Exposure CS 2021 for CS1

Chapter 14 Section 1-3 Slides

String Processing:
String Operators & Commands

PowerPoint Presentation
created by:
Mr. John L. M. Schram
and Mr. Leon Schram
Authors of Exposure
Computer Science

Section 14.1

Introduction

String Processing

Word processing term papers, sending email messages, responding to online surveys, and even writing Python programs in an IDE like jGRASP all involve *string processing*.

Every software package on the market includes string-processing components.

Every programming language has special features that facilitate the manipulation of strings, and Python is no different.

What is a string?

Is it a *Simple Data Type* or a *Data Structure*?

```
age = 50
gpa = 3.785
finished = True
title = "Exposure Computer Science"
```

Some people would say **title** is a simple data type because it stores one string value. Other people argue it is a *data structure* because **title** stores several character values. Both are actually true.

String Definition

A *string* is a set of characters that behaves as a single unit.

The characters in a string can include upper-case letters, lower-case letters, numerical digits and a large set of symbols for a variety of purposes like:

! @ # $ % ^ & * ( ) _ +

String Variables vs. String Literals

A *string literal* is a set of characters delimited with quotations.

```
name = "John Smith"
```

name is the string variable.

"John Smith" is the string literal.

Section 14.2

String Operators

Review of String Concatenation

Mathematical Addition

```
x = 100 + 200
```

```
x = 100
x += 200
```

In both cases
x now stores **300**.

String Concatenation

```
x = "100" + "200"
```

```
x = "100"
x += "200"
```

In both cases
x now stores **"100200"**.

```
 1 # StringOperators01.py
 2 # This program reviews "Concatenation"
 3 # with the overloaded <+> operator.
 4
 5
 6 s1 = "Argentine"
 7 s2 = "Tango"
 8 s3 = s1 + " " + s2
 9 print()
10 print(s3)
11
12 s4 = "100"
13 s5 = "200"
14 s6 = s4 + s5
15 print()
16 print(s6)
```

```
----jGRASP exec:

Argentine Tango

100200

----jGRASP: oper
```

```
1  # StringOperators02.py
2  # This program demonstrates that the <+=> operator
3  # is also overloaded and can be used to concatenate
4  # one string to the end of another.
5
6
7  s1 = "Argentine "
8  s2 = "Tango"
9  s1 += s2
10 print()
11 print(s1)
12
13 s4 = "100"
14 s5 = "200"
15 s4 += s5
16 print()
17 print(s4)
```

```
----jGRASP exec:

Argentine Tango

100200

----jGRASP: oper
```

```
1  # StringOperators03.py
2  # This program demonstrates that the index operator []
3  # can be used to access the individual characters in a
4  # string, as if a string were an array of characters.
5  # NOTE: Like arrays, string indexes also start with 0.
6
7
8  state = "TEXAS"
9  print()
10 print(state[2])
```

```
----jGRASP

X

----jGRASP:
```

0	1	2	3	4
T	E	X	A	S

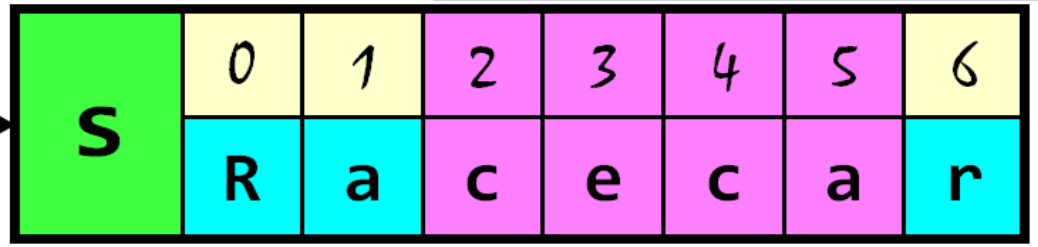
```
 1 # StringOperators04.py
 2 # This program demonstrates how to access
 3 # a piece of a string.
 4 # This is usually called a "sub-string".
 5 # It is accomplished via "String Slicing"
 6 # which is very similar to "Array Slicing".
 7
 8
 9 s = "Racecar"
10
11 print()
12 print(s[0:4])
13 print(s[1:4])
14 print(s[2:4])
15 print(s[2:6])
16 print(s[3:6])
17 print(s[4:7])
18 print(s[4:])
19 print(s[:4])
```

```
 1 # StringOperators04.py
 2 # This program demonstrates hov
 3 # a piece of a string.
 4 # This is usually called a "sul
 5 # It is accomplished via "Strir
 6 # which is very similar to "Arr
 7
 8
 9 s = "Racecar"
10
11 print()
12 print(s[0:4])
13 print(s[1:4])
14 print(s[2:4])
15 print(s[2:6])
16 print(s[3:6])
17 print(s[4:7])
18 print(s[4:])
19 print(s[:4])
```

```
----jGRASP

Race
ace
ce
ceca
eca
car
car
Race

----jGRASP:
```



