Structures: Sets and Tuples

CPSC 231: Introduction to Computer Science for Computer Science Majors I
Fall 2021

Jonathan Hudson, Ph.D.
Instructor
Department of Computer Science
University of Calgary

Wednesday, August 25 2021

UNIVERSITY OF CALGARY

Tuples?

What is a Tuple?

- A collection of values
  - Values
    - May all have the same type, or
    - May have different types
  - Each item is referred to as an element
  - Each element has an index (ORDERED)
    - Unique integer identifying its position in the tuple
  - A tuple is one type of data structure
    - A mechanism for organizing related data

Main thing to remember!

- Similar to lists, but
  - length cannot be changed
  - **Items cannot be modified (immutable)**
  - () empty tuple, (3,) length one tuple

```
aTuple = (1,"ICT",3.14)
```

Tuples

- Like a list, a tuple is a sequence type that its elements can be of any other type
- Support many of the same operations as lists
- Unlike lists, tuples are used to store data that should not be changed.
- Format
  - <tuple name> = (<value 1>, <value 2>, ... , <value n>)
- Example

```
student = ('Marc', 123456789, 9.5)
print (student)
print (student[1])
#student[2] = 10
```

```
('Marc', 123456789, 9.5)
123456789
```

TypeError: 'tuple' object does not support item assignment

UNIVERSITY OF CALGARY

Tuple

- Format:

  <list name> = (<value 1>, <value 2>, ... , <value n>)

Examples:

```
nums = (10.0, 9.0, 8.5, 5.0, 7.5)
letters = ('a', 'b', 'c', 'd', 'e', 'f', 'g’)
names = ('Marc', 'Jim', 'Ken’)
mixed = (1.0,1,”this”,True)
```

By defining the tuple memory is allocated for it

names = (x,) → Singleton tuple of one time

Regular brackets () without comma are interpreted as empty tuple

Tuple operations

Operations	Example	Description
Indexing	name[i]	Access item by index
Slicing	name[start:end:step]	Get sub-tuple
Concatenation	names1+names2	Join two tuples into larger tuple
~~Update tuple~~	Immutable	Use slicing to get sub-tuple Use concatenation to get larger tuple
Length	len(name)	Get length of tuple
Repetition	name*x	Multiply to get tuple with int x copies of its contents in order
Membership	n in name	Boolean if item is in tuple at base level
Loop	for x in name : print(x)	Iterate through each item in tuple
Index	name.index("Carl")	Returns first index of item "Carl" in tuple name

UNIVERSITY OF CALGARY

Tuple

- In effect when we return multiple values from a function we are using tuples
- The same

```
def foo():
        return x,y
def foo():
        return (x,y)
```

- A number of common languages don't have tuples a structure like tuples, and are limited to returning a single pointer of data.

Packing/Unpacking

- You can define a tuple without brackets. Python will interpret variables/expressions separated by commas.

```
x = 1,2
print(x) -> (1,2)
print(x[0]) -> 1
print(x[1]) -> 2

a,b = x
print(a) -> 1
print(b) -> 2
```

The process seen here is generally called packing, and unpacking

What is a Set?

- A collection of values
  - Values
    - May all have the same type, or
    - May have different types
  - Each item is referred to as an element
  - ~~Each element has an index~~ UNORDERED
    - ~~Unique integer identifying its position in the list~~
  - A set is one type of data structure
    - A mechanism for organizing related data

What is a Set?

- A set contains only immutable types
- A set only contains **unique!!!** elements
- A collection of values
  - Values
    - May all have the same type, or
    - May have different types
  - Each item is referred to as an element
  - ~~Each element has an index~~ UNORDERED
    - ~~Unique integer identifying its position in the list~~
  - A set is one type of data structure
    - A mechanism for organizing related data

Set

- Unlike a list/tuple, a set is unordered
- The functions for a set are very different (we can’t index/slice)
- Unlike tuples, sets can change.
- Format
  - <set name> = **{**<value>, <value>, ... , <value>**}**
- Example
  - names = {“Albert”, Brian”, ”Carl”}

Set

- Format:

<set name> = {<value 1>, <value 2>, ... , <value n>}

Examples:

```
nums = {10.0, 9.0, 8.5, 5.0, 7.5}
letters = {'a', 'b', 'c', 'd', 'e', 'f', ‘g’}
names = {'Marc', 'Jim', 'Ken’}
mixed = {1.0,1,”this”,True}
```

