Exposure CS 2021 for CS1

Chapter 14 Section 4-5 Slides

String Processing: More Commands & Common Errors

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

Section 14.4

More String Commands

```python
# StringCommands06.py
# This program demonstrates the <find> and "Reverse Find" <rfind> commands.
# <find> returns the index of the first occurrence of the substring parameter.
# <rfind> returns the index of the last occurrence of the substring parameter.
# Both return -1 if the substring parameter is not found in the original string.


s1 = "racecar"
s2 = "racecar in the car port"
s3 = "car"

index1 = s1.find(s3)
index2 = s1.rfind(s3)
index3 = s2.find(s3)
index4 = s2.rfind(s3)
index5 = s3.find("qwerty")
index6 = s3.rfind("qwerty")

print()
print("The first occurrence of",s3,"in",s1,"is at index",index1)
print("and the last occurrence is at index",index2)

print()
print("The first occurrence of",s3,"in",s2,"is at index",index3)
print("and the last occurrence is at index",index4)

print()
print("The first occurrence of qwerty in",s3,"is at index",index5)
print("and the last occurrence is at index",index6)
```

```python
# StringCommands06.py
#
#
#
#
```

s2	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
	r	a	c	e	c	a	r		i	n		t	h	e		c	a	r	p	o	r	t

s1	0	1	2	3	4	5	6
	r	a	c	e	c	a	r

```python
s1 = "racecar"
s2 = "racecar in the car port"
s3 = "car"

index1 = s1.find(s3)
index2 = s1.rfind(s3)
index3 = s2.find(s3)
index4 = s2.rfind(s3)
index5 = s3.find("qwerty")
index6 = s3.rfind("qwerty")
```

```
----jGRASP exec: python StringCommands06.py

The first occurrence of car in racecar is at index 4
and the last occurrence is at index 4

The first occurrence of car in racecar in the car port is at index 4
and the last occurrence is at index 15

The first occurrence of qwerty in car is at index -1
and the last occurrence is at index -1

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

String Functions
find & rfind

s1	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
	h	u	m	u	h	u	m	u	n	u	k	u	n	u	k	u	a	p	u	a	'	a

find returns the <u>first</u> occurrence of a substring.

`s1.find("hum")` returns **0**

`s1.find("uku")` returns **9**

`s1.find("qwerty")` returns **-1**

By the way,
it is the State
Fish of Hawaii.

rfind returns the <u>last</u> occurrence of a substring.

`s1.rfind("hum")` returns **4**

`s1.rfind("uku")` returns **13**

`s1.rfind("qwerty")` returns **-1**

```python
# StringCommands07.py
# This program demonstrates the <count>,
# <startswith> and <endswith> commands.


s = "HOW MANY BANANAS ARE ON ANA'S BANANA BOAT"
print()
print(s.count("BANANA"))
print(s.count("AN"))
print(s.count("AN",9,29))
print(s.startswith("HOW"))
print(s.startswith("BANANA"))
print(s.endswith("BOAT"))
print(s.endswith("boat"))

```

```
----jGRASP

2
6
3
True
False
True
False

----jGRASP
```

```
1 # StringCommands08.py
2 # This program demonstrates the <replace> command
3 # which can replace characters or substrings.
4 # NOTE: The <replace> command returns an altered
5 #       COPY of the original string. It does not
6 #       alter the original string in ANY way.
7
8
9 s1 = "racecar"
10 s2 = s1.replace('r','l')
11 s3 = s1.replace("ace","eally awesome ")
12
13 print()
14 print(s1)
15 print(s2)
16 print(s1)
17 print(s3)
```

```
1 # StringCommands08.p
2 # This program demon
3 # which can replace
4 # NOTE: The <replace
5 #       COPY of the
6 #       alter the or
7
8
9 s1 = "racecar"
10 s2 = s1.replace('r','l')
11 s3 = s1.replace("ace","eally awesome ")
12
13 print()
14 print(s1)
15 print(s2)
16 print(s1)
17 print(s3)
```

```
----jGRASP exec: pyth

racecar
lacecal
racecar
really awesome car

----jGRASP: operation
```

NOTE: The replace function does NOT alter s1 AT ALL. It creates and returns an altered copy of the string.

