CSCI 141
Computer Programming I

Filip Jagodzinski
Announcements

In homework 4 you are asked to calculate a tip amount. You are also asked to output the tip amount with two decimal places. Assume your calculation output is the following:

\[ \text{tipAmount} = 43.56798234234 \]

Q: How might you truncate the decimal places?
In homework 4 you are asked to calculate a tip amount. You are also asked to output the tip amount with two decimal places. Assume your calculation output is the following:

\[ \text{tipAmount} = 43.56798234234 \]

Q: How might you truncate the decimal places?

- Use a combination of `/`, `//` and the `%` operator to modify `tipAmount` so that it is 43.56 or 43.57
- Convert the `tipAmount` to a String, but retain only the first 2 decimal digits.
- Use a conversion/format function that you find online or which is in the textbook

Acceptable final “values” for the above `tipAmount` are 43.56 or 43.57
From last time

```python
aDogName = "Fido"
print((aDogName + "growls") * 2)
```

The + operator behaves differently based on the operands. If the two operands are numbers, then addition is performed. If both of the operands are a String, then + is the concatenation operator. The * operator, too, is dependent on context.
From last time

```
ADogName = "Fido"
print((ADogName + "growls") * 2)
```

The + operator behaves differently based on the operands. If the two operands are numbers, then addition is performed. If both of the operands are a String, then + is the concatenation operator. The * operator, too, is dependent on context.

```
aWord = "SuMmERs"
```

<p>| | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>S</td>
<td>u</td>
<td>M</td>
<td>m</td>
<td>E</td>
<td>R</td>
<td>s</td>
</tr>
<tr>
<td>-7</td>
<td>-6</td>
<td>-5</td>
<td>-4</td>
<td>-3</td>
<td>-2</td>
<td>-1</td>
</tr>
</tbody>
</table>
From last time

```python
aDogName = "Fido"
print((aDogName + "growls") * 2)
```

The + operator behaves differently based on the operands. If the two operands are numbers, then addition is performed. If both of the operands are a String, then + is the concatenation operator. The * operator, too, is dependent on context.

```
aWord = "SummERs"
```

To access a string’s characters (single character strings), you use square brackets. For example

```python
print(aWord[3])
```
Strings Methods
Manipulating Strings
String Methods

Recall back to the Turtles. There are functions (methods) in the Turtle class that you invoke to create and modify your turtle. For example:

```python
import turtle
aTurtle = turtle.Turtle()
aTurtle.forward(110)
aTurtle.left(90)
```
String Methods

Recall back to the Turtles. There are functions (methods) in the Turtle class that you invoke to create and modify your turtle. For example:

```python
import turtle
daTurtle = turtle.Turtle()
daTurtle.forward(110)
daTurtle.left(90)
```

These are all functions, and we invoke an existing function from a class by using the ., followed by the name of the function.
When you create a variable and assign it to “contain” a string literal ... **Q: What happens**

```python
aPhrase = "WWU is in Bellingham"
```
When you create a variable and assign it to “contain” a string literal … Q: What happens

Just like the Turtle class had methods (functions) that you could use to control a Turtle, any object of type string has methods that you can use “on the contents of” the string.
String Methods

Although you don’t need to memorize all of these methods, you should become familiar with a few of the most commonly used ones, and be sure that you can look at a list of methods and invoke them correctly.
String Methods

str

WWU is in Bellingham

```
import string
aPhrase = "WWU is in Bellingham"
print(aPhrase.upper())
```

Q: What does this print to the screen?
String Methods

**WWU is in Bellingham**

```python
str = "WWU is in Bellingham"
print(str.upper())
```

```
WWU IS IN BELLINGHAM
```

Q: Does the method `upper` change the original value of the contents of the String object?
Any string functions in the e-book, or on the lecture slides, or which you find online, you can use in completing homework assignments, labs, etc.
Task: Write a function that receives a single parameter, a string, and prints to the screen that same string except the last character is upper case.

Assuming you have the following, what code would you write?

```python
aPhrase = "WWU is in Bellingham"
```

Any ideas? How would you do it?
Task: Write a function that receives a single parameter, a string, and prints to the screen that same string except the last character is upper case.

Assuming you have the following, what code would you write?

```python
aPhrase = "WWU is in Bellingham"
```

As a first step, what is the position of the last character? Position 19
String Methods

**Task**: Write a function that receives a single parameter, a string, and prints to the screen that same string except the last character is uppercase.

