Building a logical stream network with Python

Bill Mellman

Program Manager

Clermont County Office of Environmental Quality

Ohio GIS Conference

September 24 – 26, 2018

Hyatt Regency Columbus

Columbus, Ohio

StreamNet:

- A tool to organize spatial data in a data structure more spatially aligned to the real world than a table.

- Use a binary tree to represent a stream network

- History

Is Python easy to learn?

- Javascript is easy to learn, C is easy to learn, html is easy to learn, brain surgery is easy to learn, …..

- Besides, it's the API which is the difficult part.

Key Arcpy & Python Concepts:

- Cursors
- Geometry
- Classes (Object-Oriented Programming)

Platform:

- ArcMap vs. ArcGIS Pro
  - arcpy is arcpy
    - I changed the print statements, and StreamNet ran perfectly in Pro (Python 3.x).
  - mapping is gone in pro
    - Use the ‘mp’ module instead.
- QGIS’s point manipulation is nearly identical to arcpy.
- Classes are Python not arcpy

(Full Disclosure):

- Anything you can do with OOP you can do with Procedural Programming.
- Anything you can do recursively you can do iteratively.
- This is just another way to think about it.

But…

- Everything in Python is an object.
- A table does not capture the essence of the real world.
- This is a great learning tool.

Cursors:

- Use arcpy.da

• SearchCursor	- Read
• InsertCursor	- Create
• UpdateCursor	- Modify & Delete

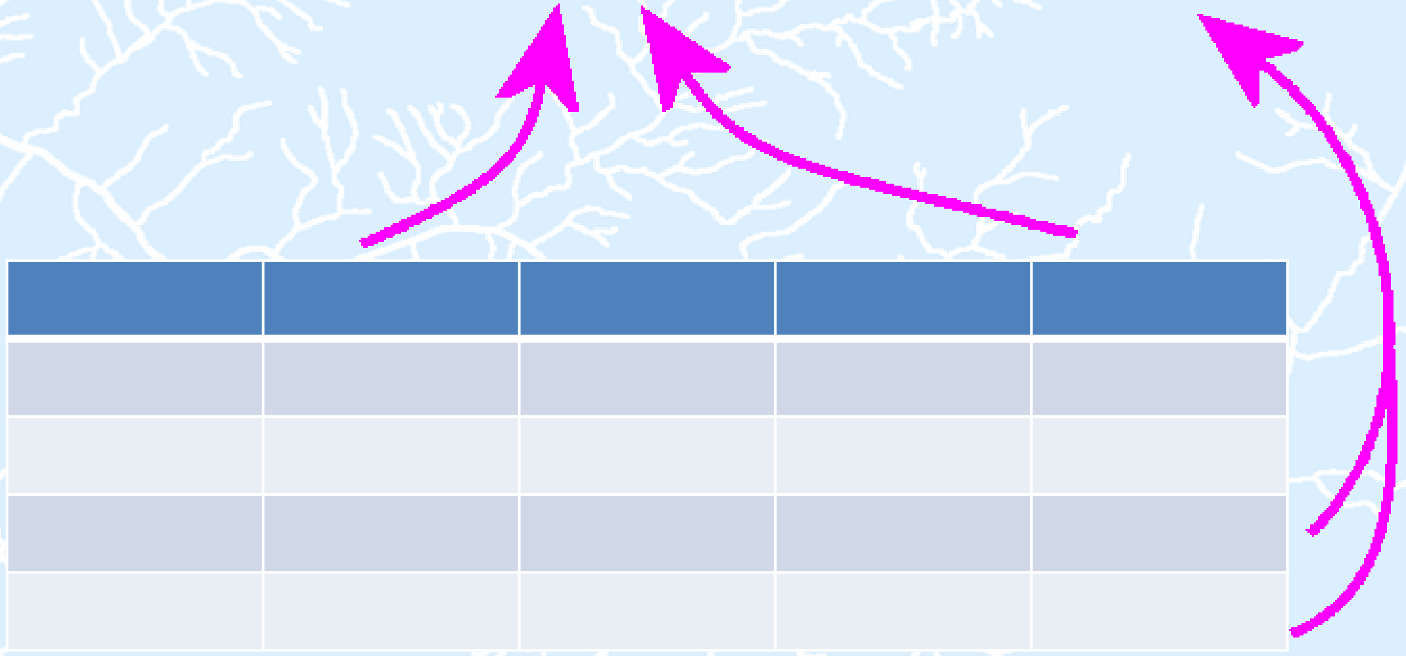
- Full load of ~40K stream lines to Python dictionary in ~ 1 second.

