11 Strings

A string is any sequence of characters such as letters, numerals, symbols and punctuation marks. A string has a certain length and with a little bit of coding, can be divided, reversed or changed to uppercase etc. We can operate on strings using functions and methods.

Aim: *To investigate the manipulation of strings in Python.*

Task 1 – String Functions

A function is a section of programming code that performs a specific task. It is common to enter some data into the function (the input) and receive some different data back (the output). There are lots of built-in functions in Python that you can make use of. You can also write your own functions to carry out a particular job.

The syntax for using a function with a string is: *function(string)* e.g.: `len(my_string)`

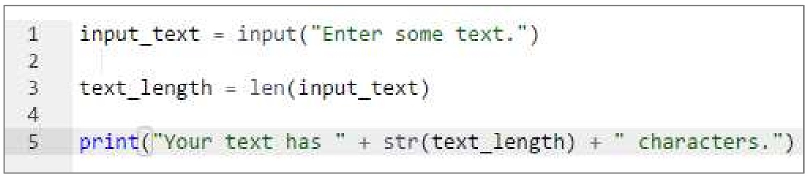
Note: *If you have worked through all tasks up to this point, you will have used the built-in functions **input**, **int**, **str**, **float** and **print**.*

len is a built-in function which tells you the length of a string.

len("rose") is 4, as there are 4 letters in the word "rose". Notice that the word *rose* has to be placed in quotes in the code.

If a variable *my_var* is assigned the word "horse", then *len(my_var)* is 5. We don't need to place quotes around a variable in the code.

Set up the program on the right in *repl.it* and see how it works. Save as '***11.1 len***'.



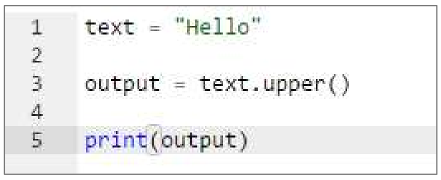
```python
input_text = input("Enter some text.")

text_length = len(input_text)

print("Your text has " + str(text_length) + " characters.")
```

Task 2 – String Methods

A method is in many ways similar to a function. In technical terms, it is a section of code that acts on a certain object such as a string. Both functions and methods are examples of *procedures*.

The syntax for using a method with a string is: *string.method()* e.g.: `my_string.upper()`

upper is a method that converts a string to uppercase letters.



```python
text = "Hello"

output = text.upper()

print(output)
```

a. Set up the program on the right in *repl.it* and see how it works. Save as '***11.2 upper***'.

b. Write down the output you get from this program. ____________________

c. What happens if you add some numbers and symbols to the original string? ______________________________________

d. Create similar programs that demonstrate each of the methods in the table below. What does each method do?

Method	Save As	Example Use	Example Inputs	Outputs
lower	11.2 lower	output = text.**lower**()	"Hello", "HELLO"	
capitalize	11.2 capitalize	output = text.**capitalize**()	"hello", "HELLO"	
islower	11.2 islower	output = text.**islower**()	"hello", "HELLO", "Hello"	
isalpha	11.2 isalpha	output = text.**isalpha**()	"Hello", "Hello1", "111"	

Strings (page 2)

Task 3 – Slicing Strings

There is a useful syntax in Python that helps you extract part of a string. You might want the first three characters of a name, for example, or the last 2 characters of a code you are using. At other times, you may need to extract characters from the middle of a string. There are a few rules to understand about how the strings are sliced.

Rules of the Slicing Syntax

1. Each character of a string has an index number. The 1st character has the index, 0; the 2nd character is 1 etc.
2. You can also count backwards from the end of the string, where the last character is -1, the second to last -2 etc.
3. You can separate a section of a string using the syntax string[a:b] where **a** is the index of the first character retrieved and **b** the index of the character after the last one retrieved. For example, *string[0:3]* gives you characters with indexes 0, 1 and 2 whereas *string[3:6]* gives you characters with indexes 3, 4 and 5.
4. If the first part of the index is missing, assume it is a 0 (i.e. the start of the string), so [:7] is the same as [0:7].
5. If the last part of the index is missing, assume it is a -1 (i.e. the end of the string), so [4:] is the same as [4:-1].

a. Read the rules of the syntax carefully and see if you can work out what the output would be from the program below.

