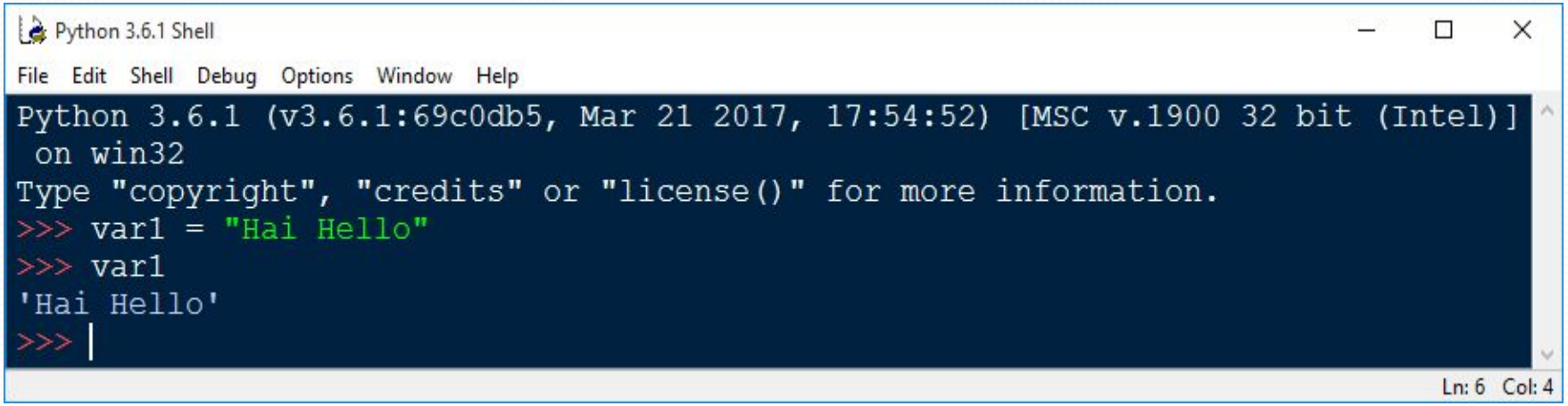
STRINGS

Agenda

- How to create a string
- Accessing values in strings
- Updating Strings
- How to change or delete a string?
- String operations
- String methods
- String Formatting operators

How to create a string

- We can create them simply by enclosing characters in quotes.
- Python treats <u>single quotes same as double quotes</u>.
- Creating strings is as simple as assigning a value to a variable.



Accessing values in strings

- Python does not support a character type; these are treated as strings of length one, thus also considered a substring.
- To access substrings, use the square brackets for slicing along with the index or indices to obtain your substring.

Continue…..

example.py - C:/Users/DELL/AppData/Local/Programs/Python/Python36-32/example.py (3.6.1)

File Edit Format Run Options Window Help

```
var1 = "Hai Hello"
var2 = "Python Programming"

print("var1[0]:",var1[0])
print("var2[1:5]:", var2[1:5])
```

Ln: 4 Col: 17

When the above code is executed, it produces the following result

Python 3.6.1 Shell

File Edit Shell Debug Options Window Help

```
Python 3.6.1 (v3.6.1:69c0db5, Mar 21 2017, 17:54:52) [MSC v.1900 32 bit (Intel)]
 on win32
Type "copyright", "credits" or "license()" for more information.
>>>
 RESTART: C:/Users/DELL/AppData/Local/Programs/Python/Python36-32/example.py
var1[0]: H
var2[1:5]: ytho
>>>
```

Ln: 7 Col: 4

Updating Strings

- You can "update" an existing string by (re)assigning a variable to another string. The new value can be related to its previous value or to a completely different string altogether.

Continue….

updated.py - C:/Users/DELL/AppData/Local/Programs/Python/Python36-32/updated.py (3.6.1)

File Edit Format Run Options Window Help

```
var1='hello world'
print("updated string:-",var1[:6]+'python')
```

Ln: 3 Col: 0

When the above code is executed, it produces the following result

Python 3.6.1 Shell

File Edit Shell Debug Options Window Help

```
Python 3.6.1 (v3.6.1:69c0db5, Mar 21 2017, 17:54:52) [MSC v.1900 32 bit (Intel)]
 on win32
Type "copyright", "credits" or "license()" for more information.
>>>
 RESTART: C:/Users/DELL/AppData/Local/Programs/Python/Python36-32/updated.py
updated string:- hello python
>>>
```

Ln: 6 Col: 4

Continue….

