Tuples

SHARP SIGHT

What you'll learn

- What are tuples
  - How they are related to lists
  - Why use tuples (vs lists)

- Working with tuples
  - creation, data retrieval

- Tuple assignment
  - assign multiple variables at once
  - swapping values of variables

Tuple basics

What are tuples?

- A tuple is like a list that you can't change

- Recall: lists *can* be changed
  - lists are 'mutable'

- Tuples *cannot* be changed
  - tuples are 'immutable'

Why we use tuples (instead of other data types)

- Tuples use less space

- Tuples are faster than lists
  - loop through them faster
  - faster code

- Tuples are good for things that won’t change
  - days of the week, months, spatial dimensions (x, y, z)

- Tuples protect your data
  - They can't be changed, so you can’t accidentally change one

Tuples contain multiple values

- They are a lot like lists!

```
days = ('Mon','Tue','Wed','Thu','Fri','Sat','Sun')
```

index:	0	1	2	3	4	5	6
days:	Mon	Tue	Wed	Thu	Fri	Sat	Sun

Tuples have indexes

```
days = ('Mon','Tue','Wed','Thu','Fri','Sat','Sun')
```

Every value of a tuple has an associated index

index:	0	1	2	3	4	5	6
days:	Mon	Tue	Wed	Thu	Fri	Sat	Sun

Tuples indexes start at 0

```
days = ('Mon','Tue','Wed','Thu','Fri','Sat','Sun')
```

Tuple indexes start at 0

index:	0	1	2	3	4	5	6
days:	Mon	Tue	Wed	Thu	Fri	Sat	Sun

Creating tuples

There are multiples ways to create tuples

- Simplest way is to enumerate items, separated by commas

- You can also enclose the group of items inside parenthesis
  - This is the "best practice"

Create tuples by separating items with a comma

Here, we've created a tuple by enumerating several values, separated by commas

```
dimensions = 'x', 'y', 'z'
print(dimensions)
('x', 'y', 'z')

type(dimensions)
tuple
```

You can also create tuples with items, inside of parenthesis

Here, we've created a tuple by enumerating several values, separated by commas, *inside of parenthesis*

```
days = ('Mon','Tue','Wed','Thu','Fri','Sat','Sun')
print(days)
('Mon', 'Tue', 'Wed', 'Thu', 'Fri', 'Sat', 'Sun')

type(days)
tuple
```

Getting data from tuples

Getting data from tuples is like getting data from other sequences

- You can use the methods you use with other sequences
  - Indexing
  - Slicing

Syntax: how to retrieve single items from a tuple

To retrieve an item from a tuple, type the name of the tuple, followed by brackets

```
your_tuple[index-to-retrieve]
```

Inside of the brackets, you have the index associated with the value you want to get

Example: how to retrieve single items from a tuple

- Use an index to retrieve specific element

```
days = ('Mon','Tue','Wed','Thu','Fri','Sat','Sun')
days[0]
'Mon'
```

Syntax: how to access a "slice" of a tuple

The name of
the tuple

```
your_tuple[start-index : stop-index]
```

The index associated with
the first tuple item you
want to retrieve

The index of the "stoping
point" of your slice
... this will not be included!

Example: access a slice of a tuple

- In this example, we're accessing the first 5 values
  - remember: the slice starts at the start index
  - the slice goes up to and *excluding* the stop index

```
days = ('Mon','Tue','Wed','Thu','Fri','Sat','Sun')
days[0:5]
('Mon', 'Tue', 'Wed', 'Thu', 'Fri')
```

Tuple assignment

You can use "tuple assignment" to assign values to variables

- We can use tuples to assign values to multiple variables in one line of code
- Sometimes called “tuple unpacking”

Example: tuple assignment

In the first line, we're creating a tuple with 3 values

```
dimensions = ('x', 'y', 'z')
(dim1, dim2, dim3) = dimensions
```

In the second line, we're creating a *second* tuple with 3 variable names, and we're assigning the values '`x`', '`y`', and '`z`' to those variables

Example: tuple assignment

```
dimensions = ('x', 'y', 'z')
(dim1, dim2, dim3) = dimensions

print(dim1)
x

print(dim2)
y

print(dim3)
z
```

The variables `dim1`, `dim2`, and `dim3` now contain the values `x`, `y`, and `z`

