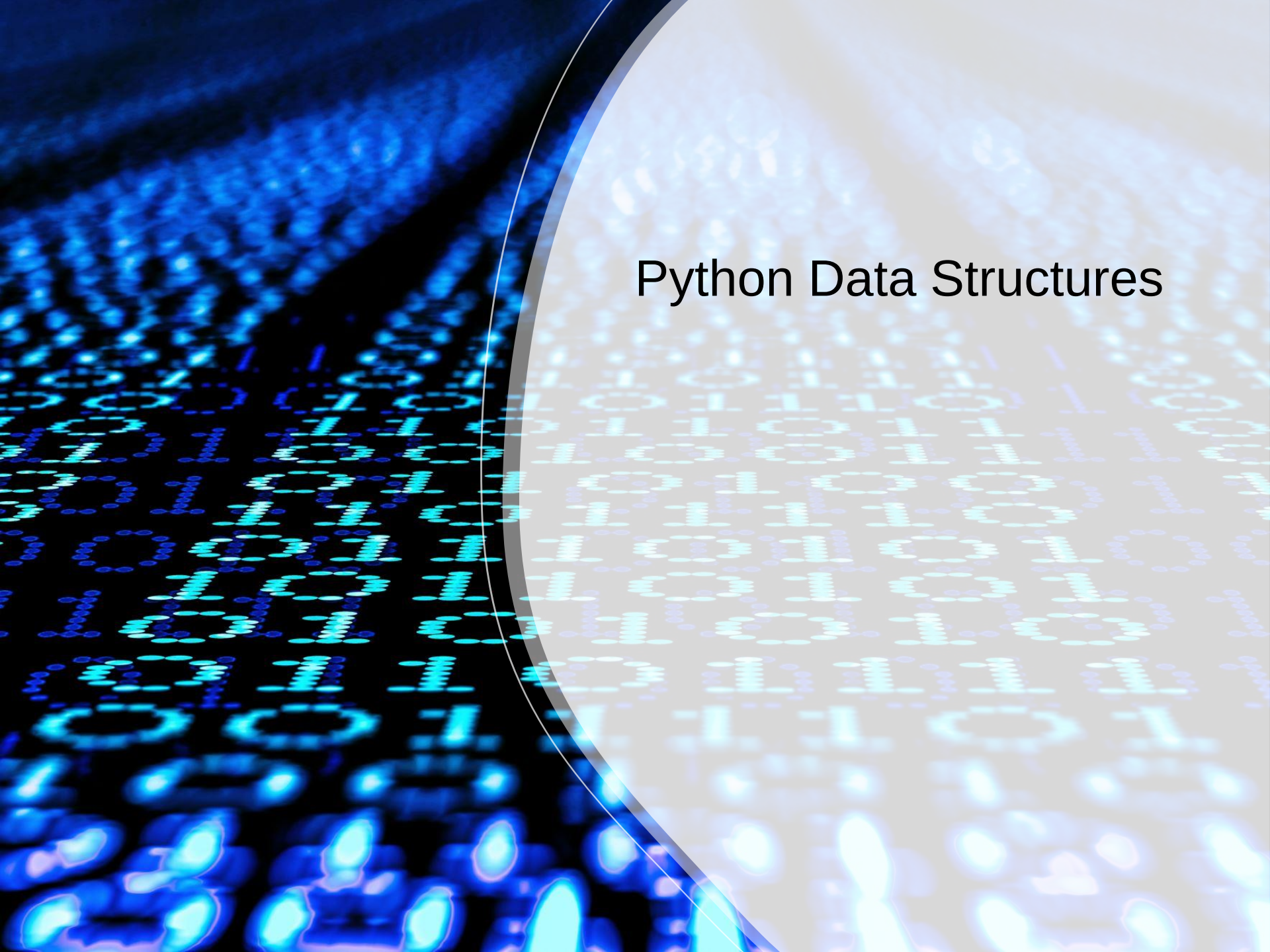

Python Data Structures

Topics

Data Structures in Python

- List
- Tuples
- Sets
- Dictionaries

Data Structures Terms

- **Data structure-**A way of organizing or storing information. So far, the main data structure we've seen is the list.
- **Ordered-**The data structure has the elements stored in an ordered sequence
- **Mutable-**The contents of the data structure can be changed.

List

used to store the sequence of various types of data.
Python lists are mutable (we can modify its element after it created)

List:

- lists are ordered.
- List elements can be accessed by index.
- lists are mutable types.
- can store the number of various elements.

