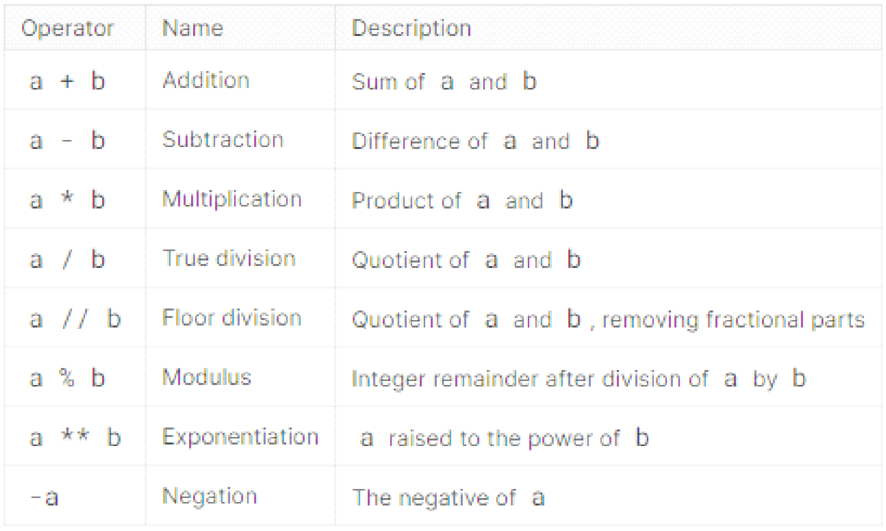
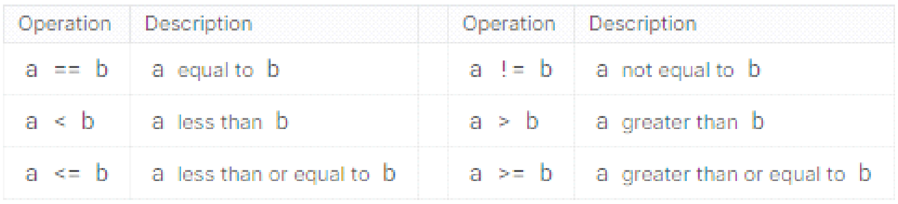
Python

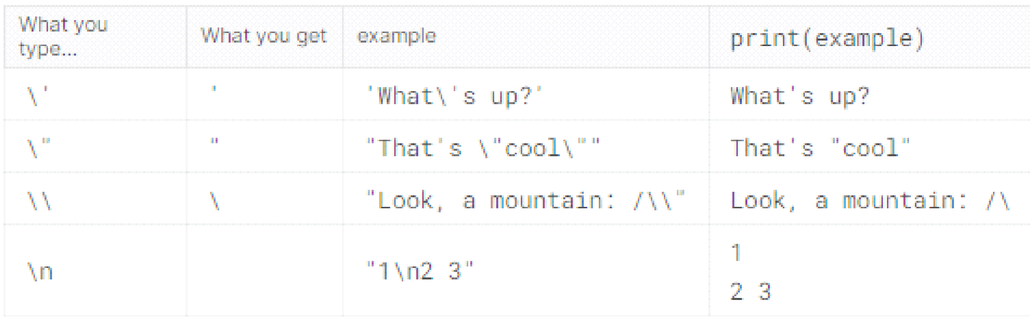
sábado, 20 de fevereiro de 2021 14:41



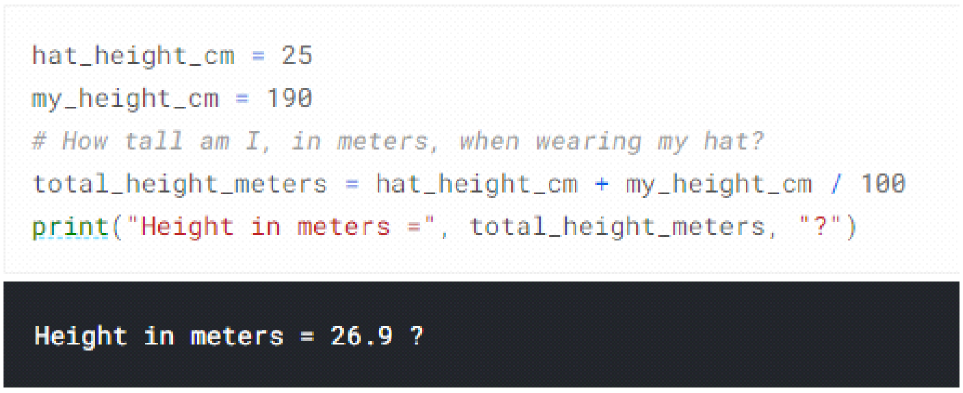
Operator	Name	Description
a + b	Addition	Sum of a and b
a - b	Subtraction	Difference of a and b
a * b	Multiplication	Product of a and b
a / b	True division	Quotient of a and b
a // b	Floor division	Quotient of a and b , removing fractional parts
a % b	Modulus	Integer remainder after division of a by b
a ** b	Exponentiation	a raised to the power of b
-a	Negation	The negative of a