```python
# StringCommands09.py
# This program demonstrates what you need to do to replace
# characters/substrings in the original string.


s1 = "racecar"
s1 = s1.replace('r','l')

s2 = "racecar"
s2 = s2.replace("ace","eally awesome ")

print()
print(s1)
print(s2)
```

```
----jGRASP exec: pyth

lacecal
really awesome car
```

```python
# StringCommands10.py
# This program demonstrates the <upper> and <lower> commands
# which convert string to all UPPERCASE or all lowercase.
# NOTE: Like the <replace> command, <upper> and <lower>
#       do not alter the original string in ANY way.


s1 = "racecar"
s2 = "RaCeCaR"
s3 = ">RACECAR-100<"

print("\nUppercased Strings:")
print(s1.upper())
print(s2.upper())
print(s3.upper())

print("\nLowercased Strings:")
print(s1.lower())
print(s2.lower())
print(s3.lower())

print("\nOriginal Strings:")
print(s1)
print(s2)
print(s3)
```

```
----jGRASP exec: pyth

Uppercased Strings:
RACECAR
RACECAR
>RACECAR-100<

Lowercased Strings:
racecar
racecar
>racecar-100<

Original Strings:
racecar
RaCeCaR
>RACECAR-100<

----jGRASP: operation
```

```python
# StringCommands11.py
# This program demonstrates how to UPPERCASE
# or lowercase the original string.


s = ">RaCeCaR<"

print()
print(s)

s = s.upper()
print(s)

s = s.lower()
print(s)

```

```
----jGRASP

>RaCeCaR<
>RACECAR<
>racecar<

----jGRASP:
```

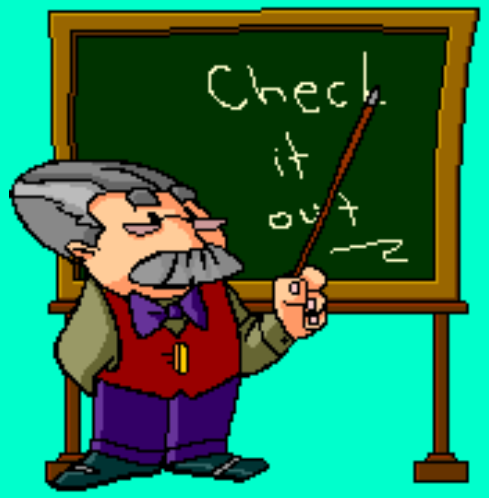
Altering the Original String

Remember, string functions do not alter the original string. They return an altered *copy* of the string.

To alter the original string, you need a statement that assigns the new copy back to the original string.

Examples:

```
s1 = s1.upper()
s2 = s2.lower()
s3 = s3.replace('m','b')
```



```
1 # StringCommands12.py
2 # This program fixes the "case" issue of StringOperators07.py
3 # by using the <upper> command.  With both strings converted
4 # to UPPERCASE, the strings can be compared properly.
5 # NOTE: Using the <lower> command would also work.
6
7 print()
8 s1 = input("Enter 1st string.  -->  ")
9 s2 = input("Enter 2nd string.  -->  ")
10 print()
11
12 if s1.upper() < s2.upper():
13    print(s1,"goes alphabetically before",s2)
14 elif s1.upper() > s2.upper():
15    print(s1,"goes alphabetically after",s2)
16 else:
17    print("Both strings are equal")
```

```
----jGRASP exec: python StringCommands12.py

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

apple goes alphabetically before ZEBRA

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

```python

if s1.upper() < s2.upper():
   print(s1,"goes alphabetically before",s2)
elif s1.upper() > s2.upper():
   print(s1,"goes alphabetically after",s2)
else:
   print("Both strings are equal")
```

```python
# StringCommands13.py
# This program demonstrates how to use <str> to convert
# integers, real numbers or Boolean values to strings.


s1 = str(1000)
s2 = str(234.432)
s3 = str(True)
s4 = s1 + s2

print()
print("s1:",s1)
print("s2:",s2)
print("s3:",s3)
print("s4:",s4)

