Intermediate Python - Data Structures

Bill Mellman

Program Manager

Clermont County Office of
Environmental Quality

Ohio GIS Conference

September 23 – 25, 2019

Hyatt Regency Columbus

Columbus, Ohio

Taking GIS to a New Level

Intermediate Python

Data Structures

Refresher

- Nearly everything in Python is an object
- Thus all variables are pointers (references)
- Objects have Properties and Methods
- All objects have a “Data Type”
  - What is Python’s approach, is it:
    - “strongly/weakly” typed?
    - “dynamically/statically” typed?
- Python calls it “Duck Typing”
  - Ignore the type, and just look for the property/method.

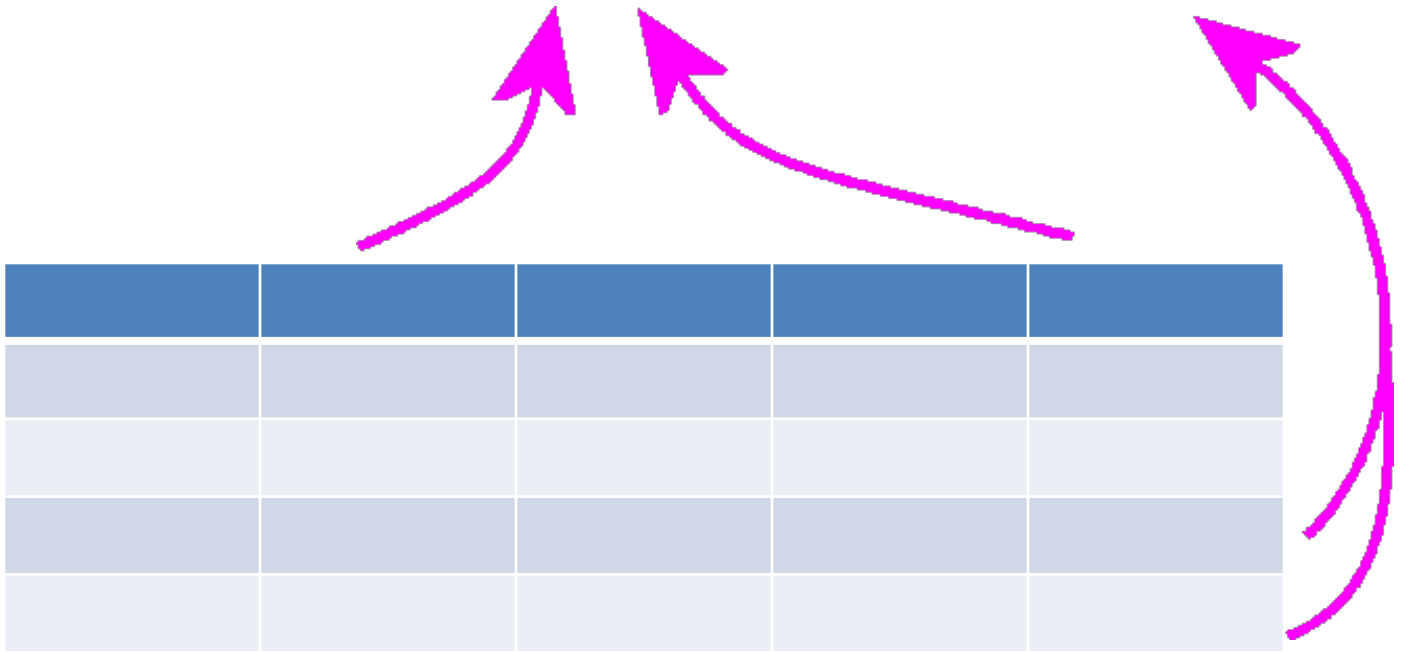
Refresher: Cursors

- Cursors are a result from a database query

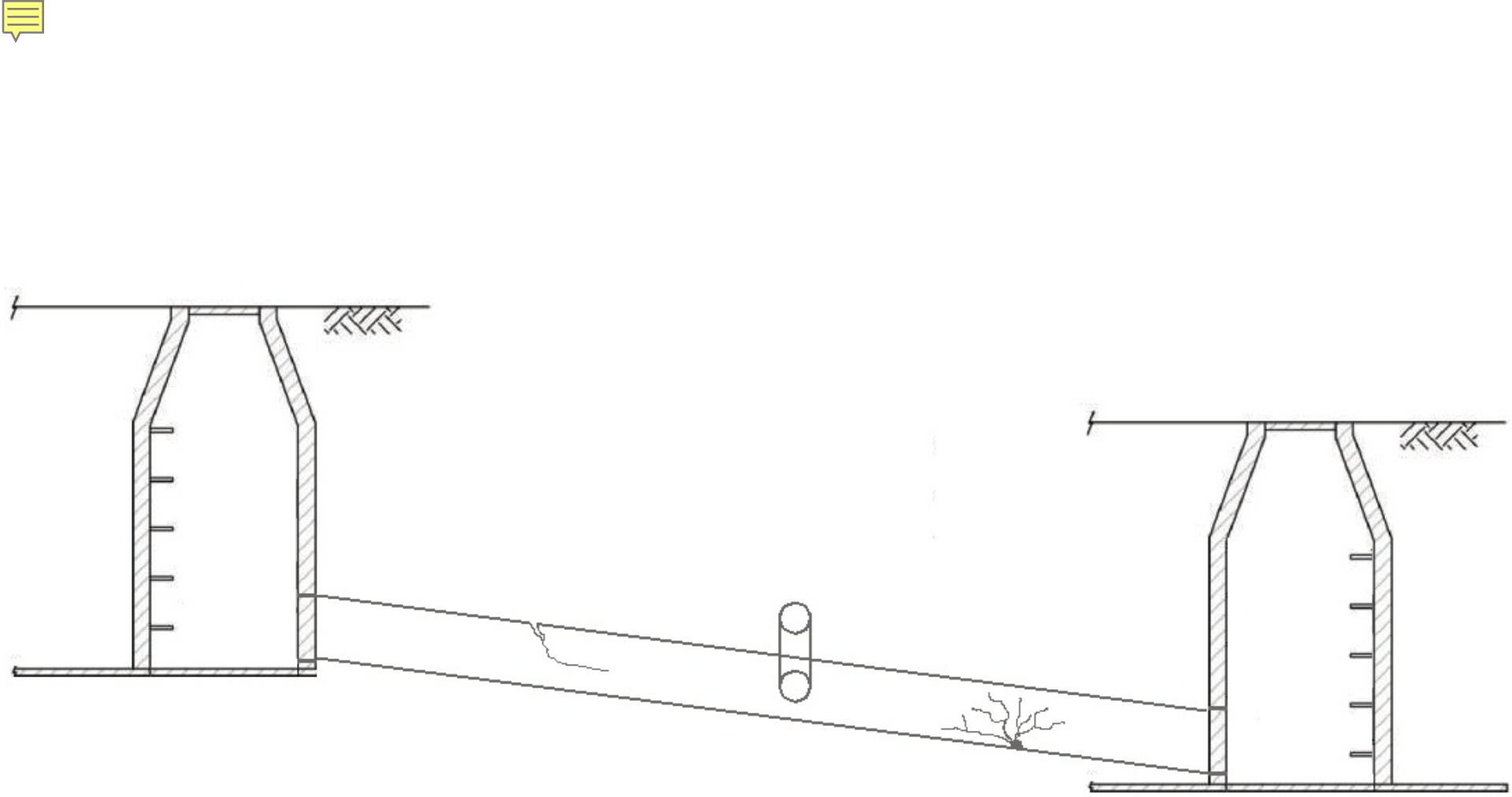
- arcpy.da.SearchCursor
  - Read an existing record
- arcpy.da.InsertCursor
  - Create a new record
- arcpy.da.UpdateCursor
  - Read and modify individual fields of an existing record
  - Delete an existing record

SearchCursor:

