Announcements

- Project
  - Remember to submit your final project proposals
  - If you complete your project (pseudocode and coding) BEFORE the due date and you want it graded, **send me an email to confirm that you want your project graded before the due date.**
  - If you complete your project BEFORE the due date and it is graded BEFORE the due date, you cannot then go back and make modifications and ask that your project be scored a second time.

- HW 5 (second to last homework) has been posted
  - 2 programming tasks: DJ name, and password verifier
Announcements

The `isInputValid` function must have a single parameter and return a boolean value (True or False). The function should check if its parameter (a string) has a single blank space, and that the first and last name are at least 4 characters (total) in length (5 including the space).

After all passwords have been received, the program should invoke the `validatePassword` function (which should take a single parameter and return True or False) for each password. Based on the output of the function `validatePassword`, your program should specify if the password is valid or invalid. If the password is invalid, the program should indicate why.

Q: Why am I imposing on you requirements for how your functions behave (what parameters they receive, and what do they output)?
The `isInputValid` function must have a single parameter and return a boolean value (True or False). The function should check if its parameter (a string) has a single blank space, and that the first and last name are at least 4 characters (total) in length (5 including the space).

After all passwords have been received, the program should invoke the `validatePassword` function (which should take a single parameter and return True or False) for each password. Based on the output of the function `validatePassword`, your program should specify if the password is valid or invalid. If the password is invalid, the program should indicate why.

```
"Filip Jag"  isInputValid  True
```

A function should perform as few tasks as possible, which makes it easy to debug ...
Announcements

The `isInputValid` function must have a single parameter and return a **boolean** value (True or False). The function should check if its parameter (a string) has a single blank space, and that the first and last name are at least 4 characters (total) in length (5 including the space).

After all passwords have been received, the program should invoke the `validatePassword` function (which should take a **single** parameter and return **True** or **False**) for each password. Based on the output of the function `validatePassword`, your program should specify if the password is valid or invalid. If the password is invalid, the program should indicate why.

```
while true
    prompt user, receive input
    if isInputValid(userInput) exit while
```
Announcements

The `isInputValid` function must have a single parameter and return a boolean value (True or False). The function should check if its parameter (a string) has a single blank space, and that the first and last name are at least 4 characters (total) in length (5 including the space).

After all passwords have been received, the program should invoke the `validatePassword` function (which should take a single parameter and return True or False) for each password. Based on the output of the function `validatePassword`, your program should specify if the password is valid or invalid. If the password is invalid, the program should indicate why.

```plaintext
pseudo code

while true
  prompt user, receive input
  if isInputValid(userInput) exit while
```
Announcements

The `isInputValid` function must have a single parameter and return a `boolean` value (True or False). The function should check if its parameter (a string) has a single blank space, and that the first and last name are at least 4 characters (total) in length (5 including the space).

After all passwords have been received, the program should invoke the `validatePassword` function (which should take a single parameter and return `True` or `False`) for each password. Based on the output of the function `validatePassword`, your program should specify if the password is valid or invalid. If the password is invalid, the program should indicate why.

```
while true
    prompt user, receive input
    if isInputValid(userInput) exit while
    print(generateDJName(userInput))
```
Could you complete the programming tasks without using functions? Yes. But then the programs are more difficult to debug if something goes wrong.

If you go on in CS, your programs will be hundreds – even thousands of lines of code – and having bite-size functions to tests makes the overall writing and debugging process MUCH easier.
A list is a collection of items, and each item is an element of the list. Just like Strings, each element has an index.

Unlike strings, elements in a list can be of different types.

What do these lists “look” like?

```
[10, 20, 30, 40]
["spam", "bungee", "CS"]
["hello", 2.0, 5, [10, 20]]
```
From Last Time

A list is a collection of items, and each item is an element of the list. Just like Strings, each element has an index.

Unlike strings, elements in a list can be of different types

This is a list made up of 4 elements, all of them are integers

```
[10, 20, 30, 40]
["spam", "bungee", "CS"]
["hello", 2.0, 5, [10, 20]]
```
A list is a collection of items, and each item is an element of the list. Just like Strings, each element has an index.

Unlike strings, elements in a list can be of different types.

This is a list made up of 4 elements, all of them are integers.