List examples

```
llist1 = [“Tim", 22, “Canada"]
list2= [1, 2, 3, 4, 5, 6]

print(list1)
print(list2[0])
```

Accessing list elements

List = [0, 1, 2, 3, 4, 5]

0	1	2	3	4	5

List[0] = 0

List[1] = 1

List[2] = 2

List[3] = 3

List[4] = 4

List[5] = 5

List[0:] = [0,1,2,3,4,5]

List[:] = [0,1,2,3,4,5]

List[2:4] = [2, 3]

List[1:3] = [1, 2]

List[:4] = [0, 1, 2, 3]

Methods of Lists

List.append(x)

#adds an item to the end of the list

List.extend(L)

#Extend the list by appending all in the given list L

List.insert(I,x)

#Inserts an item at index I

List.remove(x)

#Removes the first item from the list whose value is x

Methods of Lists

List.append(x)

adds an item to the end of the list

List.extend(L)

Extend the list by appending all in the given list L

List.insert(I,x)

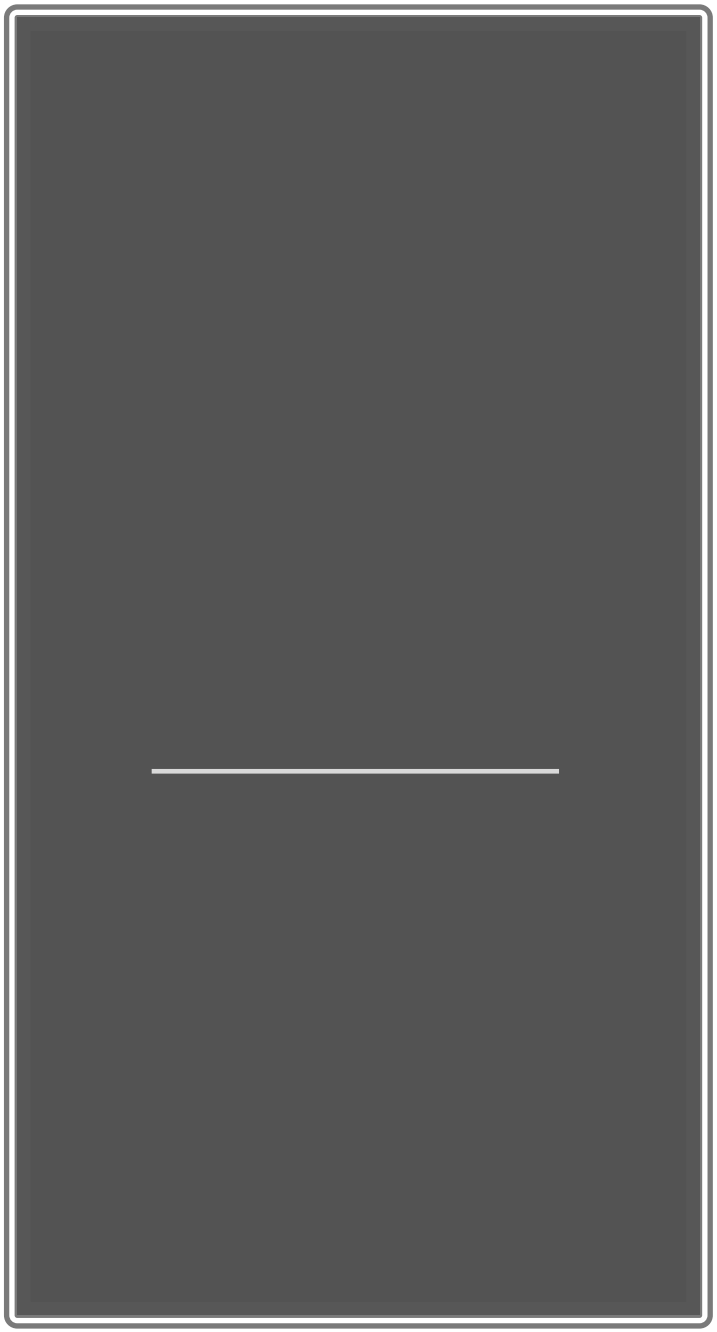
Inserts an item at index I

List.remove(x)

