Tutorial II

Intermediate Python Programming

1 – String Manipulation.

Resources created by Andy Colley (@MrAColley)

Learning Goals/Objectives

Be able to read, comprehend, trace, adapt and create Python code that:

- Uses **len** to count the number of characters in a string
- Uses *get char* to get a single character from a string
- Uses **substring** to get a sequence of characters from a string
- Uses *change case* to convert caps to lower case and vice versa

repl.it

Theory - String Manipulation

A **string**?

- **Strings** are a **data type** used by Python.
- All data stored in string variables is treated as **text**, even if it is numeric characters.
- String data is surrounded by **“quotation marks”**

```
“This is a string”
```

```
variable = “This is a string being stored in a variable”
```

repl.it

A **function**?

We have already learned how to create our own functions, but Python also has lots built in.

- They are pre-set sections of code that perform common tasks.
- They have a name and often use **parameters.**
- Functions **return** a value to the program.

→ *We are going to learn about the functions that we can use with strings.*

```
functionName(parameter1,parameter2)
```

repl.it

len()

Returns the number of characters in a string

Human **len()**

len() counts the number of characters in a string and returns the number to the program.

What would the following function calls return?

len(“orange)
len(“Computing.”)
len(“Hello World!”)

repl.it

len() into a variable

When the value is returned, we need to store it somewhere.

We use a variable for this.

```
stringLength = len("Hello World!")
```

1 - Give the variable a sensible name.

2 - Use the len function with the string as the parameter.

repl.it

Getting **len()** of input

1 - Get input just like we have done previously

```
word = input("Enter a word)
wordLength = len(word)


print(wordLength)
```

3 - The number of characters is returned into the *wordLength* variable.

2 - Use the variable with the string in it as a parameter.

4 - Output the variable with the number of characters stored in it

repl.it

Getting len() of input - More efficient code

1 - Get input just like we have done previously

```
word = input("Enter a word)

print(len(word))
```

2 - Use the len function as a parameter of the print function.

repl.it

len with selection

```
word = input("Enter a word)
wordLength = len(word)

if wordLength > 50:
    print("Wow! That's a long word!"
```

1 - Use the *wordLength* variable as part of the condition.

repl.it

len with selection - more efficient code

```
word = input("Enter a word)

if len(word) > 50:
    print("Wow! That's a long word!")
```

1 - Use the *len()* function as part of the condition.

repl.it

Len - Independent Task

Write a program that:

- Asks the user to input an 8 letter word.
- Stores the input in a variable.
- Calculates the length of the word input.
- If the word is more than 8 characters, output ‘Too long’
- If the word is less than 8 characters, output ‘Too short’
- If the word is 8 characters, output ‘Perfect, thank you!’

repl.it

Get Char - String Slicing

Get Char?

- Get Char lets us get a single character from a string
- Python treats strings like lists, each character is given its own index number, starting at 0

0	1	2	3	4	5
P	y	t	h	o	n

repl.it

Using **Slices** to get a character

- We use code called a **slice** to get a substring in Python.
- Slices treat strings like **lists.** Each character is given an index number starting with 0

1. Assign the text to a variable.

```
phrase = "Hello World!"
letter = phrase[4]
print(letter)
```

2. **Slices out character 4 of the string in the *phrase* variable.**

3. Output the variable containing the slice

Get Char - Make

Write a program that:

- Asks the user to input a word and stores it in a suitably named variable
- Asks the user to input a number and stores it in a suitably named variable.
- If the number entered is larger than the length of the word input then output an error message.
- Else output the character from the word at the position input as part of a sentence.

Eg for inputs ‘Jimi’ and ‘2’ the program outputs ‘The letter **m** is at position **2** in your name’.
For inputs ‘Jimi’ and ‘6’ the program outputs ‘The number you entered is too large’.

Substrings - String Slicing

A **substring**?

- A **substring** is part of a string.
- It is **some text from a string** exactly as it appears in the string

```
"This is a string"
```

```
This
This is
is a
is is a st
```

repl.it

A substring

Which of the examples below is a substring of:

“Computer Science”

A) `comp`
B) `uter Sci`
C) `ter Science.`
D) `CompSci`

Using **Slices** to get substrings

- We use code called a **slice** to get a substring in Python.
- Slices treat strings like **lists.** Each character is given an index number starting with 0

1. Assign the text to a variable.

```
phrase = "Hello World!"
subPhrase = phrase[2:5]
print(subPhrase)
```

2. **Slices out characters 2 to 5 of the string in the *phrase* variable.**
Character 5 is **NOT INCLUDED.**

3. Output the variable containing the slice

repl.it

Using **Slices** to get substrings - more efficient code

```
phrase = “Hello World!”
print(phrase[2:5])
```

repl.it

Task - Substrings - Make

Write a program that:

- Asks the user to input a phrase and stores it in a suitably named variable
- Asks the user to input a number between 0 and the length of the phrase and stores it in a suitably named variable.
- Asks the user to input a second number between the first number and the length of the phrase and stores it in a suitably named variable.
- Gets and outputs the substring of characters between the two numbers entered

Eg for inputs ‘I love Computing’ and numbers ‘3’ and ‘7’ the output would be ‘ove’

repl.it

https://repl.it/@MrAColley/17-Substrings-Predict-Run

Change Case

Case?

```
"THIS IS UPPER CASE"

"this is lower case"
```

repl.it

Changing To Uppercase

```
“Hello World!”.upper()

word.upper()
```

1. Put the string or string variable first.

2. Put a . then `upper()`

repl.it

Changing To Lower

```
“Hello World!”.lower()

word.lower()
```

1. Put the string or string variable first.

2. Put a `.` then `lower()`

repl.it

Change case - more efficient code

```
print(“Hello World”.upper())

print(word.upper())
```

repl.it

Change Case - Using In Conditions

```
name = “dAvE”
nameUpper = name.upper()

If nameUpper == “DAVE”:
   print(“Hi Dave”)
```

repl.it

Change Case - Using In Conditions

```
name = “dAvE”

if name.upper() == “DAVE”:
```