Operation	Description	Operation	Description
a == b	a equal to b	a != b	a not equal to b
a < b	a less than b	a > b	a greater than b
a <= b	a less than or equal to b	a >= b	a greater than or equal to b



What you type...	What you get	example	print(example)
\'	'	'What\'s up?'	What's up?
\"	"	"That's \"cool\""	That's "cool"
\\	\	"Look, a mountain: /\\"	Look, a mountain: /\
\n		"1\n2 3"	1 2 3



```python
hat_height_cm = 25
my_height_cm = 190
# How tall am I, in meters, when wearing my hat?
total_height_meters = hat_height_cm + my_height_cm / 100
print("Height in meters =", total_height_meters, "?")
```

```
Height in meters = 26.9 ?
```

```python
print(min(1, 2, 3))
print(max(1, 2, 3))
```

```
1
3
```

```python
print(abs(32))
print(abs(-32))
```

```
32
32
```

```python
print(float(10))
print(int(3.33))
# They can even be called on strings!
print(int('807') + 1)
```

```
10.0
3
808
```

```python
print(1, 2, 3, sep=' < ')
```

```
1 < 2 < 3
```

```python
a = 1
b = 0
a, b = b, a
print(a, b)
```

```
0 1
```

Função

```python
def least_difference(a, b, c):
    diff1 = abs(a - b)
    diff2 = abs(b - c)
    diff3 = abs(a - c)
    return min(diff1, diff2, diff3)
```

```
print(
    least_difference(1, 10, 100),
    least_difference(1, 10, 10),
    least_difference(5, 6, 7), # Python allows trailing comma
s in argument lists. How nice is that?
)
```

```
9 0 1
```

```
def mult_by_five(x):
    return 5 * x

def call(fn, arg):
    """Call fn on arg"""
    return fn(arg)

def squared_call(fn, arg):
    """Call fn on the result of calling fn on arg"""
    return fn(fn(arg))

print(
    call(mult_by_five, 1),
    squared_call(mult_by_five, 1),
    sep='\n', # '\n' is the newline character - it starts a new line
)
```

```
5
25
```

```
def can_run_for_president(age, is_natural_born_citizen):
    """Can someone of the given age and citizenship status run for presid
ent in the US?"""
    # The US Constitution says you must be a natural born citizen *and* a
t least 35 years old
    return is_natural_born_citizen and (age >= 35)

print(can_run_for_president(19, True))
print(can_run_for_president(55, False))
print(can_run_for_president(55, True))
```

```
False
False
True
```

Booleano

```
print(bool(1)) # all numbers are treated as true, except 0
print(bool(0))
print(bool("asf")) # all strings are treated as true, except the empty st
ring ""
print(bool(""))
# Generally empty sequences (strings, lists, and other types we've yet to
see like lists and tuples)
# are "falsey" and the rest are "truthy"
```

```
True
False
True
False
```

Bibliotecas externas

- Math

```
import math

print("It's math! It has type {}".format(type(math)))
```

```
It's math! It has type <class 'module'>
```

```
print(dir(math))
```

```
['__doc__', '__file__', '__loader__', '__name__', '__package__', '__
spec__', 'acos', 'acosh', 'asin', 'asinh', 'atan', 'atan2', 'atanh',
'ceil', 'copysign', 'cos', 'cosh', 'degrees', 'e', 'erf', 'erfc', 'e
xp', 'expm1', 'fabs', 'factorial', 'floor', 'fmod', 'frexp', 'fsum',
'gamma', 'gcd', 'hypot', 'inf', 'isclose', 'isfinite', 'isinf', 'isn
an', 'ldexp', 'lgamma', 'log', 'log10', 'log1p', 'log2', 'modf', 'na
n', 'pi', 'pow', 'radians', 'remainder', 'sin', 'sinh', 'sqrt', 'ta
n', 'tanh', 'tau', 'trunc']
```

```
print("pi to 4 significant digits = {:.4}".format(math.pi))
```

```
pi to 4 significant digits = 3.142
```

```
math.log(32, 2)
```

```
5.0
```

- Numpy

```
import numpy
print("numpy.random is a", type(numpy.random))
print("it contains names such as...",
      dir(numpy.random)[-15:]
     )
```

```
numpy.random is a <class 'module'>
it contains names such as... ['seed', 'set_state', 'shuffle', 'stand
ard_cauchy', 'standard_exponential', 'standard_gamma', 'standard_nor
mal', 'standard_t', 'test', 'triangular', 'uniform', 'vonmises', 'wa
ld', 'weibull', 'zipf']
```

```
# Roll 10 dice
rolls = numpy.random.randint(low=1, high=6, size=10)
rolls
```

```
array([3, 1, 2, 1, 1, 3, 1, 5, 1, 1])
```

```
# Or maybe I just want to get back on familiar ground, in which case I mi
ght want to check out "tolist"
rolls.tolist()
```

```
[3, 1, 2, 1, 1, 3, 1, 5, 1, 1]
```