



# Python Coding Schools

5<sup>th</sup> lesson

**Seed Academy**

# Agenda

- wk1. Installing Python, HelloWorld
- wk2. Arithmetic Operators
- wk3. Data Types : Integer, Floating point, Boolean, String
- wk4. Data Structures: List
- wk5. Data Structures: Set, Tuples
- wk6. Data Structures: Dictionary

# Agenda

- wk7. Control flows
- wk8. Conditional
- wk9. Loops
- wk10. Function
- wk11. Class
- wk12. Data Visualization

# Class materials

<https://github.com/TaeheeJeong/seedacademy>

<https://github.com/TaeheeJeong/SummerCoding2023>

# Today's topic: Tuples & Set

## Data structure

- List
- Tuples
- Set
- Dictionary

# Tuples Are Like Lists

Tuples are another kind of sequence that functions much like a list - they have elements which are indexed starting at 0

```
>>> x = ('Glenn', 'Sally', 'Joseph')
>>> print(x[2])
Joseph
>>> y = ( 1, 9, 2 )
>>> print(y)
(1, 9, 2)
>>> print(max(y))
9
```

# but... Tuples are “immutable”

Unlike a list, once you create a tuple, you cannot alter its contents - similar to a string

```
>>> x = [9, 8, 7]
>>> x[2] = 6
>>> print(x)
>>> [9, 8, 6]
```

```
>>> y = 'ABC'
>>> y[2] = 'D'
Traceback: 'str' object
does
not support item
Assignment
```

```
>>> z = (5, 4, 3)
>>> z[2] = 0
Traceback: 'tuple' object
does
not support item
Assignment
```

# Things not to do With Tuples

```
>>> x = (3, 2, 1)
```

```
>>> x.sort()
```

```
Traceback:
```

```
AttributeError: 'tuple' object has no attribute 'sort'
```

```
>>> x.append(5)
```

```
Traceback:
```

```
AttributeError: 'tuple' object has no attribute 'append'
```

```
>>> x.reverse()
```

```
Traceback:
```

```
AttributeError: 'tuple' object has no attribute 'reverse'
```



# Tuples are More Efficient

- Since Python does not have to build tuple structures to be modifiable, they are simpler and more efficient in terms of memory use and performance than lists.
- When we are making “temporary variables” we prefer tuples over lists.

# Tuples and Assignment

- We can also put a tuple on the left-hand side of an assignment statement
- We can even omit the parentheses

```
>>> (x, y) = (4, 'fred')
>>> print(y)
fred
>>> (a, b) = (99, 98)
>>> print(a)
99
```

# Tuples are Comparable

The comparison operators work with tuples and other sequences. If the first item is equal, Python goes on to the next element, and so on, until it finds elements that differ.

```
>>> (0, 1, 2) < (5, 1, 2)
True
>>> (0, 1, 2000000) < (0, 3, 4)
True
>>> ( 'Jones', 'Sally' ) < ('Jones', 'Sam')
True
>>> ( 'Jones', 'Sally') > ('Adams', 'Sam')
True
```

# Tuples and Lists

- Lists are convenient to use and used in most cases compared to Tuples.
- Tuples are used in a specific case because of its efficiency.

# Set

- Lists and tuples store values in a sequence.
- Sets are another data type that also store values.
- Sets are a mutable collection of distinct (unique) immutable values that are unordered.
- The major difference is that sets, unlike lists or tuples, cannot have multiple occurrences of the same element and store unordered values.

# Advantage of Set

- Because sets cannot have multiple occurrences of the same element,
- Set can efficiently remove duplicate values from a list or tuple.
- Set can perform unions and intersections.

# Create a Set

You can initialize an empty set by using `set()`.

```
>>> emptySet = set()
```

To initialize a set with values, you can pass in a list to `set()`.

```
>>> dataScientist = set(['Python', 'R', 'SQL', 'Git', 'Tableau', 'SAS'])
```

```
>>> dataEngineer = set(['Python', 'Java', 'Scala', 'Git', 'SQL',  
                        'Hadoop'])
```

# Create a Set

- Sets containing values can also be initialized by using curly braces.

```
>>> dataScientist = {'Python', 'R', 'SQL', 'Git', 'Tableau', 'SAS'}
```

- curly braces {} can only be used to initialize a set containing values.
- using curly braces without values is one of the ways to initialize a dictionary and not a set.

```
>>> emptySet = set()
```

```
>>> emptyDict = dict()
```

```
>>> emptyDict = {}
```



# Add Values from Sets

- Initialize set with values.

```
>>> graphicDesigner = {'InDesign', 'Photoshop', 'Acrobat', 'Premiere',  
                        'Bridge' }
```

- You can use the method add to add a value to a set.

```
>>> graphicDesigner.add('Illustrator')
```

# Remove Values from Sets

- Option 1: remove method removes a value from a set.

```
>>> graphicDesigner.remove('Illustrator')
```

- Option 2: discard method removes a value from a set. There is no error message even though the value is not in the set.

```
>>> graphicDesigner.discard('Premiere')
```

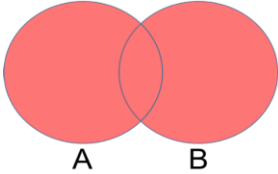
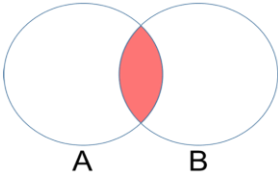
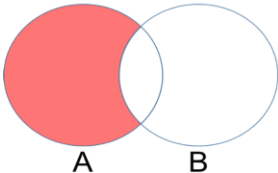
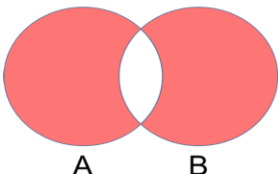
- Option 3: pop method removes and returns an arbitrary value from a set.

```
>>> graphicDesigner.pop()
```

# Remove Duplicates from a List

```
>>> list(set([1, 2, 3, 1, 7]))
```

# Set Operations

Set Operation	Venn Diagram	Interpretation
Union		$A \cup B$ , is the set of all values that are a member of $A$ , or $B$ , or both.
Intersection		$A \cap B$ , is the set of all values that are members of both $A$ and $B$ .
Difference		$A \setminus B$ , is the set of all values of $A$ that are not members of $B$
Symmetric Difference		$A \triangle B$ , is the set of all values which are in one of the sets, but not both.

# Set Operations

```
dataScientist = set(['Python', 'R', 'SQL', 'Git', 'Tableau', 'SAS'])
```

```
dataEngineer = set(['Python', 'Java', 'Scala', 'Git', 'SQL', 'Hadoop'])
```

```
dataScientist.union(dataEngineer)
```

```
dataScientist.intersection(dataEngineer)
```

```
dataScientist.difference(dataEngineer)
```

```
dataScientist.symmetric_difference(dataEngineer)
```

# Acknowledgements / Contributions



These slides are Copyright 2010- Charles R. Severance ([www.dr-chuck.com](http://www.dr-chuck.com)) of the University of Michigan School of Information and made available under a Creative Commons Attribution 4.0 License. Please maintain this last slide in all copies of the document to comply with the attribution requirements of the license. If you make a change, feel free to add your name and organization to the list of contributors on this page as you republish the materials.

Initial Development: Charles Severance, University of Michigan School of Information

Modification: Taehee Jeong, San Jose State University