

## LING115 Lecture Notes

### Session #4: Things you need to know for Homework#4

Here are additional tricks in Python that you need to know to solve Homework#4.

#### 1. `len (sequence)`

This is a function we use to calculate the length of a sequence. For example, the following prints out how many members the list `x` contains:

```
>>> x=['a','b','c']
>>> len(x)
```

Similarly, the following prints out how many characters the string `x` consists of:

```
>>> x='a string'
>>> len(x)
```

#### 2. `range (x, y, step)`

This is a function that lists a sequence of numbers from `x` to `y` with the numbers incrementing by `step`. Note that `y` is not inclusive. For example, the following returns a list of three numbers `[4, 6, 8]`:

```
>>> range(4, 9, 2)
```

By default, `x = 0` and `step = 1`. So the following is equal to `[0, 1, 2, 3, 4]`:

```
>>> range(5)
```

In conjunction with the `len()` function we learned above, the `range()` function allows to write a for-loop that iterates over the indices of a sequence. Try and see what the following does:

```
>>> a=['a','b','c']
>>> for i in range(len(a)):
>>>     print a[i]
```

#### 3. `sys.argv`

Like the shell commands we saw previously, we often want to write a Python program that allows users to specify arguments. What you enter in the command prompt can be retrieved by using `sys.argv`, which is a list of command-line arguments. The first member of the list, `sys.argv[0]` refers to the name of the program and the arguments you specify after the program name are indexed from one, so the first argument you specify would be `sys.argv[1]`. Note that each command-line argument is a string.

As you can guess, `sys.argv` is a variable named `argv` defined in the Python module `sys`, the same module as in `sys.stdin.readlines()`. Since this variable is defined in `sys`, we must first import

sys before we use the variable. See `print_input.py` under `/home/ling115/` for example. Try and see what the following does.

```
$ python print_input.py blah
```

#### 4. Type-casting

Sometimes we want to convert the data type of a value. Suppose we wanted to write a program that reads the list of numbers saved in a file and calculates the mean. To do this, our program should first read the lines from a file, identify the numbers in each line, add them up, and divide the sum by the number of lines in the file. The problem is reading a line from a file results in a string, not a number. So we cannot use the arithmetic operators as we want them to work. Recall that the following two are different:

```
>>> 2+2
>>> '2'+ '2'
```

So what do we do? We tell Python to convert the data type of the value: from string to number in this case. This is called type-casting. Here are useful built-in functions for type-casting:

```
int(x)
```

Convert `x` to an integer.

```
float(x)
```

Convert `x` to a floating point number.

```
str(x)
```

Convert `x` to a string.

```
list(x)
```

Convert `x` to a list.

Suppose we wanted to write a program that prints out the sum of two integers that the user specifies as arguments. How would you do it? See `/home/ling115/python_examples/add.py` for answer.