Computer Seminar - Strings

Please ensure you complete parts: A. indexing & slicing strings, B. String methods, C. Finding data in strings and D. Traversing through strings. Part E is an extended exercise.

A. Indexing and slicing strings

The first character in a string is position 0. To slice a string is to take a portion of or a string.

```
pizza = 'Hawaiian'
```

Position	0	1	2	3	4	5	6	7
String characters	H	a	w	a	i	i	a	n

Exercise 1. 15 minutes
Match the code with the correct output. **This exercise can be answered on Blackboard** -> Formative Tests - > **Strings - Exercise 1 (match code with answer)**

	Code	Matches a) to h)		Output
1	`print(pizza[0])`		a)	wa
2	`print(pizza[:5])`		b)	a
3	`print(pizza[0:2])`		c)	Hawai
4	`print("pizza[6]")`		d)	Hawaiian
5	`print(pizza)`		e)	n
6	`print(pizza[6])`		f)	aiian
7	`print(pizza[2:4])`		g)	pizza[6]
8	`print(pizza[3:])`		h)	Ha
9	`print(pizza[-1])`		i)	Hawaiia
10	`print(pizza[:-1])`		j)	H

Exercise 2. Write a program to ask the user for their full name. Using indexing and slicing output:

a. The whole name
b. the first character
c. the first 5 characters
d. the fourth, fifth and sixth characters

Exercise 3. **Length of a string - len().** Write a program which asks for a username of 6 characters in length. It should check the data input length and state if they have entered six characters or not.

B. String Methods

In Python, strings are objects and objects have certain built-in methods that perform a certain task. Dot (.) notation allows us to use methods belonging to an object.

Exercise 4 - Apply the string methods lower(), upper() and capitalize() to the following string, line by line, and print out the returned results.

```
str = '4cosc001W'
```

- str.**upper**() - Return a copy of the string ***str*** with characters converted to uppercase
- str.**lower**() - Return a copy of the string ***str*** with characters converted to lowercase.
- str.**capitalize**() - Return a copy of the string ***str*** with its first character capitalized.

Exercise 5 - One common task is to remove white space (spaces, tabs, or newlines) from the beginning and end of a string using the **strip** method. Run the following to check the output.

```
line = '  more white space   '
print(line)
print(line.strip())
print(line)
```

Do string methods change the string?

The string method above returns a **copy of the string** with any changes made but **does not change the original string**. This is okay if we only want to print the string returned. If the changed value is required later in the program then store the return value back to the original string or to a new string. Note the change to the program below:

```
line = '  lots of white space   '
print(line)
line = line.strip()
print(line)
```

For more string methods see: https://docs.python.org/3.9/library/stdtypes.html#string-methods

C. Finding data in strings

True or **False** will be returned when you ask Python if a string contains a value.

```
guesses = 'abcde'
letter = 'z'
if letter in guesses:        # true if in guesses
       print(letter)
else:                       # false if not found in guesses
       print(' _ ')
```

D. Traversing a string

A for loop can be used to **traverse** through a string accessing each string character one at a time.

Example 1:

```
name = "liz"
for char in name:
     print(char.upper()*3)
```

- Each time through the loop, the next character in the string is assigned to the variable `char`. Within the loop, each letter is converted to uppercase and printed three times. The loop continues until no characters are left.
- Note the string variable name replaces the range() function.

Example 2. Here we introduce a count to the loop and use '**f-strings**' formatting.

```
message = "Testing"
count = 0
for character in message:
    print(f'Index:{count}, Character:{character}')
    count += 1
```

