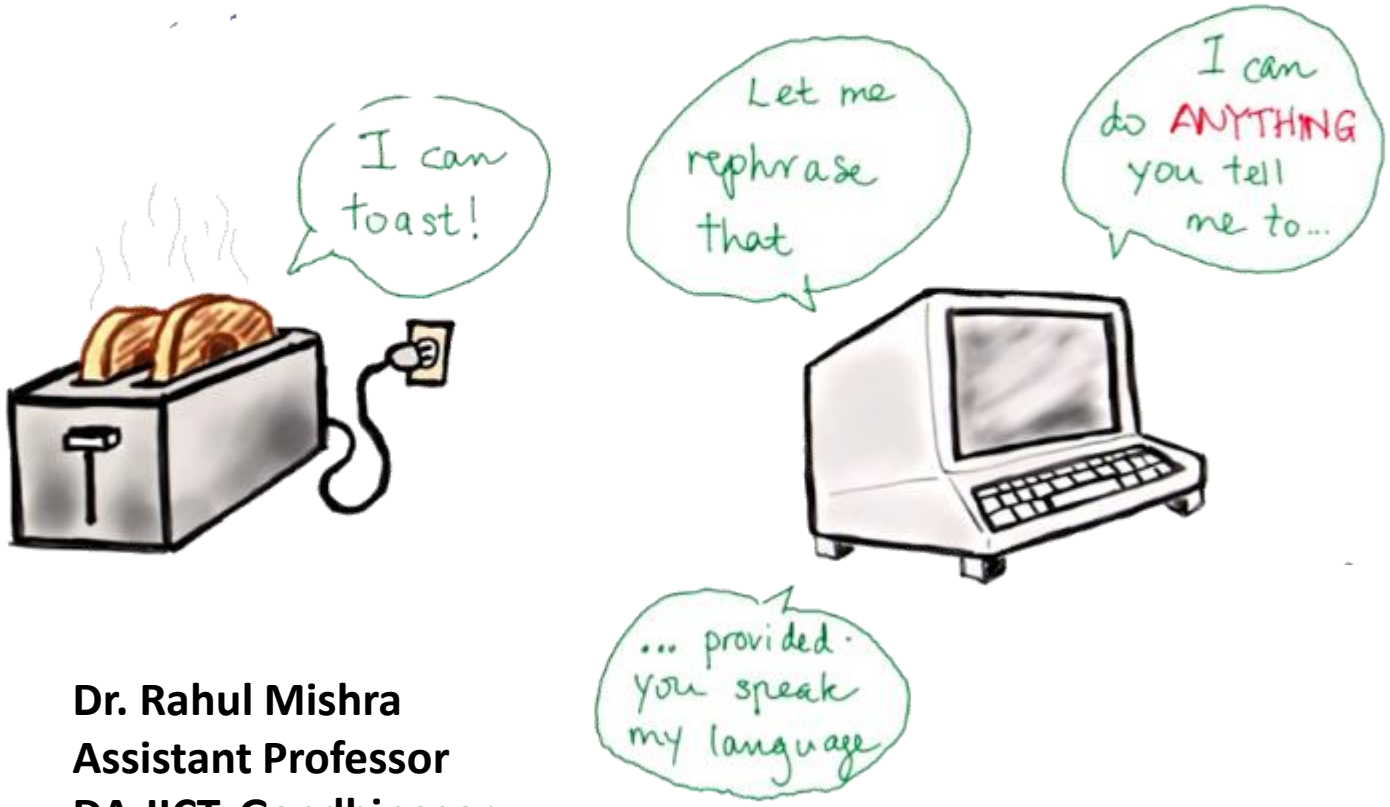
Programming Lab

Autumn Semester

Course code: PC503



Dr. Rahul Mishra
Assistant Professor
DA-IICT, Gandhinagar

Lecture 3

Strings functions

1. If you need a different string, you should create a new one:

2. The built-in function len() returns the length of a string:

```
>>> 'J' + word[1:]
'Jython PC 503'
>>> word[:2] + 'py'
'Pypy'
>>> s = 'supercalifragilisticexpialidocious'
>>> len(s)
34
>>>
```

Strings in detail:

- Textual data in Python is handled with str objects or strings.

