Lesson 8 – Strings

1. **Strings**

Strings are simply “words”, although not necessarily words that make sense. For example, "a3rt66th" is a string. Strings are denoted with either double quotes or single quotes – e.g., s = “hello” and s = ‘hello’ are equivalent.

Strings are immutable, which mean they cannot be changed. This is not a real problem in programming, you just have to keep in mind that you need to re-assign the variable as illustrated below:

s = "hello"

s = s.replace('h', 'j')

Note that "hello" did not get changed – the variable s did!

1. **String Methods and Operations**

Suppose we want to find out if a letter or sub-string is contained within another string. An example of a substring is “He”, which is part of “Hello”. We could then write something like the following:

s1 = "Hello"

s2 = "He"

if "He" in "Hello":

print("YES")

Quite often, we need to find the first occurrence of a substring (or multiple occurences). This is done as follows:

a = s1.find(s2)

or

a = s1.find(s2, startIndex)

if we want to find multiple occurences.

Also, we may want to replace letters or substrings in a string. This is done as follows:

a = s1.replace('h', 'j')

Adding two strings together is easy, we just use the + operator as follows:

s3 = s1 + s2

If you want to add a number to a string, you have to use the str(X) function where X is a number.

For changing the case of a string, you can use the following:

s3 = s1.lower() (or upper)

For testing the case of a string, you can use the following:

if s1.islower(): (or isupper())

print("YES")

For testing if a character of a string is a digit or letter, you can use the following:

if s1.isdigit(): (or isalpha)

print("YES")

There are many more methods for strings – you can look them up here: http://docs.python.org/py3k/library/stdtypes.html#string-methods

1. **Substrings**

A substring is defined as a contiguous block of characters within a given string up to and including the string itself. For example, “thema” is a substring of “mathematics”.

Substrings are indicated by a starting position (counting from 0) and an ending position. In python, a substring looks like the following examples:

s = “mathematics”

s1 = s[2:7] #this gives “thema”. Note that the 7 is exclusive!!

s2 = s[:7] #this gives “mathema”.

s3 = s[2:] #this gives “thematics”.

In python, this notation (s[a:b]) is called “slicing”. It will be the same notation that we use later on when we discuss lists.

1. **Examples**
2. The first step to writing hangman is to see if a letter is contained in the mystery word. This would be done something like this:

mystery = "Computer Science 11"

guess = input()

if guess in mystery:

print("your letter is in the word")

else:

print("guess again")

1. In HTML, formatting tags have a start tag (<b>) and an end tag (</b>). If you change your mind and want to replace bold with italics in a given text, you can write something like the following:

text = "<b> hello </b>"

text = text.replace("<b>", "<i>")

text = text.replace("</b>", "</i>")

Note that it would be faster to write it as follows:

text = text.replace("<b>", "<i>").replace("</b>", "</i>")

1. Suppose we had a numerical encryption scheme where characters were added to a numerical string in order to create confusion. For example, “g37h54jhg12k98” would be an encrypted word in this scheme. To decrypt it, you would need to remove the characters and then create a new string with only the numbers in the same order that they are above (“37541298”). This could be done as follows:

s = "g37h54jhg12k98"

newS = ""

for c in s:

if c.isdigit():

newS = newS + c

1. In hangman, you would have a string of \*’s which represents the mystery word to be guessed. When the player correctly guesses a letter, you will want each occurrence of that letter to show up in the string of \*’s. This can be done as follows:

for i in range(0, len(mystery)):

if guess == mystery[i]:

t = t[:i] + guess + t[i+1:]

1. **Programming Projects**
2. Given an integer input of a number of donuts, print a string of the form 'Number of donuts: count', where count is the actual number typed in by the user. However, if the count is 10 or more, then use the word 'many' instead of the actual count. So typing in “5” gives 'Number of donuts: 5' and typing in “50” gives 'Number of donuts: many'
3. Given a string s, print a string made up of the first 2 and the last 2 characters of s. For example, 'spring' yields 'spng'. You may assume that the length of s will be greater than 2.
4. Given a string s, print a string where all occurences of its first char have been changed to '\*', except for the first character itself. So for example, 'babble' will be changed to 'ba\*\*le'. You may assume that the string length will be greater than 2.
5. Given two strings a and b, print a **single** string with a and b separated by a space and with the first 2 characters of each string swapped. For example,

'dog', 'dinner' becomes 'dig donner'

1. Given a string s, find the first appearance of the substrings 'not' and 'bad'. If the substring 'bad' occurs after the substring 'not', replace the whole 'not'...'bad' substring with 'good'. So for example,

'This dinner is not that bad!' becomes

'This dinner is good!'

1. A word is called a palindrome if it is spelled the same backwards as well as forwards while ignoring case. An example is “Madam”. Write a program to test if a word is a palindrome.
2. Write a simple hangman program. The mystery word does not change whenever you run the program. As well, you do not need any graphics. The program should keep track of the number of incorrect guesses and end the program if you do not guess the word in time. If you do guess the word before reaching the limit of guesses, it should tell you that you have won (or something). Also, it should display the word with the correctly guessed letters and something like the underscore for letters that have not been guessed yet as well as the number of guesses left.

It is very important that you write or think out the algorithm for this game before you start programming it.