Predefined Functions

- A predefined function can be thought of as a self-contained "mini-program" which has a specific task.
- They are created by the creators of Python to help programmers create their own programs.
- Python and all other computer languages have a “library” of predefined functions which perform a number of frequently required tasks.
- These predefined functions can be called by the programmer at any time in a program they are designing.

Three important points to remember about predefined functions in Python are:

1) A predefined function returns a **single** result. Depending on the function, this result can be a string, an integer, a float, or a boolean (true or false) value.
2) Predefined functions usually include at least one **parameter** (input) which is inside the brackets. If there is more than one parameter, they are separated by commas. If there is no parameter, empty brackets () must be used.
3) Functions are often grouped in a library or collection of related functions. The name of the library precedes the name of the actual function. Sometimes, the library is part of language (such as the **string** library) while sometimes the library must be imported (such as the **math** library).

Predefined Functions Used With Strings

<u>After watching the corresponding video</u>, type the small programs following each function and record the result to make sure you understand how the function works.

1. **len (***string***)** - returns an integer representing the number of characters in a string
   - the parameter is the string whose length is to be measured

ex. 1 **print** (len ("abcdef")) *Outputs 6*

ex. 2 word = "hello there"

print (len (word)) *Outputs 11*

2. **str.find (***string*, *pattern***)** - returns an integer representing the position of the first instance of a string pattern to be searched for in the string
   - -1 is returned if the string pattern is not found in the string
   - the parameters are the string or string variable to be searched and the string pattern to be searched for
   - it <u>does not</u> return the number of times the pattern is found in the string

ex. 1 **print** (**str.find** ("abcabc", "c")) *Outputs 2*

ex. 2 word = "hello there"

print (**str.find** (word, "h")) *Outputs 0*

print (**str.find** (word, "the")) *Outputs* 6

print (**str.find** (word, "e")) *Outputs* 1

print (**str.find** (word, "ll")) *Outputs* 2

print (**str.find** (word, "zx")) *Outputs* -1

3. **ord (***string***)** - returns the ASCII code which corresponds to the string character
   - the parameter is a which must be **one** character in length

ex. 1 **print** (**ord** ("a")) *Outputs 97*

ex. 2 **print** (**ord** ("A")) *Outputs 65*

4. **chr (***ASCII value***)** - returns the string character which corresponds to the ASCII value
   - the parameter is an integer representing the ASCII value

ex. 1 **print** (**chr** (116)) *Outputs t*

ex. 2 **print** (**chr** (36)) *Outputs $*

5. **str.upper** (*string*) - returns the string completely capitalized
   - the parameter is a string value or string variable

ex. 1 **print** (**str.upper** (“hello there”)) *Outputs HELLO THERE*

ex. 2 **print** (**str.upper** (“Hello There”)) *Outputs HELLO THERE*

6. **str.lower** (*string*) - returns the string with all characters in lower case
   - the parameter is a string value or string variable

ex. 1 **print** (**str.lower** (“HELLO THERE”))*Outputs hello there*

ex. 2 **print** (**str.lower**(“Hello There”)) *Outputs hello there*

7. **str.capitalize** (*string*) - returns the string with the first character capitalized
   - the parameter is a string value or string variable

ex. 1 **print** (**str.capitalize** (“HELLO THERE”)) *Outputs Hello there*

ex. 2 **print** (**str.capitalize** (“hello there”)) *Outputs Hello there*

8. **str.count** (*string*, *pattern*) - returns the number of times the pattern is found in the string
- the parameters are the string or string variable to be searched and the string pattern to be searched for.

ex. 1 **print** (**str.count** ("HELLO THERE", "H")) *Outputs 2*

ex. 2 **print** (**str.count** ("hello there", "ll")) *Outputs 1*

ex. 3 **print** (**str.count** ("hello there", "H")) *Outputs 0*

9. **str.replace** (*string*, *oldPattern*, *newPattern*)
- returns the string with all the occurrences of *oldPattern* replaced with *newPattern*
- the parameters are the string or string variable to be searched, the string pattern to be replaced and the string pattern which will replace it.

ex. 1 **print** (**str.replace** ("abcabc", "a","z")) *Outputs zbczbc*

ex. 2 **print** (**str.replace** ("abcabc", "a","aa")) *Outputs aabcaabc*

ex. 3 **print** (**str.replace** ("abcabc", "a","zzz")) *Outputs zzzbczzzbc*

ex. 4 **print** (**str.replace** ("abcabc", "a","z z")) *Outputs z zbcz zbc*

ex. 5 **print** (**str.replace** ("abcabc", "b"," ")) *Outputs a ca c*

ex. 6 **print** (**str.replace** ("abcabc", "b","")) *Outputs acac*

What is the difference between " " and ""? *" " is a space character and "" is a null string*

ex. 7 word = "aabbcc"

biggerWord = **str.replace**(word,"bb","BBBB")

print (biggerWord) *Outputs aaBBBBcc*

Other Predefined Functions Used With Strings

For a more comprehensive list of predefined functions that can be used to manipulate lists, visit the website: http://www.tutorialspoint.com/python/python_strings.htm and move down the page until you see the Built-in String Methods section.