[301] Objects

Tyler Caraza-Harter

Learning Objectives Today

More data types

- tuple (immutable list)
- custom types: creating objects from namedtuple and recordclass

References

- Motivation
- “is” vs “==”
- Gotchas (interning and argument modification)

Read:

- **Downey Ch 10 ("Objects and Values" and “Aliasing")**
- **Downy Ch 12**

Today's Outline

New Types

- **tuple**
- namedtuple
- recordclass

References

- motivation
- unintentional argument modification
- “is” vs. “==”

Tuple Type

```
nums_list  = [200, 100, 300]
nums_tuple = (200, 100, 300)
```

if you use parentheses (round)
instead of brackets [square]
you get a tuple instead of a list

Tuple Type

```
nums_list  = [200, 100, 300]
nums_tuple = (200, 100, 300)
```

if you use parentheses (round)
instead of brackets [square]
you get a tuple instead of a list

What is a tuple?

Tuple Type

```
nums_list  = [200, 100, 300]
nums_tuple = (200, 100, 300)
```

Like a list

- for loop, indexing, slicing, other methods

Unlike a list:

- immutable (like a string)

Tuple Type

```
nums_list  = [200, 100, 300]
nums_tuple = (200, 100, 300)

print(nums_list[2])
print(nums_tuple[2])
```

Like a list

- for loop, **indexing**, slicing, other methods

Unlike a list:

- immutable (like a string)

Tuple Type

```
nums_list  = [200, 100, 300]
nums_tuple = (200, 100, 300)
```

```
print(nums_list[2])
print(nums_tuple[2])
```

both of these print 300

Like a list

- for loop, **indexing**, slicing, other methods

Unlike a list:

- immutable (like a string)

Tuple Type

```
nums_list  = [200, 100, 300]
nums_tuple = (200, 100, 300)

nums_list[0] = 22
nums_tuple[0] = 22
```

Like a list

- for loop, indexing, slicing, other methods

Unlike a list:

- **immutable** (like a string)

Tuple Type

```
nums_list  = [200, 100, 300]
nums_tuple = (200, 100, 300)

nums_list[0] = 22
nums_tuple[0] = 22
```

changes list to
[22, 100, 300]

Like a list

- for loop, indexing, slicing, other methods

Unlike a list:

- **immutable** (like a string)

Tuple Type

```
nums_list  = [200, 100, 300]
nums_tuple = (200, 100, 300)

nums_list[0] = 22
nums_tuple[0] = 22
```

changes list to
[22, 100, 300]

Crashes!

```
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'tuple' object does not support item assignment
```

Like a list
- for loop, indexing, slicing, other methods

Unlike a list:
- **immutable** (like a string)

Tuple Type

```
nums_list  = [200, 100, 300]
nums_tuple = (200, 100, 300)

nums_list[0] = 22
nums_tuple[0] = 22
```

changes list to
[22, 100, 300]

Crashes!

```
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'tuple' object does not support item assignment
```

Like a list

- for loop, indexing, slicing, other methods

Unlike a list:

- **immutable** (like a string)

Why would we ever want immutability?

1. avoid certain bugs
2. some use cases require it (e.g., dict keys)

Example: location -> building mapping

```
buildings = {
   [0,0]: “Comp Sci”,
   [0,2]: “Psychology”,
   [4,0]: “Noland”,
   [1,8]: “Van Vleck”
}
```

trying to use x,y coordinates as key

FAILS!

```
Traceback (most recent call last):
  File "test2.py", line 1, in <module>
    buildings = {[0,0]: "CS"}
TypeError: unhashable type: 'list'
```

Example: location -> building mapping

```
buildings = {
   (0,0): “Comp Sci”,
   (0,2): “Psychology”,
   (4,0): “Noland”,
   (1,8): “Van Vleck”
}
```

trying to use x,y coordinates as key

Succeeds!

(with tuples)

Today's Outline

New Types

- tuple
- **namedtuple**
- recordclass

References

- motivation
- unintentional argument modification
- “is” vs. “==”

Today's Outline

New Types

- tuple
- namedtuple
- recordclass

References

- motivation
- unintentional argument modification
- “is” vs. “==”

Today's Outline

New Types

- tuple
- namedtuple
- recordclass

References

- motivation
- unintentional argument modification
- “is” vs. “==”

Today's Outline

New Types

- tuple
- namedtuple
- recordclass

References

- motivation
- unintentional argument modification
- “is” vs. “==”

Today's Outline

New Types

- tuple
- namedtuple
- recordclass

References

- motivation
- unintentional argument modification
- “is” vs. “==”