Start at index 2.
Stop before index 6.

```
1 # StringOperators05.py
2 # This program demonstrates that you can
3 # actually "multiply" a string by an integer
4 # with the overloaded <*> operator.
5 # NOTE: This is very similar to
6 #       "Array Multiplication".
7
8
9 s1 = "Racecar"
10 s2 = s1 * 3
11
12 print()
13 print(s1)
14 print(s2)
```

```
----jGRASP exec: python StringOperators05


Racecar
RacecarRacecarRacecar


----jGRASP: operation complete.
```

```
 7
 8
 9 s1 = "Racecar"
10 s2 = s1 * 3
11
12 print()
13 print(s1)
14 print(s2)
```

```
----jGRASP exec: python StringOperators05

Racecar
RacecarRacecarRacecar

----jGRASP: operation complete.
```

```
 7
 8
 9 s1 = "Racecar"
10 s2 = s1 * 3
11
12 print()
13 print(s1)
14 print(s2)
```

NOTE: When using the terms “Array Multiplication” and “String Multiplication”, it needs to be understood that Arrays and Strings can only be “multiplied” by integer values. You cannot “multiply” an array by another array or a string by another string.

```
1 # StringOperators06.py
2 # This program demonstrates the "is equal to"
3 # operator == can be used to compare 2 strings
4 # for equality.
5
6
7 s1 = "Foxtrot"
8 s2 = "Waltz"
9 s3 = "Foxtrot"
10
11 print()
12 print(s1 == s2)
13 print(s1 == s3)
14
```

```
----jGRASP

False
True

----jGRASP:
```

```
1  # StringOperators07.py
2  # This program demonstrates that the "greater than" > and
3  # less than < operators can compare strings alphabetically.
4  # NOTE: This program will not work properly if one string
5  # starts with a CAPITAL letter and the other string does not.
6  # A later program example will fix this.
7
8
9  print()
10 s1 = input("Enter 1st string.  -->  ")
11 s2 = input("Enter 2nd string.  -->  ")
12 print()
13
14 if s1 < s2:
15     print(s1,"goes alphabetically before",s2)
16 elif s1 > s2:
17     print(s1,"goes alphabetically after",s2)
18 else:
19     print("Both strings are equal")
```

```
----jGRASP exec: python StringOperators07

Enter 1st string.  -->  NEON
Enter 2nd string.  -->  ZEBRA

NEON goes alphabetically before ZEBRA

----jGRASP: operation complete.
```

```
----jGRASP exec: python StringOperators07

Enter 1st string.  -->  Computer
Enter 2nd string.  -->  Computer

Both strings are equal

----jGRASP: operation complete.
```

```
----jGRASP exec: python StringOperators07

Enter 1st string.  -->  banana
Enter 2nd string.  -->  apple

banana goes alphabetically after apple

----jGRASP: operation complete.
```

```
----jGRASP exec: python StringOperators07

Enter 1st string.  -->  apple
Enter 2nd string.  -->  ZEBRA

apple goes alphabetically after ZEBRA

----jGRASP: operation complete.
```

NOTE: The issue with comparing strings with different cases was first introduced in the previous chapter. Later in this chapter the issue will finally be resolved.

Multiline Strings & Really Long Strings

```python
# StringOperators08.py
# This program demonstrates two ways of dealing
# with very long string literals.


s = "The quick brown fox jumps over the lazy dog" + \
    " on alternate Tuesdays during leap year."
print()
print(s)

s = ("The quick brown fox jumps over the lazy dog"
    " on alternate Tuesdays during leap year.")
print(s)

```

```
 ----jGRASP exec: python StringOperators06.py

The quick brown fox jumps over the lazy dog on alternate Tuesdays during leap year.
The quick brown fox jumps over the lazy dog on alternate Tuesdays during leap year.

 ----jGRASP: operation complete.
```

```
 1 # StringOperators09.py
 2 # This program demonstrates printing a
 3 # multi-line string literal.
 4
 5
 6 print()
 7 print("""The quick
 8           brown fox
 9           jumps over
10           the lazy dog.""")
11
12 print()
13 print("""The quick
14 brown fox
15 jumps over
16 the lazy dog.""")
17
```