The correct way to think of this is the following:

```plaintext
[10, 20, 30, 40]
["spam", "bungee", "CS"]
["hello", 2.0, 5, [10, 20]]
```
From Last Time

A list is a collection of items, and each item is an element of the list. Just like Strings, each element has an index.

Unlike strings, elements in a list can be of different types.

This is a list made up of 3 elements, all of them are strings:

```
["spam", "bungee", "CS"]
["hello", 2.0, 5, [10, 20]]
```

```
<table>
<thead>
<tr>
<th>“spam”</th>
<th>“bungee”</th>
<th>“CS”</th>
</tr>
</thead>
</table>
```
A list is a collection of items, and each item is an element of the list. Just like Strings, each element has an index.

Unlike strings, elements in a list can be of different types.

This is a list made up of 3 elements, all of them are strings.

The correct way to think of this is the following:

```
[10, 20, 30, 40]
["spam", "bungee", "CS"]
["hello", 2.0, 5, [10, 20]]
```
From Last Time

A list is a collection of items, and each item is an element of the list. Just like Strings, each element has an index.

Unlike strings, elements in a list can be of different types

This is a list made up of ...

(on the board explanation)
A list is a collection of items, and each item is an element of the list. Just like Strings, each element has an index.

Unlike strings, elements in a list can be of different types.

This is a list made up of ...

(String, float, int, int, int)

["spam", "bungee", "CS"]

["hello", 2.0, 5, [10, 20]]
Warmup

Q: Locate the errors in the below code

```python
for x in range(0,3):
    print ("The value of x is ",  x)
    print ("The value of x is " + x)

aPhrase = "an Apple"
if "a" in aPhrase :
    print ("a is in " + aPhrase)

for letter in aPhrase :
    print ("a is in ", letter)

for "a" in aPhrase :
    print ("a is in ", letter)
```
Warmup

Q: Locate the errors in the below code

```python
for x in range(0, 3):
    print("The value of x is ", x)
print("The value of x is "+ x)
```

When the `+` operator is used, either both operands must be strings, or both operands must be numbers.

The `+` is EITHER a concatenation operator or the addition operator, which python infers from context

Q: How would you fix the syntax error in the highlighted line of code?
Warmup

Q: Locate the errors in the below code

```python
for x in range(0,3):
    print ("The value of x is ", x)
print ("The value of x is "+x)

aPhrase = "an Apple"
if "a" in aPhrase :
    print ("a is in "+aPhrase)

for letter in aPhrase :
    print ("a is in ", letter)
for "a" in aPhrase :
    print ("a is in ", letter)

print ("The value of x is " + str(x))
```

When the + operator is used, either both operands must be strings, or both operands must be numbers.

The + is EITHER a concatenation operator or the addition operator, which python infers from context.
Q: Locate the errors in the below code

```python
for x in range(0, 3):
    print("The value of x is ", x)
    print("The value of x is " + x)

aPhrase = "an Apple"
if "a" in aPhrase :
    print("a is in " + aPhrase)

for letter in aPhrase :
    print("a is in ", letter)

for "a" in aPhrase :
    print("a is in ", letter)
```

Recall that a for loop iterator variable takes on different values, one-by-one, that are generated by the right argument of the `in` operator.

The for loop iterator variable MUST be a variable (so that it can take on different values).
Warmup

Q? What do the error-free lines of code print to the screen?

```python
for x in range(0, 3):
    print("The value of x is ", x)
    print("The value of x is " + x)

aPhrase = "an Apple"
if "a" in aPhrase:
    print("a is in " + aPhrase)

for letter in aPhrase:
    print("a is in ", letter)

for "a" in aPhrase:
    print("a is in ", letter)
```
When used in combination with a for loop, the in operator will assign the value of the variable (left operand), one-by-one, successive values generated by the expression/code that is the right operand.

```python
for x in range(0, 3):
    print("The value of x is ", x)
    print("The value of x is " + x)

aPhrase = "an Apple"
if "a" in aPhrase:
    print("a is in " + aPhrase)

for letter in aPhrase:
    print("a is in ", letter)

for "a" in aPhrase:
    print("a is in ", letter)
```
Warmup