Assuming you have the following, what code would you write?

```python
aPhrase = "WWU is in Bellingham"
```

As a first step, what is the position of the last character? Position 19

Now write a for loop that prints all of the characters up to but not including position 19, and the character at position 19 should be printed out in all caps.
String Methods

Task: Write a function that receives a single parameter, a string, and prints to the screen that same string except the last character is upper case.

Assuming you have the following, what code would you write?

```
aphrase = "WWU is in Bellingham"

outputString = ""
for x in range(0,19):
    outputString = outputString + aPhrase[x]
print(outputString + aPhrase[19].upper())
```

This works, but it assumes that you know the length of the string. What we need is a function that tells us the length of a string, so that our code would work for ANY word, not just the phrase that has 20 characters.

The `len` function (short for length), when applied on a string, returns the number of characters in the string.
String Methods

Task: Write a function that receives a single parameter, a string, and prints to the screen that same string except the last character is uppercase.

Assuming you have the following, what code would you write?

```
aPhrase = "WWU is in Bellingham"
outputString = ""
for x in range(0, 19):
    outputString = outputString + aPhrase[x]
print(outputString + aPhrase[19].upper())
```

Task: Rewrite the above code to use `len` in place of hard coding the 19

The `len` function (short for length), when applied on a string, returns the number of characters in the string.
Task: Write a function that receives a single parameter, a string, and prints to the screen that same string except the last character is upper case.

Assuming you have the following, what code would you write?

```python
aPhrase = "WWU is in Bellingham"

outputString = ""
for x in range(0, 19):
    outputString = outputString + aPhrase[x]
print(outputString + aPhrase[19].upper())
```

Task: Rewrite the above code to use `len` in place of hard coding the 19

```python
outputString = ""
for x in range(0, len(aPhrase)-1):
    outputString = outputString + aPhrase[x]
print(outputString + aPhrase[len(aPhrase)-1].upper())
```
Write the function `extractAllNumbers` that

- Has a single parameter (which you can assume is of type string)
- It detects all characters in the input argument that are digits, from left to right
- Returns a String that contains all of the digits in the input string

For example, the below program

```python
aString = "There are 72 months in 6 years"
print(extractAllNumbers(aString))
```

Would print 726
Another handy operation that you can use is the slice operator (in other programming languages, this is called the substring operation).

```python
aPhrase = "WWU is in Bellingham"
```

Task: for the above `aPhrase`, print to the screen “WWU is here” using a for loop and the concatenation operator

Live demo
Another handy operation that you can use is the slice operator (in other programming languages, this is called the substring operation).

```
aPhrase = "WWU is in Bellingham"
```

Task: for the above `aPhrase`, print to the screen “WWU is here” using a for loop and the concatenation operator.

```
outputString = ""
for x in range(0, 3):
    outputString = outputString + aPhrase[x]
print(outputString + " is here")
```

Doable .. But there is a more elegant way to do this ... using slice
String Methods

Another handy operation that you can use is the slice operator (in other programming languages, this is called the substring operation).

```python
aPhrase = "WWU is in Bellingham"

print(aPhrase[0:5])
```

This specifies print the characters at position 0 through 4 of the string `aPhrase`

Q: What is the output?
Another handy operation that you can use is the slice operator (in other programming languages, this is called the substring operation).

```
aPhrase = "WWU is in Bellingham"
```

```
print(aPhrase[0:5])
```

This specifies print the characters at position 0 through 4 of the string `aPhrase`

**Q: What is the output?**

```
WWU i
```

If you do not include a number before the colon, then the slice beings at position 0. If you do not include a number after the colon, then the slice ends at the last character.
What is the output of the following program?

```python
aPhrase = "WWU is in Bellingham"
print (aPhrase[:5] + aPhrase[15:])
```

A. WWU i ingham
B. WWU is ngham
C. WWU I ngham
D. WWU ingham
E. WWU gham
What is the output of the following program?

```python
aPhrase = "WWU is in Bellingham"
print (aPhrase[:5] + aPhrase[15:15])
```

A. WWU i ingham  
B. WWU is ngham  
C. WWU I ngham  
D. WWU ingham  
E. WWU gham  

Task: Be able to explain what each of the pieces of code in the blue boxes accomplishes
What is the output of the following program?

```python
aPhrase = "WWU is in Bellingham"
print (aPhrase[:5] + aPhrase[15:1])
```

- This retrieves all characters from 0 up to (but not including) 5
- A. WWU i ingham
- B. WWU is ngham
- C. WWU I ngham
- D. WWU ingham
- E. WWU gham

This retrieves all characters from position 15 to the end

This is the concatenation operator
Comparing strings

Task: Write a function, `compare2Strings`, that has 2 parameters (that you can assume are of type String), and which outputs True if both of the arguments are Strings with identical characters, else it returns False.