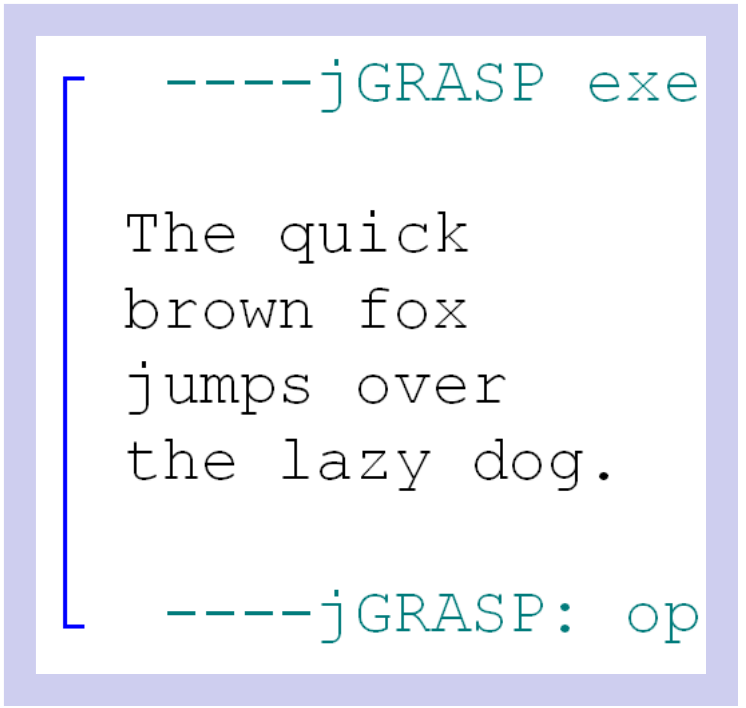
```
----jGRASP exec: pyth

The quick
          brown fox
          jumps over
          the lazy dog.

The quick
brown fox
jumps over
the lazy dog.

----jGRASP: operation
```

```
1  # StringOperators10.py
2  # This program demonstrates storing a
3  # multi-line string literal in a string
4  # variable and then printing it.
5
6
7  s = """The quick
8  brown fox
9  jumps over
10 the lazy dog."""
11
12 print()
13 print(s)
14
```



Section 14.3

String Commands

```
1  # StringCommands01.py
2  # This program demonstrates <len> command.
3  # In the same way that <len> will tell you
4  # how many items are in an array, it will also
5  # tell you how many characters are in a string.
6
7
8  s1 = "Argentine"
9  s2 = "Tango"
10 s3 = s1 + " " + s2
11
12 print()
13 print(s1,"has",len(s1),"characters.")
14 print(s2,"has",len(s2),"characters.")
15 print(s3,"has",len(s3),"characters.")
```

```
----jGRASP exec: python StringCommands01

Argentine has 9 characters.
Tango has 5 characters.
Argentine Tango has 15 characters.

----jGRASP: operation complete.
```

```
 8 s1 = "Argentine"
 9 s2 = "Tango"
10 s3 = s1 + " " + s2
11
12 print()
13 print(s1,"has",len(s1),"characters.")
14 print(s2,"has",len(s2),"characters.")
15 print(s3,"has",len(s3),"characters.")
```

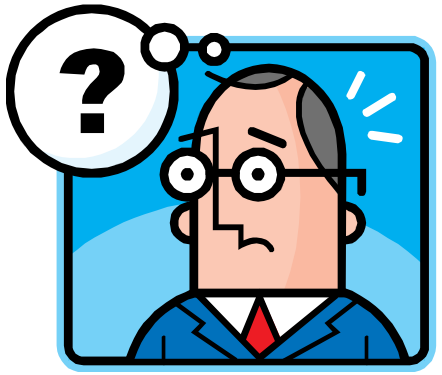
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
A	r	g	e	n	t	i	n	e		T	a	n	g	o

