420-LCU-05 Programming in Python
Lab 4 strings - Solutions
March 4, 2022

Goals for this lab:

- Practice with strings, including:
  - Indexing
  - Slicing
  - String methods

Part 1 - string indexing

1. Create a variable named `full_name` whose value should be your full name.

   ```
   full_name = "Lea Smith"
   ```

2. Create another variable named `name_len` whose value is the length of your name in characters (you may use the `len()` function to compute this).

   ```
   name_len = len(full_name)
   ```

3. Write the expression to print the first character in your name.

   ```
   print(full_name[0])
   ```

4. Write the expression to print the last character in your name.

   ```
   print(full_name[-1])
   ```

5. At the Python prompt, write the expression:

   ```
   >>> full_name[name_len]
   ```

   Describe what happens and why.

You should get an error, specifically an IndexError "exception". This is because the first character is at position zero, and so the last character is at position `name_len-1`. Attempting to access an index outside the legal range results in an error.

Part 2 - string slicing and operators

1. Using the variable `full_name` you created above, figure out the slice expression to extract your first (given) name(s) using positive indices. Compute the value of the correct index and assign it to the variable `first_name_index`. Use this variable as a constant value. Write the statement to assign the extracted string to the variable `first_name`.

   ```
   first_name_index = 3 # index of space
   first_name = full_name[:first_name_index]
   print(first_name)
   ```

2. As above, write the slice expression to extract your last (family) name(s) using *negative* indices. Compute the value of the correct index and assign it to the variable `last_name_index`. Now, assign this value to the variable name `last_name`.

   ```
   last_name_index = -5 # index of 1st letter in last name
   last_name = full_name[last_name_index:]
   ```

3. Now, using operators we have learned, write the statement to print the string "Hello, " followed by your first name.

   ```
   print("Hello, " + first_name)
   ```

4. Write an expression to check whether your name is “greater or smaller than” the string `'m'`. Print ”greater” or ”smaller”.

```
if (full_name >= 'm'):
    print("greater")
else:
    print("smaller")
```

5. Write an expression to test whether or not your name contains the string "an", Your program should print an exact message depending on the result: ”There is "an" in my name” or ”There is no "an" in my name”.

```
if ("an" in full_name):
    print('There is "an" in my name')
else:
    print('There is no "an" in my name')
```

Part 3 - string methods

In this section we'll experiment with a few string methods.

1. Write an expression to search for the first occurrence of a substring in your name (pick any substring that is actually present in your name). This result must be expressed as an integer (i.e. save value in a variable named `Index`).

```
Index = full_name.index("mi")
print(Index) # will print 5 in this example.
```

2. Use the `rindex()` method to find the position of the *final* space character in your full name. Use this method to find the position of that substring in your name, and save this value in the variable `final_space`.

```
final_space = full_name.rindex(' ')
```

3. Using the variable you saved in the previous step, write an expression to extract and print the last part of your name (The last part of my name is ...)

```
print("The last part of my name is",full_name[final_space+1:])
```

4. Using the variable you saved in the second step, write an expression or statement to extract and print the first part of your name (The first part of my name is ...).

```
print("The first part of my name is",full_name[:final_space])
```

5. Starting with your `full_name` variable above, write the code to create a new string in which all of the vowels (’a’, ’e’, ’i’, ’o’, and ’u’) have been replaced with a dash (’-’) character. You may need to write more than one line of code and create several new variables of your choice to do this. Be sure your code does *not* change your original variable, it should still contain your full name, unmodified. In the lab, we will see how to do this in 1 step.

```
a = full_name.replace('a', '-')
e = a.replace('e', '-')
i = e.replace('i', '-')
o = i.replace('o', '-')
u = o.replace('u', '-')

#you can replace all characters in one line and create new string x
x= full_name.replace('a','-').replace('e','-').replace('i','-').replace('o','-').replace('u','-')
```

6. Using your original `full_name` variable, write a single statement or expression that will display your name with a lower-case character at the beginning, and all of the rest of the characters in upper case. For example, if your `full_name` is `'Lisa Smith'`, this expression should print `'lISA SMITH'`. **Hint:** You can start by creating several expressions (lines) and then try to combine in one statement.

```
print(full_name[:1].lower() + full_name[1:].upper())
```

7. It is very common to write programs to process text from the Internet, such as email or web pages. Many email formats include a standard header line that looks like this:

```
header = 'From jsmith@gmail.com Thu Aug 31 13:32:39 2017\n'
```

Write an expression to extract the domain name (in this case, 'gmail.com') from a string of this format. Your expression should work with most, or even all, strings formatted like this. Hint: This string lies between the first and only '@' and the second space character in the line.

```
header = 'From jsmith@gmail.com Thu Aug 31 13:32:39 2017\n'
s1 = header.index(' ') #position of 1st space
s2 = header.index(' ',s1+1) #position of 2nd space
at = header.index('@') #position of @
print("domain name is",header[at+1:s2])
# In one line, it looks like this:
print("domain name is",header[header.index('@')+1:header.index(' ',header.index(' ')+1)])
```

Write a Python program that asks the user to enter a password. Decide on an appropriate message to display based on the following specifications. Your program checks the validity of the entered password based on the following criteria (in any order). Print an appropriate message and exit for any violation. Otherwise proceed to the next validation. If all validations pass, your program prints a success message.
Hint: Use nested if statements to move from one validation to the next.

1. password is composed of letters and digits only except for last character which must be a #.
2. Minimum length 8 characters and maximum length 15 characters (inclusive).
3. Password must start with a capital letter
4. At least 1 digit.
5. At least 1 small letter.
6. If all conditions pass: print "Password accepted"

```
pwd=input("enter a password:")
slice=pwd[0:-1]
if (slice.isalnum()):
    if(pwd.endswith("#")): #pwd[len(pwd)-1]=="#"
        if(8<=len(pwd)<=15):
            if(pwd[0].isupper()):#3
                if (slice.isalpha()==False):#4
                    if (slice.isupper() == False):
                        print("accepted")
                    else:
                        print("no small letters")
                else:
                    print("no digits")
            else:
                print("does not start with a capital letter")
        else:
            print("length does not satisfy requirements")
    else:
        print("does not end with #")
else:
    print("all except last character not alphanumeris")
```