As a first step, write the pseudocode for such a function, and THEN proceed to write the code.

Live demo
Comparing strings

The comparison operator determines if two strings are equal

```
aPhrase = "WWU is in Bellingham"
anotherPhrase = "WWU is in bellingham"
yetAnotherPhrase = "WWU is in bellingham"
```

Luckily, the equality operator can be used to achieve the same feat ...
Comparing strings

The comparison operator determines if two strings are equal

```python
aPhrase = "WWU is in Bellingham"
anotherPhrase = "WWU is in bellingham"
yetAnotherPhrase = "WWU is in bellingham"
```

Q: Which of the following will evaluate to True?

```python
aPhrase == anotherPhrase
aPhrase == yetAnotherPhrase
anotherPhrase == yetAnotherPhrase
```
Comparing strings

The comparison operator determines if two strings are equal

```python
aPhrase = "WWU is in Bellingham"
anotherPhrase = "WWU is in bellingham"
yetAnotherPhrase = "WWU is in bellingham"
```

Q: Which of the following will evaluate to True?

- False
- False
- True

```python
aPhrase == anotherPhrase
aPhrase == yetAnotherPhrase
anotherPhrase == yetAnotherPhrase
```

Remember that strings are case sensitive
Comparing strings

The comparison operator determines if two strings are equal

```python
aPhrase = "WWU is in Bellingham"
anotherPhrase = "WWU is in bellingham"
yetAnotherPhrase = "WWU is in bellingham"
```

The > and < operators can also be used, to determine lexicographical order

```python
if (aPhrase < anotherPhrase):
    print("aPhrase is lexicographically first")
```

Q: What is the output of the above code?
Comparing strings

A caveat for the `<` or `>` operators when lexicographically ordering strings is that ALL upper-case characters are lexicographically BEFORE all lower case characters, which is a bit different than what you are accustomed to when ordering using a dictionary.
Comparing strings

A caveat for the `<` or `>` operators when lexicographically ordering strings is that ALL upper-case characters are lexicographically BEFORE all lower case characters, which is a bit different than what you are accustomed to when ordering using a dictionary.

Q: What is the output of the below python code?

```python
wordA = "hello"
wordB = "Jello"
if (wordA < wordB):
    print("yes")
```

A. Yes  
B. No  
C. Nothing  
D. error
Comparing strings

A caveat for the < or > operators when lexicographically ordering strings is that ALL upper-case characters are lexicographically BEFORE all lower case characters, which is a bit different than what you are accustomed to when ordering using a dictionary.

Q: What is the output of the below python code?

```python
wordA = "hello"
wordB = "Jello"
if (wordA < wordB):
    print("yes")
```

A. Yes  
B. No  
C. Nothing  
D. error

Even though the letter h is before the letter j in the alphabet, all upper case letters are BEFORE all of the lower case letters, hence in this case python lexicographically orders wordB < wordA

Think of > and < as using page numbers for a dictionary. “a” < “c” because “a” is on an earlier page than is “b”
Comparing strings

Q: What is the output of the following program:

```python
print("Z" < "c")
print("aPpLe" < "aPPLe")
print("z" > "w")
```
Task: Write a function, `compareTwoStrings`, that has two parameters. You can assume that when invoked, the `compareTwoStrings` function’s both argument are of type String containing only letters.

The function should determine if among the two arguments there are none, a single, or more than a single upper case letter.

- **If no upper case letters**, the function should return a concatenation of both arguments, except all upper case
- **If a single upper case letter**, the function should return a concatenation of both arguments, except all (original) lower case letters are printed upper case, and the single upper case letter is lower case.
- **If more than a single upper case letter** is detected, the function should output the string “ERROR”

(hint, do the assigned reading – the Strings chapter)
String functions
Lists and Files