encoding 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

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

representing strings: utf-8 encoding

utf (ucs (universal character set) transformation format)
unicode: correspondence between characters and integers
utf-8: correspondence between those integers and bytes

a byte is 8 bits and can encode any integer 0-255.

```
baby
girl
woman
older woman
```

mutation operations

some objects can change
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]
>>> def mystery(s):
...     s.pop()
...     s.pop()
>>> len(four)
2
>>> four = [1, 2, 3, 4]
>>> def another_mystery(s):
...     four.pop()
...     four.pop()
>>> len(four)
4
>>> another_mystery()
# No arguments!
>>> len(four)
2
>>> four = [1, 2, 3, 4]
>>> len(four)
4
>>> another_mystery()
# No arguments!
>>> another_mystery()
# No arguments!
>>> len(four)
2
>>> four = [1, 2, 3, 4]
>>> def mystery(s):
...     s[2:] = []
>>> another_mystery
...     return len(s)
>>> another_mystery()
>>> another_mystery()
>>> another_mystery()
```

Tuples are Immutable Sequences

Immutable values are protected from mutation:

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

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

```python
>>> x = 2
>>> x = x
>>> x = 3
>>> x = x
6
```

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

```python
>>> x = [(1, 2), 3]
>>> x[0] = 4
>>> x
[(4, 2), 3]
```

Identity Operators

Identity:

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

Equality:

- `<exp0> == <exp1>` evaluates to `True` if both `<exp0>` and `<exp1>` 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=[1]):
...     s.append(5)
...     return len(s)
...     return len(s)
...     return len(s)
...     return len(s)
>>> f()  # No arguments!
1
>>> f()  # No arguments!
2
>>> f()  # No arguments!
3
```

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.

```
>>> a = [10]
>>> b = a
>>> a == b
True
>>> a[0] = 10
>>> a == b
False
```

Tuples

- Tuples are immutable sequences.
- Immutable values are protected from mutation.
- The value of an expression can change because of changes in names or objects.
- An immutable sequence may still change if it contains a mutable value as an element.

Identity Operators

Identity:

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

Equality:

- `<exp0> == <exp1>` evaluates to `True` if both `<exp0>` and `<exp1>` 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=[1]):
...     s.append(5)
...     return len(s)
...     return len(s)
...     return len(s)
...     return len(s)
>>> f()  # No arguments!
1
>>> f()  # No arguments!
2
>>> f()  # No arguments!
3
```

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.

```
>>> a = [10]
>>> b = a
>>> a == b
True
>>> a[0] = 10
>>> a == b
False
```