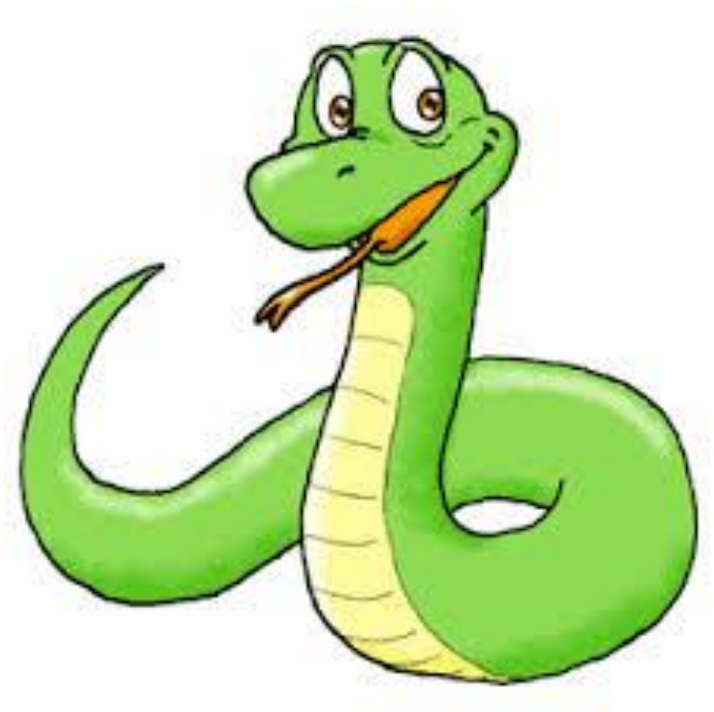
Application of GIS with Python

Chapter 5: Lists



https://www.python.org/

http://www.tutorialspoint.com/python/

Table of Content

Lists

- ➢List is a sequence of comma separated values(items) between square brackets.
- ➢Compound data types, used to group together other values (values can be any type)
  - [10, 20, 30, 40]
  - ['crunchy frog', 'ram bladder', 'lark vomit']
  - [10, 20, ‘Car’, ‘Number’, 200]

Lists

- List can contains a string, a float, an integer, and another list.
    - ['spam', 2.0, 5, [10, 20]]
- A list within another list is **nested list**.
- A list that contains no elements is called an **empty list**; can create one with empty brackets, [ ].
- List can be assigned to variables

```
>>> cheeses =['Cheddar','Edam','Gouda']

>>> numbers = [17, 123]

>>> empty = [ ]

>>> print cheeses, numbers, empty

['Cheddar', 'Edam', 'Gouda'] [17, 123] [ ]
```

Lists

- Like string indices, list indices start at 0, and lists can be sliced, concatenated and so on
  - >>> a = ['spam', 'eggs', 100, 1234]
  - >>> a

  ['spam', 'eggs', 100, 1234]
- All slice operations return a new list containing the requested elements.(returns a shallow copy)

```
>>> a[0]
'spam'
>>> a[3]
1234
>>> a[-2]
100
>>> a[1:-1]
['eggs', 100]
>>> a[:2] + ['bacon', 2*2]
['spam', 'eggs', 'bacon', 4]
>>> 3*a[:3] + ['Boo!']
['spam', 'eggs', 100, 'spam', 'eggs', 100, 'spam', 'eggs', 100,
'Boo!']
```

Lists are mutable

- Unlike strings, which are immutable, it is possible to change individual elements of a list:
  - >>> a

  ['spam', 'eggs', 100, 1234]
  - >>> a[2] = a[2] + 23
  - >>> a
  - ['spam', 'eggs', 123, 1234]

