

Data structures

Week 2, day 1

Lists

- Heterogenous array
- Sequences
 - can use `len()` on
 - can iterate over
 - uses slice notation, e.g. `[1]`, `[1,4]`, `[1:]`, `[:4]`
 - can test with `in`, `min`, `max`

Lists 2

- **Mutable**
 - can add and remove from end with `append()` and `pop()`
 - methods like strings, e.g. `count()` and `index()`
 - `append(item)` vs `append(list)` vs `extend(list)`
- Can convert to list with `list()`

List sorting

```
>>> myLst = [1, 10, 5, 15]
>>> myLst.sort()
>>> myLst
[1, 5, 10, 15]
>>> def myCompare (a, b):
        if (b < a):
            return -1
        elif (a < b):
            return 1
        else:
            return 0
>>> myLst.sort (myCompare)
```

Tuple

- Just like a list, but immutable
- Tuples are everywhere ...

- function arguments:

```
def myfunc (a, b, c): ...myfunc (3, "fasta", True)
```

- return multiple values from an argument:

```
return a, b ... x, y = myfunc()
```

Dictionary

- **Associative container**
- **keys lead to values**
 - query keys with `keys()` and `has_key()`
 - bulletproof `getitem` with `get (key, [default_value])`
- **Keys must be immutable ...**

Iterating over dictionaries

```
>>> myDict = {1: 'a', 2: 'b', 3: 'c'}
```

```
>>> for x in myDict.items():  
    print x
```

```
(1, 'a')
```

```
(2, 'b')
```

```
...
```

```
>>> for x in myDict.iteritems():  
    print x
```

```
(1, 'a')
```

```
(2, 'b')
```

```
...
```

Set

- New type
- “Like” a list
 - `set([1, 2, 3, 4])`
 - `add()`
 - no fixed order
 - all the useful set operations, e.g. `union()`, `intersection`, `&`, `|`, `issubset`, `issuperset`

Deque

- “doubled ended” queue
- `from collection import deque`
- Sequences
 - remove in the order things were added (append, pop)
 - FIFO (a stack is LIFO)

Extensions

- UserList
- UserDict
- UserString
- And objects

Python variables are references to values

```
>>> x = 3
>>> y = x
>>> x = 4
>>> x, y
(4, 3)

>>> lst1 = [1, 2, 3]
>>> lst2 = lst1
>>> lst1.append(4)
>>> lst1, lst2
([1, 2, 3, 4], [1, 2, 3, 4])
                                (need copy.copy()!)
```

Case study: amino acid data

- Have a list of amino acids and associated properties
- Need to lookup info for each amino acid

Case study: phylogeny

- A series of nodes
- Some nodes lead to other nodes
- There are distances between nodes