- A string is a sequence of characters. You can access the characters one at a time with the bracket operator

Python 3.4.3 Shell

File Edit Shell Debug Options Window Help

```
Python 3.4.3 (v3.4.3:9b73f1c3e601, Feb 24 2015, 22:43:06) [MSC v.1600 32 bit (In
tel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> fruit="banana"
>>> letter=fruit[1]
>>> print(letter)
a
>>>
```

How to change or delete a string?

Python 3.4.3 Shell

File Edit Shell Debug Options Window Help

```
Python 3.4.3 (v3.4.3:9b73f1c3e601, Feb 24 2015, 22:43:06) [MSC v.1600 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> mt_string='apssdc'
>>> mt_string[5]='a'
Traceback (most recent call last):
  File "<pyshell#1>", line 1, in <module>
    mt_string[5]='a'
TypeError: 'str' object does not support item assignment
>>> mt_string='Python'
>>> mt_string
'Python'
>>>
```

```
>>> del mt_string[1]
Traceback (most recent call last):
  File "<pyshell#4>", line 1, in <module>
    del mt_string[1]
TypeError: 'str' object doesn't support item deletion
>>> del mt_string
>>> mt_string
Traceback (most recent call last):
  File "<pyshell#6>", line 1, in <module>
    mt_string
NameError: name 'mt_string' is not defined
>>>
```

String Operations

- **Concatenation of Two or More Strings:**Joining of two or more strings into a single one is called concatenation.
- The + operator does this in Python. Simply writing two string literals together also concatenates them.
- The * operator can be used to repeat the string for a given number of times.

script.py IPython Shell

```python
str1 = 'Hello'
str2 ='World!'

# using +
print('str1 + str2 = ', str1 + str2)

# using *
print('str1 * 3 =', str1 * 3)

```

script.py IPython Shell

```
str1 + str2 =  HelloWorld!
str1 * 3 = HelloHelloHello
```

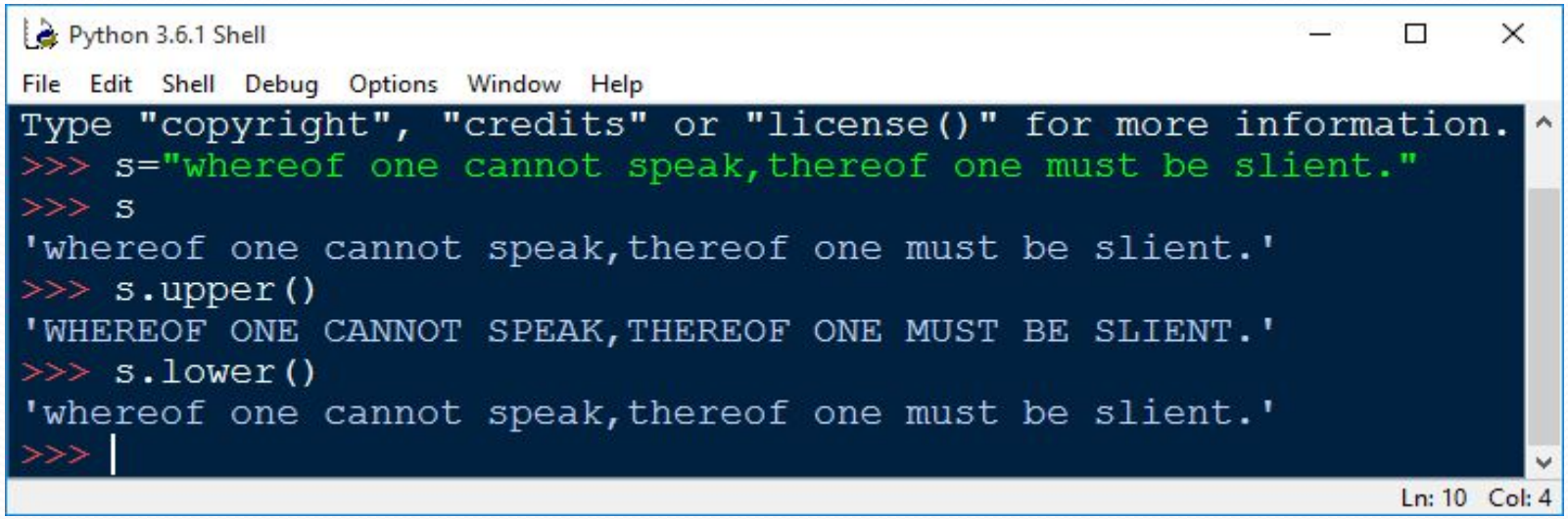
String Membership Test

- We can test if a sub string exists within a string or not, using the keyword in

```
>>> 'a' in 'program'
True
>>> 'at' not in 'battle'
False
```

String Methods

- **S.lower() and S.lower():** Returns the lowercase and upper case version of the string. The .upper() and .lower() string methods are self-explanatory.
- Performing the .upper() method on a string converts all of the characters to uppercase, whereas the lower() method converts all of the characters to lowercase.



