

Objectives

- Indefinite Loops
- Dictionaries

Mar 12, 2012

Sprenkle - CSC1111

1

Review

- How can we make something repeat when some condition is true?
- True or False: Every **for** loop can be converted into a **while** loop
- True or False: A **while** loop is more powerful than a **for** loop

Mar 12, 2012

Sprenkle - CSC1111

2

Review: While Loop Syntax

```
while condition :  
    statement1  
    statement2  
    ...  
    statementn
```

keyword

body of while loop

loop stops when condition is False

- Like a looped **if** statement
 - Execute statements **only** when condition is true

Mar 12, 2012

Sprenkle - CSC1111

3

Another Way to Read from a File

```
FILENAME="data/years.dat"  
dataFile = open(FILENAME, "r")  
line = dataFile.readline()  
while line != "":  
    line = line.rstrip()  
    print(line)  
    line = dataFile.readline()  
dataFile.close()
```

file_read_while.py

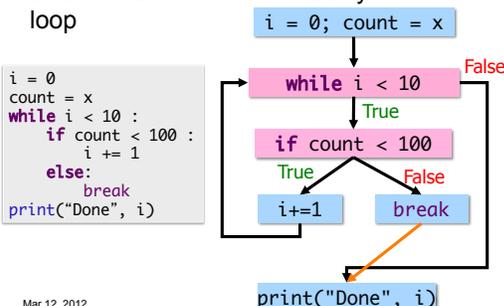
Mar 12, 2012

Sprenkle - CSC1111

4

Use of break statement

- **break** statement can "break you" out of a loop



Mar 12, 2012

5

while Loops: comparing use of break

```
# condition says when loop  
# will continue  
x=eval(input("Enter number:"))  
while x % 2 != 0 :  
    print("Error!")  
    x = eval(input("Try  
again: "))  
print(x, "is an even number.")  
  
# have to look inside loop to  
# know when it stops  
while True :  
    x = eval(input("Enter number:"))  
    if x % 2 == 0 :  
        break  
    print("Error")  
print(x, "is an even number.")
```

Says when to keep going

Says when to stop

Using break statements:
Best when loop has to
execute at least once.

Mar 12, 2012

Sprenkle - CSC1111

6

Flipping Coins

- Problem: How many flips does it take to get 3 consecutive heads?

`consecutiveHeads.py`

Mar 12, 2012

Sprenkle - CSCI111

7

How Does `in` Work for Lists?

- Example: `guess in prevGuesses`, where `prevGuesses` is a list object
 - For each element in list, checks if element equals (`==`) `guess`
- In the worst case, how many elements does `in` have to check?
 - How could we improve the search?

Mar 12, 2012

Sprenkle - CSCI111

8

Faster Lookups

- In my phone's contacts app, if I wanted to know my friend's phone number, ...
 - Would I search through an ordered list of phone numbers?
 - No, I would look up my friend and find the phone number **associated** with my friend
- This type of data structure is known as a **dictionary** in Python
 - Maps a **key** to a **value**
 - Contacts' key: "Friend's name", value: phone number

Mar 12, 2012

Sprenkle - CSCI111

9

Examples of Dictionaries

Dictionary	Keys	Values
Dictionary		
Textbook's index		
Cookbook		
URL (Uniform Resource Locator)		

- Any other things we've done/used in class?

Mar 12, 2012

Sprenkle - CSCI111

10

Examples of Dictionaries

Dictionary	Keys	Values
Dictionary	Word	Definition
Textbook's index	Keyword	Page number
Cookbook	Food type	Recipes
URL (Uniform Resource Locator)	URL	Web page

- Any other things we've done/used in class?

Mar 12, 2012

Sprenkle - CSCI111

11

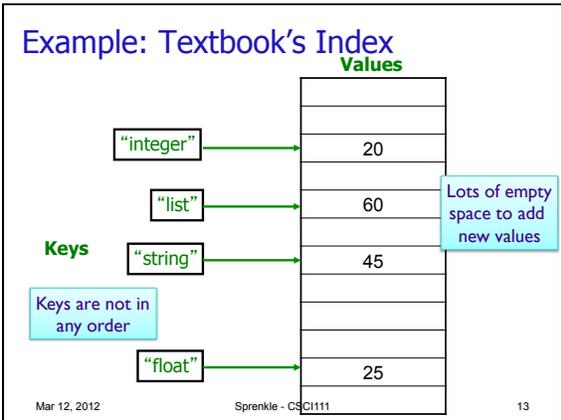
Examples of Dictionaries

- Real-world:
 - Dictionary
 - Textbook's index
 - Cookbook
 - URL (Uniform Resource Locator)
- Examples from class
 - Variable name → value
 - Function name → function definition
 - ASCII value → character