All three cursors use these parameters:

- The table or feature class
- The fields
  - Restricting columns
- An optional where clause
  - Restricting rows
  - Not used by the Insert Cursor

SearchCursor:

SearchCursor(table, field_names, {where_clause})



Fields:

- Use a list or tuple of quoted elements.
  - [“Name”, “Comment”]
- A tuple would be more meaningful since order is important.
- Regardless, a list is more common

Use "constants" for readability:

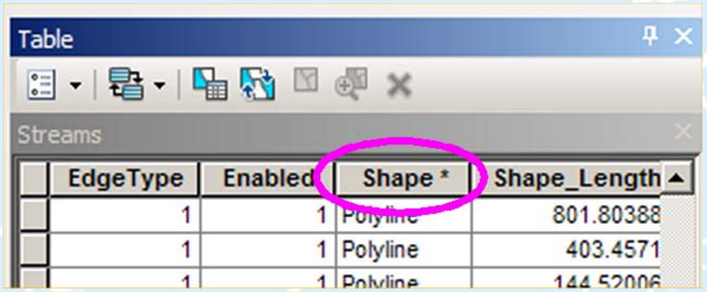
- Fields = ["Address", "City", "State"]
- Cursor rows indexed by position
  - "Address" is row[0]
- Use a "constant"
  - (a variable constrained by standards, e.g. all caps)

```
ADDRESS = 0
Row[ADDRESS]
```

Arcpy Tokens:

• "OID@"	(an Esri OBJECTID)
• "SHAPE@"	(a geometry)
• (This is the "Shape" column)	
• "SHAPE@X"	(a float)
• "*"	(everything)

What’s in a Geometry?:



Table

Streams

	EdgeType	Enabled	Shape *	Shape_Length
	1	1	Polyline	801.80388
	1	1	Polyline	403.4571
	1	1	Polyline	144.52006

```
# Use "contants" for field position
SHAPE = 0
OID   = 1

StreamsFC   = r'C:\gis\ClermontCounty.gdb\hyd_streams'

with arcpy.da.SearchCursor(StreamsFC,["SHAPE@","OID@"],
                                     "OBJECTID = 10552") as streams:
    for stream in streams:
        Question1 = stream[SHAPE]
        Question2 = Question1[0]
        Question3 = Question2[0]
        Question4 = Question3.X
```

```
<class 'arcpy.arcobjects.geometries.Polyline'>
<class 'arcpy.arcobjects.arcobjects.Array'>
<class 'arcpy.arcobjects.arcobjects.Point'>
<type 'float'>
```

Geometries:

- The Shape column contains a Geometry object
- A Geometry object contains an Array (polyline & polygon FCs) or a Point (Point FC)
- An Array is a list-like structure
  - Probably a polymorph of a list.

Multi-part features:

- An Array is composed of Points or other Arrays or both

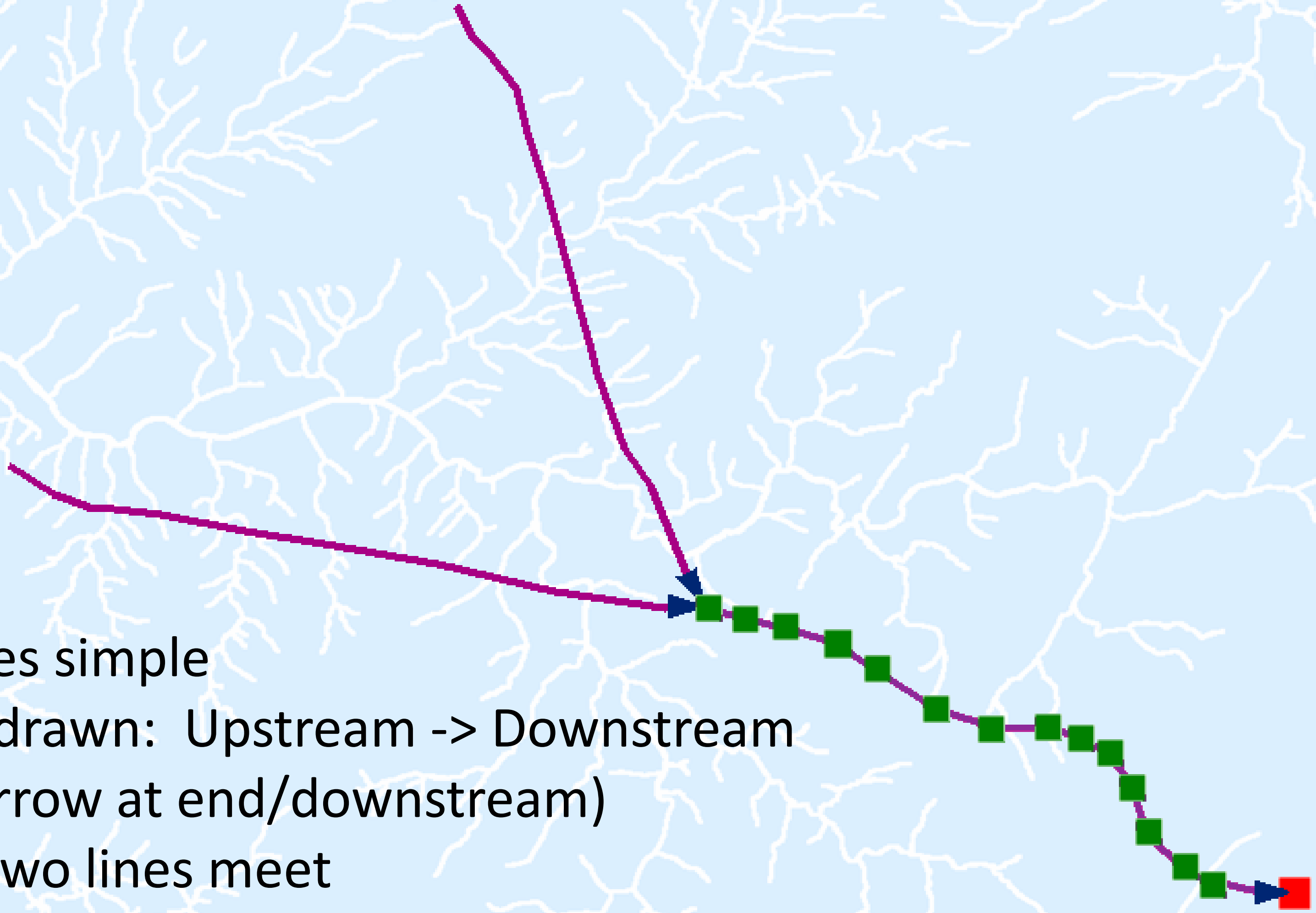
- The rabbit hole isn't very deep

- Features nest only one layer down
  - No Arrays of Arrays of Arrays.
  - If you merge two multipart features you get ONE multipart feature with the sum of the parts.

Just a thought, if it were otherwise:

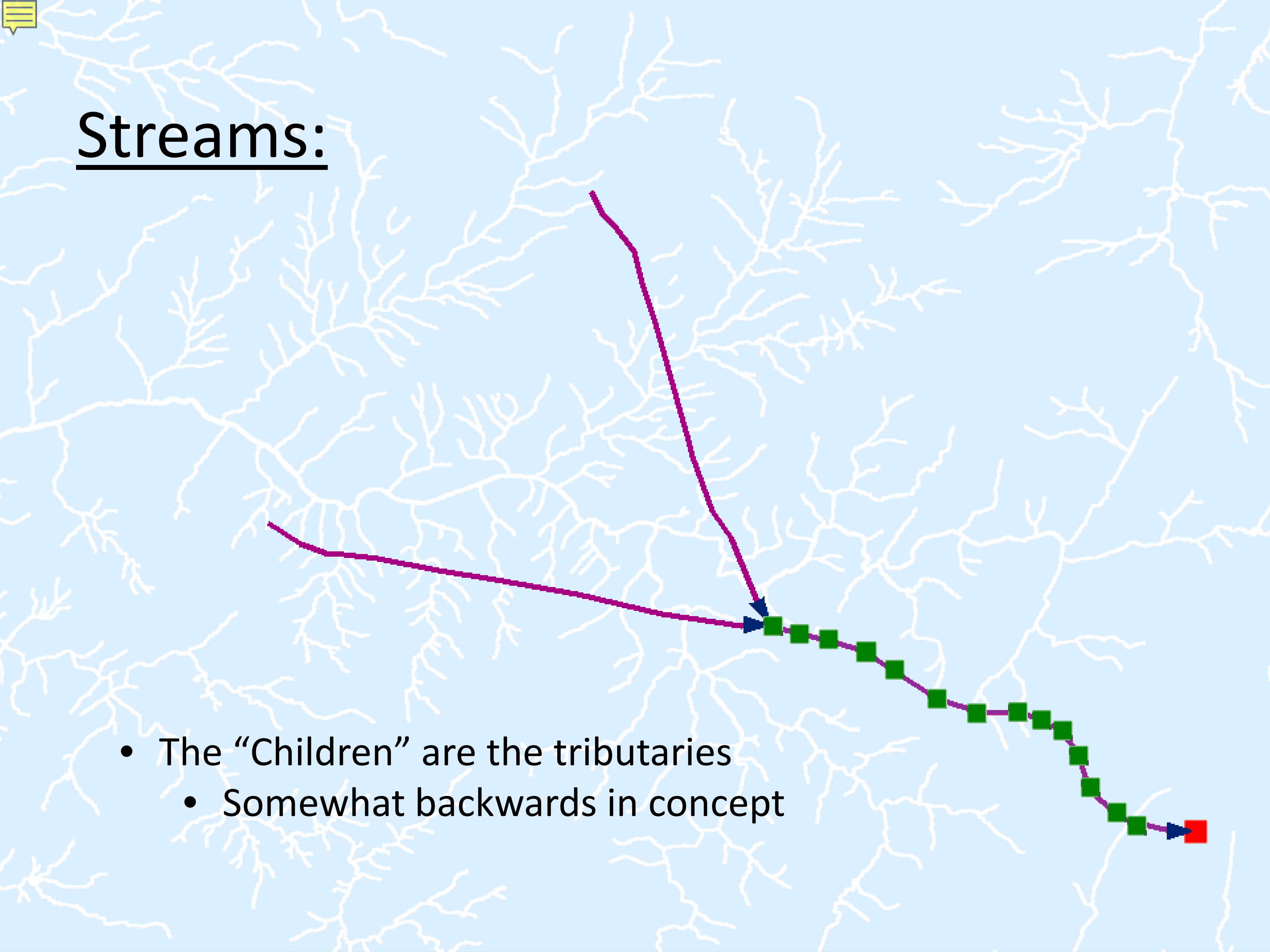
- If Arrays contained arrays of arrays of arrays of arrays …
- Then recursion would be the way to think of the problem.

Streams:



- All lines simple
- Lines drawn: Upstream -> Downstream
  - (arrow at end/downstream)
- Only two lines meet
- Endpoints snapped to Startpoints

