

# INLS 560

## Programming for Information Professionals

# Lists



UNC  
SCHOOL OF INFORMATION  
AND LIBRARY SCIENCE

Joan Boone

*jpboone@email.unc.edu*

# Part 1: Overview

Part 2: Iteration, functions

Part 3: Reading file contents into a list

Part 4: Concatenation, Slicing, Finding

Part 5: Add, Remove, Copy

Part 6: Example and Exercise

# Why simple variables are not enough

- So far we've covered most of the Python language fundamentals
  - Variables and expressions
  - Decision structures (**if/else**)
  - Repetition structures (**for** and **while** loops)
  - Functions
  - Files and exceptions
- Data in your programs has been represented by simple variables
- But more realistic, practical problems usually require more robust representations of data, often in the form of collections of data

# Python Data Structures

## Lists and Dictionaries are most important

### List

- Ordered sequence of multiple items: `[1, 2, 3]`
- Mutable, and often contains homogeneous items (items of same data type)

### Tuple

- Similar to lists: `(1, 2, 'hello')`
- But, contents are immutable, and often heterogeneous

### Dictionary

- Unordered set of key-value pairs: `{ 'AAPL': 'Apple Inc', 'AMZN': 'Amazon.com Inc.' }`
- Keys must be unique

### Set

- Unordered collection with no duplicate elements. Often used to remove duplicates from a list : `set([1, 2, 3, 3]) → set([1, 2, 3])`

Reference: Python Tutorial on [Data Structures](#)

# Lists

- A list is an object that contains multiple items
- May contain items of different types, but usually the items all have the same type, and are ordered
- Lists are mutable (unlike tuples) which means their contents can be changed
- There are many operations available to manipulate the contents of a list: indexing, slicing, add, remove, sort, etc.
- How to create lists of
  - numbers: `temps = [45.6, 33.0, 78.5, 54.0]`
  - strings: `planets = ['Mars', 'Saturn', 'Venus']`
  - or both: `course = ['Proposal Development', 781, 1.5]`

Part 1: Overview

**Part 2: Iteration, functions**

Part 3: Reading file contents into a list

Part 4: Concatenation, Slicing, Finding

Part 5: Add, Remove, Copy

Part 6: Example and Exercise

# Iterating over a list with a **for** loop

```
temps = [45.6, 33.0, 78.5, 54.0]
total = 0

for temp in temps:
    total = total + temp
    print(temp)

print('Total temps:', total)
```

Output

```
45.6
33.0
78.5
54.0
Total temps: 211.1
```

- As in every programming language, there are multiple ways to do the same thing...
- ...the accumulator **for** loop can be replaced by a simple call to the built-in **sum** function

```
print('Total temps:', sum(temps))
```

- Recall the list of [built-in Python functions](#)

# Useful List Functions:

`sum()`, `max()`, `min()`, `len()`

`sum(temps)` returns the sum of all items in the list

`max(temps)` returns the largest item in a list

`min(temps)` returns the smallest item in a list

`len(temps)` returns the length of a list

```
temps = [45.6, 33.0, 78.5, 54.0]
```

```
print('Total temps:', sum(temps))
```

```
print('Max temp:', max(temps))
```

```
print('Min temp:', min(temps))
```

```
print('Length:', len(temps))
```

```
average = sum(temps) / len(temps)
```

```
print('Average temp:', average)
```

## Output

```
Total temps: 211.1
```

```
Max temp: 78.5
```

```
Min temp: 33.0
```

```
Length: 4
```

```
Average temp: 52.775
```

list\_sum\_max\_min\_len.py



# Using indexes to access individual list items

- Indexing starts at 0, i.e., the first item is 0
- Index of the last item is 1 less than the number of items in the list

```
temps = [45.6, 33.0, 78.5, 54.0]

print(temps[0], temps[1], temps[2], temps[3])
```