Lists are mutable

➢Assignment to slices is also possible, and this can even change the size of the list or clear it entirely

```
>>> a[0:2] = [1, 12] # Replace some items
>>> a
[1, 12, 123, 1234]
>>>a[0:2] = [] # Remove some:
>>> a
[123, 1234]
>>>a[1:1] = ['bletch', 'xyzzy'] # Insert some:
.>>> a
[123, 'bletch', 'xyzzy', 1234
```

```
>>> a[:0] = a     # Insert (a copy of) itself at the beginning
>>>  a
[123, 'bletch', 'xyzzy', 1234, 123, 'bletch', 'xyzzy', 1234]

>>> a[:] = [ ]  # Clear the list: replace all items with an empty list
>>> a
[ ]
```

Nested list

- List can contain other lists

```
>>> q = [2, 3]
>>> p = [1, q, 4]
>>> len(p)
3
>>> p[1]
[2, 3]
>>> p[1][0]
2
```

```
>>> p[1].append('xtra')
>>> p
[1, [2, 3, 'xtra'], 4]

>>> q
[2, 3, 'xtra']
```

Nested list

➢ They are a powerful tool but they need to be used carefully

```
>>> mat = [
            [1, 2, 3],
            [4, 5, 6],
            [7, 8, 9],
              ]
>>> print [[row[i] for row in mat] for i in [0, 1, 2]]    #swaps rows and
columns
[[1, 4, 7], [2, 5, 8], [3, 6, 9]]
```

‘in’ operator

- If we want to know whether a list contains a certain item but we're not interested in it's position then we can use the in operator:

```
>>>person = ['name', 'country', 'os']
>>>'name' in person
True
```

Lists and strings

- A string is a sequence of characters and a list is a sequence of values, but a list of characters is not the same as a string.

- from a string to a list of characters; **list** function

```
>>> s = 'spam'
>>> t = list(s)
>>> print t
['s', 'p', 'a', 'm']
```

- string into words; **split** method

```
>>> s = 'pining for the fjords'
>>> t = s.split()
>>> print t
['pining', 'for', 'the', 'fjords']
```

- An optional argument called a **delimiter** specifies which characters to use as word boundaries.

```
>>> s = 'spam-spam-spam'
>>> delimiter = '-'
>>> s.split(delimiter)
['spam', 'spam', 'spam']
```

- **Join;** string method is the inverse of split.

```
>>> t = ['pining', 'for', 'the', 'fjords']
>>> delimiter = ' '
>>> delimiter.join(t)
'pining for the fjords'
```

```
>>> a = 'banana'
>>> b = 'banana'
>>> a is b
True
```

```
>>> a = [1, 2, 3]
>>> b = [1, 2, 3]
>>> a is b
False
```

Lists methods

- **list.append(x)** : Add an item to the end of the list.
- **list.extend(L)** : Extend the list by appending all the items in the given list L.
- **list.insert(i, x)** : Insert an item at a given position. The first argument is the index of the element before which to insert
- **list.remove(x)** : Remove the first item from the list whose value is x. It is an error if there is no such item.
- **list.pop():** Remove the item at the given position in the list, and return it
- **list.index(x)** : Return the index in the list of the first item whose value is x. It is an error if there is no such item.
- **list.count(x)** :Return the number of times x appears in the list.
- **list.sort()** : Sort the items of the list, in place
- **list.reverse()** : Reverse the elements of the list, in place

Lists methods

Adding

```
>>> li = ['a', 'b', 'c']
>>> li.extend(['d', 'e', 'f'])
>>> li
['a', 'b', 'c', 'd', 'e', 'f']
>>> len(li)
6
>>> li[-1]
'f'
>>> li = ['a', 'b', 'c']
>>> li.append(['d', 'e', 'f'])
>>> li
['a', 'b', 'c', ['d', 'e', 'f']]
>>> len(li)
4
>>> li[-1]
['d', 'e', 'f']
```

Searching

```
>>> li=['a', 'b',
'new', 'mpilgrim',
'z', 'example', 'new',
'two', 'elements']
>>>
li.index("example")
5
>>> li.index("new")
2
>>> li.index("c")
Traceback
(innermost last):
  File "<interactive
input>", line 1, in ?
ValueError:
list.index(x): x not
in list
>>> "c" in li
False
```