```

```
----jGRASP exec:

s1: 1000
s2: 234.432
s3: True
s4: 1000234.432

----jGRASP: oper
```

```python
# StringCommands14.py
# This program demonstrates how to use <int>, <float> and
# <bool> to convert string values to integers, real numbers
# (a.k.a. floating point numbers) and Boolean values.


n1 = int("1000")
n2 = float("234.432")
n3 = n1 + n2
b1 = bool("True")

print()
print("n1:",n1)
print("n2:",n2)
print("n3:",n3)

if b1:
    print("b1:",b1)
```

```
----jGRASP

n1: 1000
n2: 234.432
n3: 1234.432
b1: True

----jGRASP:
```

```python
# StringCommands15.py
# This program demonstrates how to use <ord> & <chr>.
# <ord> gives you the ASCII value of a character.
# <chr> gives you the character of an ASCII value.


letter = 'A'
number = 66

print()
print(ord(letter))
print(chr(number))

```

```
----jGRASP

65
B

----jGRASP
```

int & str vs. ord & chr

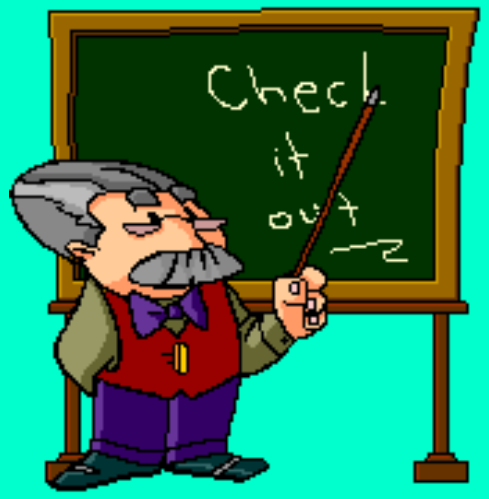
`int('1') = 1`	`ord('1') = 49`
`str(65) = "65"`	`chr(65) = 'A'`
`int('A') = Error`	`ord('A') = 65`
`str(300) = "300"`	`chr(300) = Error`
`int("65") = 65`	`ord("65") = Error`

Coding your own **upper** function

Consider this:

The ASCII value of **'A'** is **65** and the ASCII value of **'a'** is **97**.	**97 – 65 = 32**
The ASCII value of **'B'** is **66** and the ASCII value of **'b'** is **98**.	**98 – 66 = 32**
: : : : : : :	: : :
The ASCII value of **'Z'** is **90** and the ASCII value of **'z'** is **122**.	**122 – 90 = 32**

The difference between the ASCII values of a CAPITAL letter and its corresponding lowercase letter is always **32**.



```python
# StringCommands16.py
# This program attempts to use <ord> and <chr> to convert
# a string to all CAPITAL letters, like the <upper> command.
# The problem is it performs the conversion on all of the
# characters, not just the lowercase letters.


def allCAPS(strOld):
    strNew = ""
    for k in range(len(strOld)):
        ascii = ord(strOld[k])
        strNew += chr(ascii - 32)
    return strNew


##########
## MAIN ##
##########

title = "exposure computer science for CS1 & CS1-Honors"
newTitle = allCAPS(title)
print()
print(title)
print(newTitle)
```

```
----jGRASP exec: python StringCommands16.py

exposure computer science for CS1 & CS1-Honors
EXPOSURE COMPUTER SCIENCE FOR #3 0011   0006  #3 0011
(ONORS
```

```python
# StringCommands17.py
# This program fixes the issue of the previous program
# by first checking if a character is a lowercase letter
# before capitalizing it.


def allCAPS(strOld):
    strNew = ""
    for k in range(len(strOld)):
        if strOld[k] >= 'a' and strOld[k] <= 'z':
            ascii = ord(strOld[k])
            strNew += chr(ascii - 32)
        else:
            strNew += strOld[k]
    return strNew


###########
## MAIN ##
###########

title = "exposure computer science for CS1 & CS1-Honors"
newTitle = allCAPS(title)
print()
print(title)
print(newTitle)
```

```
----jGRASP exec: python StringCommands17.py

exposure computer science for CS1 & CS1-Honors
EXPOSURE COMPUTER SCIENCE FOR CS1 & CS1-HONORS
```

