Working with Lists

```python
>>> digits = [1, 8, 2, 8]
>>> add([2, 7], mul(digits, 2))
[2, 7, 1, 8, 2, 8, 1, 8, 2, 8]
```

The number of elements

```python
>>> len(digits)
4
```

An element selected by its index

```python
>>> digits[3]
8
```

Concatenation and repetition

```python
>>> [2, 7] + digits * 2
[2, 7, 1, 8, 2, 8, 1, 8, 2, 8]
```

Nested lists

```python
>>> pairs = [[10, 20], [30, 40]]
>>> pairs[1][0]
30
```
Containers

Built-in operators for testing whether an element appears in a compound value

```python
>>> digits = [1, 8, 2, 8]
>>> 1 in digits
True
>>> 8 in digits
True
>>> 5 not in digits
True
>>> not(5 in digits)
True
```

(Demo)

Sequence Iteration

```python
def count(s, value):
    total = 0
    for element in s:
        if element == value:
            total = total + 1
    return total
```

Name bound in the first frame of the current environment (not a new frame)

(Demo)

For Statements
For Statement Execution Procedure

```python
for <name> in <expression>:
    <suite>
```

1. Evaluate the header `<expression>`, which must yield an iterable value (a sequence)

2. For each element in that sequence, in order:
   A. Bind `<name>` to that element in the current frame
   B. Execute the `<suite>`

Sequence Unpacking in For Statements

```python
>>> pairs = [[1, 2], [2, 2], [3, 2], [4, 4]]
>>> same_count = 0

>>> for x, y in pairs:
...     if x == y:
...         same_count = same_count + 1

>>> same_count
2
```

The Range Type

A range is a sequence of consecutive integers.*

```
..., -5, -4, -3, -2, -1, 0, 1, 2, 3, 4, 5, ...
```

Length: ending value - starting value

Element selection: starting value + index

```
>>> list(range(-2, 2))
[-2, -1, 0, 1]
```

List constructor

```
>>> list(range(4))
[0, 1, 2, 3]
```

Range with a 0 starting value

* Ranges can actually represent more general integer sequences.
List Comprehensions

>>> letters = ['a', 'b', 'c', 'd', 'e', 'f', 'm', 'n', 'o', 'p']
>>> [letters[i] for i in [3, 4, 6, 8]]
['d', 'e', 'm', 'o']

Example: Promoted

Implement `promoted`, which takes a sequence `s` and a one-argument function `f`. It returns a list with the same elements as `s`, but with all elements `e` for which `f(e)` is a true value placed first. Among those placed first and those placed after, the order stays the same.

```python
def promoted(s, f):
    """Return a list with the same elements as s, but with all elements e for which f(e) is a true value placed first. Among those placed first and those placed after, the order stays the same."
    return [e for e in s if f(e)] + [e for e in s if not f(e)]
```

```
>>> promoted(range(10), odd)  # odds in front
[1, 3, 5, 7, 9, 0, 2, 4, 6, 8]
```