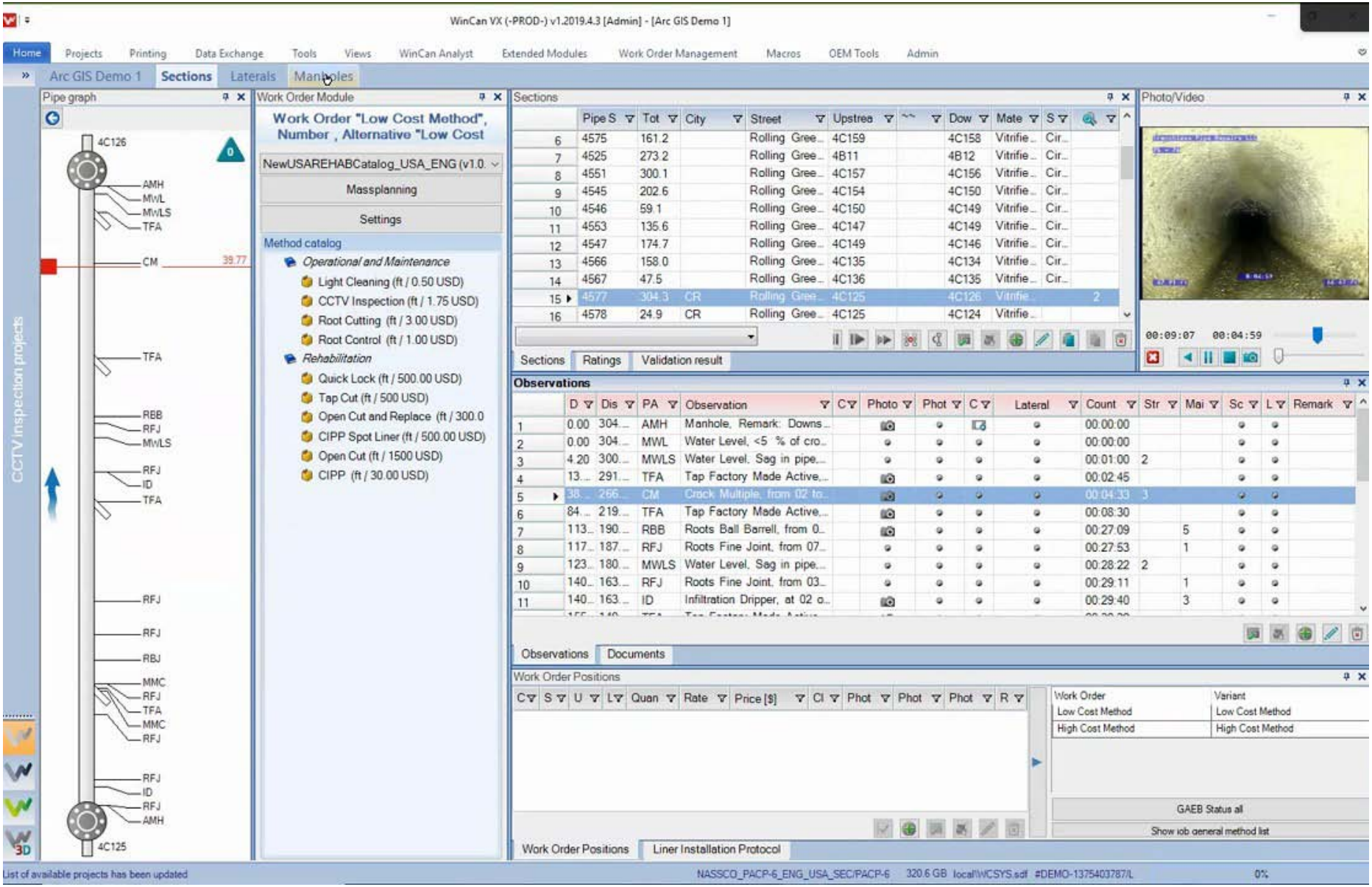
SearchCursor(table, field_names, {where_clause})



Camera Inspections



WinCan


WinCan VX (-PROD-) v1.2019.4.3 [Admin] - [Arc GIS Demo 1]
Home
Projects
Printing
Data Exchange
Tools
Views
WinCan Analyst
Extended Modules
Work Order Management
Macros
OEM Tools
Admin
Arc GIS Demo 1
Sections
Laterals
Manholes
Pipe graph
Work Order Module
Work Order "Low Cost Method", Number , Alternative "Low Cost
NewUSAREHABCatalog_USA_ENG (v1.0.
Massplanning
Settings
Method catalog
Operational and Maintenance
Light Cleaning (ft / 0.50 USD)
CCTV Inspection (ft / 1.75 USD)
Root Cutting (ft / 3.00 USD)
Root Control (ft / 1.00 USD)
Rehabilitation
Quick Lock (ft / 500.00 USD)
Tap Cut (ft / 500 USD)
Open Cut and Replace (ft / 300.0
CIPP Spot Liner (ft / 500.00 USD)
Open Cut (ft / 1500 USD)
CIPP (ft / 30.00 USD)
Sections
Photo/Video
Sections
Ratings
Validation result
Observations
Documents
Work Order Positions
Work Order
Variant
Low Cost Method
High Cost Method
GAEB Status all
Show job general method list
Work Order Positions
Liner Installation Protocol
CCTV inspection projects
List of available projects has been updated

