CISS450: Artificial Intelligence
Lecture 7: Tuples

Yihsiang Liow

Agenda

- Study tuples. Tuples are very similar to lists.

Tuples

- Tuples are similar to lists except that they are immutable. So they are pretty easy once you know lists.

First Examples

- Try:

```
xs = (1,2,3,4)
print(xs, type(xs))
xs = ()
print(xs, type(xs))
xs = (1)
print(xs, type(xs))
```

- Is there anything wrong with the above? Can you explain? Now try

```
xs = (1,)
print(xs, type(xs))
```

Operators

- Try:

```
xs = (1,2,3)
ys = ("four", 5)
zs = xs + ys
print(zs, type(zs))
print(zs[0], zs[-2])
print(zs[1:4:2])
print(2*zs)
```

Immutability

- You cannot change a tuple in-place:

```
xs = (1, 2, 3)
print(xs[0])
xs[0] = 4
```

- If you want to change the 0^{th} element to 4, you have to do the same trick as for strings:

```
xs = (1, 2, 3)
xs = (4,) + xs[1:]
```

Type Conversion

- You can convert between lists and tuples:

```
xs = [1,2,3]
xs = tuple(xs)
print(xs, type(xs))
xs = list(xs)
print(xs, type(xs))
```

Functions/Methods

- I will not go through all the functions/methods. They are similar to the functions/methods of lists except that when if a list function/method changes the list in-place, then that function/method does not exist for tuples.
- For instance you know that you can sort a list (in-place). So there is no sort method for tuples. The following gives an error.

```
xs = (3,2,1)
xs.sort()
```

Lists vs Tuples

- If you're using an “array” structure where the elements are static, then use tuples. Otherwise use lists.
- Tuples provide integrity and security.
- Later when we cover dictionaries, you will see that tuples can be used as keys but not lists (for the obvious reason).

Zipping

- Try this:

```
names = ("john", "jane", "tom")
heights = (5.85, 5.73, 5.56)
xs = zip(names, heights)
print(xs)
print(list(xs))
```