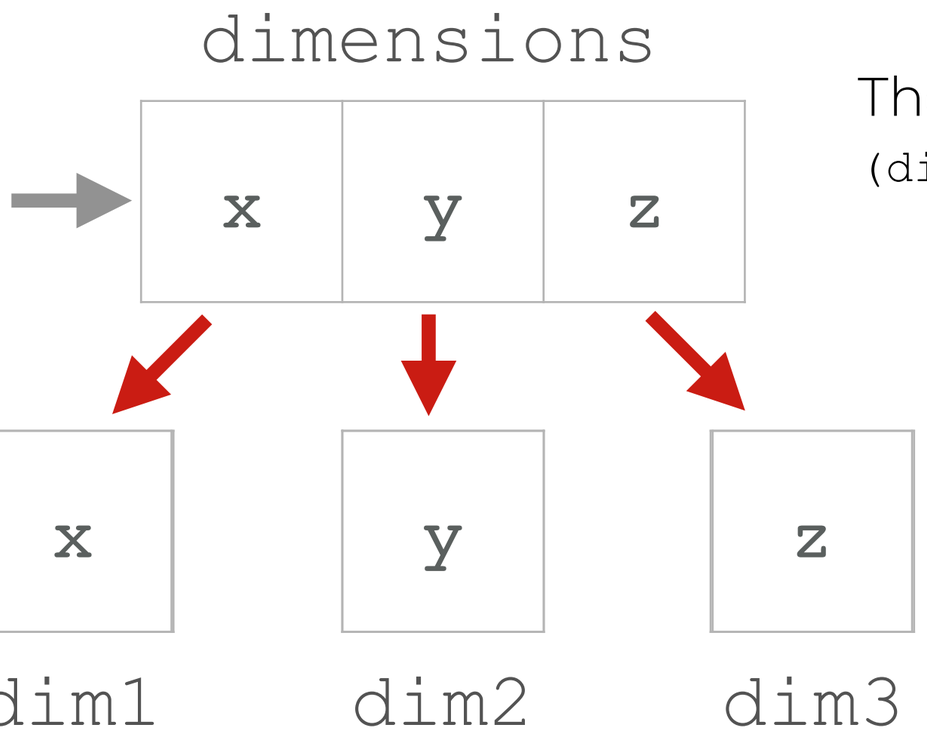
Example: tuple assignment

```
dimensions = ('x', 'y', 'z')
(dim1, dim2, dim3) = dimensions
```

The code

`dimensions = ('x', 'y', 'z')`

creates a new tuple



The code

`(dim1, dim2, dim3) = dimensions`

assigns the values of the tuple `dimensions` to the new variables

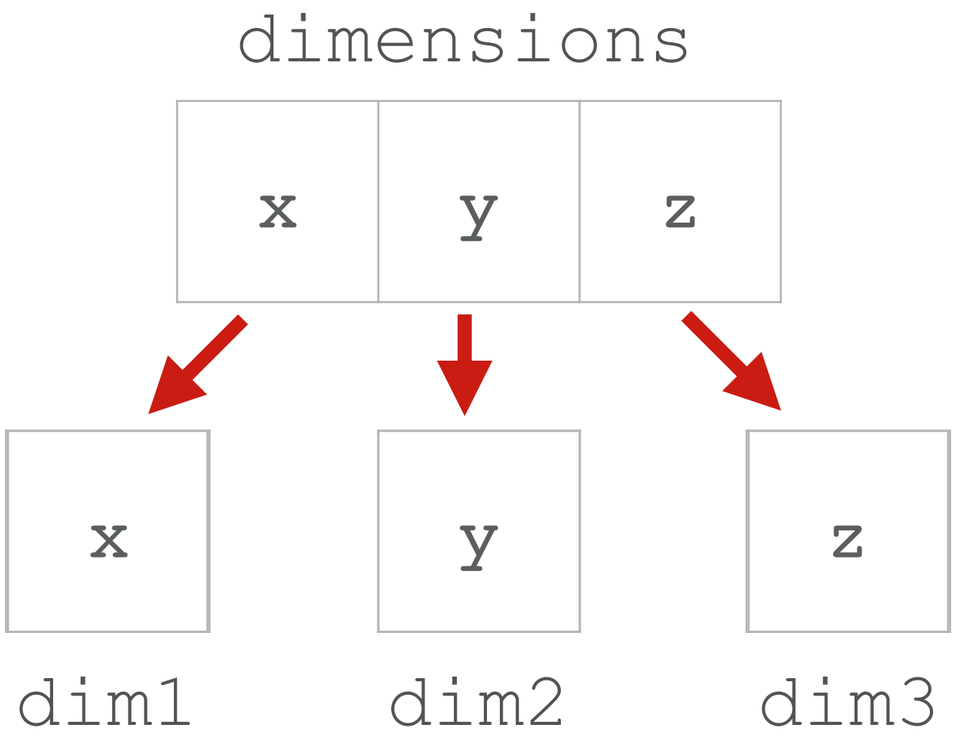
The code

`(dim1, dim2, dim3)`

creates three new variables

Example: tuple assignment

```
dimensions = ('x', 'y', 'z')
(dim1, dim2, dim3) = dimensions
```



In tuple assignment, the values on the right hand side of code are assigned to the variable names on the left hand side

You can use tuple assignment to swap values between variables

- Here, we're swapping values between `var1` and `var2`

```
var1 = 55
var2 = 99

var1, var2 = (var2, var1)
var1
99
```

Recap

Recap of what we learned

- Create tuples by enumerating items inside parenthesis

- Access tuple values using bracket notation
  - get individual values by index
  - access "slices" with slicing notation

- Tuple assignment
  - assign multiple variables at once
  - swapping values of variables