ECMM455 Python Worksheet 4: Handling strings

Prof Hywel Williams

September 12, 2019

1 Aims

- Practice using built-in functions to handle strings
- Learn some more techniques for strings
- Try some problem-solving exercises involving strings

2 Further reading

The material in this worksheet is covered in Chapter 8 of *Think Python*.

3 Strings: The story so far...

In the Code Academy tutorial, you have already learned a lot about string variables:

- Three ways to create a string:
  - Double quotes: *"my string"*
  - Single quotes: *'my string'*
  - Converting another data type using *str*: *str(2.34)*
- Concatenation using +: *"first string" + "second string"*
- Finding the length of a string using the *len()* function: *len("my string")*
- Accessing a character in a string using its index (recall that index positions always count up from zero), e.g. the index of the fifth character (t) in "my string" is 4, so we could use: *"my string"[4]*
- Converting a string to all lower-case letters using the string variable method *lower()*: *"my string".lower()*
- Converting a string to all upper-case letters using the string variable method *upper()*: *"my string".upper()*

Hopefully all this is familiar. If not, go through the tutorial "Strings & Console Output" on Code Academy again.

3.1 Exercises

1. Write a Python program called *first_name.py* that:

   (a) Creates a variable called *firstname* and assigns to it a string containing your first name.

   (b) Displays the string stored in *firstname*. (Hint: Use the *print()* function.)

   (c) Displays the number of characters in your first name.

2. Save a copy of *first_name.py* with new filename *whole_name.py* . Alter *whole_name.py* so that it:

   (a) Creates a variable called *firstname* and assigns to it a string containing your first name.

   (b) Creates a variable called *surname* and assigns to it a string containing your surname.

   (c) Concatenates both variables into a single string, containing a space, and displays the result. (Hint: Use +)

4 Getting user input

Consider the code snippet below, which uses the *input()* function (remember that *input()* in Python 3 is the same as *raw_input()* in Python 2):

```
print("How old are you?")
age = input()
print("How tall are you?")
height = input()
print("So, you're %s years old and %s tall." % ( age, height))
```

What do you think it does? Here is some sample output:

```
ECMM429$ python example.py
How old are you?
104
How tall are you?
2.5m
So, you're 104 years old and 2.5m tall.
ECMM429$
```

4.1 Exercises

1. Write a Python program called *personal_info.py* . Enter the code above exactly as it is written. Run the program.

2. Extend *personal_info.py* so that the program begins by asking your name, then uses your name in the final output. E.g. "Hi Hywel, you are 104 years old and 2.5m tall."

5 String "slices"

It is easy to extract part of an existing string (known as a substring or "slice") using a range of indexes. To specify a range of indexes you need to use a colon (:). Look at the example in interactive mode below.

```
>>> test_string = "extracting a slice is easy"
>>> print(test_string)
extracting a slice is easy
>>> my_slice = test_string[6:10]
>>> print(my_slice)
ting
>>> another_slice = "here is another string"[8:15]
>>> print(another_slice)
another
>>> end_slice = test_string[13:]
>>> print(end_slice)
slice is easy
>>> start_slice = test_string[:18]
>>> print(start_slice)
extracting a slice
>>>
```

There are a couple of things to note about slices:

- The index range has form *[start:end]* , where *start* is the first index to include and *end* is the first index not to include (i.e. the range is inclusive at its beginning and exclusive at its end - why it does it like this is a mystery!).
- If you leave out the *start* index, you get all characters until the *end* index.
- If you leave out the *end* index, you get all characters including and after the *start* index.
- You can specify a slice of a string variable or directly from a string.
- Spaces are counted as characters in strings - don't miss them out counting index positions.

5.1 Exercises

1. Write a Python program called *get_a_word.py* . This program should ask the user to enter a word, then display the word they entered.
2. Write a Python program called *first_and_last.py* . This program should ask the user to enter a word, then display the first and last letters of the word they entered. (Hint: Use *len()* to find the string length and remember that indexes start from zero.)
3. Write a Python program called *first_half.py* . This program should ask the user to enter a word with an even number of letters, then display the first half of the word they entered. (E.g. Entering the string "WooHoo" yields "Woo".)

6 More string methods

Here are three more useful methods that can be used with string variables. Recall that a "method" is a function specific to a particular data type; these methods are specific to strings. They can be used with any *string*, but not with *int* or *float* or other data types.

