Level 0x06

Fancy Containers / Tricks and Techniques

Topics

- Events
- Hacker History
- Fancy Containers
  - Lists
  - Sets
  - Stack
  - Maps / Dictionary

Upcoming Events

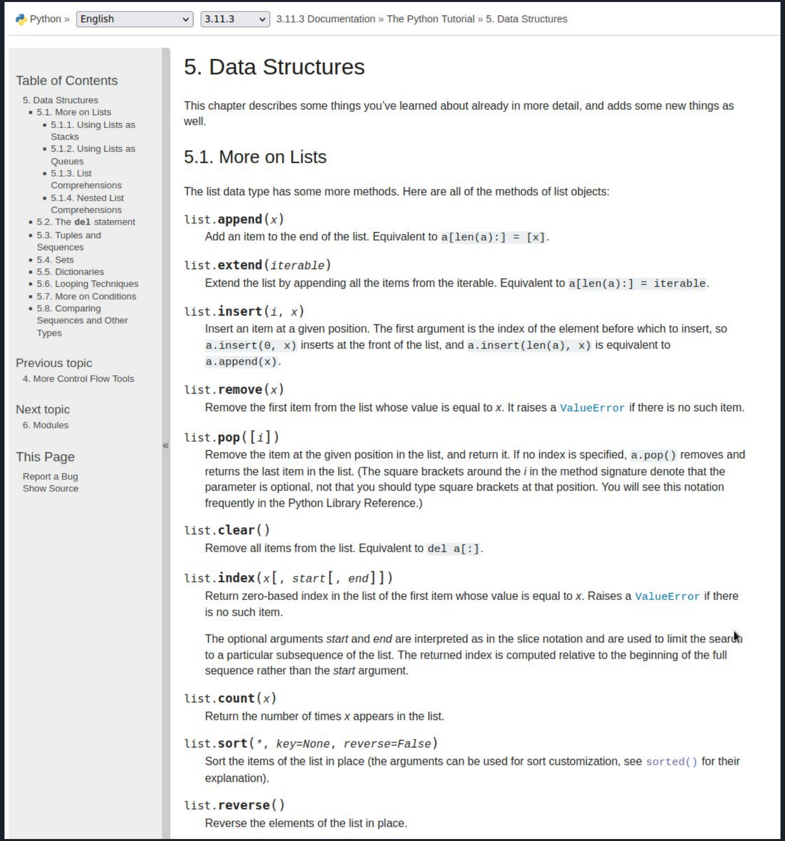
- Code Quest (Lockheed Martin, Orlando)
    - Saturday, April 22nd
    - 5 teams for West Shore

Python Lists

Lists are a list of items. Keeps track of order of items. You can change them (mutable), sort, add to them.

```
>>> mylist = [ "one", "two", "three" ]
>>> print(mylist)
['one', 'two', 'three']
>>> print(mylist[1])
two
>>> # Lists are changeable (mutable)
>>> mylist[1] = "dos"
>>> print(mylist)
['one', 'dos', 'three']
>>> mylist.insert(0,"zero")
>>> print(mylist)
['zero', 'one', 'dos', 'three']
```

Python List Documentation



Python » English 3.11.3 3.11.3 Documentation » The Python Tutorial » 5. Data Structures

Table of Contents

This Page
Report a Bug
Show Source

5. Data Structures

This chapter describes some things you've learned about already in more detail, and adds some new things as well.

5.1. More on Lists

The list data type has some more methods. Here are all of the methods of list objects:

`list.append(x)`
Add an item to the end of the list. Equivalent to `a[len(a):] = [x]`.

`list.extend(iterable)`
Extend the list by appending all the items from the iterable. Equivalent to `a[len(a):] = iterable`.

`list.insert(i, x)`
Insert an item at a given position. The first argument is the index of the element before which to insert, so `a.insert(0, x)` inserts at the front of the list, and `a.insert(len(a), x)` is equivalent to `a.append(x)`.