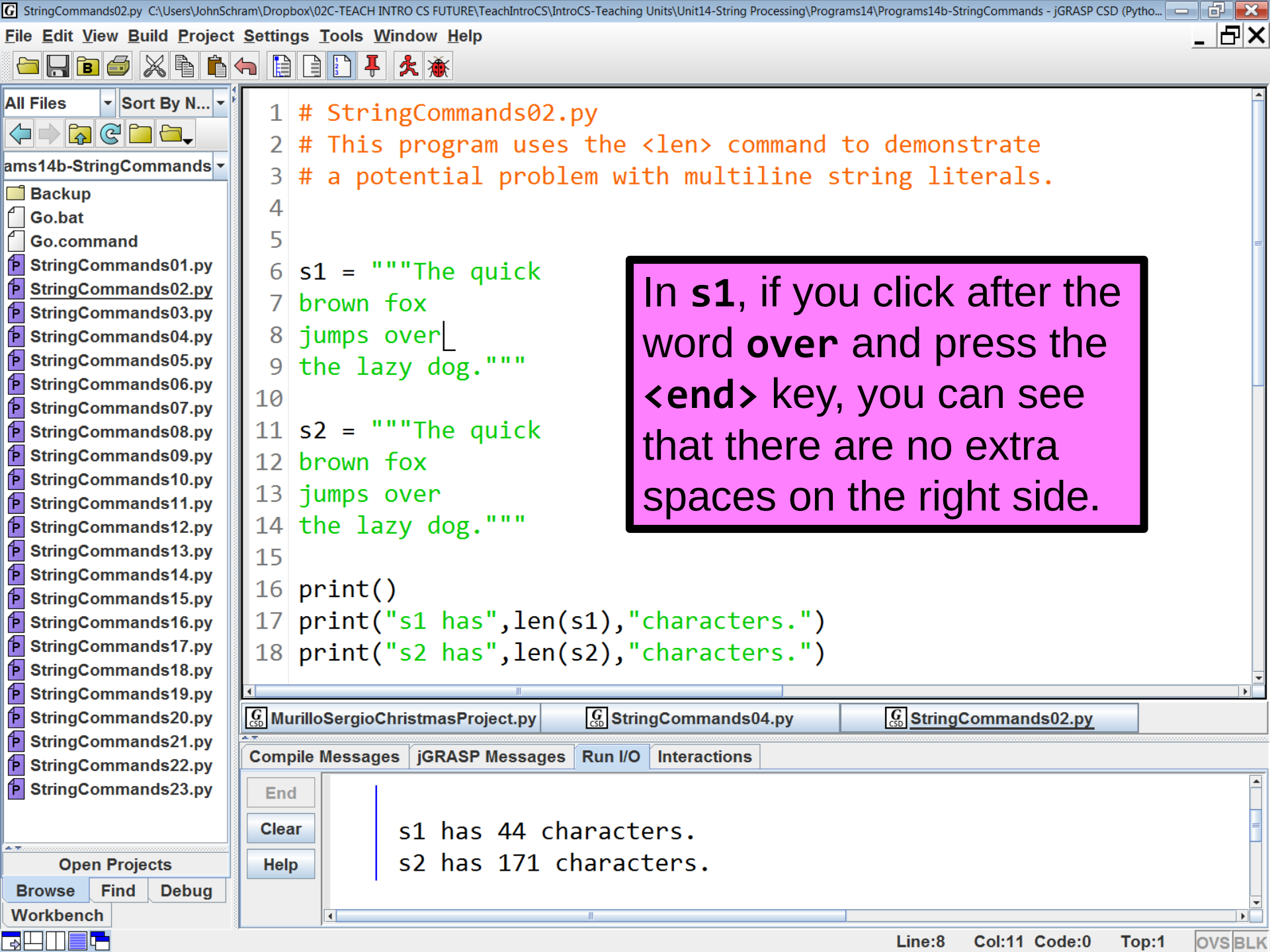
```
1  # StringCommands02.py
2  # This program uses the <len> command to demonstrate
3  # a potential problem with multiline string literals.
4
5
6  s1 = """The quick
7  brown fox
8  jumps over
9  the lazy dog."""
10
11 s2 = """The quick
12 brown fox
13 jumps over
14 the lazy dog."""
15
16 print()
17 print("s1 has",len(s1),"characters.")
18 print("s2 has",len(s2),"characters.")
```

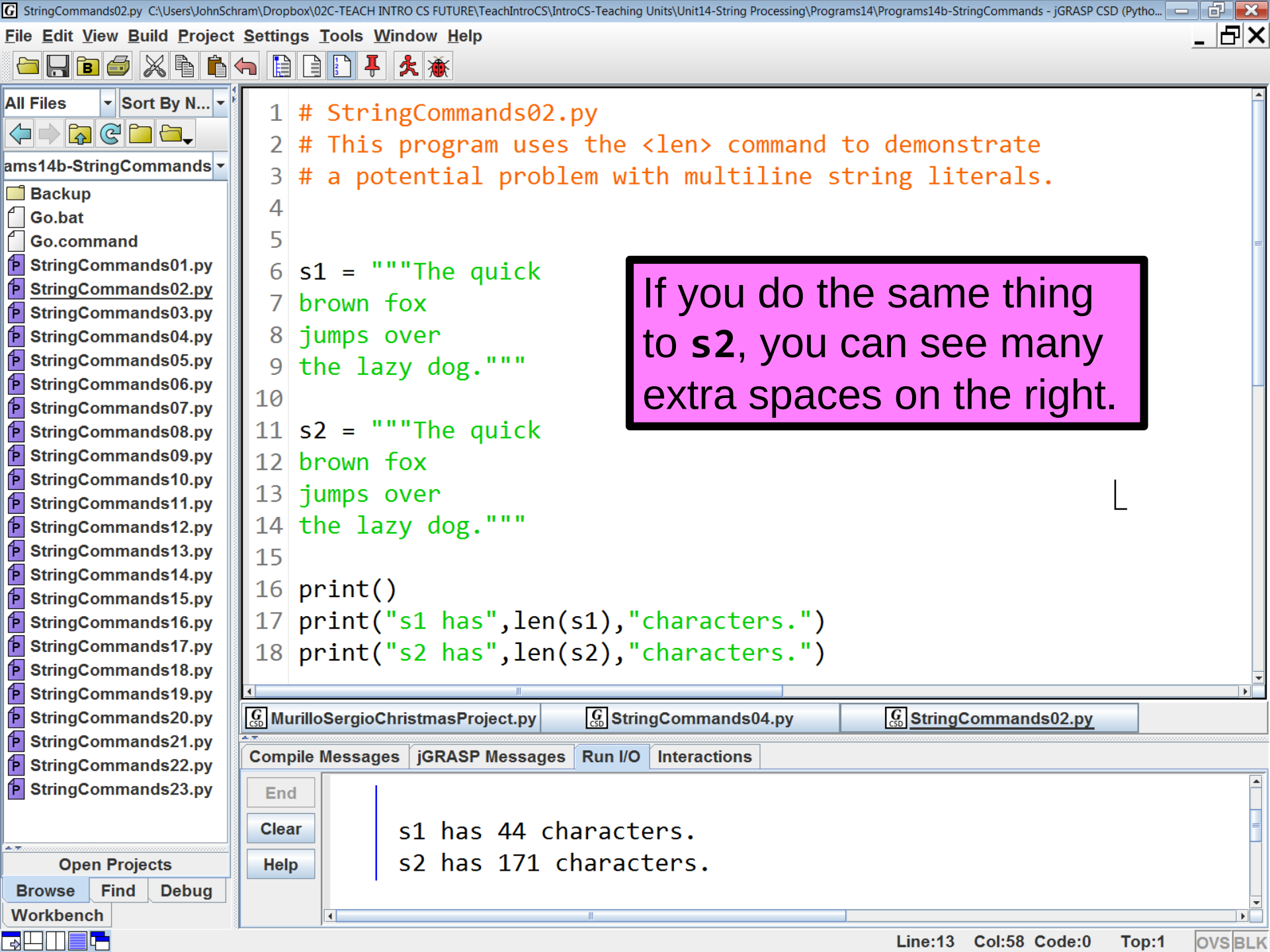
```
  ----jGRASP exec: pyth

 s1 has 44 characters.
 s2 has 171 characters.

  ----jGRASP: operation
```