Replace

- **S.replace('old', 'new'):** Returns a string where all occurrences of 'old' have been replaced by 'new'

```
Python 3.6.1 Shell
File  Edit  Shell  Debug  Options  Window  Help
Python 3.6.1 (v3.6.1:69c0db5, Mar 21 2017, 17:54:52) [
MSC v.1900 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more in
formation.
>>> s="I intend to live forever,or die trying."
>>> s.replace("to","three")
'I intend three live forever,or die trying.'
>>> s.replace("fore","five")
'I intend to live fivever,or die trying.'
>>> |
Ln: 8  Col: 4
```

Split

- At some point, you may need to break a large string down into smaller chunks, or strings. This is the opposite of concatenation which merges or combines strings into one. To do this, you use the split function.
- **S.split('delim')** : Split() splits or breakup a string and add the data to a string array using a defined separator.
- If no separator is defined when you call upon the function, whitespace will be by default. In simpler terms, the separator is a defined character that will be placed between each variable.

Example

split.py - C:/Users/DELL/AppData/Local/Programs/Python/Python36-32/split.py (3.6.1)

File Edit Format Run Options Window Help

```
str="Line1-abcdef \nLine2-abc \nLine4-abcd"

print(str.split())
print(str.split(' ',2))
```

Ln: 5 Col: 0

When the above code is executed, it produces the following result

Python 3.6.1 Shell

File Edit Shell Debug Options Window Help

```
Python 3.6.1 (v3.6.1:69c0db5, Mar 21 2017, 17:54:52) [MSC v.1900 32 bit (Intel)]
 on win32
Type "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:/Users/DELL/AppData/Local/Programs/Python/Python36-32/split.py =
['Line1-abcdef', 'Line2-abc', 'Line4-abcd']
['Line1-abcdef', '\nLine2-abc', '\nLine4-abcd']
>>>
```

Ln: 7 Col: 4

JION

- s.join(list): The method join () returns a string in which the string elements of sequence have been joined by str separator. Opposite of split(),joins the elements in the given list together using the string as the delimiter.
- **Syntax**
- ```
  str.join(sequence)
  ```
  ```

Example

joi n.py - C:/Users/DELL/AppData/Local/Programs/Python/Python36-32/joi n.py (3.6.1)

File Edit Format Run Options Window Help

```
s="-"
seq = ("a", "b", "c") # This is sequence of strings.

print (s.join( seq ))
```

Ln: 5 Col: 0

When the above code is executed, it produces the following result

Python 3.6.1 Shell

File Edit Shell Debug Options Window Help

```
Python 3.6.1 (v3.6.1:69c0db5, Mar 21 2017, 17:54:52) [MSC v.1900 32 bit (Intel)]
 on win32
Type "copyright", "credits" or "license()" for more information.
>>> s="-";
>>>
= RESTART: C:/Users/DELL/AppData/Local/Programs/Python/Python36-32/joi n.py =
a-b-c
>>>
```

Ln: 7 Col: 4

String capitalize() Method

- It returns a copy of the string with only its first character capitalized.

Syntax

```
str.capitalize()
```

```
>>> name="sree rama"
>>> name.capitalize()
'Sree rama'
>>> frndname="amala"
>>> frndname.capitalize()
'Amala'
>>>
```

Find and len()

Python 3.4.3 Shell

File Edit Shell Debug Options Window Help

```
Python 3.4.3 (v3.4.3:9b73f1c3e601, Feb 24 2015, 22:43:06) [MSC v.1600 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> str="HAPPY NEW YEAR"
>>> str.find('EW')
7
>>> len(str)
14
>>>
```

Reversing String

- By using the reverse function, you can reverse the string. For example, if we have string "selenium" and then if you apply the code for the reverse function as shown below.

rev.py - C:/Users/DELL/AppData/Local/Programs/Python/Python36-32/rev.py (3.6.1)

File Edit Format Run Options Window Help