By defining the set memory is allocated for it

names = set() → Only way to declare an empty set

{} -> is interpreted as a empty dictionary

Set operations

Operations	Example	Description
Unique	x = {1,1,1,2,2,2,2}	x = {1,2}
Membership	n in name	Boolean if item is in set at base level
~~Concatenation~~	~~names1+names2~~	~~Join two sets into larger set~~
Update set	add(item) update(set) remove(item) discard(item) pop()	no change if duplicate add all items from other set error if no item no error random remove
Length	len(name)	Get length of tuple
~~Repetition~~	~~name*x~~	~~Multiply to get set with int x copies of its contents in order~~
Loop	for x in name : print(x)	Iterate through each item in tuple

UNIVERSITY OF CALGARY

Sets

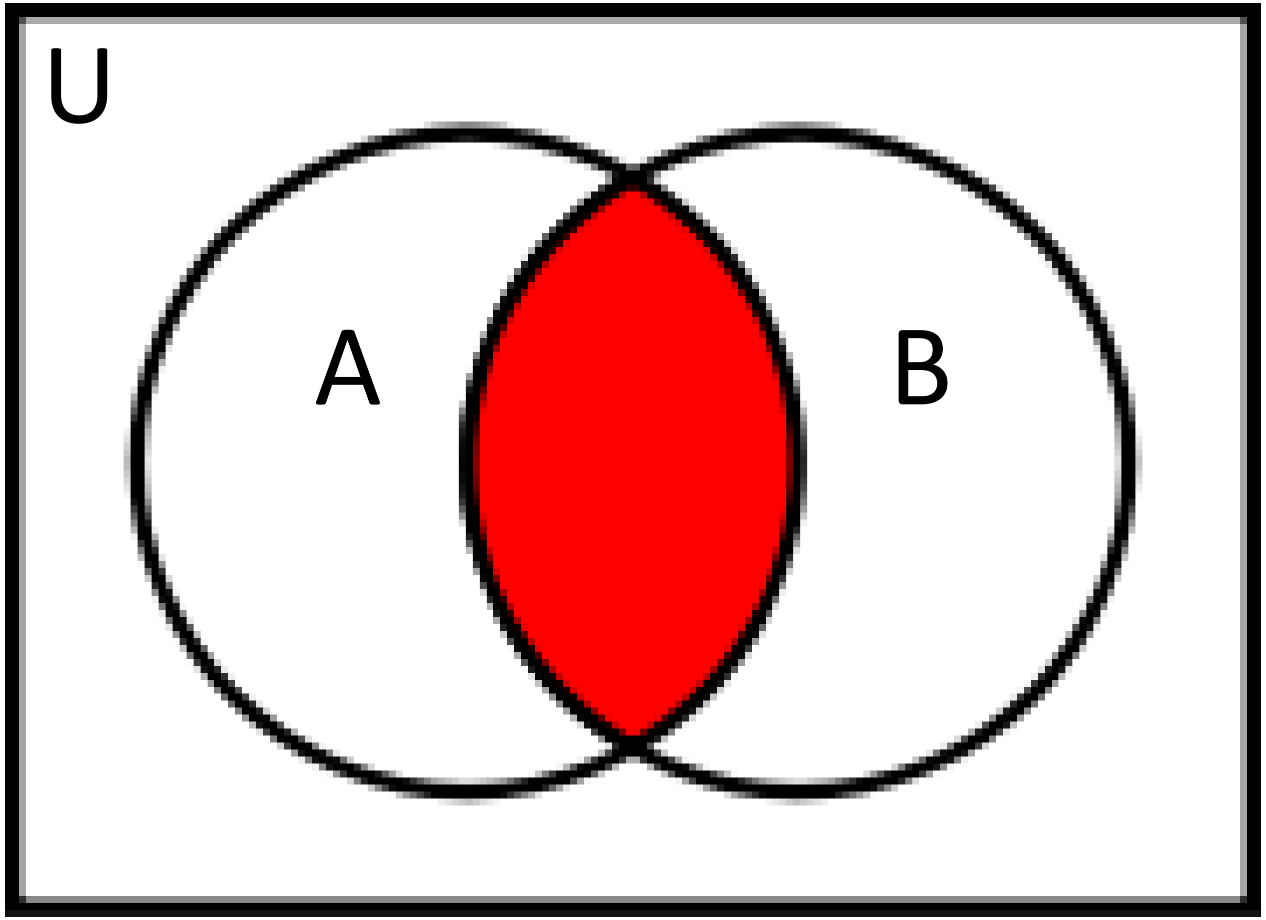
- Why do we use sets?
  - Natural uniqueness can make some things quick (we can skip membership checks)
  - Sets are rather common in many pure mathematics, logic, philosophy, and computer science (especially AI)
  - Where have you seen sets visualized (Venn Diagrams!)

