Python

Session 7

By Declan Fox
With thanks to Al Sweigart

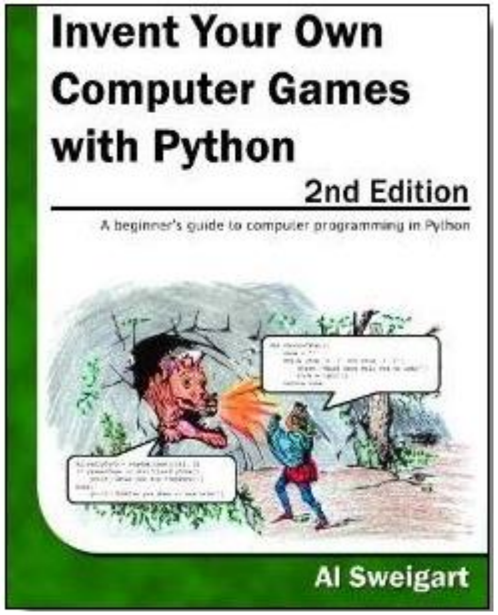
“Above all, be cool.”

Wi-Fi Name: CoderDojo

Password: coderdojowireless

Website: http://cdathenry.wordpress.com/

Remember to check out “Invent your own computer games with Python” by Al Sweigart.


Invent Your Own Computer Games with Python
2nd Edition
A beginner's guide to computer programming in Python
Al Sweigart

The book is free online at

http://inventwithpython.com

New Topics

- Multi-line strings
- Lists
- for loops
- Methods
    - list methods
    - string methods
- elif statements
- The dictionary data type.
- Multiple variable assignment, such as a, b, c = [1, 2, 3]

The range() and list() Functions

The range() function is easy to understand.

You can call it with either one or two integer arguments.

When called with one argument, range() will return a range object of integers from 0 up to (but not including) the argument.

If you pass two arguments to range(), the list of integers it returns is from the first argument up to (but not including) the second argument.

range objects can be converted to the more familiar list data type with the list() function.

for Loops

- The for loop is very good at looping over a list of values.
- This is different from the while loop, which loops as long as a certain condition is true.
- A for statement begins with the for keyword, followed by a variable name, followed by the in keyword, followed by a sequence (such as a list or string) or a range object (returned by the range() function), and then a colon.
- Each time the program execution goes through the loop (that is, on each **iteration** through the loop) the variable in the for statement takes on the value of the next item in the list.

Slices and Slicing

If we want to get a shorter copy of some of the items in a list, we can use list slicing.

Slicing creates a duplicate list out of some or all of the items in another list. In code, we can create a slice of a list by specifying two indexes (the beginning and end) and a colon.

```
Contestants =  [‘olly’, ‘1D’, ‘matt’, ‘james’, ‘leona’]
Winners = Contestants [2:4]
```

elif ("Else If") Statements

We’re familiar with if else statements

```
if year == 2005:
        winner = ‘shayne’
else:
        print(‘Shayne didn’t win)
```

If we want to test for other years we could do the following

```
if year == 2005:
        winner = ‘Shayne’
else:
        if year == 2006:
                winner = ‘Leona’
        else:
                if year == 2007:
                        winner = ‘Leon’
```

And so on.

elif ("Else If") Statements

But if we want more things, then the code starts to have a lot of indentation. Typing all those spaces means you have more chances of making a mistake. So Python has the elif keyword. Using elif, the above code looks like this:

```
if year == 2005:
   print(‘Winner  was Shayne')
elif  year == 2006
   print(‘Winner was Leona.')
elif year == 2007
   print(‘Winner  was Leon')
elif  year == 2008
   print(‘Winner was Alexandra')
else:
   print(‘Your year is out of range')
```

Dictionary

- Dictionary is another data type in Python.
- Dictionaries are collections of items that have a “key” and a “value”.
- They are just like lists, except instead of having an assigned index number,
  you make up the index.
- Dictionaries are unordered, so the order that the keys are added doesn’t necessarily reflect what order they may be reported back.
- Use {} curly brackets to construct the dictionary.
- Look up the value associated with a key using []
- Provide a key and a value.
- A colon is placed between key and value (key:value)
- Each key must be unique and each key can be in the dictionary only once.

Dictionary

How to create a Dictionary

To create a dictionary, provide the key and the value and make sure that each pair is separated by a colon.

```
# This is a list
mylist = ["first","second","third"]
# This is a dictionary
mydictionary = {0:"first",1:"second",2:"third"}
```

Dictionary

To access dictionary elements, you can use the square brackets along with the key

to obtain its value.

```
print(mydictionary [0])
>>
first
```