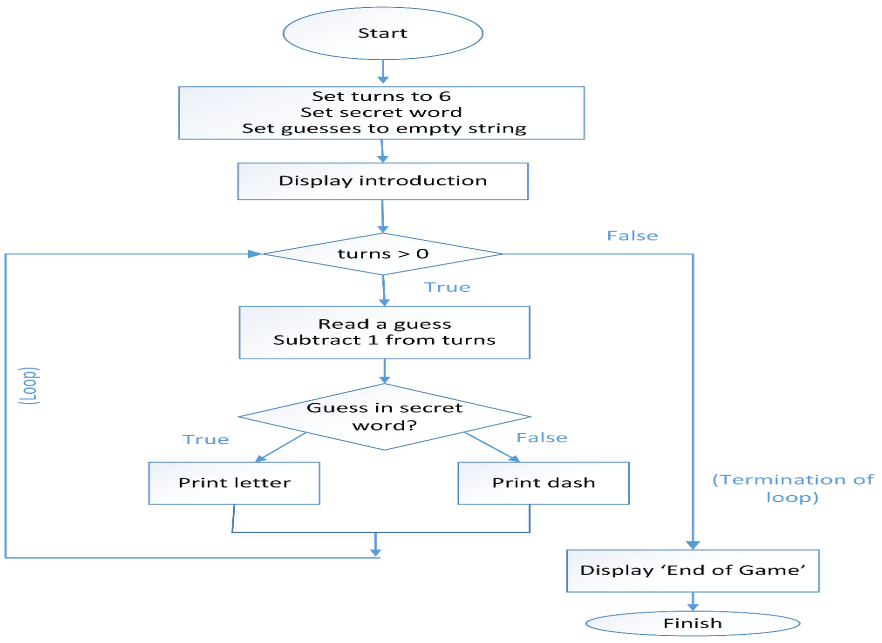
E. Extended Exercise: Guess the word

This exercise will use *flowcharts*, *strings*, *for* loops, *while* loops and *if-else* conditions.

Create a python program to play 'Guess the Word'.

- The program will store a secret word (variable *secret*)
- The player will guess a letter in the word (variable *guess*).
- The program will store each guess entered by the player (variable *guesses*).
- The player will be allowed a number of guesses (variable *turns*).
- The program will print a dash (_) for each letter in the secret word not yet found.
- If the players guess is found in the word, display the letter in replace of the appropriate dash.
- The program will display appropriate introduction and end of game messages.

Sample Flowchart – <u>This relates to exercises 1, 2 and 3 only.</u> You should work out how to design and implement exercise 4 and the additional exercises.



1. Getting started (shown on flowchart)
   - Store the string 'water' in a variable called *secret*.
   - Set variable *turns* to 6 (then you can subtract 1 from turns each time the player takes a guess).
   - Store an empty string in a variable called guesses (to store a string of all the guesses entered by the player).
   - Display introduction:
     - Print a message that announces the start of the guess the word game.
     - Print the number of turns the player has.
     - Print the letters in the secret word replaced by a dash (underscore).
       - **Hint:** In Python the * operator can be used to performs repetition on strings. Therefore, you can print ' _ ' multiplied by the length of the string

   - Example output:

```
Let's play Guess the Word
You have 6 turns to guess the word!
 _  _  _  _  _
```

- Save and run the program to check that it works as expected.

2. Ask player to guess a letter (shown on flowchart)
   A) Update the program to let the player enter a letter and store it in a variable named *guess*. Then concatenate (join) the guess to the *guesses* variable.
   B) The program should show where any guessed letters are. Use a *for* loop to traverse the string variable *secret* letter by letter. For each loop, check if the letter in the secret word is stored in variable *guesses*.
      a. If it is in *guesses*, print the letter.
      b. Otherwise, print a dash (underscore)

```
for letter in secret:
    if letter in guesses:
         print('', letter, '', end='')
    else:
         print(' _ ', end='')
```

- Add an end of game message.
- Example output (user input in bold):

```
Let's play Guess the Word
You have 6 turns to guess the word!
 _  _  _  _  _
Guess a letter: w
w  _  _  _  _
End of Game
```

- Remember to use the correct indentation for the loops and conditions.
- Save and run file. The program should allow the user to guess a letter and show where it is in the string if it is found.

3) Allow the player to take multiple guesses (shown on flowchart)
- Use a while loop to repeat the program until the player is out of turns. Earlier you created a variable called *turns* and set it to 6.
- Control the while loop with a condition that checks if *turns* is above zero.
  - Within the loop subtract one from *turns*.
  - When turns is finally zero, the "while" will stop repeating.

Example output (user input in bold):

```
Let's play Guess the Word
You have 6 turns to guess the word!
 _  _  _  _  _
Guess a letter: e
 _  _  _ e _
Guess a letter: i
 _  _  _ e _
```

```
Guess a letter: w
 w  _  _ e _

Guess a letter: x
 w  _  _ e _

Guess a letter: y
 w  _  _ e _

Guess a letter: z
 w  _  _ e _

End of Game
```

- Run your program and test that it works.

4) Allow the program to work with input that is lowercase or uppercase.
5) The program should base the number of turns based on the length of the word and try different secret words.
6) Try different adding different words to the variable **secret**. If a user enters a word instead of a character, how could you just take the first letter of the word (think indexing).

Additional Exercises - Improve the program

7) Player should win when word is guessed (NOT shown on flowchart)

The current flowchart and program show the program loops until the number of turns reaches zero.

How can we check if the word has been guessed and then terminate the loop early? **Hints:**

- We know that the word has not been guessed if dashes are printed for missing letters. Therefore, keep a count of each dash printed in a variable called *missed*.
- If the number of dashes becomes zero then the word has been found and you can terminate the loop. Which keyword can be used to terminate a loop early?

8) The player should only lose a turn on a wrong guess. Let the user know how many turns are left.