Removes the first item from the list whose value is x

Python List Built-in functions

SN	Function	Description	Example
1	cmp(list1, list2)	It compares the elements of both the lists.	This method is not used in the Python 3 and the above versions.
2	len(list)	It is used to calculate the length of the list.	L1 = [1,2,3,4,5,6,7,8] print(len(L1)) 8
3	max(list)	It returns the maximum element of the list.	L1 = [12,34,26,48,72] print(max(L1)) 72
4	min(list)	It returns the minimum element of the list.	L1 = [12,34,26,48,72] print(min(L1)) 12
5	list(seq)	It converts any sequence to the list.	str = "Johnson" s = list(str) print(type(s)) <class list>



Operator	Description	Example
Repetition	The repetition operator enables the list elements to be repeated multiple times.	L1*2 = [1, 2, 3, 4, 1, 2, 3, 4]
Concatenation	It concatenates the list mentioned on either side of the operator.	l1+l2 = [1, 2, 3, 4, 5, 6, 7, 8]
Membership	It returns true if a particular item exists in a particular list otherwise false.	print(2 in l1) prints True.
Iteration	The for loop is used to iterate over the list elements.	for i in l1: print(i) **Output**1 2 3 4
Length	It is used to get the length of the list	len(l1) = 4

Examples of other methods

#Defines List
a = [66.25, 333, 333, 1, 1234.5]

print(a.count(333) #calls method
2 //output

a.index(333) /#Returns the first index where the given value appears
1 //ouput

a.reverse() /#Reverses order of list

Tuples

- Tuples are :-
  - ordered,
  - immutable collections of elements.

- The only difference between a tuple and a list is that once a **tuple has been made, it can't be changed!**

Tuples

- Making a tuple:
  - a = (1, 2, 3, 4)
- Accessing a tuple:
  - number = a[0]
- The syntax for access is exactly like a list. However, you can't reassign things. ~~**(a[0] = 5)**~~

Returning Tuples

- Funtion can return tuple
  - def myFunc():
  - 	return 1, 2
  - result = myFunc()
  - print(result)
- When you return multiple things and store it in a single variable, it comes back as a tuple!

When to use Tuples

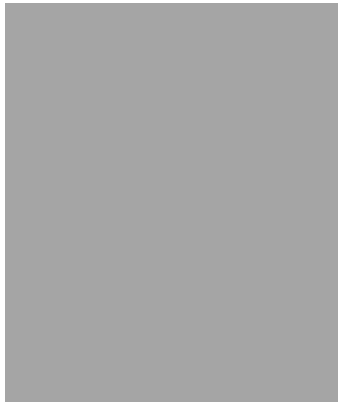
- Sometimes it's important that the contents of something not be modified in the future.

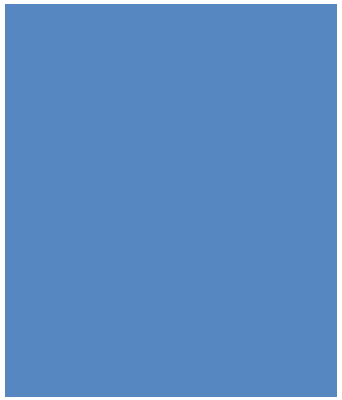
- Instead of trying to remember that you shouldn't modify something, just put it in a tuple! A lot of programming is learning to protect you from yourself.

Sets

Set is

- unordered collection of elements
- each element must be unique.
- Attempts to add duplicate elements are ignored.



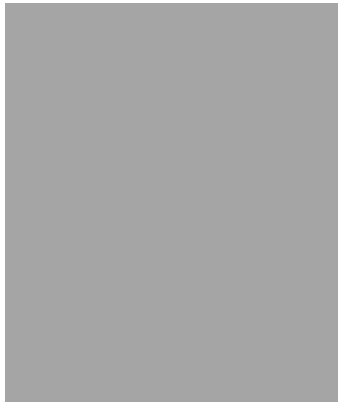


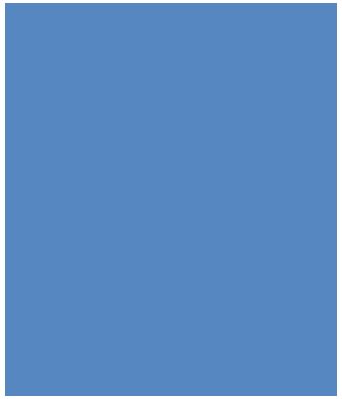
Sets exmples

- s1 = set(['a', 'b', 'c', 'd'])

- Or:
  - myList = [1, 2, 3, 1, 2, 3, 4 ,4 ,4]
  - s2 = set(myList)

- Note that in the second example, the set would consist of the elements {1, 2, 3,4}





Sets

- mySet = set(['a'])

- # Add an element:
- mySet.add('b')

- #Remove an element:
- mySet.remove('b')

- # Remove and return a random element:
- mySet.pop()

Sets

- There is also support for combining sets.
- Returns a new set with all the elements from both sets:
  - mySet.union(someOtherSet)
- Returns a new set with elements that were in both sets:
  - mySet.intersection(someOtherSet)
- Tons more methods can be found here:
- https://docs.python.org/3/tutorial/datastructures.html

Dictionaries

Up until now our storage has been done in lists.