`list.remove(x)`
Remove the first item from the list whose value is equal to *x*. It raises a `ValueError` if there is no such item.

`list.pop([i])`
Remove the item at the given position in the list, and return it. If no index is specified, `a.pop()` removes and returns the last item in the list. (The square brackets around the *i* in the method signature denote that the parameter is optional, not that you should type square brackets at that position. You will see this notation frequently in the Python Library Reference.)

`list.clear()`
Remove all items from the list. Equivalent to `del a[:]`.

`list.index(x[, start[, end]])`
Return zero-based index in the list of the first item whose value is equal to *x*. Raises a `ValueError` if there is no such item.

The optional arguments *start* and *end* are interpreted as in the slice notation and are used to limit the search to a particular subsequence of the list. The returned index is computed relative to the beginning of the full sequence rather than the *start* argument.

`list.count(x)`
Return the number of times *x* appears in the list.

`list.sort(*, key=None, reverse=False)`
Sort the items of the list in place (the arguments can be used for sort customization, see `sorted()` for their explanation).

`list.reverse()`
Reverse the elements of the list in place.

Python Tuples

Tuples are just like lists, but are unchangeable (immutable)

```
>>> myTuple = ( "four", "five", "six" )
>>> print(myTuple)
('four', 'five', 'six')
>>> print(myTuple[1])
five
>>> myTuple[1] = "cinco"
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'tuple' object does not support item assignment
```

2-D List / Nested Lists

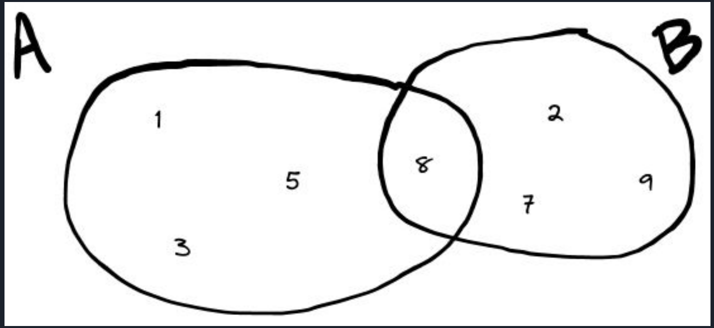
The items in a list can be of any type. We can even make a list of lists aka 2-D list.

```
>>> threeByThree = [ [ 1, 2, 3 ],
...                  [ 4, 5, 6 ],
...                  [ 7, 8, 9 ] ]
>>> print(threeByThree)
[[1, 2, 3], [4, 5, 6], [7, 8, 9]]
>>> print(threeByThree[0][1])
```

Python Sets

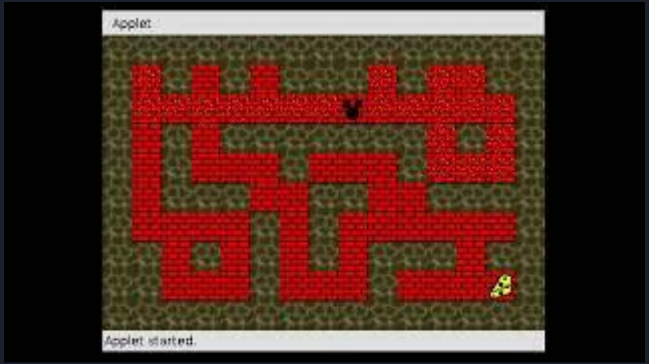
Sets are collection of items. Order doesn’t matter. Ignores duplicate items.

```
>>> setA = { 1, 3, 5, 8 }
>>> setB = set([2, 8, 9, 7])
>>> print(setA)
{8, 1, 3, 5}
>>> print(setB)
{8, 9, 2, 7}
>>> print(setA.intersection(setB))
{8}
>>> print(setA.union(setB))
{1, 2, 3, 5, 7, 8, 9}
```