StringCommands02.py C:\Users\JohnSchram\Dropbox\02C-TEACH INTRO CS FUTURE\TeachIntroCS\IntroCS-Teaching Units\Unit14-String Processing\Programs14\Programs14b-StringCommands - jGRASP CSD (Pytho...
File Edit View Build Project Settings Tools Window Help
All Files
Sort By N...
ams14b-StringCommands
Backup
Go.bat
Go.command
StringCommands01.py
StringCommands02.py
StringCommands03.py
StringCommands04.py
StringCommands05.py
StringCommands06.py
StringCommands07.py
StringCommands08.py
StringCommands09.py
StringCommands10.py
StringCommands11.py
StringCommands12.py
StringCommands13.py
StringCommands14.py
StringCommands15.py
StringCommands16.py
StringCommands17.py
StringCommands18.py
StringCommands19.py
StringCommands20.py
StringCommands21.py
StringCommands22.py
StringCommands23.py
Open Projects
Browse
Find
Debug
Workbench
1 # StringCommands02.py
2 # This program uses the <len> command to demonstrate
3 # a potential problem with multiline string literals.
4
5
6 s1 = """The quick
7 brown fox
8 jumps over
9 the lazy dog."""
10
11 s2 = """The quick
12 brown fox
13 jumps over
14 the lazy dog."""
15
16 print()
17 print("s1 has",len(s1),"characters.")
18 print("s2 has",len(s2),"characters.")
In s1, if you click after the word over and press the <end> key, you can see that there are no extra spaces on the right side.
MurilloSergioChristmasProject.py
StringCommands04.py
StringCommands02.py
Compile Messages
jGRASP Messages
Run I/O
Interactions
End
Clear
Help
s1 has 44 characters.
s2 has 171 characters.
Line:8 Col:11 Code:0 Top:1
OVS BLK