The Problem

- MakeQueryTable in the old version
  - Joined four tables
- Complex table relationships
  - Now six tables with non-sequential relationships
- Solution: Use Python rather than arcpy

Data Structures

- Numbers and Strings
- Lists
  - sets, array module
- Tuples - immutable, so may be used as dictionary keys.
  - N- dimensional “array”
- Dictionaries
- Collections module
- Classes

List-Like Data Structures

Structure	Difference from List
Tuple	Immutable
Set	Unique
Array	Homogenous

- Use tuples when the position is important.
  - (lat, long) != (long, lat)
- Use sets when you need to know if something exists.
  - e.g. Think of sets as a group of flags
- Arrays (a module) are rare, and for special cases.
  - Not to be confused with arcpy.Array

A Note About Indexing

- # **Poor** Using “Magic Numbers”

FirstPoint = currentRow[1][0][0]

- *currentRow* is a feature
- *FirstPoint* is the first point in the list of points defining the shape of that feature.

A Note About Indexing

- # Poor Using “Magic Numbers”

```
FirstPoint = currentRow[1][0][0]
```

- # Better **# Field Names**

```
OID_     = 0   # Allows you to rearrange or add/subtract
SHAPE_   = 1
SIZE_    = 2
```

 # Object Description

```
ARRAY_   = 0   #  Helps identify the components as you
POINT_   = 0   #      go down the rabbit hole.
```

```
FirstPoint = currentRow[SHAPE_][ARRAY_][POINT_]
```

A Note About Indexing

- # A feature:
- FirstPoint = currentRow

- # An arcpy Geometry: (The Shape column of the feature table)
- FirstPoint = currentRow[SHAPE_]

- # An arcpy Array: (A list of feature parts & vertices)
- FirstPoint = currentRow[SHAPE_][ARRAY_]

- # A single point (an arcpy.Point)
- FirstPoint = currentRow[SHAPE_][ARRAY_][POINT_]

