Discrete Structures: CMPSC 102

Oliver BONHAM-CARTER

Fall 2018
Week 5

Types of Sequences?

- Strings, which are sequences of characters.
- Files contain a sequence of lines and the lines are sequences of characters.
- Objects, over which the `range()` function, can iterate

Examples

```
for element in [1, 2, 3]: # lists
    print(element)
for element in (1, 2, 3): #sets
    print(element)
for key in {'one':1, 'two':2}: #dictionaries
    print(key)
for char in "123": #strings
    print(char)
for line in open("myfile.txt"): # open, read a file
    print(line, end='')
```

Building Tuples

To cover this again...

Building Tuples in Python

```
# Creating non-empty tuples
myTuple = 'tea', 'coffee'
print(myTuple)
print(type(myTuple))
```

Or, Use Parenthesis to Build Tuples in Python

```
myOtherTuple = ('Bagels', 'Donuts')
print(myOtherTuple)
print(type(myOtherTuple))
```

Tuples and n-Tuples

Mathematically Speaking...

- In mathematics, a *tuple* is a finite ordered list (sequence) of elements
- An n-tuple is a sequence (or ordered list) of n elements (n is a positive integer).
  - Ex: (2, 7, 4, 1, 7) denotes a 5-tuple.

General Rule About Equality

- The general rule for the identity of two n-tuples is

$$(a_1, a_2, ..., a_n) = (b_1, b_2, ..., b_n) \text{ if and only if}$$
$$a_1 = b_1, a_2 = b_2, ..., a_n = b_n$$

```
a = (1,2,3)
b = (1,2,3)
a == b # test?
b = (1,3,2)
a == b # test?
```

Tuples and n-Tuples
Mathematically Speaking...

1. A tuple may contain multiple instances of the same element,
   tuple $(1, 2, 2, 3) \neq (1, 2, 3)$ but,
   set $\{1, 2, 2, 3\} = \{1, 2, 3\}$
2. Tuple elements are ordered,
   tuple $(1, 2, 3) \neq (3, 2, 1)$ but,
   set $\{1, 2, 3\} = \{3, 2, 1\}$
3. A **tuple** has a finite number of elements (also known as n-tuples), while a set or a **multiset** may have an infinite number of elements.

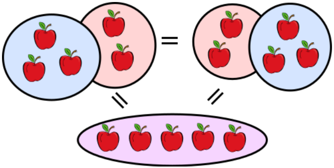
```
a = (1,2,3)
s = set({1,2,3})
a == s # test?
```

Elements of Tuples

- Sequences are not generic: they usually contain similar types of elements.
  - Ex: Lists contain same types of data structures, strings contain chars, files contain lines
- Sequences and n-tuples
  - n-tuples: An ordered set with n elements
    - Ex: File sequences are not n-tuples because they can contain any number of lines

Commutative

- The term "commutative" is used in several related senses.
- A binary operation $*$ on a set S is called *commutative* if: $x * y = y * x$ for all $x, y \in S$
  - An operation that does not satisfy the above property is called *non-commutative*.
- One says that x *commutes* with y under $*$ if: $x * y = y * x$
- A binary function $f : A \times A \to B$ is called *commutative* if: $f(x, y) = f(y, x)$ for all $x, y \in A$



Properties of Sequences

Formal Definition of Identity

- **Identity**: There exists an element $e \in S$ such that for any $a \in M$, $e * a = a * e = a$

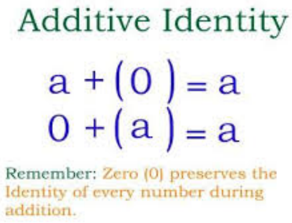
Identity

- An identity is an equality relation $a = b$,
- Ex: a and b equal some numeric value.
  - a + b == a + b
  - a + b == b + a
  - a * b == a * b
  - a * b == b * a

- $a = a + e$
- $a + e = a$
- a is non-empty, contains some element
- e must be an empty sequence or is equal to 0
  - e has an *identity* property, meaning that it does not influence the operations
- $a * e = a$ or $a = a * e$, (what is e, the identity here?)

Types of Sequences
Properties of Sequences
Commutative
Identity
Concatenation
Associative

Properties of Sequences
Identity




```
a = 1
b = 0
a == a + b #make a truth test
a + b == a #make another truth test
```

Properties of Sequences

Non-Commutative operations

- Washing and drying clothes resembles a noncommutative operation; washing and then drying produces a markedly different result to drying and then washing.
- Putting on left and then right socks on feet is commutative
- Putting on shirt and then sweater is not-commutative

Strings

```
a = "face"
b = "book"
a + b == b + a # run the test!
"facebook" != "bookface"
```

- Definition: a series of interconnected things or events. The concatenation is to place one string after another. The order of placement is significant to the final product.

Ex: Concatenation of sequences

```
a = ("This", "Is")
type(a)
b = ('Loads', 'Of', 'Fun', ':-)')
type(b)
c = a + b
print(c)
(  'This', 'Is', 'Loads', 'Of', 'Fun', :-)''   )
type(c)
```

Properties of Sequences

Formal Definition of Associativity

- **Associativity Addition**: For any $a, b, c \in S, a + (b + c) = (a + b) + c$
- **Associativity Multiplication**: For any $a, b, c \in S, a * (b * c) = (a * b) * c$

Properties of Sequences

Associative Property

- Definition: The associative property states that you can add or multiply regardless of how the numbers are grouped.
- Concatenation of sequences with the associative property

- $(a + b) + c = a + (b + c)$ for any strings a, b and c.

```
a, b, c = 1, 2, 3
(a + b) + c == a + (b + c)
```

- $(a * b) * c = a * (b * c)$ for any strings a, b and c.

```
a, b, c = 1, 2, 3
(a * b) * c == a * (b * c)
```

Properties of Sequences

Associative Property

Generalized Associative Law: Keep variables in same order

- ((ab)c)d
- (ab)(cd)
- (a(bc))d
- a((bc)d)
- a(b(cd))

To Note:

- **Associative**: Variables kept in same order, operators may change order
- **Commutative**: Variables may change order, operators kept in same order.

Problem:

Put x_0 money in a bank at year 0. What is the value after N years if the interest rate is p percent per year?

Solution:

The fundamental information relates the value at year n, x_n to the value of the previous year, x_{n-1}.

$$x_n = x_{n-1} + \frac{p}{100} * x_{n-1}$$

Start with x_0 and then calculate x_1, then x_2, and onward...

The output of the program?

- $x_0 = 100000$ # initial amount
- $p = 3.92$ # interest rate
- $N = 6$ # number of years

```
At year =  0 Current value is =  103920.0
At year =  1 Current value is =  107993.664
At year =  2 Current value is =  112227.0156288
At year =  3 Current value is =  116626.3146414896
At year =  4 Current value is =  121198.06617539376
At year =  5 Current value is =  125949.0303694692
```

Test these values online

- For example: `http://www.moneychimp.com/calculator/compound_interest_calculator.htm`