Lecture 17: Introduction to Computer Programming Course - CS1010

DEPARTMENT OF COMPUTER SCIENCE | 03/25/2019

Rensselaer

Announcements

- There will be no-class on April 8
- Exam Grades will be available shortly:
  - Still grading
  - Solutions will be posted after the makeup exam on Wednesday

Goals for today

- Controlling Loops
- List comprehensions
- Lambda function

Loops Review

- **for** loops are counted loops and have fixed number of iterations
- **while** loops can have an indefinite termination, determined by the condition specified.
- Most Python **for** loops are easily rewritten as **while** loops, but not vice-versa.

List Comprehensions

- Comprehensions are an efficient way of creating lists.
- Syntactically, list comprehensions consist of an iterable containing an expression followed by a ‘for’ clause.
- When using for loops with append, list comprehensions are specifically useful.
- list_variable = [x for x in iterable]

Comparison

- shark_letters = [letter for letter in 'shark']
- print(shark_letters)

- shark_letters = []
- for letter in 'shark':
- shark_letters.append(letter)
- print(shark_letters)

Lambda

- The lambda operator or lambda function is a way to create small anonymous functions, i.e. functions without a name.
- These functions are throw-away functions, i.e. they are just needed where they have been created.
- Lambda functions are mainly used in combination with the functions filter(), map() and reduce().

Lambda

- The general syntax of a lambda function is quite simple:

- lambda argument_list: expression

- The argument list consists of a comma separated list of arguments and the expression is an arithmetic expression using these arguments.
- You can assign the function to a variable to give it a name.
- Lets check in spyder

Map function

- The advantage of the lambda operator can be seen when it is used in combination with the map() function.
- map() is a function which takes two arguments:

- r = map(func, seq)
- The first argument *func* is the name of a function and the second a sequence (e.g. a list) *seq*. *map()* applies the function *func* to all the elements of the sequence *seq*.

Filter

- The function filter() offers an elegant way to filter out all the elements of a sequence , for which a function returns True.
  - filter(function, sequence)
- The function filter(f,l) needs a function f as its first argument.
  - f has to return a Boolean value, i.e. either True or False.
  - This function will be applied to every element of the list *l*.
  - Only if f returns True will the element be produced by the iterator, which is the return value of filter(function, sequence).

Controlling the execution of loops

- **break** sends the flow of control immediately to the first line of code outside the current loop.
- **continue**, immediately sends control back to the “top” of the loop, skipping the rest of the code.
- Infinite loops:
  - When working with a while loop one always needs to ensure that the loop will terminate! Otherwise we have an *infinite loop*.

Reading from files

- Python uses file objects to interact with external files on your computer.
- These file objects can be any sort of file you have on your computer, whether it be an audio file, a text file, emails, Excel documents, etc.
- You will probably need to install certain libraries or modules to interact with those various file types, but they are easily available.

Create a .txt file and read it in Python

- First find the working directory and save your file there :
- Type pwd to check the working directory and save your file there

- Alternatively, to grab files from any location on your computer, simply pass in the entire file path.

- For Windows you need to use double \ so python doesn't treat the second \ as an escape character, a file path is in the form:
- myfile = open("C:\\Users\\YourUserName\\Home\\Folder\\myfile.txt")

- For MacOS and Linux you use slashes in the opposite direction:
- myfile = open("/Users/YouUserName/Folder/myfile.txt")

Open a file

- To open a file in Python, we first need some way to associate the file on disk with a some variable in Python.
- This process is called *opening* a file.
- We begin by telling Python where the file is.
- The location of your file is often referred to as the file *path*. In order for Python to open your file, it requires the path.

Open file continued

- The open() function requires as its first argument the file path.
- The function also allows for many other parameters.
- However, most important is the optional mode parameter.
- Mode is an optional string that specifies the mode in which the file is opened.
- The mode you choose will depend on what you wish to do with the file.

Mode Options

- Some of the mode options:
- 'r' : use for reading
- 'w' : use for writing
- 'x' : use for creating and writing to a new file
- 'a' : use for appending to a file
- 'r+' : use for reading and writing to the same file
- Syntax:
  - open(path/file ,'r')

Read a file

- We can read an existing file using the following command:
  - Filename.read()
- Before reading we need to open it:
  - open(path/file ,'r')
- Since our file has been opened, we can now manipulate it (i.e. read from it) through the variable we assigned to it.
- Python provides three related operations for reading information from a file. (read,readline,readlines)

Write a file

- We can write text into a file using:
  - The ‘w’ mode
- A new file can be created or an existing file can be over-written.

Closing a file

- Closing a file makes sure that the connection between the file on disk and the file variable is finished.
- Closing files also ensures that other programs are able to access them and keeps your data safe.
- So, always make sure to close your files.

Next Class

- Some problems on loops and reading files