StringCommands02.py C:\Users\JohnSchram\Dropbox\02C-TEACH INTRO CS FUTURE\TeachIntroCS\IntroCS-Teaching Units\Unit14-String Processing\Programs14\Programs14b-StringCommands - jGRASP CSD (Pytho...
File Edit View Build Project Settings Tools Window Help
All Files
Sort By N...
ams14b-StringCommands
Backup
Go.bat
Go.command
StringCommands01.py
StringCommands02.py
StringCommands03.py
StringCommands04.py
StringCommands05.py
StringCommands06.py
StringCommands07.py
StringCommands08.py
StringCommands09.py
StringCommands10.py
StringCommands11.py
StringCommands12.py
StringCommands13.py
StringCommands14.py
StringCommands15.py
StringCommands16.py
StringCommands17.py
StringCommands18.py
StringCommands19.py
StringCommands20.py
StringCommands21.py
StringCommands22.py
StringCommands23.py
Open Projects
Browse Find Debug
Workbench
1 # StringCommands02.py
2 # This program uses the <len> command to demonstrate
3 # a potential problem with multiline string literals.
4
5
6 s1 = """The quick
7 brown fox
8 jumps over
9 the lazy dog."""
10
11 s2 = """The quick
12 brown fox
13 jumps over
14 the lazy dog."""
15
16 print()
17 print("s1 has",len(s1),"characters.")
18 print("s2 has",len(s2),"characters.")
If you do the same thing to s2, you can see many extra spaces on the right.
MurilloSergioChristmasProject.py
StringCommands04.py
StringCommands02.py
Compile Messages
jGRASP Messages
Run I/O
Interactions
End
Clear
Help
s1 has 44 characters.
s2 has 171 characters.
Line:13 Col:58 Code:0 Top:1
OVS BLK

```
1  # StringCommands03.py
2  # This program demonstrates how to "traverse"
3  # a string using a <for> loop.  Note that this
4  # can be done both forwards and backwards.
5
6
7  s = "COMPUTER"
8  n = len(s)
9
10 print()
11 for k in range(n):
12     print(s[k])
13 print()
14
15 # Count from the last index (n-1)
16 # to the first index (0) backwards.
17 for k in range(n-1,-1,-1):
18     print(s[k], end = "")
19 print()
```

```
 1 # StringCommands03.py
 2 # This program demonstrates h
 3 # a string using a <for> loop
 4 # can be done both forwards a
 5
 6
 7 s = "COMPUTER"
 8 n = len(s)
 9
10 print()
11 for k in range(n):
12     print(s[k])
13 print()
14
15 # Count from the last index (n-1)
16 # to the first index (0) backwards.
17 for k in range(n-1,-1,-1):
18     print(s[k], end = "")
19 print()
```

```
----jGRASP

C
O
M
P
U
T
E
R

RETUPMOC

----jGRASP:
```

```
 1 # StringCommands04.py
 2 # This program demonstrates how to take
 3 # one string and create a second string
 4 # that is a reverse of the first.
 5
 6
 7 s1 = "Madam I'm Adam"
 8 s2 = ""
 9 n = len(s1)
10
11 # Count from the last index (n-1)
12 # to the first index (0) backwards.
13 for k in range(n-1,-1,-1):
14     s2 += s1[k]
15
16 print()
17 print(s1)
18 print(s2)
```

```
 1 # StringCommands04.py
 2 # This program demonstrates how to take
 3 # one string and create a second string
 4 # that is a reverse of the first.
 5
 6
 7 s1 = "Madam I'm Adam"
 8 s2 = ""
 9 n = len(s1)
10
11 # Count from the last index (n-1)
12 # to the first index (0) backwards.
13 for k in range(n-1,-1,-1):
14     s2 += s1[k]
15
16 print()
17 print(s1)
18 print(s2)
```

n
14

	0	1	2	3	4	5	6	7	8	9	10	11	12	13
s1	M	a	d	a	m		I	'	m		A	d	a	m
s2														

```
 1 # StringCommands04.py
 2 # This program demonstrates how to take
 3 # one string and create a second string
 4 # that is a reverse of the first.
 5
 6
 7 s1 = "Madam I'm Adam"
 8 s2 = ""
 9 n = len(s1)
10
11 # Count from the last index (n-1)
12 # to the first index (0) backwards.
13 for k in range(n-1,-1,-1):
14     s2 += s1[k]
15
16 print()
17 print(s1)
18 print(s2)
```

n	k
14	13

	0	1	2	3	4	5	6	7	8	9	10	11	12	13
s1	M	a	d	a	m		I	'	m		A	d	a	m
s2	m													

```
 1 # StringCommands04.py
 2 # This program demonstrates how to take
 3 # one string and create a second string
 4 # that is a reverse of the first.
 5
 6
 7 s1 = "Madam I'm Adam"
 8 s2 = ""
 9 n = len(s1)
10
11 # Count from the last index (n-1)
12 # to the first index (0) backwards.
13 for k in range(n-1,-1,-1):
14     s2 += s1[k]
15
16 print()
17 print(s1)
18 print(s2)
```

n	k
14	12

	0	1	2	3	4	5	6	7	8	9	10	11	12	13
s1	M	a	d	a	m		I	'	m		A	d	a	m
s2	m	a												

```
 1 # StringCommands04.py
 2 # This program demonstrates how to take
 3 # one string and create a second string
 4 # that is a reverse of the first.
 5
 6
 7 s1 = "Madam I'm Adam"
 8 s2 = ""
 9 n = len(s1)
10
11 # Count from the last index (n-1)
12 # to the first index (0) backwards.
13 for k in range(n-1,-1,-1):
14     s2 += s1[k]
15
16 print()
17 print(s1)
18 print(s2)
```