Index	0	1	2	3	4	5	6	7	8	9	10
Character	H	e	l	l	o		W	o	r	l	d
From End	-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1

```
1   text = "Hello World"
2
3   print(text[0])
4   print(text[4])
5   print(text[-3])
6   print(text[0:3])
7   print(text[2:7])
8   print(text[3:])
9   print(text[:4])
10  print(text[1:-2])
11  print(text[2:-5])
12  print(text[:-1])
```

L 3. H

L 4. ______________________

L 5. ______________________

L 6. Hel

L 7. ______________________

L 8. ______________________

L 9. ______________________

L 10. ______________________

L 11. ______________________

L 12. ______________________

b. Create the program above in *repl.it* and check your answers against the output. Save as '***11.3 Slicing***'.

c. Create a program that allows you to input some text and then outputs only the first three characters. Save as '***11.3 Left 3***'.

d. Create a program that asks you to input some text, then asks you to input a number. You should output this number of characters from the start of the text. For example, the inputs "This is my text" and "7" should output "This is". You will need to place a variable inside the square brackets to show the number of characters to return. Add comments to explain what you are doing. Save as '***11.3 Left Select***'.

```
print(text[:num_chars])
```

Strings (page 3)

e. Copy and paste the last program into a new session called '***11.3 Right Select***'. Adapt the code so that you are now displaying the required number of characters from the end of the text. For example, the inputs "This is my text" and "7" should output "my text". You will need to find the negative value of the number entered. You could multiply by -1 using code something like that below. Add comments to explain what you are doing.

```
end_num_chars = -num_chars
```

Extension 1

Create a program that takes your input and then outputs the last half of the string, rounded down. If the inputted string has 4 or 5 characters, then display the last 2; if it has 8 or 9 characters, then display the last 4 etc.

You will need the ***len*** and ***int*** functions along with a division (= num_chars/2), a multiplication by -1 and a slice. Add comments to explain what you are doing. Save as '***11.3 Half***'.

Extension 2

Try and make your program from Extension 1 more efficient without changing the way it works. You can do this by placing functions and operations together etc. For example:

```
num_chars = len(text)

half_chars = num_chars/2
```

```
half_chars = len(text)/2
```

You should be able to remove several lines of code. Our solution (partly shown on the right) uses only two. Bear in mind that complicated lines of code which carry out many functions can be difficult for anyone to understand; this is simply an exercise. Save as '***11.3 Efficient***'.

```
text=input("Enter
print(text[-int(le
```

Task 4 – Searching Strings

There are several methods of finding strings within strings. For example, you may want to know if a piece of text contains a certain word you are looking for. Or you may need to find the position of a particular letter in a string. Alternatively, you may want to count how many times a certain word or letter is used. These can all be achieved with Python string methods.

a. Create the short program on the right. It uses the *string.find* method to look for the letter 'a' in the text you input. What outputs do you get when inputting the following pieces of text?

Note: *An index of -1 means that the search term has not been found.*

```
text = input("Enter some text")

print(text.find("a"))
```

1. apple ________ **2.** pear ________ **3.** lemon ________ **4.** banana ________ **5.** GRAPE ________

b. Expand the program so that it also does the things listed below. Save as '***11.4 Searching***'.

- Uses the code *text.rfind("a")* to find the index of the last letter 'a' in the string.
- Uses the code *text.count("a")* to find out how many letter 'a's there are in the string and display this.
- Use the functions *max(text)* and *min(text)* to display the highest and lowest alphabetical characters.
- Uses the code *text.isnumeric()* to see whether all the characters entered are numbers.

Add comments to your <u>output</u> explaining the numbers and letters being displayed.

```
ar_index = text.
print("The last

num_a = text.cou
print("There are

high_alpha = max
print("The highe

low_alpha = min(
print("The lowes

is_numbers = tex
print("Is this t
```