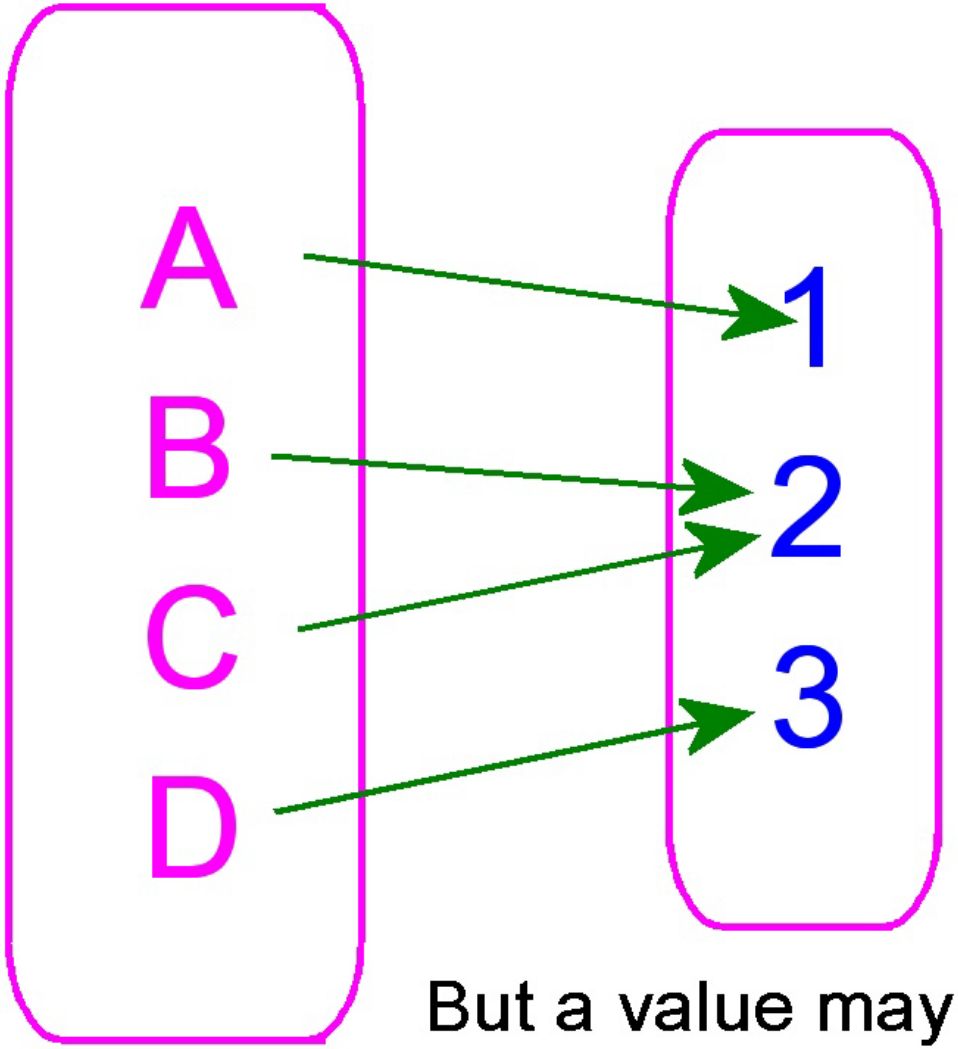
Python Dictionaries

- Hash, associative array, key-value pairs

- Key -> Value
  - A key only points to one value
  - A value may be pointed to by 1 or more keys

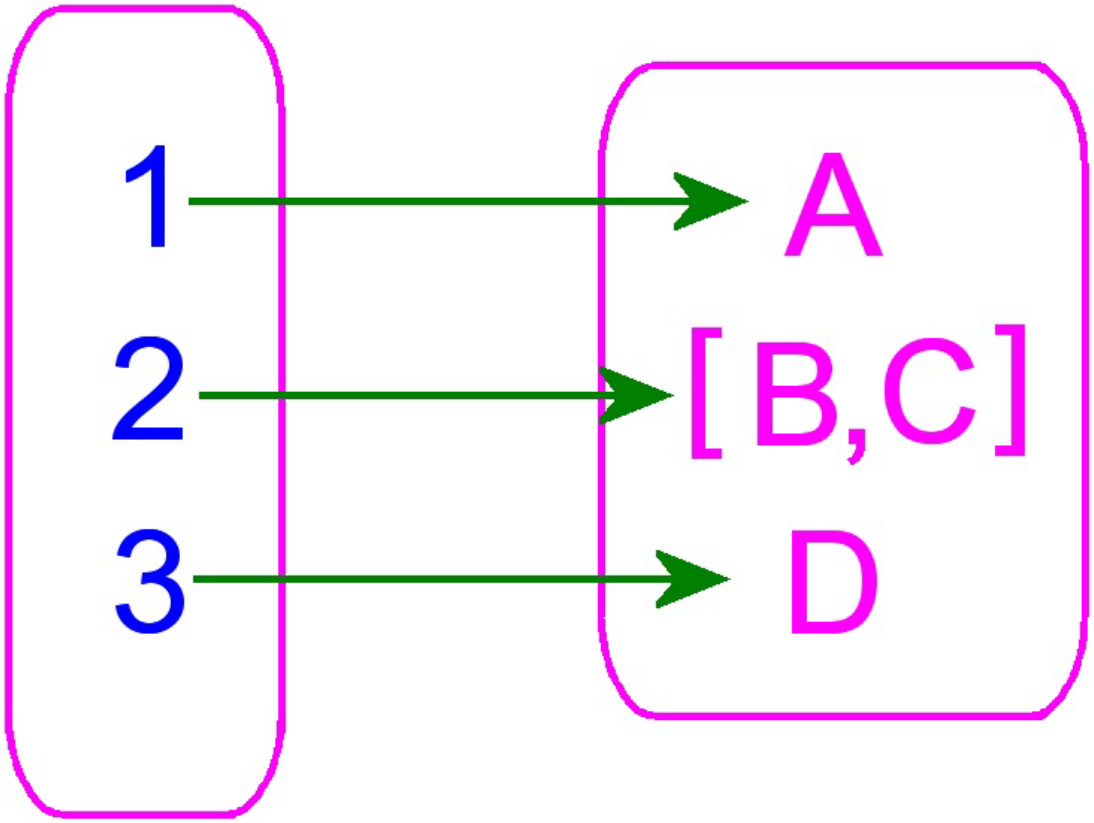
- Keys must be immutable (strings are immutable)
- Values may be any type (strings, lists, dictionaries,…)