The "is" Commands

```python
# StringCommands18.py
# This program demonstrates <capitalize>, <title>,
# <islower>, <isupper> and <istitle>.
# Note that <capitalize> is not the same thing as <upper>.


s = "exposure computer science 2020 in python for CS1 & CS1-honors"
print("\n" + s)
print(s.islower(), s.isupper() ,s.istitle())

s = s.capitalize()
print("\n" + s)
print(s.islower(), s.isupper() ,s.istitle())

s = s.upper()
print("\n" + s)
print(s.islower(), s.isupper() ,s.istitle())

s = s.lower()
print("\n" + s)
print(s.islower(), s.isupper() ,s.istitle())

s = s.title()
print("\n" + s)
print(s.islower(), s.isupper() ,s.istitle())
```

islower isupper istitle

```
----jGRASP exec: python StringCommands18.py

exposure computer science 2020 in python for CS1 & CS1-honors
False False False

Exposure computer science 2020 in python for cs1 & cs1-honors
False False False

EXPOSURE COMPUTER SCIENCE 2020 IN PYTHON FOR CS1 & CS1-HONORS
False True False

exposure computer science 2020 in python for cs1 & cs1-honors
True False False

Exposure Computer Science 2020 In Python For Cs1 & Cs1-Honors
False False True

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

```python
# StringCommands19.py
# This program demonstrates <isdecimal>, <isalpha>, <isalnum>
# and <isidentifier>.


s = "Exposure Computer Science 2020 in Python for CS1 & CS1-Honors"
print("\n" + s)
print(s.isdecimal(), s.isalpha(), s.isalnum(), s.isidentifier())

s = "ExposureComputerScience"
print("\n" + s)
print(s.isdecimal(), s.isalpha(), s.isalnum(), s.isidentifier())

s = "1996"
print("\n" + s)
print(s.isdecimal(), s.isalpha(), s.isalnum(), s.isidentifier())

s = "3.14159"
print("\n" + s)
print(s.isdecimal(), s.isalpha(), s.isalnum(), s.isidentifier())

s = "11A2B"
print("\n" + s)
print(s.isdecimal(), s.isalpha(), s.isalnum(), s.isidentifier())

s = "A_108"
print("\n" + s)
print(s.isdecimal(), s.isalpha(), s.isalnum(), s.isidentifier())
```

isdecimal isalpha isalnum isidentifier

```
----jGRASP exec: python StringCommands19.py

Exposure Computer Science 2020 in Python for CS1 & CS1-Honors
False False False False

ExposureComputerScience
False True True True

1996
True False True False

3.14159
False False False False

11A2B
False False True False

A_108
False False False True

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

```python
# StringCommands20.py
# This program demonstrates how <isdigit> and
# <isnumeric> are different from <isdecimal>.
# NOTE: This program will not execute in jGRASP.
# You will need to use the Python Shell.

s = "10"
print(s.isdecimal(), s.isdigit(), s.isnumeric())

s = "10¹²³"
print(s.isdecimal(), s.isdigit(), s.isnumeric())

s = "10¹²³¼½¾"
print(s.isdecimal(), s.isdigit(), s.isnumeric())

s = "3.14159"
print(s.isdecimal(), s.isdigit(), s.isnumeric())