Intersection (and)

Set Notation
$A \cap B$

Python

A & B



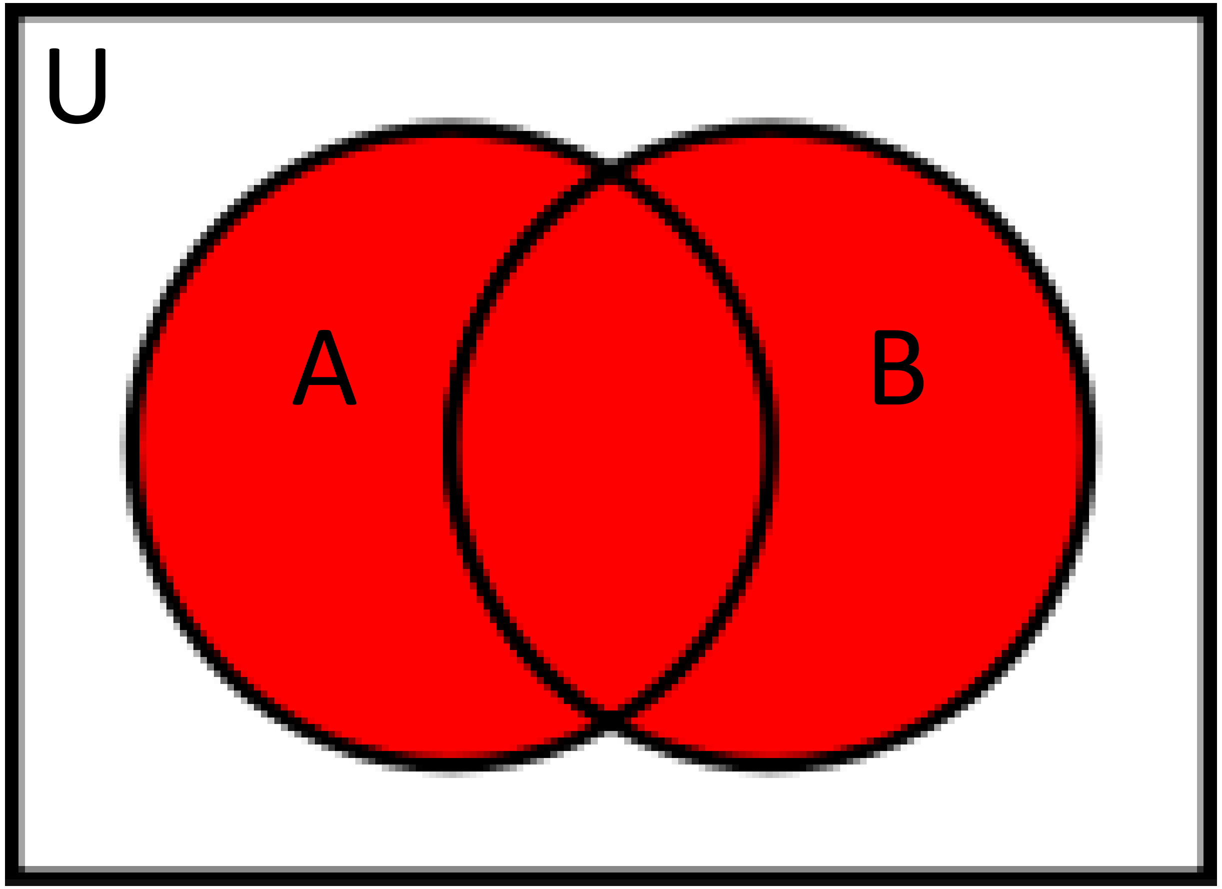
Union (or)

Set Notation

$A \cup B$

Python

A | B



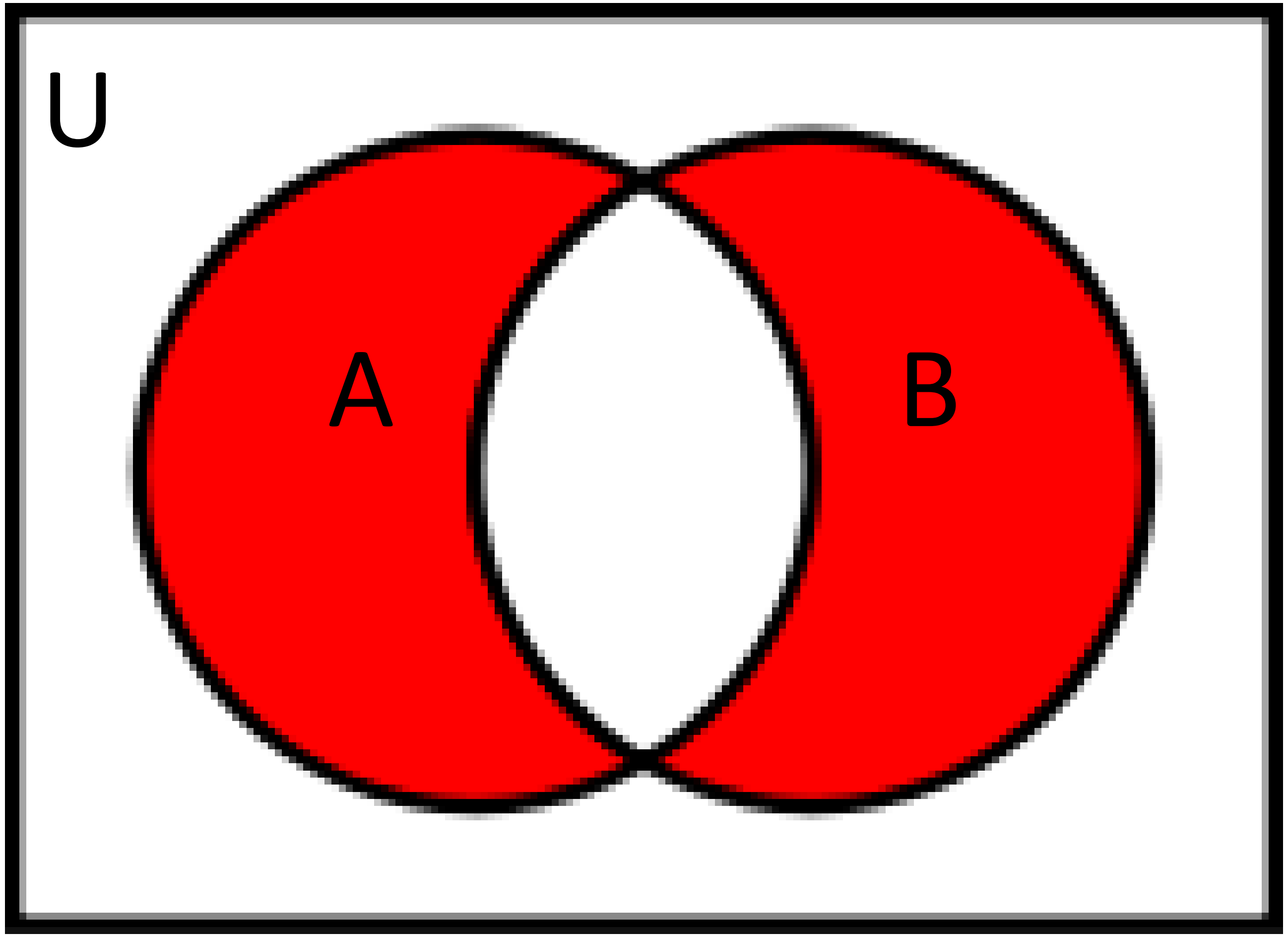
UNIVERSITY OF CALGARY

Symmetric Difference (not and)

Set Notation
$A \triangle B$

Python

A ^ B



Complement Difference

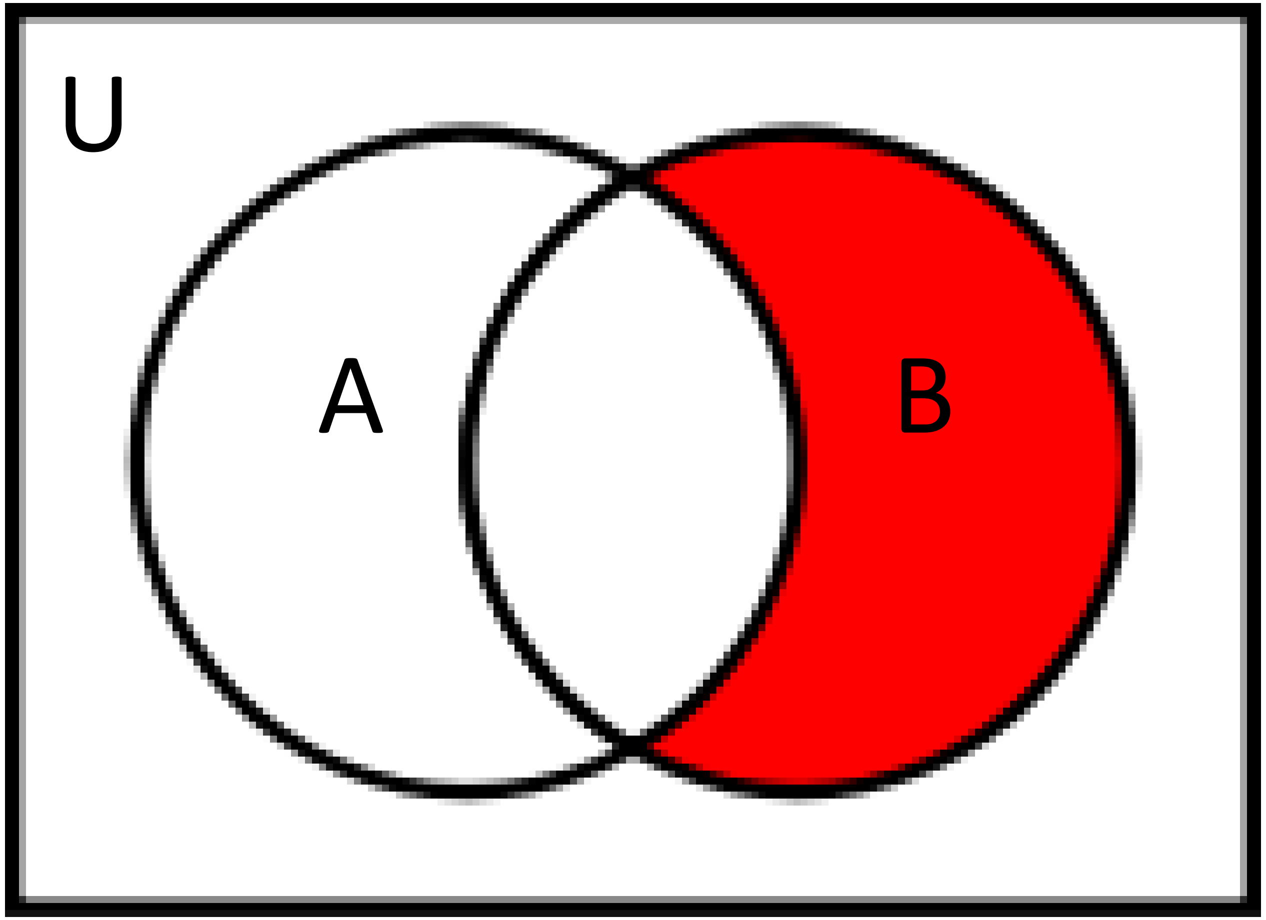
Set Notation

B \ A

$A^C \cap$ B

Python

B - A



Set questions

Operations	Example	Description
Is disjoint	x.isdisjoint(y)	True if neither x,y share an element
Is subset	x.issubset(y) OR x <= y	True if all elements in x are in y
Is superset	x.issuperset(y) OR x >= y	True if all in elements in y are in x
Equal	x==y	True if all elements in x are in y, all elements in y are in x
Not equal	x != y	True if at least one element is not in both x and y
Proper subset	x < y	x<=y and x!=y
Proper superset	x > y	x>=y and x!=y

Onward to … dictionaries.

Jonathan Hudson
jwhudson@ucalgary.ca
https://pages.cpsc.ucalgary.ca/~hudsonj/

UNIVERSITY OF CALGARY