n	k
14	11

	0	1	2	3	4	5	6	7	8	9	10	11	12	13
s1	M	a	d	a	m		I	'	m		A	d	a	m
s2	m	a	d											

```
 1 # StringCommands04.py
 2 # This program demonstrates how to take
 3 # one string and create a second string
 4 # that is a reverse of the first.
 5
 6
 7 s1 = "Madam I'm Adam"
 8 s2 = ""
 9 n = len(s1)
10
11 # Count from the last index (n-1)
12 # to the first index (0) backwards.
13 for k in range(n-1,-1,-1):
14     s2 += s1[k]
15
16 print()
17 print(s1)
18 print(s2)
```

n	k
14	10

	0	1	2	3	4	5	6	7	8	9	10	11	12	13
s1	M	a	d	a	m		I	'	m		A	d	a	m
s2	m	a	d	A										

```
 1 # StringCommands04.py
 2 # This program demonstrates how to take
 3 # one string and create a second string
 4 # that is a reverse of the first.
 5
 6
 7 s1 = "Madam I'm Adam"
 8 s2 = ""
 9 n = len(s1)
10
11 # Count from the last index (n-1)
12 # to the first index (0) backwards.
13 for k in range(n-1,-1,-1):
14     s2 += s1[k]
15
16 print()
17 print(s1)
18 print(s2)
```

n	k
14	9

	0	1	2	3	4	5	6	7	8	9	10	11	12	13
s1	M	a	d	a	m		I	'	m		A	d	a	m
s2	m	a	d	A										

```
 1 # StringCommands04.py
 2 # This program demonstrates how to take
 3 # one string and create a second string
 4 # that is a reverse of the first.
 5
 6
 7 s1 = "Madam I'm Adam"
 8 s2 = ""
 9 n = len(s1)
10
11 # Count from the last index (n-1)
12 # to the first index (0) backwards.
13 for k in range(n-1,-1,-1):
14     s2 += s1[k]
15
16 print()
17 print(s1)
18 print(s2)
```

n	k
14	8

	0	1	2	3	4	5	6	7	8	9	10	11	12	13
s1	M	a	d	a	m		I	'	m		A	d	a	m
s2	m	a	d	A		m								

```
 1 # StringCommands04.py
 2 # This program demonstrates how to take
 3 # one string and create a second string
 4 # that is a reverse of the first.
 5
 6
 7 s1 = "Madam I'm Adam"
 8 s2 = ""
 9 n = len(s1)
10
11 # Count from the last index (n-1)
12 # to the first index (0) backwards.
13 for k in range(n-1,-1,-1):
14     s2 += s1[k]
15
16 print()
17 print(s1)
18 print(s2)
```

n	k
14	7

	0	1	2	3	4	5	6	7	8	9	10	11	12	13
s1	M	a	d	a	m		I	'	m		A	d	a	m
s2	m	a	d	A		m	'							

```
 1 # StringCommands04.py
 2 # This program demonstrates how to take
 3 # one string and create a second string
 4 # that is a reverse of the first.
 5
 6
 7 s1 = "Madam I'm Adam"
 8 s2 = ""
 9 n = len(s1)
10
11 # Count from the last index (n-1)
12 # to the first index (0) backwards.
13 for k in range(n-1,-1,-1):
14     s2 += s1[k]
15
16 print()
17 print(s1)
18 print(s2)
```

n	k
14	6

	0	1	2	3	4	5	6	7	8	9	10	11	12	13
s1	M	a	d	a	m		I	'	m		A	d	a	m
s2	m	a	d	A		m	'	I						

```
 1 # StringCommands04.py
 2 # This program demonstrates how to take
 3 # one string and create a second string
 4 # that is a reverse of the first.
 5
 6
 7 s1 = "Madam I'm Adam"
 8 s2 = ""
 9 n = len(s1)
10
11 # Count from the last index (n-1)
12 # to the first index (0) backwards.
13 for k in range(n-1,-1,-1):
14     s2 += s1[k]
15
16 print()
17 print(s1)
18 print(s2)
```

n	k
14	5

	0	1	2	3	4	5	6	7	8	9	10	11	12	13
s1	M	a	d	a	m		I	'	m		A	d	a	m
s2	m	a	d	A		m	'	I						

```
 1 # StringCommands04.py
 2 # This program demonstrates how to take
 3 # one string and create a second string
 4 # that is a reverse of the first.
 5
 6
 7 s1 = "Madam I'm Adam"
 8 s2 = ""
 9 n = len(s1)
10
11 # Count from the last index (n-1)
12 # to the first index (0) backwards.
13 for k in range(n-1,-1,-1):
14     s2 += s1[k]
15
16 print()
17 print(s1)
18 print(s2)
```