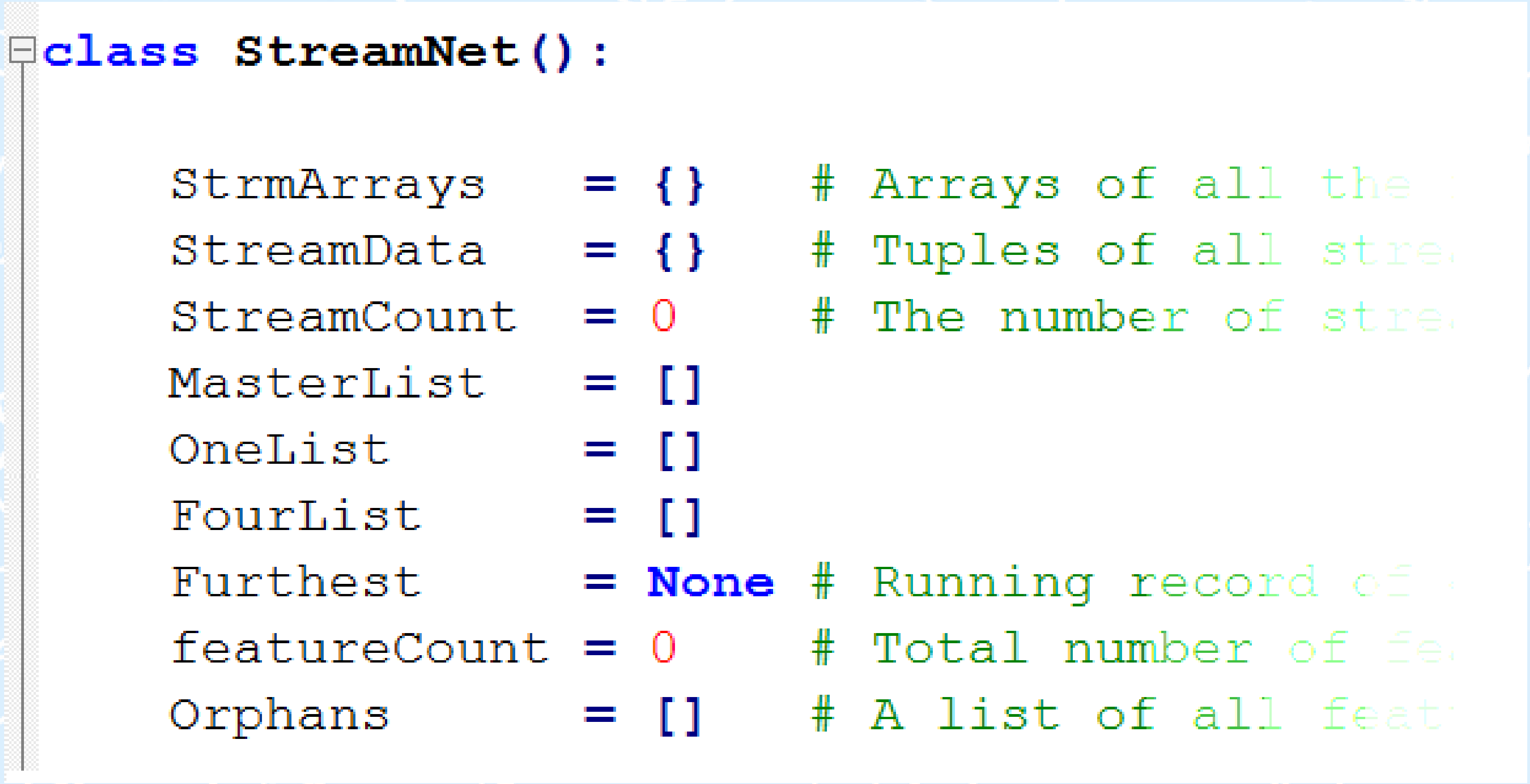
Streams:



- The “Children” are the tributaries
  - Somewhat backwards in concept

Classes:

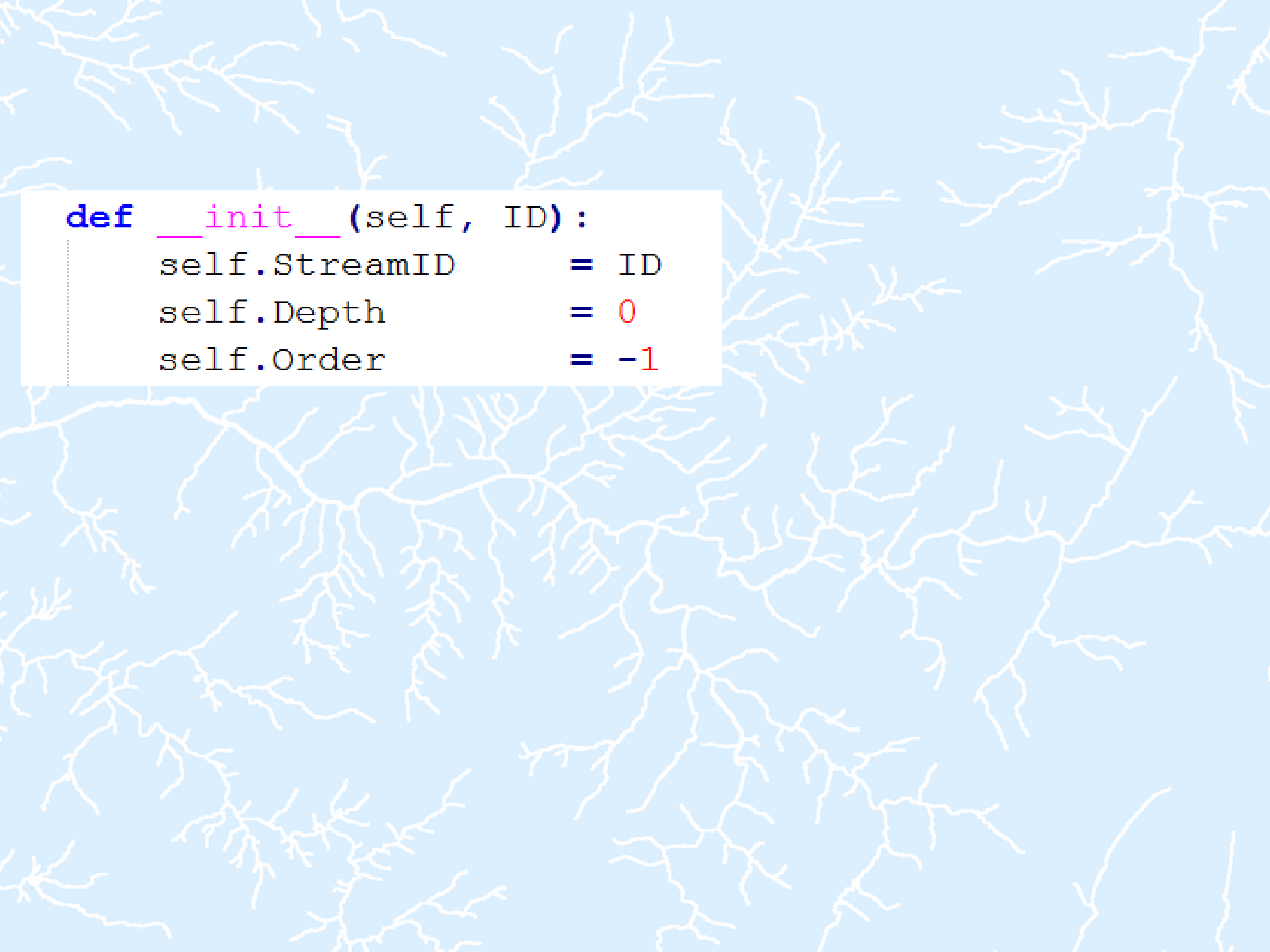
- Data-centric
- Encapsulates all the things you want to do with your data.
- Each piece of data is an object with its own methods and properties.



