Introduction to Python

The Python Data Model

10th November 2016

The Function Call Stack

- You start with a ‘global scope‘.
- This is a set of variables, class names etc. available to you at all points in the program.

The Function Call Stack

- You start with a ‘global scope‘.
- This is a set of variables, class names etc. available to you at all points in the program.
- When you call a function you ‘push‘ a new scope onto the ‘call stack‘.

The Function Call Stack

- You start with a ‘global scope‘.
- This is a set of variables, class names etc. available to you at all points in the program.
- When you call a function you ‘push‘ a new scope onto the ‘call stack‘.
- When a function returns, it is ‘popped‘ off the stack.

The Function Call Stack

- You start with a ‘global scope‘.
- This is a set of variables, class names etc. available to you at all points in the program.
- When you call a function you ‘push‘ a new scope onto the ‘call stack‘.
- When a function returns, it is ‘popped‘ off the stack.
- **Demo Time!**

How does Python Store Data?

- Recall that everything in Python is an object
- Functions, numbers, strings, python code...

How does Python Store Data?

- Recall that everything in Python is an object
- Functions, numbers, strings, python code...
- Every object has a unique ID number. In CPython (standard Python on Linux) this is the address in memory. (C Pointer if you know what that means)

How does Python Store Data?

- Recall that everything in Python is an object
- Functions, numbers, strings, python code...
- Every object has a unique ID number. In CPython (standard Python on Linux) this is the address in memory. (C Pointer if you know what that means)

```
x = 2
y = 2
x == y # True
x is y # maybe true? don't depend on it!
```

How does Python Store Data?

- Every object has a reference count - how many things are pointing to me?

How does Python Store Data?

- Every object has a reference count - how many things are pointing to me?
- If you assign a name to an object the reference count goes up. If you remove that assignment it goes down.

How does Python Store Data?

- Every object has a reference count - how many things are pointing to me?
- If you assign a name to an object the reference count goes up. If you remove that assignment it goes down.
- If the reference count is zero, Python deletes the object for you and frees up the memory.
- This is called **Garbage Collection**.

Mutable vs. Immutable Data Types

- Some objects cannot be changed in place. These are **Immutable Objects**.

```
x = 2
y = x
x is y #True! guaranteed
x += 2
print(y) # prints 2
```

Mutable vs. Immutable Data Types

- Some objects cannot be changed in place. These are **Immutable Objects**.

```
x = 2
y = x
x is y #True! guaranteed
x += 2
print(y) # prints 2
```

- Strings, ints and floats are immutable. This makes sense (I think).

Mutable vs. Immutable Data Types

- Other objects can be changed in place. These are **Mutable Objects**.

```
x = [1,2,3]
y = x
x is y #True! guaranteed
x.append(2)
print(y) # [1,2,3,2]
```

Mutable vs. Immutable Data Types

- Other objects can be changed in place. These are **Mutable Objects**.

```
x = [1,2,3]
y = x
x is y #True! guaranteed
x.append(2)
print(y) # [1,2,3,2]
```

- Lists and dictionaries are mutable types.

Mutable vs. Immutable Data Types

- Other objects can be changed in place. These are **Mutable Objects**.

```
x = [1,2,3]
y = x
x is y #True! guaranteed
x.append(2)
print(y) # [1,2,3,2]
```

- Lists and dictionaries are mutable types.
- Use round brackets instead of square brackets to create a tuple - an immutable list.

Shameless theft

- http://nedbatchelder.com/text/names1.html
- I will now steal this guys talk
- This weeks exercises all stolen from reddit.com/r/dailyprogrammer
- Use this site for practice!