Lists can be viewed as a structure that map indexes to values.

If I make the list:

```
myList = ['a', 'b', 'c']
```

I have created a mapping from 0 to 'a', 1 to 'b', and so on. If I put in 0, I'll get 'a' back.

Dictionaries

- Dictionaries let use whatever kind of keys we want!
- Instead of having 1 correspond to 'b', I can have "hello" correspond to 'b'.
- Before: Now I can do things like:
- 0 → ‘a’ "Hello" → 'a’
- 1 → 'b' 1 → 'b’
- 2 → 'c' 3.3 → 'c'

Dictionaries

- Creating a dictionary
  - myDict = {}
  - myDict["hello"] = 'a'
  - myDict[1] = 'b'
  - myDict[3.3] = 'c'
  - print(myDict["hello"])
- Prints:
- 'a'

Dictionaries

- Imagine you have a bunch of university students, and you're storing their grades in all their classes.

- Use a dictionary to look up grades by name:

  - student = “Tim"
  - grades = ["A","B","C+","D",“A","C"]
  - gradeDict = {}
  - gradeDict[student] = grades

  - print(gradeDict[“Tim"])

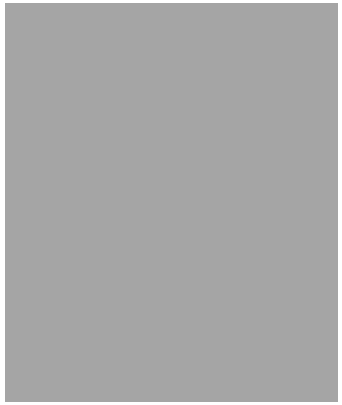
Dictionaries

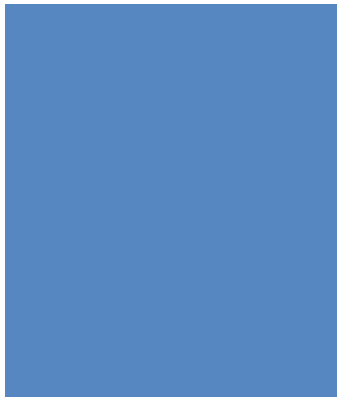
- When you look up something in a dictionary, the thing you're putting in (like the index in a list) is called a **key**. What we get out is called a **value**. A dictionary **maps** keys to values.

```
myDict["hello"] =   10
                ^       ^
              Key     Value
```

Dictionaries

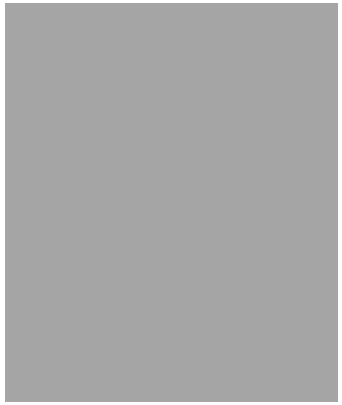
- Just like in a list, if you do this:
  - myDict["hello"] = 10
  - myDict["hello"] = 11
  - print(myDict["hello"])
- Prints:
- 11

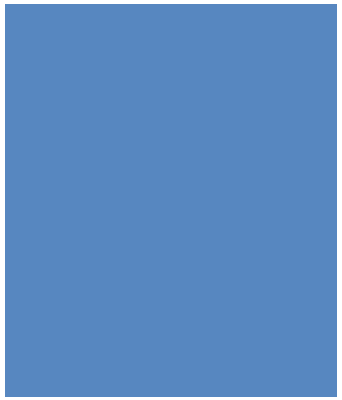




Dictionaries

- If we want to get just the keys, or just the values, there's a function for that!
  - listOfKeys = myDict.keys()
  - listOfValues = myDict.values()
  - listOfPairs = myDict.items()





References

https://docs.python.org/3/tutorial/index.html
https://docs.python.org/3/tutorial/introduction.html#lists
https://www.javatpoint.com/python-lists