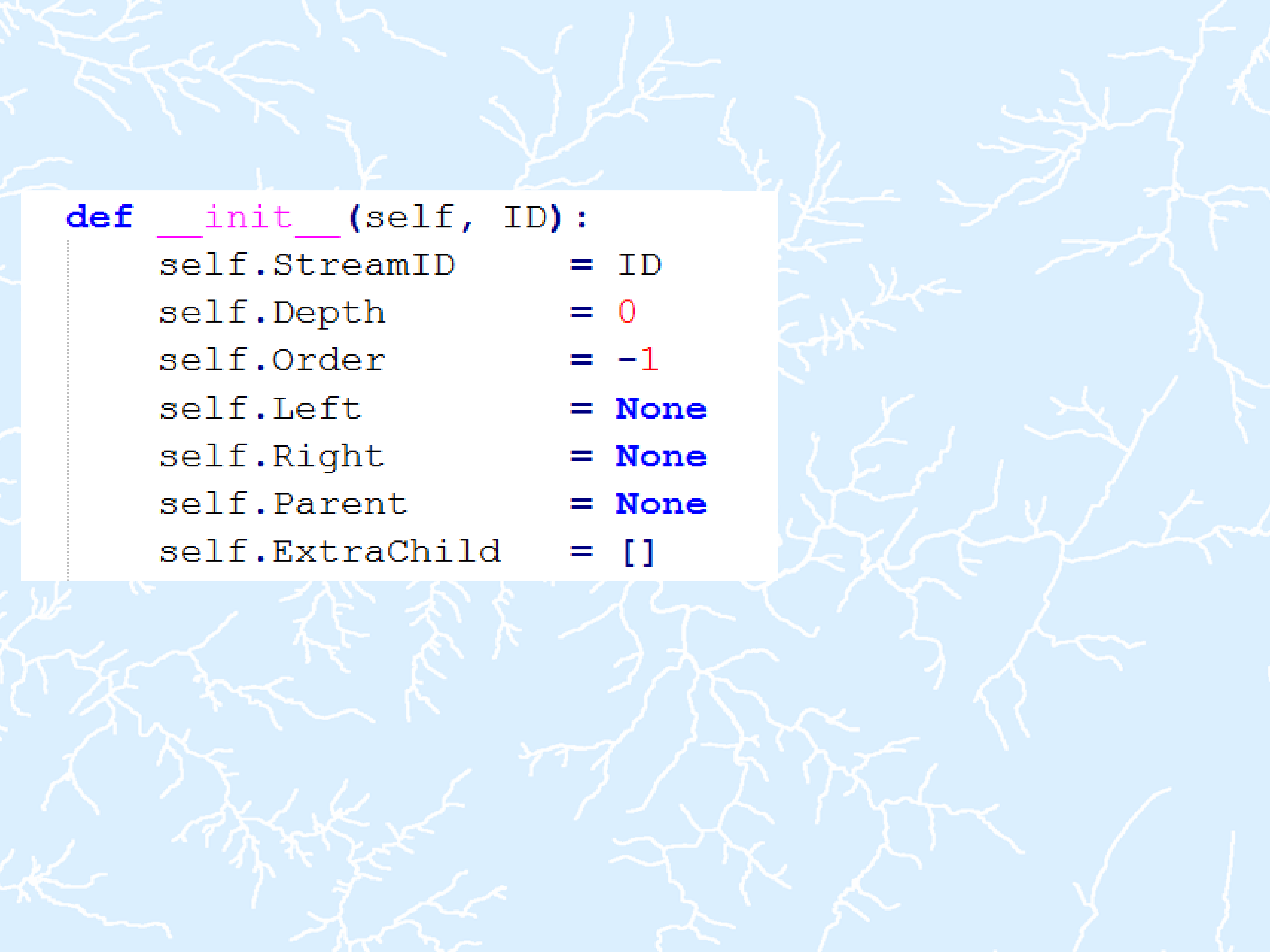
```
class StreamNet():

    StrmArrays   = {}   # Arrays of all the
    StreamData   = {}   # Tuples of all stre
    StreamCount  = 0    # The number of stre
    MasterList   = []
    OneList      = []
    FourList     = []
    Furthest     = None # Running record of
    featureCount = 0    # Total number of fe
    Orphans      = []   # A list of all feat
```

```
  def __init__(self, ID):
```

```
def __init__(self, ID):
    self.StreamID       = ID
```

