Obroni Computer Club – Lists in Python

Joachim Breitner

September 26th 2006

As I am not well, I won't be talking all the time today. I have written everything down, for you to read *and try*. If you have any questions, just ask me.

Today, we will have a look at *lists* in python. Lists are a way of storing more than one value in a variable, and allow us to treat them efficiently by looping over them.

While looking at that, we will write a simple text-based maze game. You should feel free to extend that and add more features.

1 Using lists in python

1.1 Creating lists

Lists are very essential in python programming, and hardly any useful program can do without. A list is an ordered collection of values of arbitary type, and you create one by putting the values in square brackets and separating by a comma:

```
list = [1, 'text', 50, [ 'bla', 'blubb' ], 4 ]
```

This list now contians a number, a string, a list with two strings and a number. As you can see, you can put lists inside lists without problems.

If you **print** such a list, it will appear with the square bracket and commas. This is good to debug a program, but for "real" output for the user, we need something nicer.

1.2 Accessing list element

You can access a specific list element using it's *index*. Assuming the list from above, you can use:

```
print list[0] # gives 1
print list[2] # gives 50
print list[3] # gives ['bla','blubb']
print list[3][1] # gives 'blubb
print list[-1] # gives 4
```

Note that the first element in the list has the index 0, as it is almost alwas the case in computer science. If you use a negative number, python counts the elements from the back.

1.3 Altering a list

A list is also a variable, meaning that you can change it. You can either change single elements just like you were accessing it, or you can use certain *methods* to change the list.

```
list[0] = 2 # sets the first element
list[-1] = 5 # sets the last element
list.append('tail') # appends an element to the end
list.extend([1,2,3]) # appends the elements of another list to the end
list.remove(50) # removes a element
```

After all these operations, the list should be:

```
[2,'text',['bla','blubb'],5,'tail',1,2,3]
```

You can also sort and reverse a list, using list.sort and list.reverse.

1.4 Getting more information from lists

Often, you want to know the length of a list, or other properties. Here some useful methods:

```
print len(list) # number of elements, here 8
print min(list) # smallest element, here 1
print max(list) # largest element, here 'text'
'text' in list # tests if something is in the list, here true
'foo' not in list # tests if something is not in the list, here true
list.index('tail') # gives 4, as list[4] is 'tail'
```

1.5 Putting lists together

If you have two lists, you can concatenate them using the +-operator:

```
list1 = [1,2,3]
list2 = ['a','b','c']
new_list = list1 + list2
print new_list # gives [1,2,3,'a','b','c']
```

2 Looping over lists

To efficiently work with lists, you often have to do something once per entry. For that, we can use the **for**-loop:

```
fruits = ['apple','peach','orange']
for fruit in fruits:
  print 'I like to eat a '+fruit+'.'
```

This code outputs

```
I like to eat a apple.
I like to eat a peach.
I like to eat a orange.
```

What happens here is that the variable name after ther **for** statement gets set to each entry in the list, and the idented code is run.

3 Our maze

This is enough to start our simple maze. We begin with a list of rooms we have. I keep the list short in this text, you might want to have a larger maze, or have a different setting.

```
rooms = ['kitchen', 'bedroom', 'bathroom', 'hallway', 'living room']
```

Then, we need to know what room is connected to what room. So we create a list of lists, with all the rooms a room is connected to. The order of the lists correspond with with the order in the list rooms. For better readability, I spread it on several lines.

```
connected = [
        ['hallway','living room'],
        ['bathroom','hallway'],
        ['bedroom'],
        ['kitchen','bedroom','living room'],
        ['kitchen','hallway'],
]
```

Of course, every maze has a start and a target:

```
start = 'living room'
end = 'bathroom'
```

We'd like to keep track of where the user went, so we have a list for that, too. Initially, it only contains the start. Note the missing quotes, we are talking about the variable start, not the text 'start'!

```
path = [start]
```

Now it's time to start the game. For that, we have the position of the player, and as long as that is not the end, we keep tell him where he can go, wait for his input, validate that and record that in the path.

```
position = start
while position != end:
```

```
        print 'You are in '+position+'.'
        # Find the possible ways
        room_number = rooms.index(position)
        ways = connected[room_number]
        print 'You can go to these rooms:'
        for room in ways:
                print ' * '+room
        next_room = None
        # We ask until a proper room was given
        while next_room not in ways:
                next_room = raw_input('Where do you want to go? ')
        # Ok, let's go there
        path.append(next_room)
        position = next_room
        print '----------------------------------------'
print 'Congratulations, you have finished the maze'
print 'You took this path:'
print ' -> '.join(path) # you will see what this does
```

If you put these lines in a program and run it, you should be able to play it. Of course, don't forget the *#!/usr/bin/python* in the first line!

3.1 Extending the maze

Starting from this point, you can easily extend your program with your own ideas. Here are some suggestions:

- Monsters. Some room have monsters, if you enter them, you are dead.
- Time limit. Allow only a certain amount of moves before the target has to be reached (hint: len(path)).
- Make it a more adventure-like game. For that:
  - Add descriptions to the rooms that are printed upon entering.
  - Put things in various rooms that the player can pick up and use in other places.
  - Some of these things might be required to open some of the doors, like keys.
- Make a maze of the school/the hostels/your home.

You now have already the skills to write very complex programs. It is your now your imagination that limits what you can do. You just have to do it!