- Strings are immutable sequences of Unicode code points.

- String literals are written in a variety of ways:
  - Single quotes: 'allows embedded "double" quotes
  - Double quotes: "allows embedded 'single' quotes"
  - Triple quoted: '''Three single quotes''', """Three double quotes”””

- Triple quoted strings may span multiple lines - all associated whitespace will be included in the string literal.

- There is also no mutable string type, but str. join() or io.StringIO can be used to efficiently construct strings from multiple fragments.

str.join()

The join() method takes all items in an iterable and joins them into one string.

A string must be specified as the separator.

```
>>> myTuple = ("John", "Peter", "Vicky")
>>>
>>> x = "#".join(myTuple)
>>>
>>> print(x)
John#Peter#Vicky
>>> y="__".join(myTuple)
>>> y
'John__Peter__Vicky'
```

io.StringIO

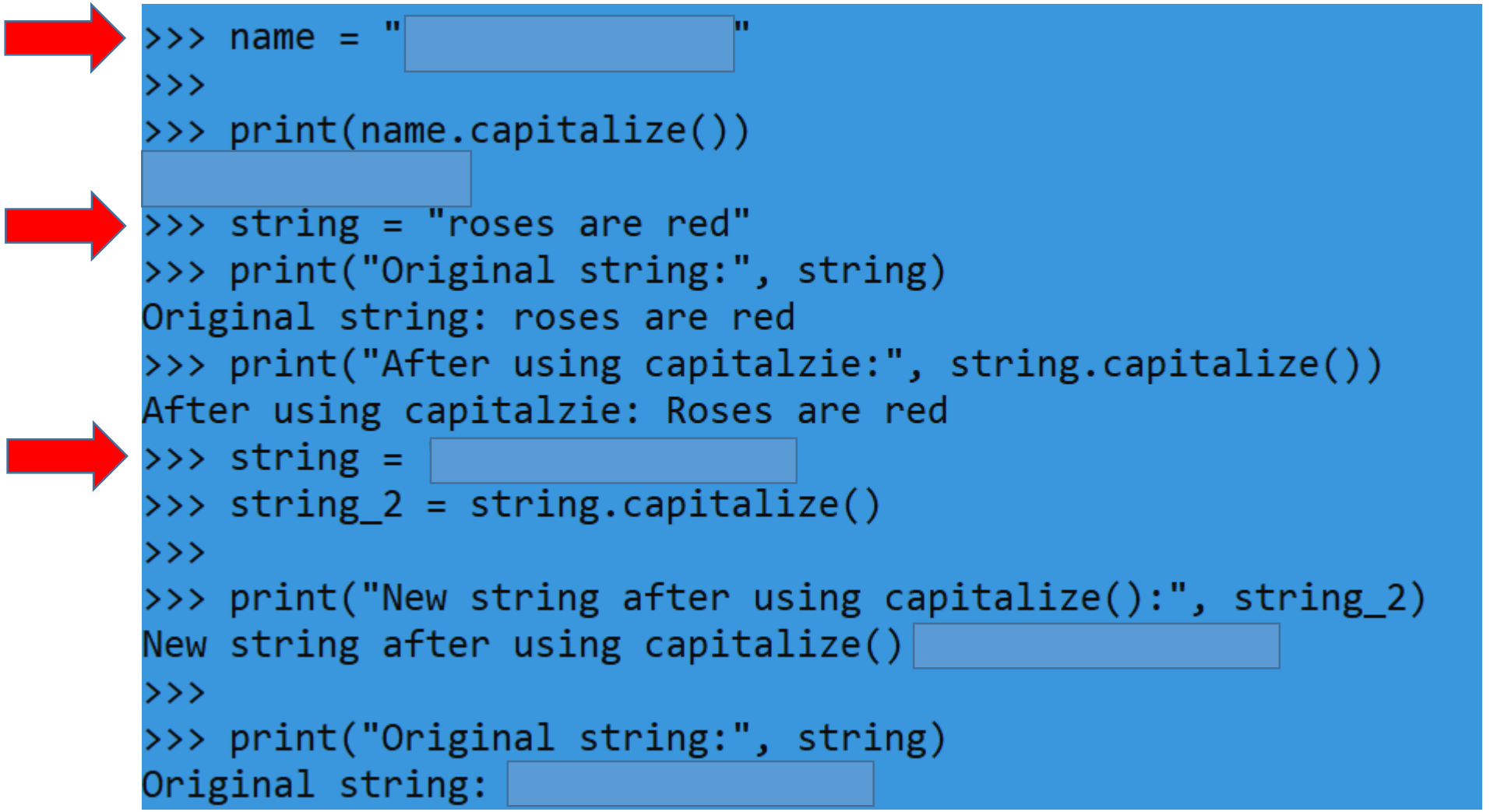
```
>>> import io
>>> output = io.StringIO()
>>> output.write('First line.\n')
12
>>> print('Second line.', file=output)
>>> # Retrieve file contents -- this will be
>>> # 'First line.\nSecond line.\n'
>>> contents = output.getvalue()
>>> output.close()
>>> contents
'First line.\nSecond line.\n'
>>> contents=output.getvalue()
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ValueError: I/O operation on closed file
>>>
```

