UNIVERSITY of WISCONSIN
LA CROSSE
COMPUTER SCIENCE

CS 224 Introduction to Python

Iteration

1

Preliminaries: Additional List Operations

- List membership: `in`
  - returns Boolean
  - `e in L`
- List of consecutive integers: `range`
  - returns a list
  - `range(10)` → `[0, 1, …, 9]`
  - `range(100, 200)` → `[100, 101, …, 199]`
- Assignment: `=`
  - creates an alias

2

Preliminaries: Random

- random provides a number of helpful methods
  - `random()` returns a float in [0, 1)
  - `randint(x, y)` returns an int in [x, y]
  - `shuffle(L)` permutes L in place
- Multiple ways to import
  - `import random`
    - `random.random, random.randint, random.shuffle`
  - `import random as rd`
    - `rd.random, rd.randint, rd.shuffle`
  - `from random import random, randint, shuffle`
    - `random, randint, shuffle`

3

Iteration

- while loops
  - similar to other languages with minor syntactic differences
- for loops
  - primarily list based

```
i = 0
while i < 10:
    instructions
    i += 1


for i in range(x):
    instructions
```

4

Iteration

- for loops with range and len
  - `range(len(L))` returns list with an int value, beginning with 0, for each index in list `L`
- iterator
  - operate on each element of list
  - `e` takes value of each element in list in turn

```
for i in range(len(L)):
    instructions
```

```
for e in L:
    instructions
```

5

Important note about iterators

- Consider this code fragment

```
for e in L:
    e = 2 * e
```

  - `e` is really a reference to each element in list
  - `e = 2 * e` reassigns the reference but doesn't affect the value stored in the list
  - Thus `L` is unchanged

6

A big BUT concerning previous note

- Let L be a list of lists
- Consider code this code fragment

```
for e in L:
    e[0] = 2 * e[0]
```

  - `e` is a reference to each element in list
  - `e[0] = 2 * e[0]` does not reassign the reference
  - Thus `L` is updated

7