Numbers

`int`

In [ ]:
```python

```

In [ ]:
```python
my_int = 6
print('value: {}, type: {}'.format(my_int, type(my_int)))
```

`float`

In [ ]:
```python
my_float = float(my_int)
print('value: {}, type: {}'.format(my_float, type(my_float)))
```

Note that division of `int`s produces `float`:

In [ ]:
```python
print(1 / 1)
print(6 / 5)
```

Be aware of the binary floating-point pitfalls (see Decimal for workaround):

In [ ]:
```python
val = 0.1 + 0.1 + 0.1
print(val == 0.3)
print(val)
```

Floor division `//`, modulus `%`, power `**`

In [ ]:
```python
7 // 5
```

In [ ]:
```python
7 % 5
```

In [ ]:
```python
2 ** 3
```

`decimal.Decimal`

In [ ]:
```python
from decimal import Decimal
```

In [ ]:
```python
from_float = Decimal(0.1)
from_str = Decimal('0.1')
print('from float: {}\nfrom string: {}'.format(from_float, from_str))
```

In [ ]:
```python
my_decimal = Decimal('0.1')
sum_of_decimals = my_decimal + my_decimal + my_decimal
print(sum_of_decimals == Decimal('0.3'))
```

Operator precedence in calculations

Mathematical operator precedence applies. Use brackets if you want to change the execution order:

In [ ]:
```python
print(1 + 2**2 * 3 / 6) # 1 + 4 * 3 / 6 == 1 + 12 / 6 == 1 + 2
print((1 + 2**2) * 3 / 6)
```