, a função **int()** converte a *string* "1000"em um número inteiro 1000. Isso permite que você realize operações matemáticas com o valor convertido.

NOTA: É importante notar que, ao converter uma string em um número inteiro, a string deve conter apenas dígitos (0-9) e, opcionalmente, um sinal de positivo (+) ou negativo (-) no início.

Converter um número de ponto flutuante em um número inteiro:

```
1 >>> valor_pi = int(3.14)
2 >>> print(valor_pi)  # Saída:  3
```

No exemplo acima, a função **int()** converte o número de ponto flutuante 3.14 em um número inteiro 3. A conversão de um número de ponto flutuante para um número inteiro trunca a parte fracionária, descartando qualquer informação após o ponto decimal.

3.2 Números flutuantes ou decimais (float)

Os números flutuantes são representados pelo tipo de dado float e assim como os inteiros é possível converter strings e inteiros em números flutuantes com a função **float()**. Eles são usados para representar valores com casas decimais. Por exemplo:

```
>>> x = 1.5
>>> y = 0.0
>>> z = -1.5
>>> # convertendo string em número
>>> v = float("1.5")
>>> u = float("-1.5")
```

NOTA: Esteja sempre iente de que operações com números de ponto flutuante podem resultar em imprecisões devido à forma como os números são armazenados e manipulados internamente pelo computador. Portanto, pode ser necessário arredondar os resultados, se necessário, usando a funções ou formatar a saída de acordo com as necessidades do seu programa.

3.3 Operações com números

Você pode realizar diversas operações com números. Além das operações aritméticas básicas (adição, subtração, multiplicação e divisão), existem outras operações matemáticas disponíveis. Segue abaixo os operadores mais utilizados em operações matemáticas de números inteiros e flutuantes:

Soma:

```
>>> 3+2 #soma
5
```

Subtração:

```
>>> 7-3 #subtracao
4
```

Multiplicação:

```
>>> 7*2 #multiplicacao
14
```

Potenciação: utiliza o operador ** para elevar um número a uma potência.

```
>>> 5**2 #exponenciacao
25
```

Divisão:

```
>>> 8/4 #divisao
2
```

Divisão inteira: utiliza o operador // para realizar a divisão inteira, retornando apenas a parte inteira do resultado.

```
>>> 7//2 #divisao inteira
3
```

Resto da divisão (módulo): utiliza o operador % para obter o resto da divisão entre dois números.

```
>>> 15 % 4
3
```

Arrendondamento:

```
>>> round(1.3)
1
>>> round(1.5)
2
```

Potenciação:

```
>>> pow(2, 4)
16
```

Conversão pra números decimaisL

```
>>> float(2)
2.0
```

Valor absoluto:

```
>>> abs(-2)
2
```

3.3.1 Módulo math

O módulo *math* em Python fornece várias funções matemáticas. Aqui estão algumas das principais funções disponíveis:

Funções trigonométricas:

- math.sin(x): retorna o seno de x (em radianos).
- math.cos(x): retorna o cosseno de x (em radianos).
- math.tan(x): retorna a tangente de x (em radianos).

Funções exponenciais e logarítmicas:

- math.exp(x): retorna a exponencial de x (eˆx).
- math.log(x): retorna o logaritmo natural (base e) de x.
- math.log10(x): retorna o logaritmo de base 10 de x.
- math.log2(x): retorna o logaritmo de base 2 de x.
- math.pow(x, y): retorna x elevado à potência y.
- math.sqrt(x): retorna a raiz quadrada de x.

Funções de arredondamento:

- math.ceil(x): retorna o menor inteiro maior ou igual a x.
- math.floor(x): retorna o maior inteiro menor ou igual a x.
- math.trunc(x): retorna a parte inteira de x.
- round(x): arredonda x para o número inteiro mais próximo.

Constantes:

- math.pi: a constante matemática π (pi).
- math.e: a constante matemática e (base do logaritmo natural).

Essas são apenas algumas das funções disponíveis na biblioteca 'math' do Python. Existem outras funções e constantes matemáticas que você pode explorar consultando a documentação oficial do Python ou experimentando a biblioteca em seu código.