```
# Why Python is Great: Namedtuples
# Using namedtuple is way shorter than
# defining a class manually:
>>> from collections import namedtuple
>>> Car = namedtuple('Car', 'color
mileage')

# Our new "Car" class works as expected:
>>> my_car = Car('red', 3812.4)
>>> my_car.color
'red'
>>> my_car.mileage
3812.4

# We get a nice string repr for free:
>>> my_car
Car(color='red' , mileage=3812.4)

# Like tuples, namedtuples are immutable:
>>> my_car.color = 'blue'
AttributeError: "can't set attribute"
```

```
# The get() method on dicts
# and its "default" argument

name_for_userid = {
    382: "Alice",
    590: "Bob",
    951: "Dilbert",
}

def greeting(userid):
    return "Hi %s!" %
name_for_userid.get(userid, "there")

>>> greeting(382)
"Hi Alice!"

>>> greeting(333333)
"Hi there!"
```

```
# How to sort a Python dict by value
# (== get a representation sorted by
value)

>>> xs = {'a': 4, 'b': 3, 'c': 2, 'd': 1}

>>> sorted(xs.items(), key=lambda x: x[1])
[('d', 1), ('c', 2), ('b', 3), ('a', 4)]

# Or:

>>> import operator
>>> sorted(xs.items(),
key=operator.itemgetter(1))
[('d', 1), ('c', 2), ('b', 3), ('a', 4)]
```

```
# Different ways to test multiple
# flags at once in Python
x, y, z = 0, 1, 0

if x == 1 or y == 1 or z == 1:
    print('passed')

if 1 in (x, y, z):
    print('passed')

# These only test for truthiness:
if x or y or z:
    print('passed')

if any((x, y, z)):
    print('passed')
```

```
# How to merge two dictionaries
# in Python 3.5+

>>> x = {'a': 1, 'b': 2}
>>> y = {'b': 3, 'c': 4}

>>> z = {**x, **y}

>>> z
{'c': 4, 'a': 1, 'b': 3}
```

```
# In Python 2.x you could
# use this:
>>> z = dict(x, **y)
>>> z
{'a': 1, 'c': 4, 'b': 3}

# In these examples, Python merges
dictionary keys
# in the order listed in the expression,
overwriting
# duplicates from left to right.
```