```
print('hello world'[::-1])
```

Ln: 3 Col: 0

When the above code is executed, it produces the following result

Python 3.6.1 Shell

File Edit Shell Debug Options Window Help

```
Python 3.6.1 (v3.6.1:69c0db5, Mar 21 2017, 17:54:52) [MSC v.1900 32 bit (Intel)]
 on win32
Type "copyright", "credits" or "license()" for more information.
>>>
== RESTART: C:/Users/DELL/AppData/Local/Programs/Python/Python36-32/rev.py ==
dlrow olleh
>>>
```

Ln: 6 Col: 4

String Slices

- Slice Operator is used to extract part of a string or a list. The syntax is simple.
- Actually it looks a little bit like accessing a single element with an index, but instead of just one number we have more, separated with a colon ":".
- We have a start and an end index, one or both of them may be missing. It's best to study the mode of operation of slice by having a look at

Example

```
Python 3.6.1 Shell
File  Edit  Shell  Debug  Options  Window  Help
Python 3.6.1 (v3.6.1:69c0db5, Mar 21 2017, 17:54:52) [MSC v.1900 32 bit (Intel)]
 on win32
Type "copyright", "credits" or "license()" for more information.
>>> s='Don Quijote'
>>> s[4]
'Q'
>>> |
Ln: 6  Col: 4
```

If you want to start counting from the end of the string, instead of the beginning, use a negative index. For example, an index of -1 refers to the right-most character of the string

```
Python 3.6.1 Shell
File  Edit  Shell  Debug  Options  Window  Help
Python 3.6.1 (v3.6.1:69c0db5, Mar 21 2017, 17:54:52) [MSC v.1900 3
2 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> s='Don Quijote'
>>> s[-1]
'e'
>>> s[-7]
'Q'
>>> |
Ln: 8  Col: 4
```

Continue…

Python 3.6.1 Shell

File Edit Shell Debug Options Window Help

```
Python 3.6.1 (v3.6.1:69c0db5, Mar 21 2017, 17:54:52) [MSC v.1900 32 bit (Intel)]
 on win32
Type "copyright", "credits" or "license()" for more information.
>>> s='Don Quijote'
>>> s[4:8]
'Quij'
>>> |
```

Python 3.6.1 Shell

File Edit Shell Debug Options Window Help

```
Python 3.6.1 (v3.6.1:69c0db5, Mar 21 2017, 17:54:52) [MSC v.1900 32 bit (Intel)]
 on win32
Type "copyright", "credits" or "license()" for more information.
>>> s = 'Don Quijote'
>>> s[4:]
'Quijote'
>>> s[:4]
'Don '
>>> s[:]
'Don Quijote'
>>> |
```

Ln: 10 Col: 4

String Operators

- Assume string variable **a =10 and b=20, then**

Operator	Description	Example
+ Addition	Adds values on either side of the operator.	a + b = 30
- Subtraction	Subtracts right hand operand from left hand operand.	a – b = -10
* Multiplication	Multiplies values on either side of the operator	a * b = 200
/ Division	Divides left hand operand by right hand operand	b / a = 2
% Modulus	Divides left hand operand by right hand operand and returns remainder	b % a = 0
** Exponent	Performs exponential (power) calculation on operators	a**b =10 to the power 20

Continue…..

//	Floor Division - The division of operands where the result is the quotient in which the digits after the decimal point are removed. But if one of the operands is negative, the result is floored, i.e., rounded away from zero (towards negative infinity):	9//2 = 4 and 9.0//2.0 = 4.0, -11//3 = -4, -11.0//3 = -4.0

String Formatting Operator

- Python uses C-style string formatting to create new, formatted strings. The "%" operator is used to format a set of variables enclosed in a "tuple" (a fixed size list), together with a format string, which contains normal text together with "argument specifiers", special symbols like "%s" and "%d"

```
>>> "Leader {} who say  {}!".format("Ram","HI Everyone")
'Leader Ram who say  HI Everyone!'
...
```

Format Symbol	Conversion
%c	Character
%s	string conversion via str() prior to formatting
%i	signed decimal integer
%d	signed decimal integer
%u	unsigned decimal integer
%o	octal integer
%x	hexadecimal integer (lowercase letters)
%X	hexadecimal integer (UPPERcase letters)
%e	exponential notation (with lowercase 'e')
%E	exponential notation (with UPPERcase 'E')
%f	floating point real number
%g	the shorter of %f and %e