What does this piece of code print to the screen?

```python
for x in range(0, 3):
    print ("The value of x is ", x)
    print ("The value of x is " + x)

aPhrase = "an Apple"
if "a" in aPhrase:
    print ("a is in " + aPhrase)

for letter in aPhrase:
    print ("a is in ", letter)

for "a" in aPhrase:
    print ("a is in " , letter)
```
Q: What does this program print to the screen?

```python
for x in range(0, 3):
    print ("The value of x is ", x)
    print ("The value of x is "+x)

aPhrase = "an Apple"
if "a" in aPhrase:
    print ("a is in "+aPhrase)

for letter in aPhrase:
    print ("a is in ", letter)
for "a" in aPhrase:
    print ("a is in ", letter)
```

When used in combination with `if`, the `in` operator will return True or False depending if the String that is the left operand is a substring in the string that is the right operand.
Q: What does this program print to the screen?

```python
for x in range(0, 3):
    print("The value of x is ", x)
    print("The value of x is " + x)

aPhrase = "an Apple"
if "a" in aPhrase:
    print("a is in " + aPhrase)

for letter in aPhrase:
    print("a is in ", letter)

for "a" in aPhrase:
    print("a is in ", letter)
```

What does this piece of code print to the screen?
Q: What does this program print to the screen?

```python
for x in range(0, 3):
    print("The value of x is ", x)
    print("The value of x is " + x)

aPhrase = "an Apple"
if "a" in aPhrase:
    print("a is in "+aPhrase)

for letter in aPhrase:
    print("a is in ", letter)

for "a" in aPhrase:
    print("a is in ", letter)
```

The `in` operator is being used in conjunction with a `for` loop, and the right operand of `in` is a String (we’ve seen this before), then the variable `letter` will be assigned successive letters (strings) in `aPhrase`, and for EACH of them, the body of the `for` loop will be executed.

Take home message: The `in` operator’s behavior is determined from its context ... whether it is used in combination with a `for` loop or an `if` statement.
Today

Lists
Files
Lists

Much of the functionality available to you in Strings is also available for your use on lists, including accessing elements, determining membership, concatenation, repetition, and slicing.
Much of the functionality available to you in Strings is also available for your use on lists, including **accessing elements, determining membership, concatenation, repetition, and slicing**

```python
aSent = "We're off to see the wizard"
print(aSent[4])
if "a" in aSent:
    print("woo hoo 'a' is in aSent")
aNewSentence = aSent + "!!!
print(aSent * 3)
print(aSent[0:6] + aSent[17:25] + "'s")
```

**Task:** Be able to explain line-by-line what each line of code accomplished
Lists

Much of the functionality available to you in Strings is also available for your use on lists, including **accessing elements**, **determining membership**, **concatenation**, **repetition**, and **slicing**.

- Prints the string at position 4
- If the string “a” is a substring of `aSent`, then print “woo hoo…”
- Repetition

```python
aSent = "We're off to see the wizard"
print(aSent[4])
if "a" in aSent:
    print("woo hoo 'a' is in aSent")
aNewSentence = aSent + "!!!"
print(aSent * 3)
print(aSent[0:6] + aSent[17:25] + "s")
```

String

"We're off to see the wizard"
Lists

Much of the functionality available to you in Strings is also available for your use on lists, including **accessing elements, determining membership, concatenation, repetition, and slicing**

```python
aSent = "We're off to see the wizard"
print(aSent[4])
if "a" in aSent:
    print("woo hoo 'a' is in aSent")
aNewSentence = aSent + "!!!"
print(aSent * 3)
print(aSent[0:6] + aSent[17:25] + "s")
```

**Q: What does this print to the screen?**
Much of the functionality available to you in Strings is also available for your use on lists, including **accessing elements**, **determining membership**, **concatenation**, **repetition**, and **slicing**

```python
aSent = "We're off to see the wizard"
print(aSent[4])
if "a" in aSent:
    print("woo hoo 'a' is in aSent")
aNewSentence = aSent + "!!!"
print(aSent * 3)
print(aSent[0:6] + aSent[17:25] + "s")
```

Q: What does this print to the screen?  