Using a loop to print all items in a list

```
temps = [45.6, 33.0, 78.5, 54.0]

index = 0
while index < len(temps):
    print(temps[index])
    index = index + 1
```

**Output**

```
45.6
33.0
78.5
54.0
```

Part 1: Overview

Part 2: Iteration, functions

**Part 3: Reading file contents into a list**

Part 4: Concatenation, Slicing, Finding

Part 5: Add, Remove, Copy

Part 6: Example and Exercise

# Reading File Contents into a List using a **for** loop

Use a **for** loop to read each line in the file, and **append** to the list.

```
def main():
    city_list = [] # Initialize list

    city_file = open('cities.txt', 'r')

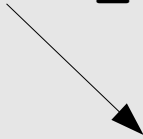
    # Read the contents of the file and append each line
    # as an item to the list. Use strip() to remove '\n'
    for city in city_file:
        city_list.append(city.strip())

    city_file.close()

    print(city_list)

main()

['Chicago', 'Boise', 'Toledo', 'Tampa', 'Santa Fe']
```



# Exercise: Rainfall Summary using a List

- Write a program that reads the contents of `rain_data.txt` and creates a list with the rain amounts
- Start with the draft version of the program, `rain_stats_draft.py`
- Calculate the total, average, maximum, and minimum rainfall
- Display the list of rain amounts, total, average, max, and min:

Rain data:

```
[2.5, 3.0, 5.6, 3.1, 2.0, 4.1, 0.5, 1.2, 3.2, 6.6, 7.2, 2.8]
```

```
Total rainfall: 41.80
```

```
Average rainfall: 3.48
```

```
Maximum rainfall: 7.20
```

```
Minimum rainfall: 0.50
```

Part 1: Overview

Part 2: Iteration, functions

Part 3: Reading file contents into a list

**Part 4: Concatenation, Slicing, Finding**

Part 5: Add, Remove, Copy

Part 6: Example and Exercise

# Many, many List operations

- Concatenation
- Slicing
- Finding items in a list
- Copying lists
- Built-in methods to
  - `append` and `insert` items
  - `remove` and `del` items
  - Find `index` of an item
  - `sort` items
  - `reverse` the order of items

# Concatenating and Slicing Lists

- Concatenating lists: use **+** operator

```
list1 = ['a', 'b', 'c']  
list2 = [1, 2, 3, 4]  
list2 = list1 + list2
```



```
['a', 'b', 'c', 1, 2, 3, 4]
```

- List slicing selects a range of items from a list

```
list_name[start : end]
```

returns a list containing a copy of `list_name` from **start** index, up to, but not including, **end** index

```
days = ['Sunday', 'Monday', 'Tuesday', 'Wednesday',  
         'Thursday', 'Friday', 'Saturday']  
weekdays = days[1:6]
```



```
['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday']
```

Other variations:

```
somedays = days[:2]  
somedays = days[4:]  
somedays = days[:]
```

# Finding items using `index` method

- Use the `index` method when you want to know whether an item is in the list and where it is located
- Returns the index of the first item that is equal to the argument
- `ValueError` exception is occurs if the item is not found

```
def main():  
    # Create a list of cities  
    cities = ['Chicago', 'Boise', 'Toledo', 'Tampa']  
  
    # Prompt a city to search for.  
    city_name = input('Enter a city: ')  
  
    # Determine whether the city is in the list  
    try:  
        city_index = cities.index(city_name)  
        print(search, 'is item #', city_index+1, ' in the list.')  
    except ValueError:  
        print(city_name, 'was not found in the list.')  
  
main()
```



# Finding items in a list using `in` operator

- Use the `in` operator to determine whether an *item* is contained in a list: `item in list`
- Returns a boolean value of `True` or `False`

```
def main():
    # Create a list of cities
    cities = ['Chicago', 'Boise', 'Toledo', 'Tampa']

    # Get a city to search for.
    city_name = input('Enter a city: ')

    # Determine whether the city is in the list.
    if city_name in cities:
        print(city_name, 'was found in the list.')
    else:
        print(city_name, 'was not found in the list.')

main()
```