- *mystring.find(x)*
  - Returns the lowest index in *mystring* where string *x* is found. Search string *x* can be a single character or a string of characters. Returns -1 if *x* is not found.
- *mystring.replace(x,y)*
  - Returns a version of *mystring* in which each instance of string *x* has been replaced by string *y*.
- *mystring.count(x)*
  - Returns a count of all instances of string *x* in *mystring*.

Have a look at the examples below which use each method in interactive mode.

```
>>> test_string = 'banana'
>>> print(test_string)
banana
>>> print(test_string.count('b'))
1
>>> print(test_string.count('n'))
2
>>> print(test_string.count('a'))
3
>>> print(test_string.count('z'))
0
>>> print(test_string.find('a'))
1
>>> print(test_string.find('n'))
2
>>> print(test_string.replace('a','i'))
binini
>>> print(test_string.replace('n','p'))
bapapa
>>> print(test_string.find('x'))
-1
>>> print(test_string.replace('x','p'))
banana
>>>
```

6.1 Exercises

1. Open Python in interactive mode.
2. Count the number of instances of "x" in the word "saxifrage".
3. Count the number of instances of "a" in the word "saxifrage".
4. Replace each "a" with an "o" in the word "copacabana".
5. Find the index of the first "a" in "copacabana".

7 Problem-solving exercises

These are taken from Chapter 2 of *Python for Biologists* by Martin Jones. [A pdf of the old (free) version of this book and the solutions to the exercises are now provided on ELE.] They can all be completed in <10 lines of code, using the string handling techniques you have learned. The descriptions of the exercises are deliberately terse and may be somewhat ambiguous (just like requirements for programs you will write in real life). See the solutions in *Python for Biologists* for in-depth discussions of the exercises. BUT try to solve them yourself first - you will learn very little just from reading someone else's solution. The aim here is to practice breaking the problem down into a sequence of steps and then constructing a solution (program) iteratively, checking each stage as you go. You may want to cut-and-paste the DNA sequences into your program - if so, take care to correct for line-breaks etc.

7.1 Calculating AT content

Here's a short DNA sequence:

ACTGATCGATTACGTATAGTATTTGCTATCATACATATATATCGATGCGTTCAT

The AT content is the proportion of the bases that are A or T. Write a program that will display the AT content of this DNA sequence. *[Correct answer: AT content = 0.686]*

7.2 Complementing DNA

Here's a short DNA sequence:

ACTGATCGATTACGTATAGTATTTGCTATCATACATATATATCGATGCGTTCAT

The complement of a DNA sequence is another DNA sequence that replaces A with T, T with A, G with C, C with G. Write a program that will display the complement of this sequence. *[Correct answer: manual check.]*

7.3 Restriction fragment lengths

Here's a short DNA sequence:

ACTGATCGATTACGTATAGTAGAATTCTATCATACATATATATCGATGCGTTCAT

The sequence contains a recognition site for the *EcoRI* restriction enzyme, which cuts at the motif G*AATTC (the position of the cut is indicated by an asterisk). That is, when it finds that motif in a sequence, it cuts the sequence at the indicated position to leave two fragments. Write a program which will calculate the size of the two fragments that will be produced when the DNA sequence is digested with *EcoRI*. *[Correct answer: 22 and 33]*

7.4 Splicing out introns - part one

Here's a short section of genomic DNA:

ATCGATCGATCGATCGACTGACTAGTCATAGCTATGCATGTAGCTACTCGATC
GATCGATCGATCGATCGATCGATCGATCGATCATGCTATCATCGATCGATATC
GATGCATCGACTACTAT

(Note that this is a single unbroken sequence, the line breaks shown here are just for display purposes.) It comprises two exons (coding regions) and an intron (non-coding region). The first exon runs from the start of the sequence to the sixty-third character, and the second exon runs from the ninety-first character to the end of the sequence. Write a program that will print just the coding regions of the DNA sequence.

Correct answer:

ATCGATCGATCGATCGACTGACTAGTCATAGCTATGCATGTAGCTACTCGATC
GATCGATCGATCATCGATCGATATCGATGCATCGACTACTAT

7.5 Splicing out introns - part two

Using the data from part one, write a program that will calculate what percentage of the DNA sequence is coding. *[Correct answer: 77.2%]*

7.6 Splicing out introns - part three

Using the data from part one, write a program that will print out the original genomic DNA sequence with coding bases in uppercase and non-coding bases in lowercase.

Correct answer:

ATCGATCGATCGATCGACTGACTAGTCATAGCTATGCATGTAGCTACTCGATC
GATCGATCGatcgatcgatcgatcgatcgatcatgctATCATCGATCGATATC
GATGCATCGACTACTAT