Objects

- Objects represent information
- They consist of data and behavior, bundled together to create abstractions
- Objects can represent things, but also properties, interactions, & processes
- A type of object is called a class; classes are first-class values in Python

Object-oriented programming:
- A metaphor for organizing large programs
- Special syntax that can improve the composition of programs
- In Python, every value is an object
  - All objects have attributes
  - A lot of data manipulation happens through object methods
  - Functions do one thing; objects do many related things

Example: Strings

Representing Strings: the ASCII Standard
- American Standard Code for Information Interchange
- 8 rows: 3 bits
- 16 columns: 4 bits
- Layout was chosen to support sorting by character code
- Rows indexed 2-5 are a useful 6-bit (64 element) subset
- Control characters were designed for transmission
  - "Line feed" (\n)
  - "Bell" (\a)

Representing Strings: the Unicode Standard
- 109,000 characters
- 93 scripts (organized)
- Enumeration of character properties, such as case
- Supports bidirectional display order
- A canonical name for every character
- U+0058 LATIN CAPITAL LETTER X
- U+263a WHITE SMILING FACE
- U+2639 WHITE FROWNING FACE

Mutation Operations
Some Objects Can Change

The same object can change in value throughout the course of computation. All names that refer to the same object are affected by a mutation.

Mutation Can Happen Within a Function Call

A function can change the value of any object in its scope.

```python
>>> four = [1, 2, 3, 4]
>>> len(four)
4
>>> mystery(four)
3
>>> len(four)
4
>>> another_mystery() # No arguments!
>>> len(four)
4
```

Tuples

Immutable values are protected from mutation.

```python
>>> turtle = (1, 2, 3)
>>> new(turtle)
>>> turtle
(1, 2, 3)
>>> change(turtle)'s binding
Anything could be inside!
```

Sameness and Change

As long as we never modify objects, a compound object is just the totality of its pieces.

A rational number is just its numerator and denominator.

This view is no longer valid in the presence of change.

A compound data object has an "identity" in addition to the pieces of which it is composed.

A list is still "the same" list even if we change its contents.

Conversely, we could have two lists that happen to have the same contents, but are different.

Identity Operators

- **Identity**
  - `<exp1> == <exp2>`
  - Evaluates to True if both `<exp1>` and `<exp2>` evaluate to the same object.

- **Equality**
  - `<exp1> is <exp2>`
  - Evaluates to True if both `<exp1>` and `<exp2>` evaluate to equal values.

  Identical objects are always equal values.

Mutable Default Arguments are Dangerous

A default argument value is part of a function value, not generated by a call.

```python
>>> def f(s=[]):
...     s.append(4)
...     return len(s)
...     2

>>> f()
2
```

```python
>>> def mystery(s):
...     s[2] = []
...     s.pop()
...     s
...

>>> mystery(four)
[1, 2, 4]
```

Immutable values are protected from mutation.

A function can change the value of any object in its scope.

```python
>>> f() # `f` is called
4
>>> len(f())
4
>>> len(s)
2
```

```python
>>> def another_mystery():
...     four.pop()
...     four
...

>>> another_mystery() # No arguments!
[1, 2, 3]
```