A range is a sequence of consecutive integers.

\[ \ldots, -5, -4, -3, -2, -1, 0, 1, 2, 3, 4, 5, \ldots \]

Length. A sequence has a finite length.

Element selection. A sequence has an element corresponding to any non-negative integer index less than its length, starting at 0 for the first element.

Generator expressions

\{ \text{map exp} \text{ for } \text{name} \text{ in } \text{iter exp} \text{ if } \text{filter exp} \}

- Evaluates to an iterable object.
- \text{iter exp} is evaluated when the generator expression is evaluated.
- Remaining expressions are evaluated when elements are accessed.

List comprehensions

\{ \text{map exp} \text{ for } \text{name} \text{ in } \text{iter exp} \text{ if } \text{filter exp} \}

Short version: \{ \text{map exp} \text{ for } \text{name} \text{ in } \text{iter exp} \}

Unlike generator expressions, the map expression is evaluated when the list comprehension is evaluated.

```python
>>> suits = ['heart', 'diamond', 'spade', 'club']
>>> from unicodedata import lookup
>>> [lookup('WHITE ' + s.upper() + ' SUIT') for s in suits]
["\u2663", "\u2664", "\u2665", "\u2666"]
```

A function with a parent frame

```
def make_withdraw(balance):
    def withdraw(amount):
        if amount > balance:
            return 'Insufficient funds'
        balance -= amount
        return balance
    return withdraw

withdraw = make_withdraw(100)
withdraw(50)
```

Mutable values can be changed without a nonlocal statement.

```
def pig_latin(w):
    if starts_with_a_vowel(w):
        return w[:1] + w[0].lower() + 'ay'
    return w

def pig_latin(w1):...
To evaluate a dot expression, `<expression>.*<name>`:

1. Evaluate the expression to the left of the dot, which yields the object of the dot expression.
2. `<name>` is matched against the instance attributes of that object.
3. If an attribute with that name exists, its value is returned.
4. If not, `<name>` is looked up in the class, which yields a class attribute value.

That value is returned unless it is a function, in which case a `bound method` is returned instead.

To look up a name in a class:

1. If it names an attribute in the class, return the attribute value.
2. Otherwise, look up the name in the base class, if there is one.

Assignment statements with a dot expression on their left-hand side assign attribute for the object of that dot expression:

- If the object is an instance, then assignment sets an instance attribute.
- If the object is a class, then assignment sets a class attribute.

Assignment affects instance attributes:

- If callable(value):
  - If `method`(*`args`):
    - `return` value.
  - `return` method.
- If `__init__`(*`args`):
  - `return` instance.
- If `instance`(*`args`):
  - `return` instance.
- If `cls`(*`args`):
  - `return` constructor.
- If `cls.get_value`(*`args`):
  - `return` value.
- If `cls.set_value`(*`args`):
  - `return` value.

When a class is called:

1. A new instance of that class is created.
2. The constructor `__init__` of the class is called with the new object as its first argument (`self`), along with additional arguments provided in the call expression.

The class of the instance.

Special constructor name is fixed here.

Class attribute lookup.

Dispatch dictionary.

Type dispatching: Define a different function for each possible combination of types for which an operation is valid.
The interface for sets:
- Membership testing: Is a value an element of a set?
- Adjunction: Return a set with all elements in set1 and set2.
- Union: Return a set with all elements in set1 or set2.
- Intersection: Return a set with any elements in set1 and set2.

Proposal 1: A set is represented by a recursive list that contains no duplicate items.

Proposal 2: A set is represented by a recursive list with unique elements ordered from least to greatest.

Proposal 3: A set is represented as a Tree. Each entry is:
- Larger than all entries in its left branch and
- Smaller than all entries in its right branch.

<table>
<thead>
<tr>
<th>Union</th>
<th>Intersection</th>
<th>Adjunction</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 2 3</td>
<td>1 4 5 3</td>
<td>2 3 1 2</td>
</tr>
<tr>
<td>4 3 2</td>
<td>1 3 2</td>
<td>1 2 3</td>
</tr>
</tbody>
</table>

If 9 is in the set, it is somewhere in this branch.

Exceptions are raised with a raise statement.

raise <expression>

<expression> must evaluate to an exception instance or class.

Exceptions are constructed like any other object; they are just instances of classes that inherit from BaseException.

try: <try suite>
except <exception class> as <name>:
    <except suite>
...

The <try suite> is executed first;

The <except suite> is executed, with <name> bound to the exception.

Scheme programs consist of expressions, which can be:
- Primitive expressions: 2, 3.3, true, +, quotient, ...
- Combinations: (quotient 10 2), (not true), ...
- Numbers are self-evaluating; symbols are bound to values.
- Call expressions have an operator and 0 or more operands.

A combination that is not a call expression is a special form:
- If expression: (if <predicate> <consequent> <alternative>)
- Lambda expressions evaluate to anonymous functions.

(lambda (<formal-parameters>) <body>)

Two equivalent expressions:
- (define (plus4 x) (+ x 4))
- (define (plus4 (lambda (x) (+ x 4)))

An operator can be a call expression too:

((lambda (y z) (+ y (square z))) 1 2 3)

In the late 1950s, computer scientists used confusing names.
- cons: Two-argument procedure that creates a pair.
- car: Procedure that returns the first element of a pair.
- cdr: Procedure that returns the second element of a pair.
- nil: The empty list.

They also used a non-obvious notation for recursive lists.
- A (recursive) Scheme list is a pair in which the second element is nil or a Scheme list.
- Scheme lists are written as space-separated combinations.
- A dotted list has an arbitrary value for the second element of the last pair. Dotted lists may not be well-formed lists.

Exceptions are constructed like any other object; they are just instances of classes that inherit from BaseException.

try: <try suite>
except <exception class> as <name>:
    <except suite>
...

The <try suite> is executed first;

If, during the course of executing the <try suite>, an exception is raised that is not handled otherwise, and

If the class of the exception inherits from <exception class>, then

The <except suite> is executed, with <name> bound to the exception.