Mar 12, 2012

Sprenkle - CSCI111

12



Dictionaries in Python

- Map **keys** to **values**
 - Keys are probably **not** alphabetized
 - Mappings are from **one** key to **one** value
 - Keys are **unique**, Values are not necessarily unique
 - Example: student id → last name
 - Keys must be **immutable** (numbers, strings)
- Similar to Hashtables/Hashmaps in other languages

How would we handle if there is more than one *value* for a given key?

Mar 12, 2012 Sprenkle - CSCI111 14

Why Dictionaries?

- Another way to store data
- Allow fast lookup of data
 - Requires keys, unique keys
 - Data may not have a natural mapping

Pros	Cons
Fast lookup (<i>much</i> faster than lists if a lot of elements)	Requires a lot of space, unique keys

Mar 12, 2012 Sprenkle - CSCI111 15

Creating Dictionaries in Python

Syntax:

```
{<key>:<value>, ..., <key>:<value>}
```

```
empty = {}
ascii = { 'a':97, 'b':98, 'c':99, ..., 'z':122 }
```

Mar 12, 2012 Sprenkle - CSCI111 16

Dictionary Operations

Indexing	<dict>[<key>]
Length (# of keys)	len(<dict>)
Iteration	for <key> in <dict>:
Membership	<key> in <dict>
Deletion	del <dict>[<key>]

Unlike strings and lists, doesn't make sense to do slicing, concatenation, repetition for dictionaries

Mar 12, 2012 Sprenkle - CSCI111 17

Dictionary Methods

Method Name	Functionality
<dict>.clear()	Remove all items from dictionary
<dict>.keys()	Returns a <i>copy</i> of dictionary's list of keys
<dict>.values()	Returns a <i>copy</i> of dictionary's list of values
<dict>.get(x [, default])	Returns <dict>[x] if x is a key; Otherwise, returns None (or default value)

Mar 12, 2012 Sprenkle - CSCI111 18

Accessing Values using Keys

- Syntax:
`<dictionary>[<key>]`
- Examples:

```
ascii['z']  
contacts['friendname']
```

- **KeyError** if key is not in dictionary
 - Runtime error; exits program

Mar 12, 2012

Sprenkle - CSC1111

19

Accessing Values Using get Method

- `<dict>.get(x [, default])`
 - Returns `<dict>[x]` if `x` is a key; Otherwise, returns `None` (or default value)

```
ascii.get('z')  
directory.get('friendname')
```

- If no mapping, get **None** back instead of **KeyError**

Mar 12, 2012

Sprenkle - CSC1111

20

Accessing Values

- Typically, you will check if dictionary has a key before trying to access the key

```
if 'friend' in contacts:  
    number = contacts['friend']
```

- Or handle if get default back

```
number = contacts.get('friend')  
if number is None: # do something ...
```

Know mapping exists before trying to access
No phone number exists

Mar 12, 2012

Sprenkle - CSC1111

21

Recall: Special Value None

- Special value we can use
 - E.g., Return value from function when there is an error
- Similar to **null** in Java
- If you execute

```
list = list.sort()  
print(list)
```

 - Prints `None` because `list.sort()` does **not** return anything

Mar 12, 2012

Sprenkle - CSC1111

22

Example Using None

```
# returns the lowercase letter translated by the key.  
# If letter is not a lowercase letter, returns None  
def translateLetter( letter, key ):  
    if letter < 'a' or letter > 'z':  
        return None  
    #As usual ...
```

```
# example use  
encletter = translateLetter(char, key)  
if encletter is None:  
    print("Error in message: ", char)
```

Mar 12, 2012

Sprenkle - CSC1111

23

Inserting Key-Value Pairs

- Syntax:
`<dictionary>[<key>] = <value>`
- `ascii['a'] = 97`
 - Creates new mapping of 'a' → 97

ascii_dictionary.py

Mar 12, 2012

Sprenkle - CSC1111

24

