Tutorial 7 Worksheet

COMP1117B Computer Programming I 2018-2019

|  |
| --- |
| **Tutorial objectives:**   * To practice strings in Python.   **Tutorial 7 Exercises Deadline: 26 March, 2019 11:20am** |

1. **A useful tool for Assignment 2. Question 1: The join method of python string**

To add something to the end of one string, there are two mainly used function:

(a) list.append(object) # Add an object object to the list

(b) list.extend(sequence) # Add the contents of a sequence seq to the list

To see the difference between these two functions, consider the following program.

|  |
| --- |
| music\_media = ['compact disc', '8-track tape', 'long playing record']  new\_media = ['DVD Audio disc', 'Super Audio CD']  music\_media.append(new\_media)  print music\_media  >>>['compact disc', '8-track tape', 'long playing record', ['DVD Audio disc', 'Super Audio CD']] |

When using append, new\_media is treated as an object, and the whole package is added to the music\_media object.

|  |
| --- |
| music\_media = ['compact disc', '8-track tape', 'long playing record']  new\_media = ['DVD Audio disc', 'Super Audio CD']  music\_media.extend(new\_media)  print music\_media  >>>['compact disc', '8-track tape', 'long playing record', 'DVD Audio disc', 'Super Audio CD'] |

When using extend, new\_media is treated as a sequence, and the sequence is merged with the music\_media sequence and placed behind it.

1. **Demo: Interswitch between string and list of chars**

Consider the following program.

|  |
| --- |
| def main():  message = input()  mask = input()  print(encrypt(message, mask))  main() |

Complete the function encrypt() that accepts a string message and a single character mask, then replace the first 9 occurrence of the character mask by number 1 to 9 accordingly. For example, if message is bilibalaba and mask is b, encrypt() should return 1ili2ala3a.

To implement this function, we need a loop to find the occurrences of the character mask, then replace that character by a count.

|  |
| --- |
| def encrypt(message, mask):  count = 1  for i in range(len(message)):  if message[i] == mask:  message[i] = str(count) # !!! doesn't work!  count += 1  return message |

If you try the code above, it doesn’t work as string is immutable!

We can convert the string to a list of characters, which makes it mutable.

|  |
| --- |
| ch = list(message) |

To convert it back to a string, we can use the join function of string.

|  |
| --- |
| message = ''.join(ch) |

Now we can rewrite the function.

|  |
| --- |
| def encrypt(message, mask):  count = 1  ch = list(message)  for i in range(len(message)):  if ch[i] == mask:  ch[i] = str(count)  count += 1  return ''.join(ch) |

**Exercise 7.1 Substitution Cipher**

Write a simple cipher program to perform encryption and decryption. The plain text and the corresponding cipher as follows:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Plaintext alphabets | a | b | c | d | e | f | g | h | i | j | k | l | m |
| Cipher alphabets | c | g | i | n | e | l | o | s | u | r | y | h | p |
|  | | | | | | | | | | | | | |
| Plaintext alphabets | n | o | p | q | r | s | t | u | v | w | x | y | z |
| Cipher alphabets | v | a | b | z | j | k | x | d | t | f | w | m | q |

The user can input 'E' for encryption and 'D' for decryption. And then input the message. Your program will perform the encryption or decryption accordingly.

For further information: <https://en.wikipedia.org/wiki/Substitution_cipher>

In this question, assume that the input message will only consists of alphabets a to z.

|  |  |  |
| --- | --- | --- |
| **Test case** | **Input** | **Output** |
| Decryption | D  u hate mad | i love you |
| Encryption | E  i love you | u hate mad |

**Exercise 7.2 Printing Pattern from String**

Consider the following Python code that read an input string from user and construct a triangle using the string.

|  |
| --- |
| def main():  patternBase = input()  printTriangle(patternBase)    main() |

Implement the function printTriangle(patternBase) so that it prints a triangle pattern from the patternBase following the examples below.

|  |  |  |
| --- | --- | --- |
| **Test case** | **Input** | **Output** |
| Hello | Hello | HelloHello  elloHell  lloHel  loHe  oH |
| foo | foo | foofoo  oofo  of |

**Exercise 7.3 Substring counting**

Consider the following Python code.

|  |
| --- |
| def main():  message = input()  word = input()  print(countWord(message, word))    main() |

Implement the countWord() function that accepts two strings, message and word, as input argument and return the number of times word appears in message.

|  |  |  |
| --- | --- | --- |
| **Test case** | **Input** | **Output** |
| Zero | banana  c | 0 |
| One | banana  banana | 1 |
| Two | banana  na | 2 |

**Exercise 7.4 DNA sequence matching**

Consider the following Python code.

|  |
| --- |
| def main():  dna = input()  pattern = input()  printMatch(dna, pattern)  main() |

Implement the printMatch() function that accepts two input arguments:

* dna - string of ACGT (no need to validate its content)
* pattern - a string of ACGT (no need to validate its content)

The function prints two lines:

* First line: the dna
* Second line: A ‘\*’ at the position that pattern occurs in the dna sequence, and a ‘\_’ at the other positions.

String library reference:

String slicing techniques [https://docs.python.org/3/tutorial/introduction.html?#strings](https://docs.python.org/3/tutorial/introduction.html?" \l "strings)

|  |  |
| --- | --- |
| **Test case** | **Input** |
| **Output** |
| 1 | ATACTCGTCGTCGATCGATACTCGTCTGTCGTCGAGTCGTTCGTCTCGTC  TCGTC |
| ATACTCGTCGTCGATCGATACTCGTCTGTCGTCGAGTCGTTCGTCTCGTC  \_\_\_\_\*\_\_\*\_\_\_\_\_\_\_\_\_\_\_\_\_\*\_\_\_\_\_\_\*\_\_\_\_\_\_\_\_\_\_\_\*\_\_\_\_\*\_\_\_\_ |
| 2 | ATACTCGTCGTCGATCGATACTCGTCTGTCGTCGAGTCGTTCGTCTCGTC  TCG |
| ATACTCGTCGTCGATCGATACTCGTCTGTCGTCGAGTCGTTCGTCTCGTC  \_\_\_\_\*\_\_\*\_\_\*\_\_\_\*\_\_\_\_\_\_\*\_\_\_\_\_\_\*\_\_\*\_\_\_\_\*\_\_\_\*\_\_\_\_\*\_\_\_\_ |
| 3 | ATACTCGTCGTCTTTGATCGATACTTTTCTGTCGTCGAGTCGTTTTTCTCGTC  TT |
| ATACTCGTCGTCTTTGATCGATACTTTTCTGTCGTCGAGTCGTTTTTCTCGTC  \_\_\_\_\_\_\_\_\_\_\_\_\*\*\_\_\_\_\_\_\_\_\_\_\*\*\*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\*\*\*\*\_\_\_\_\_\_\_ |

**Exercise 7.5 Run length coding**

Read an input string from user then print the corresponding run-length of the characters in the string. For example, if the input is *aabccca*, the output should be:

a 2

b 1

c 3

a 1

|  |  |  |
| --- | --- | --- |
| **Test case** | **Input** | **Output** |
| Given | aabccca | a 2  b 1  c 3  a 1 |
| Alphanumeric | a11b22c | a 1  1 2  b 1  2 2  c 1 |
| Single char | x | x 1 |