Lecture 5: Introduction to Computer Programming Course - CS1010

DEPARTMENT OF COMPUTER SCIENCE | 09/17/2019

Rensselaer

Announcements

- Homework 2 posted today
- Homework 1 grades will be available next week

Goals for Today

- Strings in Python
- Manipulation
- Print Formatting
- Approach Programming Problems
- Practice Exercises

Object Types (Table from Previous Lecture)

Name	Type (representation)	Example
Integers	int	Whole Numbers: 1, 5 , 7500
Floating Point	float	Decimal: 2.3, 4.6, 23.15
Strings	str	Ordered sequence of characters: “hello” “Sam” “2000”
Booleans	bool	Logical Value: True, False

Strings

- Ordered Sequence of characters
- Strings in Python are actually a *sequence*, which basically means Python keeps track of every element in the string as a sequence.
- Syntax is either double quote or single quote
- Examples:
  - “Hello”
  - ‘Hello’
  - ‘I don’t know that’

Strings

- We can print strings
- We can assign them to variables, for example:
  - X = ‘Rensselaer Polytechnic Institute’
  - print(X)
- Printing the variable is different from actually calling the variable.

Combining Single and Double quotes

- A string that starts with double quotes must end with double quotes, and therefore we can have single quotes inside.
- A string that starts with single quotes must end with single quotes and therefore we can have double quotes inside.
- Extending strings across multiple lines:
  - Either begin with “”” and end with “”” (Or ‘’’ alternately)
  - Use Escape Characters (\)

Escape Characters

- – \n—end the current line of text and start a new one.
- – \t—skip to the next “tab stop” in the text. This allows output in columns.
- – \'—do not interpret the ' as a string delimiter.
- – \"—do not interpret the " as a string delimiter.
- – \\—put a true back-slash character into the string.

String Manipulations

- Because strings are ordered sequences of characters we can manipulate these using:
  - Indexing
  - Slicing
- Use the len(s) function to get the length of a string
- Use square brackets to access individual chars inside the string. The chars are numbered starting with 0, and running up to length-1.
- For example s = 'Hello’ , len(s) will be 5

Strings (Continued)

- Manipulating String elements
- Python strings are "immutable" which means a string can never be changed once created. Use + between two strings to put them together to make a larger string
  - s = 'Hello’
  - t = 'Hello' + ' hi!’
  - print(t)
- Python has a built-in string class named "str" with many features

Indexing in Strings

- Characters within a string are numbered starting with 0
- For example, s= ‘Table’
  - s[0] ‘T’
  - s[1] ‘a’
  - s[4] ‘e' -- last char is at length-1 (len(s)-1)
  - s[5] ## ERROR, index out of bounds OR string index out of range
- Reverse index: s[1] == s[-4]; s[2] ==s[-3]…last element will be s[-1], -2 is the next to last, and so on.

Slicing in Strings

- A "slice" in Python is way of referring/pointing to sub-parts of a string.
- The syntax is **s[i:j]**meaning the substring starting at index i, running up to but not including index j.
- Complete syntax for slicing is:
  - [begin:stop:step]
  - ‘begin’ is the numerical index where you will start slicing
  - ‘stop’ is the index where you will go up to
  - ‘step’ size of the jump you take
- Let’s practice in Spyder

Concatenation

- Two (or more) strings may be concatenated to form a new string, either with or without the + operator.
- X=‘Hello’
- Y=‘World’
- X+Y

Replication

- You can replicate strings by multiplying them by an integer
  - S='lol'
  - print(S*10)
- Cannot multiply by a floating point

Basic Built in Methods

- Objects in Python have built-in methods.
- These methods are programs inside the object that can perform actions on the object.
- We call methods with a period and then the method name. Methods are in the form:
  - object.method(parameters)
- Where parameters are extra arguments we can pass into the method. Don't worry if the details don't make 100% sense right now. Later on we will be creating our own objects and functions!

Some Important methods

- variable.find()
- variable.lower()
- variable.upper()
- variable.captalize()
- variable.title()
- variable.find()
- variable.count()

Print Formatting

- Insert a variable into your string for printing
- There are many ways of doing print formatting
- Also called string interpolation
- For example
  - My_name=‘Uzma’
  - print(“My name is ”+ My_name)

Print Formatting

- There are three ways to perform string formatting.
  - The oldest method involves placeholders using the modulo % character.
  - An improved technique uses the .format() string method.
  - The newest method, introduced with Python 3.6, uses formatted string literals, called *f-strings*.

Formatting: Method 1 (%)

- Two methods %s and %r convert any python object to a string using two separate methods: str() and repr()
- \t inserts a tab into a string.
- The %s operator converts whatever it sees into a string, including integers and floats.
- The %d operator converts numbers to integers first, without rounding.
- Floating point numbers use the format %5.2f. Here, 5 would be the minimum number of characters the string should contain; these may be padded with whitespace if the entire number does not have this many digits. Next to this, .2f stands for how many numbers to show past the decimal point.

Formatting Method 2 (.format())

- Another way to format objects into your strings for print statements is with the string .format() method. The syntax is:
- 'String here {} then also {}'.format('something1','something2’)
- Advantages:
  - Inserted objects can be called by index position
  - Inserted objects can be assigned keywords
  - Inserted objects can be reused, avoiding duplication

- Let’s try in Spyder

Formatting Method 3 (f-strings)

- Introduced in Python 3.6, f-strings
- You can bring outside variables immediately into to the string rather than pass them as arguments through .format(var).
- Floating points: With the .format() method you might see {value:10.4f}, with f-strings this can become {value:{10}.{6}}

- For more info:
- https://docs.python.org/3/reference/lexical_analysis.html#f-strings

Reminding the Methodology (Lecture 1)

- U – Understand the Problem:
  - Write down the inputs you have and explain the problem to yourself in plain English, use examples.
- D – Devise a Good Plan to Solve:
  - Write down the Algorithm you will use
  - Given input X, what are the steps necessary to return output Y?: Use comments!
  - Reduce the problem to something simpler.

Problem Solving Methodology

- I – Implement the Plan:
  - Translate Algorithm to code
  - Algorithm, Pseudo-Code, Flow Chart
- E – Evaluate the Solution:
  - Run for a few test cases
  - Simplify the code if possible
- Debugging: When running into error, re-thing of what you were asked and what you made the program do.

Problem 1

Write a Python program to change a given string to a new string where the first and last characters have been exchanged.

Step 1: INPUTS – Take a string of characters e.g. ‘I am at work’

Step 2: If I input ‘I am at work’ then my output must be ‘k am at worI’

To get this output I can :

- slice the string to get the first character and save it somewhere
- slice the last character and save it somewhere
- with the remaining string (without first and last character) I can concatenate the saved characters (by swapping them) to this string.

Write the actual code

Problem 2

- Write a program that returns True if the first character appears again somewhere in the string. Otherwise it must return False.
- For example ‘calcium’ : Output True; ‘exam’ Output False
- Step 1: My input will be a string like ‘Hello World Hey’.
- Step 1: If I input ‘Hello World Hey’, my output will be True else false.
- Step 2: What should I do:
  - Slice the first element and save it somewhere (in a variable)
  - Check this element against all others (excluding the first element)
  - If there is a match print True else print false.

Write the actual code

Problem 3

- Write a program that takes two strings as input and replaces all elements of the longer string that match with the last element of the shorter string with ‘$’ sign. If both strings are of equal length then the program must output ‘Both are equal’.
- For example if I input ‘absghtsscde’ and ‘qts’. These can be in any order.
- Output must be ‘ab$ght$$cde’

Problem 3 (Solution)

Next Week

- Functions and Modules