NAMED TUPLES

Modifying and Extending

Named Tuples are Immutable

So how can we "change" one or more values inside the tuple?

Just like with strings, we have to create a new tuple, with the modified values

```
Point2D = namedtuple('Point2D', 'x y')
pt = Point2D(0, 0)
```

Suppose we need to change the value of the x coordinate:

Simple approach: `pt = Point2D(100, pt.y)`

Note that the memory address of `pt` has now changed

Drawback

This simple approach can work well, but it has a major drawback

```
Stock = namedtuple('Stock', 'symbol year month day open high low close')

djia = Stock('DJIA', 2018, 1, 25, 26_313, 26_458, 26_260, 26_393)
```

Suppose we only want to change the `close` field

```
djia = Stock(djia.symbol,
             djia.year,
             djia.month,
             djia.day,
             djia.open,
             djia.high,
             djia.low,
             26_394)
```

painful!

Maybe slicing or unpacking?

```
djia = Stock('DJIA', 2018, 1, 25, 26_313, 26_458, 26_260, 26_393)
```

```
current = djia[:7]  current → ('DJIA', 2018, 1, 25, 26_313, 26_458, 26_260)
```

```
*current, _ = djia  current → ['DJIA', 2018, 1, 25, 26_313, 26_458, 26_260]
```

```
djia = Stock(*current, 26_394)
```

We can also use the `_make` class method – but we need to create an iterable that contains all the values first:

```
new_values = current + (26_394,)    new_values = current.append(26_394)
```

```
new_values → 'DJIA', 2018, 1, 25, 26_313, 26_458, 26_260, 26_394
```

iterable

```
djia = Stock._make(new_values)
```

This still has drawbacks

```
djia = Stock('DJIA', 2018, 1, 25, 26_313, 26_458, 26_260, 26_393)
```

What if we wanted to change a value in the middle, say `day`?

Cannot use extended unpacking (only one starred value in extending unpacking)

```
*pre, day, *post = djia
```

makes no sense...

Slicing will work:

```
pre = djia[:3]
post = djia[4:]
```

```
new_values = pre + (26,) + post
```

```
new_values → ('DJIA', 2018, 1, 26, 26_313, 26_458, 26_260, 26_394)
```

```
djia = Stock(*new_values)
```

But even this still has drawbacks!


```
djia = Stock('DJIA', 2018, 1, 25, 26_313, 26_458, 26_260, 26_393)
```

How about modifying both the `day` and the `high` values?

```
new_values = djia[:3] + (26,) + djia[4:5] + (26_459,) + djia[6:]

djia = Stock(*new_values)
```

This is just unreadable and extremely error prone!

There has to be a better way!

The _replace instance method

Named tuples have a very handy instance method, _replace

It will copy the named tuple into a new one, replacing any values from keyword arguments

The keyword arguments are simple the field names in the tuple and the new value

The keyword name must match an existing field name

```
Stock = namedtuple('Stock', 'symbol year month day open high low close')
djia = Stock('DJIA', 2018, 1, 25, 26_313, 26_458, 26_260, 26_393)

djia = djia._replace(day=26, high=26_459, close=26_394)
```

djia → 'DJIA', 2018, 1, 26, 26_313, 26_459, 26_260, 26_394

Note that the memory address of djia has now changed

Extending a Named Tuple

Sometimes we want to create named tuple that extends another named tuple, appending one or more fields

```
Stock = namedtuple('Stock', 'symbol year month day open high low close')
```

We want to create a new named tuple class, `StockExt` that adds a single field, `previous_close`

When dealing with classes, this is sometimes done by using subclassing.

But this not easy to do with named tuples

and there's a cleaner way of doing it anyway

Extending a Named Tuple

```
Point2D = namedtuple('Point2D', 'x y')
```

Let's say we want to create a `Point3D` named tuple that has an extra parameter

Yes, the obvious, and simplest approach here is best:

```
Point3D = namedtuple('Point3D', 'x y z')
```

But what happens if you have a lot of fields in the named tuple? Code is not as clean anymore...

```
Stock = namedtuple('Stock', 'symbol year month day open high low close')
```

```
StockExt = namedtuple('Stock', 'symbol year month day open high low close previous_close')
```

How about re-using the existing field names in `Stock`?

Extending a Named Tuple

```
Stock = namedtuple('Stock', 'symbol year month day open high low close')
```

```
Stock._fields → 'symbol', 'year', 'month', 'day', 'open', 'high', 'low', 'close'
```

We can then create a new named tuple by "extending" the `_fields` tuple

```
new_fields = Stock._fields + ('previous_close', )

StockExt = namedtuple('StockExt', new_fields)
```

Extending a Named Tuple

We can also easily use an existing `Stock` instance to create a new `StockExt` instance with the same common values, adding in our new `previous_close` value:

```
Stock = namedtuple('Stock', 'symbol year month day open high low close')

StockExt = namedtuple('StockExt', Stock._fields + ('previous_close', )

djia = Stock('DJIA', 2018, 1, 25, 26_313, 26_458, 26_260, 26_393)

djia_ext = StockExt(*djia, 26_000)
```

or

```
djia_ext = StockExt._make(djia + (26_000, ))
```

Code