e APIs.Mas valores de bigint podem ter milhares ou até milhões de
Dígitos, se você precisa trabalhar com números tão grandes.(Observação,
No entanto, que as implementações do BIGINT não são adequadas para criptografia
Porque eles não tentam evitar ataques de tempo.)
Os literais bigint são escritos como uma série de dígitos seguidos por um minúsculo
Carta n.Por padrão, estão na base 10, mas você pode usar o 0b, 0o e
0x prefixos para bigints binários, octais e hexadecimais:
1234n // um literal não-so-big bigint
0B111111n // um bigint binário
0o7777n // um bigint octal
0x8000000000000000n // => 2n ** 63N: um número inteiro de 64 bits
Você pode usar o bigint () como uma função para converter regularmente
Números de JavaScript ou Strings para valores Bigint:
Bigint (número.max_safe_integer) // => 9007199254740991N
Seja String = "1" + "0" .Resepat (100);// 1 seguido por 100
zeros.
Bigint (string) // => 10n ** 100n: um
Googol
Aritmética com valores bigint funciona como aritmética com regular
Números de JavaScript, exceto que a divisão cai qualquer restante e
Recunda (em direção a zero):
1000n + 2000n // => 3000n
3000N - 2000N // => 1000N
2000n * 3000n // => 6000000n
3000N / 997N // => 3N: O quociente é 3
3000n % 997n // => 9n: e o restante é 9
(2n ** 131071n) - 1n // Um ??Mersenne Prime com 39457 decimal
dígitos