```

Python 3.6.4 Shell

File Edit Shell Debug Options Window Help

```
Python 3.6.4 (v3.6.4:d48eceb, Dec 19 2017, 06:54:40) [MSC v.1900
64 bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
>>> s = "10"
>>> print(s.isdecimal(), s.isdigit(), s.isnumeric())
True True True
>>>
>>> s = "10¹²³"
>>> print(s.isdecimal(), s.isdigit(), s.isnumeric())
False True True
>>>
>>> s = "10¹²³¼½¾"
>>> print(s.isdecimal(), s.isdigit(), s.isnumeric())
False False True
>>>
>>> s = "3.14159"
>>> print(s.isdecimal(), s.isdigit(), s.isnumeric())
False False False
>>>
```

Ln: 20 Col: 4

```python
# StringCommands21.py
# This program demonstrates <isspace> and <isprintable>.


s = "Exposure Computer Science 2020 in Python for CS1 & CS1-Honors"
print("\n" + s)
print(s.isspace(), s.isprintable())

s = " "
print("\nSpace")
print(s.isspace(), s.isprintable())

s = ""
print("\nEmpty String")
print(s.isspace(), s.isprintable())

s = "\t"
print("\nTab Escape Sequence")
print(s.isspace(), s.isprintable())

s = "\n"
print("\nNew Line Escape Sequences")
print(s.isspace(), s.isprintable())
```

isspace isprintable

```
----jGRASP exec: python StringCommands21.py

Exposure Computer Science 2020 in Python for CS1 & CS1-Honors
False True

Space
True True

Empty String
False True

Tab Escape Sequence
True False

New Line Escape Sequences
True False

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

" "

""

"\t"

"\n"

Breaking Up Sentences into Words

```python
# StringCommands22.py
# This program demonstrates how to use the <split> command to
# "split" a something like a sentence into an array of words.


title = "Exposure Computer Science 2020 for CS1 & CS1-Honors"
words = title.split()
print()
print(words)

title = "Exposure, Computer, Science, 2020, for, CS1, &, CS1-Honors"
words = title.split()
print()
print(words)

title = "Exposure, Computer, Science, 2020, for, CS1, &, CS1-Honors"
words = title.split(',')
print()
print(words)

title = "Exposure, Computer, Science, 2020, for, CS1, &, CS1-Honors"
words = title.split(', ')
print()
print(words)

title = "Exposure Computer Science 2020 for CS1 & CS1-Honors"
words = title.split('&')
print()
print(words)

title = "Exposure Computer Science 2020 for CS1 & CS1-Honors"
words = title.split('%')
print()
print(words)
```

```python
# StringCommands22.py
# This program demonstrates how to use the <split> command to
# "split" a something like a sentence into an array of words.


title = "Exposure Computer Science 2020 for CS1 & CS1-Honors"
words = title.split()
print()
print(words)

title = "Exposure, Computer, Science, 2020, for, CS1, &, CS1-Honors"
words = title.split()
print()
print(words)
```

```
----jGRASP exec: python StringCommands22.py

['Exposure', 'Computer', 'Science', '2020', 'for', 'CS1', '&', 'CS1-Honors']

['Exposure,', 'Computer,', 'Science,', '2020,', 'for,', 'CS1,', '&,', 'CS1-Honors']

['Exposure', ' Computer', ' Science', ' 2020', ' for', ' CS1', ' &', ' CS1-Honors']

['Exposure', 'Computer', 'Science', '2020', 'for', 'CS1', '&', 'CS1-Honors']

['Exposure Computer Science 2020 for CS1 ', ' CS1-Honors']

['Exposure Computer Science 2020 for CS1 & CS1-Honors']

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

```python

title = "Exposure, Computer, Science, 2020, for, CS1, &, CS1-Honors"
words = title.split(',')
print()
print(words)

title = "Exposure, Computer, Science, 2020, for, CS1, &, CS1-Honors"
words = title.split(', ')
print()
print(words)

title = "Exposure Computer Science 2020 for CS1 & CS1-Honors"
words = title.split('&')
print()
print(words)

title = "Exposure Computer Science 2020 for CS1 & CS1-Honors"
words = title.split('%')
print()
print(words)
```

```
['Exposure', ' Computer', ' Science', ' 2020', ' for', ' CS1', ' &', ' CS1-Honors']

['Exposure', 'Computer', 'Science', '2020', 'for', 'CS1', '&', 'CS1-Honors']

['Exposure Computer Science 2020 for CS1 ', ' CS1-Honors']

['Exposure Computer Science 2020 for CS1 & CS1-Honors']

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