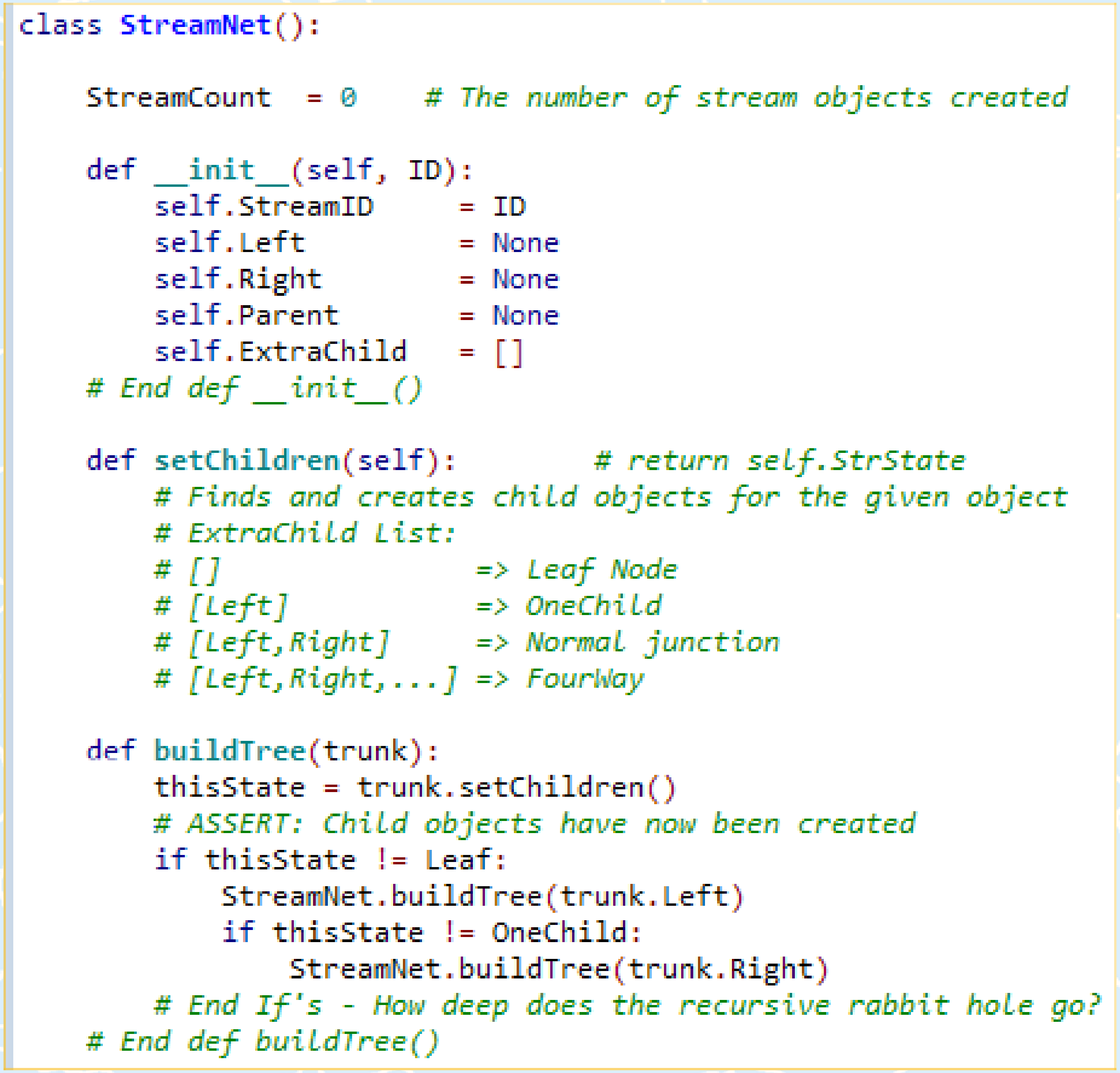


```python
def __init__(self, ID):
    self.StreamID     = ID
    self.Depth        = 0
    self.Order        = -1
```



```python
def __init__(self, ID):
    self.StreamID      = ID
    self.Depth         = 0
    self.Order         = -1
    self.Left          = None
    self.Right         = None
    self.Parent        = None
    self.ExtraChild    = []
```

```
def __init__(self, ID):
    self.StreamID      = ID
    self.Depth         = 0
    self.Order         = -1
    self.Left          = None
    self.Right         = None
    self.Parent        = None
    self.ExtraChild    = []
    self.StrState      = 0
    self.Path          = ""
    self.Length        = arcpy.Polylir
    self.Source2Start  = 0.0    # Maxir
    self.End2Mouth     = 0.0
    StreamNet.StreamCount += 1
# End def __init__()
```



```python
class StreamNet():

    StreamCount  = 0    # The number of stream objects created

    def __init__(self, ID):
        self.StreamID      = ID
        self.Left          = None
        self.Right         = None
        self.Parent        = None
        self.ExtraChild    = []
    # End def __init__()

    def setChildren(self):         # return self.StrState
        # Finds and creates child objects for the given object
        # ExtraChild List:
        # []                 => Leaf Node
        # [Left]             => OneChild
        # [Left,Right]       => Normal junction
        # [Left,Right,...] => FourWay

    def buildTree(trunk):
        thisState = trunk.setChildren()
        # ASSERT: Child objects have now been created
        if thisState != Leaf:
            StreamNet.buildTree(trunk.Left)
            if thisState != OneChild:
                StreamNet.buildTree(trunk.Right)
        # End If's - How deep does the recursive rabbit hole go?
    # End def buildTree()
```

“Variables are just names”:

- Variables are just pointers

In other languages:

```
B = A
# You’ve just created a new object ‘B’
# B is a copy of A
```

In Python:

```
B = A
# You’ve just given A a second name.
```

“Duck Typing”:

- Type checking is only performed when the call is made to a method
- “Show me the Method!”
  - Python just calls the method
  - This either works or it doesn’t

Questions?

- Slides and code will be here:
- http://stormwatermapping.com/GIS/StreamNet/