Write a program in a similar manner that holds your short bio of around 50 words and displays it on the screen.

`str.capitalize()`

- Return a copy of the string with its first character capitalized and the rest lowercase.
- capitalize() only capitalizes the first letter of a string and lowers all the remaining characters.
- Python String capitalize() Method Doesn't Modify the Original String

Strings Methods



`str.casefold()`

- Return a case-folded copy of the string. Case-folded strings may be used for caseless matching.
- The case fold () method returns a string where all the characters are lowercase.
- This method is similar to the lower() method, but the case fold () method is stronger, and more aggressive, meaning that it will convert more characters into lower case and will find more matches when comparing two strings and both are converted using the casefold() method.

```
>>> txt = "Hello, And Welcome To My World!"
>>>
>>> x = txt.casefold()
>>>
>>> print(x)
hello, and welcome to my world!
>>>
```

str.**center**(*width*[, *fillchar*])

- Return centered in a string of length width.
- Padding is done using the specified fillchar (default is an ASCII space). The original string is returned if the width is less than or equal to len(s).
- The following are the parameters for the Python string center() method.
  - **width** − This parameter is an integer value that represents the total length of the string along with the padding characters.
  - **fillchar** − This parameter specifies the filling characters. Only the single-length character is accepted. The default filling character is the ASCII space.

```
>>> str = "Welcome to Programming Course"
>>> output=str.center(40, '.')
>>> print("The string after applying the center() function is:", output)
The string after applying the center() function is: .....Welcome to Programming Course......
>>>
```

str.**center**(*width*[, *fillchar*])

- If an alphabet is taken as fill char, the input string is centered in the given width and the extra characters are padded using the alphabet that is specified in the parameter in the center() function.

```
>>> output=str.center(40, 's')
>>> print("The string after applying the center() function is:", output)
The string after applying the center() function is: ssssWelcome to Programming Coursesssssss
>>>
```

- If the fillchar is not specified in the parameter of the center() function, then the default fillchar which is the ASCII space is considered as the padding value.

```
>>> output=str.center(40)
>>> print("The string after applying the center() function is:", output)
The string after applying the center() function is:      Welcome to Programming Course
>>>
```

str.**center**(*width*[, *fillchar*])

- This function does not modify the original string if the parameter width that is mentioned is less than the length of the original input string.

```
>>> output=str.center(5)
>>> print("The string after applying the center() function is:", output)
The string after applying the center() function is: Welcome to Programming Course
>>>
```

- This function does not accept a string fillchar. It only accepts one character long fillchar. If the fillchar specified does not obey this condition, a type error is occured.

```
>>> output=str.center(40, 'aa')
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: The fill character must be exactly one character long
>>>
```