Part 1: Overview

Part 2: Iteration, functions

Part 3: Reading file contents into a list

Part 4: Concatenation, Slicing, Finding

**Part 5: Add, Remove, Copy**

Part 6: Example and Exercise

# Adding items to a list

**append** (*item*) adds an item to the end of a list

```
cities = ['Chicago', 'Boise', 'Toledo', 'Tampa']  
cities.append('Santa Fe')
```

→ ['Chicago', 'Boise', 'Toledo', 'Tampa', 'Santa Fe']

**insert** (*index*, *item*) inserts an *item* into the list at a specified *index*

```
cities = ['Chicago', 'Boise', 'Toledo', 'Tampa']  
cities.insert(2, 'Santa Fe')
```

→ ['Chicago', 'Boise', 'Santa Fe', 'Toledo', 'Tampa']

# Removing items from a list

**remove** (*item*) removes the first occurrence of an *item* from a list

```
cities = ['Chicago', 'Boise', 'Toledo', 'Tampa']  
cities.remove('Boise')
```

→ ['Chicago', 'Toledo', 'Tampa']

**del** (*index*) removes an element from a specific *index*

```
cities = ['Chicago', 'Boise', 'Toledo', 'Tampa']  
del cities[2]
```

→ ['Chicago', 'Boise', 'Tampa']

# Copying Lists: 2 techniques

Using a loop to append items from one list to another

```
# Original list
cities = ['Chicago', 'Boise', 'Toledo', 'Tampa']
# Create an empty list
new_cities = []

# Copy elements from cities to new_cities
for item in cities:
    new_cities.append(item)
```

Using concatenation to append one list to an empty list

```
# Original list
cities = ['Chicago', 'Boise', 'Toledo', 'Tampa']

# Create a copy of the list
new_cities = [] + cities
```

Note: you cannot copy a list by assignment, i.e., `list2 = list1`

Part 1: Overview

Part 2: Iteration, functions

Part 3: Reading file contents into a list

Part 4: Concatenation, Slicing, Finding

Part 5: Add, Remove, Copy

**Part 6: Example and Exercise**

# Example with multiple list functions

```
def main():
    menu = ['Salad', 'Pizza', 'Pie', 'Tea', 'Shrimp', 'Spaghetti', 'BBQ']
    print("What's on the menu:", menu)

    item = input('Add a new item:')
    menu.append(item)
    print("Updated menu:", menu)

    item = input('\nFind an item on the menu:')
    if item in menu:
        print(item, 'is on the menu')
        item_index = menu.index(item)
        print(item, ' is item #', item_index+1, ' on the menu', sep='')
    else:
        print(item, 'is not on the menu')

    print('\n', menu[0], ' is the first item on the menu', sep='')
    menu.sort()
    print("Sorted menu:", menu)
    print(menu[0], ' is now the first item on the menu', sep='')

    item = input('\nRemove an item:')
    try:
        menu.remove(item)
        print("Updated menu:", menu)
    except ValueError:
        print('That item was not found on the list')
```

```
main()
```

# Exercise: NCAA Basketball Champions

- The `NCAA_BB_Champions.txt` file contains a chronological list of the team names that won the championship from 1939 through 2021.
- Write a program that prompts the user for a team name, and then displays the number of times that team has won the championship.

```
Enter the name of a team: North Carolina
North Carolina won the NCAA Basketball Championship 6 times
between 1939 and 2022.
```

```
Enter the name of a team: Duke
Duke won the NCAA Basketball Championship 5 times between 1939
and 2022.
```

```
Enter the name of a team: North Carolina State
North Carolina won the NCAA Basketball Championship 2 times
between 1939 and 2022.
```

```
Enter the name of a team: ucla
ucla never won a NCAA Basketball Championship.
```