Objects

- Objects represent information
- They consist of data and behavior, bundled together to create abstractions
- 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 list 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 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

- First example in the course of an object changing state
- The same object can change in value throughout the course of computation
- Only objects of mutable types can change: lists & dictionaries

Mutation Can Happen Within a Function Call

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

Tuples
Tuples are Immutable Sequences

Immutable values are protected from mutation:

```python
turtle = (1, 2, 3)
```

The value of an expression can change because of changes in names or objects:

```python
turtle[0] = 4
```

An immutable sequence may still change if it contains a mutable value as an element:

```python
turtle[0] = 5
```

Identity Operators

Identity

```python
<exp0> is <exp1>
evaluates to True if both <exp0> and <exp1> evaluate to the same object
```

Equality

```python
<exp0> == <exp1>
evaluates to True if both <exp0> and <exp1> evaluate to equal values:
```

Identical objects are always equal values:

```python
Identity

<exp0> is <exp1>
evaluates to True if both <exp0> and <exp1> evaluate to the same object
```

Equality

```python
<exp0> == <exp1>
evaluates to True if both <exp0> and <exp1> evaluate to equal values:
```

Identical objects are always equal values:

```python
Identity

<exp0> is <exp1>
evaluates to True if both <exp0> and <exp1> evaluate to the same object
```

Equality

```python
<exp0> == <exp1>
evaluates to True if both <exp0> and <exp1> evaluate to equal values:
```

Identical objects are always equal values:

```python
Identity

<exp0> is <exp1>
evaluates to True if both <exp0> and <exp1> evaluate to the same object
```

Equality

```python
<exp0> == <exp1>
evaluates to True if both <exp0> and <exp1> evaluate to equal values:
```

Identical objects are always equal values:

```python
Identity

<exp0> is <exp1>
evaluates to True if both <exp0> and <exp1> evaluate to the same object
```

Equality

```python
<exp0> == <exp1>
evaluates to True if both <exp0> and <exp1> evaluate to equal values:
```

Identical objects are always equal values:

```python
Identity

<exp0> is <exp1>
evaluates to True if both <exp0> and <exp1> evaluate to the same object
```

Equality

```python
<exp0> == <exp1>
evaluates to True if both <exp0> and <exp1> evaluate to equal values:
```

Identical objects are always equal values: