Composites, Sets: Part 5

A composite type that is a programming implementation of a mathematical set.

James Tam

Sets: An Overview

- When to use:
  - You need a composite where **elements are unique** (not duplicates).
  - Python: adding an element that already exists will fail (no error)
- Another composite where the data can be accessed in different ways:
  - individual elements,
  - the entire collective entity **{ }**
- **Mutable** like a list: elements can change and elements are heterogeneous (don't have to be the same type).
- Associated operations are different from a list (likely because it's not ordered) e.g. indexing, slicing, concatenation, repetition operations are not available.
- Duplicates cannot be added.
  - Attempting to do this via the 'add' method will not add the duplicate element.
  - In this way it provides an automatic form of error prevention (should the data being modeled not allow for duplicates e.g. students in a university).

James Tam

Set Operations

- **Name of the full online example**: `2set_operations.py`
- Operations covered in the example: creating a set, iterating, adding, discarding, remove, popping, updating, iterating/creating a set dynamically, membership
- Invalid set operations (not operations in mathematical sets): concatenation, repetition.

James Tam

Creating A Set

- **Format:**

  `<name> = {<element1>, <element2>,... <element n>,}`

- **Example:**

```
cpsc231L01Set = {"Humid","Kin","Wai"}
```

- Note: this is not the same syntax as creating a dictionary!
  - Recall: This latter type has a key and pair for each element.

```
cpsc231L01Dictionary = {1:"Humid",
                        2:"Kin",
                        3:"Wai"}
```

- **Creating an empty set:**
  - A function must be called: `set()`

```
aSet = set()  #aSet = {} Creates a dictionary not a set
              <class 'dict'>

Note: the later examples occur after a set has already been
created.
```

James Tam

Adding New Elements

- Creating elements one at a time.

```
set1 = set()
set1.add("1")
set1.add(True)
set1.add(2)
set1.add(3)
set1.add(3)  #Duplicate not added.
```

```
True 2 3 1
```

```
1 2 3 True
```

James Tam

Iterating Through A Set

```
for element in set1:
    print(element,end=" ")
```

```
3 True 2 1
```

Recall: it is not ordered

```
set1 = set()
set1.add("1")
set1.add(True
set1.add(2)
set1.add(3)
set1.add(3)
```

James Tam

Discarding (A Single) Specified Element

```
set2 = {"Red","Blue","Green"}
set2.discard("Red")
```

- Nothing is discarded if the **element is not already in the set**

```
set2.discard("Orchre")
```

```
{'Blue', 'Green', 'Red'}
{'Blue', 'Green'}
```

James Tam

Removing (A Single) Specified Element

```
set2 = {"Red","Blue","Green"}
set2.remove("Red")
```

- Runtime error results if the element is not already in the set

```
set2.remove("Orcher jelly")
```

```
KeyError: 'Orcher jelly'
```

James Tam

Popping (Discard) An Element (Randomly Determined)

```
set2 = {"Red","Blue","Green"}
set2.pop() #Run multiple times and different element removed
```

```
Press enter to see an element popped out of the set

{'Red', 'Green'}

Press enter to see an element popped out of the set

{'Blue', 'Red'}

Press enter to see an element popped out of the set

{'Red', 'Blue'}
```

James Tam

Discarding (A Single) Element

```
set2 = {"Red","Blue","Green"}
set2.discard("Orchre")
```

- Nothing is discarded if the element is not already in the set

James Tam

Checking Length

```
set2 = {"Red","Blue","Green"}
len(set2)
```

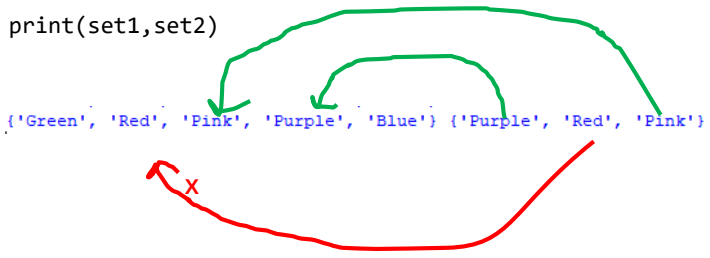
James Tam

Updating A Set (Adding One Set To Another)

```
set1 = {"Red","Blue","Green"}
set2 = {"Red","Pink","Purple"}
print(set1,set2)   {'Green', 'Red', 'Blue'} {'Purple', 'Red', 'Pink'}
set1.update(set2)
```

- Duplicates won't be added.
- The second set (the method argument, in this case set2) will be unchanged.

```
print(set1,set2)
```



James Tam

Iterating And Dynamically Creating New Elements

```
set3 = set()
for i in range(0,4,1):
    set3.add(i)
```

```
{0, 1, 2, 3}
```

James Tam

Checking For Membership (Using The 'In' Operator)

```
set3 = {1,2,3}
if 1 in set3:
    print("1 in set3")
```

```
{1, 2, 3}
1 in set3
```

James Tam

Terminology: Subset Vs. Proper Subset

- A proper subset cannot be identical to superset.
- `Set1 = {1}, Set2 = {1}`
  - Set1 is a subset of Set2
  - Set1 is NOT a proper subset of Set2 because they contain the same values.

```
Is set1 a proper subset of set2?
{1} {1}
set1 NOT a proper subset of set2
-
```

- `Set1 = {1}, Set2 = {1,2}`
  - Set1 is a subset of Set2
  - Set1 IS a proper subset of Set2 because the values they contain are not identical.

```
Is set1 a proper subset of set2?
{1} {1, 2}
set1 a proper subset of set2
```

James Tam

Why Use Python Sets?

- When mathematical set operations are needed e.g. checking if one set is a superset of another.

James Tam

Set Operations

- **Name of the full online example**: `3set_comparisons.py`
- Comparisons covered in the example: intersection, union, symmetric difference, complement difference, subset, superset, equality, inequality, proper subset, proper superset, disjoint sets.

James Tam

Terminology: Superset Vs. Proper Superset

- A proper superset cannot be identical to subset.
- `Set1 = {1}, Set2 = {1}`
  - Set1 is a superset of Set2
  - Set1 is NOT a proper subset of Set2 because they contain the same values.

```
Is set1 a proper superset of set2?
{1} {1}
set1 NOT a proper superset of set2
```

- `Set1 = {1,2}, Set2 = {1}`
  - Set1 is a subset of Set2
  - Set1 IS a proper subset of Set2 because the values they contain are not identical.

```
Is set1 a proper superset of set2?
{1, 2} {1}
set1 a proper superset of set2
```

James Tam

Other Set Comparisions

- Equality
  ```
  if(set1 == set2):
  ```
- Inequality (not equal)
  ```
  if(set1 != set2):
  ```
- Disjoint (do not share an element)

James Tam

Intersection ($A \cap B$)

```
set1 = {1}
set2 = {1,2}
set3 = set1 & set2
```

```
Intersection of set1, set2
{1} {1, 2}
{1}
```

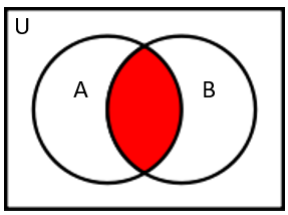


Image: Richard Zhao, Jonathan Hudson

James Tam

Union ($A \cup B$)

```
set1 = {1}
set2 = {1,2}
set3 = set1 | set2
```

```
Union of set1, set2
{1} {1, 2}
{1, 2}
```

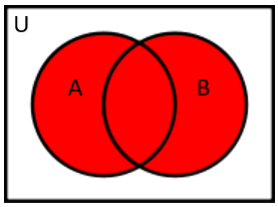


Image: Richard Zhao, Jonathan Hudson

James Tam

Symmetric Difference ($A \triangle B$)

The symmetric difference of Set1 and Set2 is the set of elements which are in either of Set1 and Set2, but not in their intersection.

```
set1 = {1,2,3}
set2 = {3,4}
set3 = set1 ^ set2
```

```
Symmetric difference of set1, set2
{1, 2, 3} {3, 4}
{1, 2, 4}
```

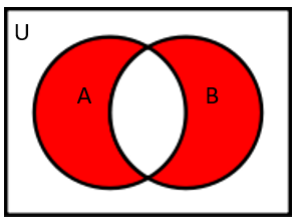


Image: Richard Zhao, Jonathan Hudson

James Tam

Complement Difference (B \ A $A^C \cap B$)

Set subtraction

```
set1 = {1,2,3}
set2 = {3,4}
set3 = set1 - set2
```

```
Complement difference of set1, set2
{1, 2, 3} {3, 4}
{1, 2}
```

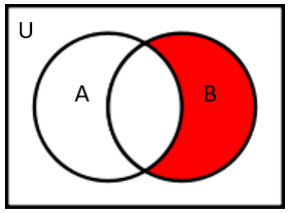


Image: Richard Zhao, Jonathan Hudson

James Tam

Subset

```
set1 = {1}
set2 = {1,2}

if(set1.issubset(set2)):
OR
if(set1 <= set2):
```

```
Is set1 a subset of set2?
{1} {1, 2}
issubset: set1 is a subset of set2
<=: set1 is a subset of set2
```

James Tam

Superset

```
set1 = {1}
set2 = {1,2}

if(set1.issuperset(set2)):
OR
if(set1 >= set2):
```

```
Is set1 a superset of set2?
{1} {1, 2}
issuperset: set1 is NOT a superset of set2
>=: set1 is NOT a superset of set2
```

James Tam

Equality

```
set1 = {1}
set2 = {1,2}

if(set1 == set2):
```

```
Inequality test: Is set1 NOT equal to set2?
{1} {1, 2}
set1 is not equal to set2
```

James Tam

Inequality

```
set1 = {1}
set2 = {1,2}

if(set1 != set2):
```

```
Equality test: Is set1 equal to set2?
{1} {1, 2}
set1 is NOT equal to set2
```

James Tam

Proper Subset

```
set1 = {1}
set2 = {1,2}

if((set1 <= set2) and (set1 != set2)):
OR
if(set1 < set2):
```

```
Is set1 a proper subset of set2?
{1} {1, 2}
set1 a proper subset of set2
```

```
set1 = {1}
set2 = {1}
```

```
Is set1 a proper subset of set2?
{1} {1}
set1 NOT a proper subset of set2
```

James Tam

Proper Superset

```
set1 = {1}
set2 = {1,2}

if((set1 >= set2) and (set1 != set2)):
OR
if(set1 > set2):
```

```
Is set1 a proper superset of set2?
{1, 2} {1}
set1 a proper superset of set2
```

```
set1 = {1}
set2 = {1}
```

```
Is set1 a proper superset of set2?
{1} {1}
set1 is NOT a proper superset of set2
```

James Tam

Disjoint Sets (Do Not Share Elements)

```
set1 = {1,2}
set2 = {2}

if(set1.isdisjoint(set2)):
```

```
Is set1 disjoint (don't share elements) from set2?
{1, 2} {2}
The sets DO share elements
```

```
set1 = {1,2}
set2 = {3}
```

```
Is set1 disjoint (does share elements) from set2?
{1, 2} {3}
The sets do NOT share elements
```

James Tam

Review Of The Composites

	String	List	Array	Dictionary	Tuple	Set
Method of creation	`""`	`[]`	`numpy.array()` or `np.array()`	`{}`	`()`	`{}`
Can be heterogeneous?	No	Yes	No	Yes	Yes	Yes
Ordered and index able?	Yes	Yes	Yes	No	Yes	No
Mutable[1]	No	Yes	Yes	Yes	No	Yes
Duplicates allowed?	Yes	Yes	Yes	Keys=no, values=yes	Yes	No

- The simple types covered thus far (int, float, bool) are all immutable.
- The final composite type (classes) is significantly different so it's covered separately (introduced at the end of the term and covered in depth in the next course).

James Tam

After This Section You Should Now Know

- Set operations: creating a set, iterating, adding, discarding, remove, popping, updating, checking for length, iterating/creating a set dynamically, membership
- Operations from mathematical sets implemented in python: intersection, union, symmetric difference, complement difference, subset, superset, equality, inequality, proper subset, proper superset, disjoint sets.
- Differences between the composite types covered thus far:
  - How to create (distinguish) between strings, lists, dictionaries, tuples, sets and arrays.
  - Which ones are mutable and which ones are immutable (you should know this for non-composites as well).
  - Which types are ordinal (elements have an ordered) and which ones can be indexed.

James Tam

After This Section You Should Now Know (2)

- Differences between the composite types covered thus far (continued):
  - Which ones allow for duplicate elements.
  - Which ones can be heterogeneous (elements don't have to all be of the same type).

James Tam

Copyright Notification

- Unless otherwise indicated, all images in this presentation were provided courtesy of James Tam.

James Tam