Each key has only one value.



But a value may have more than one key.

The "inverse" of a dictionary (if it's still a dictionary).



Python Dictionaries

- { key1 : val1, key2 : val2, …}
- Name[key] = value
- .keys() # a **list** of all the keys
- .values() # a **list** of all the values
- .items() # a list of (key,value) **tuples**

```
MyDictionary = {}              # Create an empty dictionary
MyDictionary = {'c':'cat', 'd':'dog'}      # Create non-empty
MyDictionary.values()
        [‘cat', 'dog']

MyDictionary['c'] = 'dog'          # Set/change one value

MyDictionary.keys()                #  A list of the keys
        ['c','d']
MyDictionary.values()              #  A list of the values
        ['dog', 'dog']
```

Very JSON-like

- Well, not really…

- JSON is just a string with syntax reminiscent of a dictionary's syntax.
- A dictionary is a true data structure.
- Still, if you know dictionaries,

  then you essentially know JSON.

Lookup Tables

```
StateAbrev = {}

StateAbrev['Alaska'] = 'AK'
StateAbrev['Alabama'] = 'AL'
…
StateAbrev['Wyoming'] = 'WY‘
```

```
print StateAbrev['Ohio']
```

OH

Pull a cursor into a dictionary

```
fieldsList = []
fieldsList.append('OID@');      OID_    = 0   # Arcmap's …
fieldsList.append('SHAPE@');    SHAPE_  = 1   # Geometry
fieldsList.append('Size');      SIZE_   = 2   # Diam, inch
fieldsList.append('Length');    LEN_    = 3   # Camera's …

Sections = {}
with arcpy.da.SearcchCursor\
        (SectionsTable, fieldsList) as tableRows:
    for Row in tableRows:
        Sections[Row[OID_]] = Row      # *if* the OID is unique
```

Count words in a file

```
WordsCount = {}
with open(fileName,'r') as theFile:
    for line in theFile:
        for word in line.split():
            if word in WordsCount.keys():
                WordsCount[word] += 1
            else:
                WordsCount[word] = 1
```

A Common Dictionary Error

- Why don't we just use this:

```
for word in line.split():
    WordsCount[word]  += 1
```

  - KeyError
  - DefaultDict

Copying Dictionaries

- dictionary.copy()
- The copy module
  - copy.copy
  - copy.deepcopy

Useful built-in functions for objects

- type() Data type
- dir() List of methods
- id() Unique identifier for life of object
- is True if *references* are the same

Dictionary Comprehensions

- Same idea as a List Comprehension

```
fruits = ['apple','mango','banana','cherry']
D = {f:len(f) for f in fruits if len(f)>5}
        {'banana': 6, 'cherry': 6}

D1 = {'a':1, 'b':2, 'c':3, 'd':4}
DD1 = {k:v*2 for (k,v) in D.items()}
       {'a':2, 'b':4, 'c':6, 'd':8}
```

Adding Complexity

- Dictionaries (and Lists and other structures) can be multi-dimensional and complex.
- Keys could be 2-tuples or higher
- Values can be complex structures
  - Lists
  - Lists of lists
  - Lists of lists and tuples and dictionaries of lists

- WARNING:
  - Highly complex data structures can lead to hair loss…

Thank you and Happy Coding

Bill Mellman, GISP
Project Manager
Office of Environmental Quality
Clermont County Water Resources
4400 Haskell Lane
Batavia, OH 45103
Phone: (513) 732-7874
Mobile: (513) 309-4836
Fax: (513) 732-7310
Email: bmellman@clermontcountyohio.gov
Web: http://www.oeq.net/