Deleting

```
>>> li=['a', 'b', 'new', 'mpilgrim', 'z',
'example', 'new', 'two', 'elements']
>>> li.remove("z")
>>> li
['a', 'b', 'new', 'mpilgrim', 'example',
'new', 'two', 'elements']
>>> li.remove("new")
>>> li
['a', 'b', 'mpilgrim', 'example', 'new',
'two', 'elements']
>>> li.remove("c")
Traceback (innermost last):
  File "<interactive input>", line 1, in
?
ValueError: list.remove(x): x not in
list
>>> li.pop()
'elements'
>>> li
['a', 'b', 'mpilgrim', 'example', 'new',
'two']
```

del statement also deletes list values as

```
>>> a = [-1, 1, 66.25,
333, 333, 1234.5]
>>> del a[0]
>>> a
[1, 66.25, 333, 333,
1234.5]
>>> del a[2:4]
>>> a
[1, 66.25, 1234.5]
>>> del a[:]
>>> a
[]
```

List as matrices

- Matrix by List of lists; nested lists

```
matrix = [
        [1, 2, 3, 4],
        [5, 6, 7, 8],
        [9, 10, 11, 12],
      ]
```

Visualized as 3 X 4 matrix

```
    1   2   3    4
    5   6   7    8
    9    10  11  12
>>>mattrix [0][1]
2
```

- Transposed matrix result

```
>>> transposed = [ ]
>>> for i in range(4):
                    transposed.append([row[i] for row in matrix])
>>> transposed
[[1, 5, 9], [2, 6, 10], [3, 7, 11], [4, 8, 12]]
```

- Alternative

```
>>> [[row[i] for row in matrix] for i in range(4)]
[[1, 5, 9], [2, 6, 10], [3, 7, 11], [4, 8, 12]]
```

List as matrices

- Matrix addition computation

```
>>>m1 = [ [1, 2, 3, 0], [4, 5, 6, 0], [7, 8, 9, 0] ]        #3 X 4 matrix
>>>m2 = [ [2, 4, 6, 0], [1, 3, 5, 0], [0, -1, -2, 0] ]          #3 X 4 matrix
>>>m3= [ 4*[0] for i in range(3) ]                 #3 X 4 matrix initialized to 3 rows of 4 zeros
>>>for i in range(3):                                              #iterate through all rows
            for j in range(4):                                     #iterate through all columns
            m3[i][j]= m1[i][j]+m2[i][j]                  #compute addition and assign to m3
```

Tuple and Set

- A **tuple** consists of a number of values separated by commas, may be enclosed by ( ).
- Immutable, can be nested, sliced, concatenate

```
>>> a=(2,3,1,4)
>>> type(a)
<type 'tuple'>
>>> b=1,2,3,4,5
>>> type(b)
<type 'tuple'>
```

- A **set** is an unordered collection with no duplicate elements
- Support mathematical operations like union, intersection, difference, and symmetric difference

```
>>> a = set('abracadabra')
>>> b = set('alacazam')
>>> a                        # unique letters in a
set(['a', 'r', 'b', 'c', 'd'])
>>> a - b                     # letters in a but not in b
set(['r', 'd', 'b'])
>>> a | b                     # letters in either a or b
set(['a', 'c', 'r', 'd', 'b', 'm', 'z', 'l'])
>>> a & b                     # letters in both a and b
set(['a', 'c'])
>>> a ^ b                # letters in a or b but not both
set(['r', 'd', 'b', 'm', 'z', 'l'])
```

Assignment 5:

- What Is LIST? Describe the LIST using Python Example.
- List are Mutable, Justify the statement with suitable python example.
- What do you understand by Nested List? Write python example to illustrate Nested list.
- Compare List and Strings with suitable Python Example.
- List out and Describe List Methods with suitable Python Example.
- Write Short notes:
- Tuple and Set
- ‘in’ operator