Python Games

Session 4

By Declan Fox

Rules

“Above all, be cool.”

General Information

Wi-Fi Name: CoderDojo

Password: coderdojowireless

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

Useful Links

Recommended reading:
http://inventwithpython.com

Reference Guide

http://www.tutorialspoint.com/python/

Social Media

Like our new Facebook page at
www.facebook.com/CoderDojoAthenry

Or if you are on twitter follow us on
@coderdojoathenr

Installation

As we will be moving on to graphical games we will need to install both Python and Pygame*

* If you have Python 3.x.x and Pygame installed you can ignore the next slide

Installation

We are using version 3.2 of Python go to

https://www.python.org/download/releases/3.2.5/

Select Windows x86 MSI Installer (3.2.5)

To install Pygame go to

http://pygame.org/download.shtml

Select pygame-1.9.2a0.win32-py3.2.msi

Today's topics

- Multi-line strings
- Lists
- for loops
- Methods
  - list methods
  - string methods
- elif statements
- The dictionary data type.

Multi-line strings

Ordinarily when you write strings in your source code, the string has to be on one line.

- However, if you use three single-quotes instead of one single-quote to begin and end the string, the string can be on several lines

Lists

Lists are a new data type.

- A list value can contain several other values in it.
- Lists are a good way to store several different values into one variable.
- The individual values inside of a list are also called **items**.

Lists

Creating a list
Judges = [‘Simon’, ‘Sharon’, ‘Danni’, ‘Louis’]

Printing to screen
Print (judges)

To print individual items from the list
Print (judges[1])
This would print out Sharon

The number in the square brackets is called the index
In Python, the first index is the number 0 instead of the number 1
So to get Simon we would need to do the following
Print (judges[0])

List Concatenation

We can use the + sign to join lists together.

```
judges = ['Simon', 'Louie', 'Sharon', 'Danni']
print(judges)

presenters = ['Dermot', 'Holly',]
print(presenters)

people = judges + presenters
print(people)
```

List in Operator

The in Operator

The in operator makes it easy to see if a value is inside a list or not. Expressions that use the in operator return a Boolean value:

True if the value is in the list and False if the value is not in the list.

Removing Items

You can remove items from a list with a del statement. ("del" is short for "delete.")

```
del judges[0]
```

del Statement

You can remove items from a list with
a del statement. ("del" is short for "delete.")

```
del judges[0]
```

Lists of lists

Lists are a data type that can contain other values as items in the list.

But these items can also be other lists.

For example we could make up a list of judges, a list of presenters and a list of winners and make a list to contain these lists called xFactor.

xFactor = [judges, presenters, winners]

To get an item inside the list of lists, you would use *two* sets of square brackets like this: listOfLists[1][2]

Methods

Methods are just like functions, but they are always attached to a value.

For example, all string values have a lower() method, which returns a copy of the string value in lowercase.

You cannot just call lower()by itself and you do not pass a string argument to lower() by itself (as in lower('Hello')).

You must attach the method call to a specific string value using a dot (full stop).

Functions vs. methods

- Why does Python have methods, since they seem to act just like functions. Methods are functions associated with values of that data type. For example, string methods are functions that can be called on any string.

- You cannot call string methods on values of other data types. For example, [1, 2, 'apple'].upper() would cause an error because [1, 2, 'apple'] is a list and upper() is a string method.

- The values of data types that have methods are also called objects. Object-oriented programming is a bit advanced for us, and you don't need to understand it to make games. Just understand that objects are another name for values of data types that have methods. For example, all strings and lists are objects.

List methods

The list data type also has methods.
The reverse() method will reverse the order of the items in the list.

The most common list method you will use is append(). This method will add the value you pass as an argument to the end of the list.

Split method

The split method splits a string of words into a list of words.

Judges = ‘ Simon Sharon Danni Louis’

Judges = ‘ Simon Sharon Danni Louis’.split()

Will give us

Judges = [‘Simon’, ’Sharon’, ’Danni’, ’Louis’]

Range() and List()

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.

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.

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)

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)

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

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

Next session

CoderDojo Athenry will relocate next Saturday to NUIG for the **Massive Open Dojo Session** and **Game Jam workshop** starting at 11:30 and ending around 2:30