Maze Traversal

- Common type of challenge problem
  - Wall-follower method for real-life maze
  - Many ways for computer to solve
- Simple Algorithm
  - Keep track of places you visited (breadcrumbs on map, a set of locations)
  - Have a list of 'moves' you made
  - Try going up, down, left, or right
    - Can't pass over walls or breadcrumbs
    - Add our move to the move list
    - Add our location to breadcrumbs
    - Each turn see if we won?
    - If we get to dead end, go back to where we came from (reverse our last move)



DRRDDRRDRDDDRRUURRRRDDR

Python Dictionary (map, associative array)

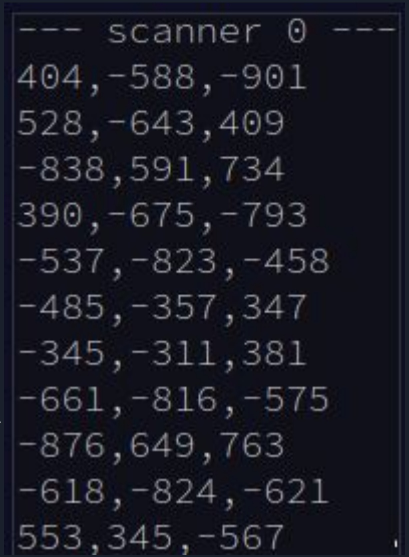
Collection of items that are stored as key:value pairs. Keys are unique, have 1 value.
Dictionaries are changeable (mutable). The keys are immutable!!

```
>>> morseCode = { 'a':".-", 'b':"-...", 'c':"-.-.",
...               'd':"-..", 'e':"." }
>>> print(morseCode)
{'a': '.-', 'b': '-...', 'c': '-.-.', 'd': '-..', 'e': '.'}
>>> print(morseCode['a'])
.-
>>> morseCode['m']="--"
>>> print(morseCode)
{'a': '.-', 'b': '-...', 'c': '-.-.', 'd': '-..', 'e': '.', 'm': '--'}
```

Sparse Structure

- Advent of Code 2021 Day 19
  - Data points in a 3-D space that was -1000 to 1000 in size.
  - Vector[2000][2000][2000] = 8GB…. Whoah…
  - Most of the space empty though…
- Use a dictionary with tuple keys

```
>>> scannerData = {}
>>> scannerData[ (404,-588,-901) ] = "scan0"
>>> scannerData[ (528,-643,409) ] = "scan1"
>>> print(scannerData)
{(404, -588, -901): 'scan0', (528, -643, 409): 'scan1'}
>>> scannerData[ [1,1,1] ] = "wont work"
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: unhashable type: 'list'
```



```
--- scanner 0 ---
404,-588,-901
528,-643,409
-838,591,734
390,-675,-793
-537,-823,-458
-485,-357,347
-345,-311,381
-661,-816,-575
-876,649,763
-618,-824,-621
553,345,-567
```

Code Quest Rules FAQ

- 1 computer per team
- Languages
  - Java 1.8 (Eclipse IDE)
  - Python 3.7 (IDLE)
  - C++ 17 (Eclipse IDE)
  - C# 6.0 (VS Code Community Edition)
- Don’t put spaces in filenames
- Web Access Help
  - Official Documentation for programming language - OK
  - Stack Overflow - NOT OK
  - Basic tutorials / learning sites like W3Schools - OK
  - Chat GPT - NOT OK
  - Github / Gitlab - NOT OK (even if it is your own repo)
- Prep guide has a solution template
  - All other pre-written code is not allowed

Get Started Today

- Decide who is bringing what computers
  - Is the OS ready?
  - If windows, do you want to install WSL too?
  - Do we want to use the Dell XPS from Florida Tech?
  - What language are you going to use?
- Read prep guide - Setup template
- Try sample challenges…

Links

- https://www.lockheedmartin.com/en-us/who-we-are/communities/codequest.html
- https://www.lockheedmartin.com/content/dam/lockheed-martin/eo/documents/code-quest/2023/2023CodeQuestPreparationEN.pdf
- https://www.lockheedmartin.com/en-us/who-we-are/communities/codequest/code-quest-official-rules.html