```python
# StringCommands23.py
# This program demonstrates how to use the <join> which is
# essentially the opposite of <split>.


words = ["The","quick","brown","fox","jumps","over","the","lazy","dogs."]

space = " "
sentence = space.join(words)
print()
print(sentence)

sentence = ",".join(words)
print()
print(sentence)

sentence = "<~>".join(words)
print()
print(sentence)

sentence = "".join(words)
print()
print(sentence)
```

```
The quick brown fox jumps over the lazy dogs.

The,quick,brown,fox,jumps,over,the,lazy,dogs.

The<~>quick<~>brown<~>fox<~>jumps<~>over<~>the<~>lazy<~>dogs.

Thequickbrownfoxjumpsoverthelazydogs.
```

Section 14.5

Common Errors with Strings

```python
# StringErrors01.py
# This program is almost identical to StringOperators03.py
# The difference is now the index value is too large.
# The causes a "string index out of range" run-time error
# which is very similar to the "index out of range" run-time
# error that you get when you make the same mistake with arrays.


state = "TEXAS"
print()
print(state[5])
```

0	1	2	3	4
T	E	X	A	S

```
 ----jGRASP exec: python StringsErrors01.py

Traceback (most recent call last):
  File "StringsErrors01.py", line 11, in <module>
    print(state[5])
IndexError: string index out of range

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

```python
# StringErrors02.py
# This program tries to use the <int> command
# to convert the string value in <n1> to an
# integer which is not possible.


n1 = "3.14159"
n2 = int(n1)

print()
print(n1)
print(n2)
```

```
 ----jGRASP exec: python StringsErrors02.py
Traceback (most recent call last):
  File "StringsErrors02.py", line 8, in <module>
    n2 = int(n1)
ValueError: invalid literal for int() with base 10: '3.14159'

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

```python
# StringErrors03.py
# This program is similar to the previous program.
# Now it tries to use the <float> command to convert
# the string value in <n1> to a  real number which
# is also not possible.


n1 = "Qwerty"
n2 = float(n1)

print()
print(n1)
print(n2)
```

```
 ----jGRASP exec: python StringsErrors03.py
Traceback (most recent call last):
  File "StringsErrors03.py", line 8, in <module>
    n2 = float(n1)
ValueError: could not convert string to float: 'Qwerty'

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

```python
# StringErrors04.py
# This program is supposed to CAPITALIZE all of the letters
# in string <s> and then replace every 'A' with 'U'; however,
# nothing happens. This Logic Error occurs when students
# forget that the string commands do NOT alter the original
# string at all.  Instead, they create an altered copy of
# the string -- which this program does not use.


s = "banana"
print()
print(s)
s.upper()
print(s)
s.replace('A','U')
print(s)

```

```
----jGRASP


banana
banana
banana


----jGRASP:
```

```python
# StringErrors05.py
# This program demonstrates how to fix the logic error
# of the previous program.  It also reviews what is
# necessary to change the original <String> object.


s = "banana"
print()
print(s)
s = s.upper()
print(s)
s = s.replace('A','U')
print(s)

```

```
----jGRASP


banana
BANANA
BUNUNU


----jGRASP:
```

```
1 # StringErrors06.py
2 # This program demonstrates what happens when you create
3 # substrings with index values that are too large.
4 # Surprisingly, there is no error message.
5 # If the second number is too large, the substring will just
6 # go to the end of the string, just as if it were omitted.
7 # If the first number is too big, the substring is simply empty.
8
9
10 s = "Racecar"
11
12 print()
13 print(s[4:])
14 print(s[4:10])
15 print(s[10:])
16
```

s	0	1	2	3	4	5	6
	r	a	c	e	c	a	r

```
----jGRASP

car
car

----jGRASP:
```