**Python Lab 6a: Python's string operations**

|  |  |
| --- | --- |
| 1. String operations create a new string based on the first.  **word1 = 'Pizza'**  **print(word1.upper())**  **# prints PIZZA**  **print(word1.lower())**  **# prints pizza**  Two strings can be checked for equality using the == operator.  **print ('house' == 'hat')**  **# prints out False** | Write a code that asks the user to enter a phrase. If the phrase equals the same phrase in uppercase, tell the user 'Stop shouting please!'  **Enter a phrase: Nice to see you**  **Enter a phrase: NICE TO SEE YOU**  **Stop shouting please!** |
| 2. The **in** command can check to see if a certain character is in a string.  **if ('e' in 'house'):**  **count = count + 1** | Write a code that asks the user to enter a string.  Count the number of different vowels ( a, e, i, o, u) that are in the string and print out the total.  You may need to write 5 different **if** statements, one for each vowel.  **Enter a string: mouse**  **mouse has 3 different vowels** |
| 3. Relational operators, such as < and > can be used with strings. One string is "less than" another if it comes before the string when they are placed in alphabetical order.  **print ('mouse' < 'cat')**  **# prints out False** | Write code that asks the user to enter two strings. Store the strings in two different variables.  Print out the string that comes first in alphabetical order.  **Enter a string: mouse**  **Enter another string: cat**  **cat comes before mouse** |
| 4. Two strings can be checked for inequality using the **!=** operator.  **print ('house' != 'hat')**  **# prints out True**  **print ('house' != 'House')**  **# prints out True** | A certain website asks people making an account to enter their email address twice. Write code that asks the user to enter their email address. Ask them to enter it again. Then, while the two strings are not equal, print out a message stating that the two inputs are not equal, and ask the user to enter each again. Print out 'Thank You!' at the end of the program.  **Enter your email address:** [**person@here.com**](mailto:tperson@here.com)  **Enter your email address again:** [**persn@here.com**](mailto:tpersn@here.com)  **The two inputs did not match.**  **Enter your email address:** [**person@here.com**](mailto:tperson@here.com)  **Enter your email address again:** [**person@here.com**](mailto:tpersn@here.com)  **Thank you!** |

**Python Lab 6b: for loops with Strings with cryptography  
While it is not necessary to memorize the ASCII table, it will be helpful to look at one during this lab.  
Search the web for 'ASCII table’**

|  |  |
| --- | --- |
| 1. The chr() function takes a single integer and returns the corresponding ASCII character.  **print (chr(72)) # prints H** | Use a while loop to print out the characters corresponding to the ASCII values between 60 and 70, inclusive. |
| 2. The for command allows the programmer to loop through each character of a string. **word = 'hello'**  **word = word.upper() # change to upper for letter in word:  print (letter)**  Enter the code above and run it. | The variable name letter is not a reserved word, and this code works the same if you change this variable to something else.  Change the variable letter to another name.  Run the code to make sure it still works.  Then change it back to letter. |
| 3 The ord() function takes a single character and returns its ASCII value.  **print (ord('A')) # prints 65** | Inside your for loop, change the print statement so that it prints out both the letter and the ASCII value.  **H 72**  **E 69**  **L 76**  **L 76**  **O 79** |
| 4. One well-known way to encrypt messages is to use what is known as a Caesar cipher. In this cipher, each letter is shifted by the same number.  Complete this code which shifts a letter by 4. | Change your print statement to the following:  **print (letter, chr(ord(letter) + 4)** |
| 5. We can build a new string, letter by letter, by starting with an empty string and adding one letter at a time.  **Change your code to match the example at the right.** | **new\_word = ''**  **for letter in word:**  **new\_letter = chr(ord(letter) + 4)**  **print (letter, new\_letter)**  **new\_word = new\_word + new\_letter**  **print (new\_word)** |
| 6 . To make your Caesar cipher complete, ask the user to enter the word they want to encode, and enter the shift amount. | **enter a phrase to encode: hello there**  **enter the shift: 6**  **the new phrase is: NKRRU&ZNKXK** |
| 7. Challenge: Can you find a way to allow spaces to remain spaces? | **enter a phrase to encode: hello there**  **enter the shift: 6**  **the new phrase is: NKRRU ZNKXK** |