n	k
14	4

	0	1	2	3	4	5	6	7	8	9	10	11	12	13
s1	M	a	d	a	m		I	'	m		A	d	a	m
s2	m	a	d	A		m	'	I		m				

```
 1 # StringCommands04.py
 2 # This program demonstrates how to take
 3 # one string and create a second string
 4 # that is a reverse of the first.
 5
 6
 7 s1 = "Madam I'm Adam"
 8 s2 = ""
 9 n = len(s1)
10
11 # Count from the last index (n-1)
12 # to the first index (0) backwards.
13 for k in range(n-1,-1,-1):
14     s2 += s1[k]
15
16 print()
17 print(s1)
18 print(s2)
```

n	k
14	3

	0	1	2	3	4	5	6	7	8	9	10	11	12	13
s1	M	a	d	a	m		I	'	m		A	d	a	m
s2	m	a	d	A		m	'	I		m	a			

```
 1 # StringCommands04.py
 2 # This program demonstrates how to take
 3 # one string and create a second string
 4 # that is a reverse of the first.
 5
 6
 7 s1 = "Madam I'm Adam"
 8 s2 = ""
 9 n = len(s1)
10
11 # Count from the last index (n-1)
12 # to the first index (0) backwards.
13 for k in range(n-1,-1,-1):
14     s2 += s1[k]
15
16 print()
17 print(s1)
18 print(s2)
```

n	k
14	2

	0	1	2	3	4	5	6	7	8	9	10	11	12	13
s1	M	a	d	a	m		I	'	m		A	d	a	m
s2	m	a	d	A		m	'	I		m	a	d		

```
 1 # StringCommands04.py
 2 # This program demonstrates how to take
 3 # one string and create a second string
 4 # that is a reverse of the first.
 5
 6
 7 s1 = "Madam I'm Adam"
 8 s2 = ""
 9 n = len(s1)
10
11 # Count from the last index (n-1)
12 # to the first index (0) backwards.
13 for k in range(n-1,-1,-1):
14     s2 += s1[k]
15
16 print()
17 print(s1)
18 print(s2)
```

n	k
14	1

	0	1	2	3	4	5	6	7	8	9	10	11	12	13
s1	M	a	d	a	m		I	'	m		A	d	a	m
s2	m	a	d	A		m	'	I		m	a	d	a	

```
 1 # StringCommands04.py
 2 # This program demonstrates how to take
 3 # one string and create a second string
 4 # that is a reverse of the first.
 5
 6
 7 s1 = "Madam I'm Adam"
 8 s2 = ""
 9 n = len(s1)
10
11 # Count from the last index (n-1)
12 # to the first index (0) backwards.
13 for k in range(n-1,-1,-1):
14     s2 += s1[k]
15
16 print()
17 print(s1)
18 print(s2)
```

n	k
14	0

	0	1	2	3	4	5	6	7	8	9	10	11	12	13
s1	M	a	d	a	m		I	'	m		A	d	a	m
s2	m	a	d	A		m	'	I		m	a	d	a	M

```
----jGRASP exec: python StringCommands04.py

Madam I'm Adam
madA m'I madaM

----jGRASP: operation complete.
```

```
 7 s1 = "Madam I'm Adam"
 8 s2 = ""
 9 n = len(s1)
10
11 # Count from the last index (n-1)
12 # to the first index (0) backwards.
13 for k in range(n-1,-1,-1):
14     s2 += s1[k]
15
16 print()
17 print(s1)
18 print(s2)
```

n
14

	0	1	2	3	4	5	6	7	8	9	10	11	12	13
s1	M	a	d	a	m		I	'	m		A	d	a	m
s2	m	a	d	A		m	'	I		m	a	d	a	M

```
----jGRASP exec: python StringCommands04.py

Madam I'm Adam
madA m'I madaM

----jGRASP: operation complete.
```

```
 7 s1 = "Madam I'm Adam"
 8 s2 = ""
 9 n = len(s1)
10
11 # Count from the last index (n-1)
12 # to the first index (0) backwards.
13 for k in range(n-1,-1,-1):
14     s2 += s1[k]
15
16 print()
17 print(s1)
18 print(s2)
```

NOTE: You may want to refer to this program when you do Lab 14A or 14B.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13
s1	M	a	d	a	m		I	'	m		A	d	a	m
s2	m	a	d	A		m	'	I		m	a	d	a	M

```
1  # StringCommands05.py
2  # This program demonstrates that
3  # <reverse> does not work with strings.
4  # NOTE: Many array commands work with
5  # strings, but not all of them.
6
7
8  s1 = "Madam I'm Adam"
9  s2 = s1.reverse()
10
11 print()
12 print(s1)
13 print(s2)
14
```

```
  ----jGRASP exec: python StringCommands05.py
Traceback (most recent call last):
  File "StringCommands05.py", line 9, in <module>
    s2 = s1.reverse()
AttributeError: 'str' object has no attribute 'reverse'

  ----jGRASP wedge2: exit code for process is 1.
  ----jGRASP: operation complete.
```

```
 7
 8 s1 = "Madam I'm Adam"
 9 s2 = s1.reverse()
10
11 print()
12 print(s1)
13 print(s2)
14
```