We're the wizas
Lists

```python
myList = ["hello", 2.0, 5, [10, 20]]
```

Q: How should you “visualize” this `myList` object?
Now you can perform the following:

```python
print(myList[1])
print(myList[3])
if "hi" in myList:
    print("hi is in myList")
print(len(myList))
```
Lists

```python
myList = ["hello", 2.0, 5, [10, 20]]
```

Now you can perform the following:

```python
print(myList[1])
print(myList[3])
if "hi" in myList:
    print("hi is in myList")
print(len(myList))
```

On the board walk through of executing the code on the left

```
2.0
[10, 20]
4
```
Assuming that `myList` has been declared as shown above, what do the two lines of code on the right accomplish?

```python
myList[0] = "Fred"
myList.append("Susan")
```
Lists

myList = ["hello", 2.0, 5, [10, 20]]

The first entry (index 0) has been assigned to point to a “new” string, while another string has been appended to the end of the list.

Recall that strings are immutable, so the string object with the content hello has NOT been modified.
What is the output of the program on the right?

```python
print(len(myList))
print(myList[0:4])
```
What is the output of the program on the right?

```python
print(len(myList))
print(myList[0:4])
```

```
5
['Fred', 2.0, 5, [10, 20]]
```
Lists

You can also delete items from a list, by using the `del` statement:

```
del myList[0]
print(myList)
```

Q: What is the output of the above code?
Lists

You can also delete items from a list, by using the `del` statement

```python
del myList[0]
print(myList)
```

```
[2.0, 5, [10, 20], 'Susan']
```
split

Now that we’ve taken a close(er) look at lists, there is a function, \texttt{split} that is very useful when you need to “break up” a String.
Now that we’ve taken a close(er) look at lists, there is a function, \texttt{split} that is very useful when you need to “break up” a String.

\texttt{firstManInSpace} = “Yuri Alekseyevich Gagarin”

\textbf{Task: for the above String} \texttt{firstManInSpace}, \textit{print out on separate lines} \textit{Yuri, Alekseyevich, and Gagarin}.
Now that we’ve taken a close(er) look at lists, there is a function, **split** that is very useful when you need to “break up” a String.

```
firstManInSpace = “Yuri Alekseyevich Gagarin”
```

Task: for the above String `firstManInSpace`, print out on separate lines *Yuri, Alekseyevich, and Gagarin.*

Does the following accomplish this task?

```
for aWord in firstManInSpace:
    print(aWord)
```
Now that we’ve taken a close(er) look at lists, there is a function, `split` that is very useful when you need to “break up” a String.

`firstManInSpace = “Yuri Alekseyevich Gagarin”`

Task: for the above String `firstManInSpace`, print out on separate lines `Yuri`, `Alekseyevich`, and `Gagarin`.

Does the following accomplish this task?

```
for aWord in firstManInSpace:
    print(aWord)
```

No. This would print out each letter (Y, u, r, etc.) on a separate line.
Now that we’ve taken a close(er) look at lists, there is a function, **split** that is very useful when you need to “break up” a String.

```
firstManInSpace = “Yuri Alekseyevich Gagarin”
```

**Task:** for the above String `firstManInSpace`, print out on separate lines `Yuri, Alekseyevich, and Gagarin`.

You could write non trivial code to make a list of words, and then print each element in the list:

```
singleWords = []
aWord = ""
for anIndex in range(0, len(firstManInSpace)):  
    if (firstManInSpace[anIndex] == " "):  
        singleWords.append(aWord)  
        aWord = ""
    elif anIndex == (len(firstManInSpace) - 1):  
        singleWords.append(aWord)
    else:  
        aWord = aWord + firstManInSpace[anIndex]
for element in singleWords:  
    print(element)
```
Now that we’ve taken a close(er) look at lists, there is a function, `split` that is very useful when you need to “break up” a String.

`firstManInSpace = “Yuri Alekseyevich Gagarin”`

Task: for the above String `firstManInSpace`, print out on separate lines `Yuri`, `Alekseyevich`, and `Gagarin`.

The function `split` can be invoked on a string, and it will generate a list whose elements are the space-separated words in the String.

```python
aList = firstManInSpace.split()
print(aList)
print(aList[2])
```

Q: What does the code on the left print to the screen?
Now that we’ve taken a close(er) look at lists, there is a function, `split` that is very useful when you need to “break up” a String.

```
firstManInSpace = “Yuri Alekseyevich Gagarin”
```

Task: for the above String `firstManInSpace`, print out on separate lines Yuri, Alekseyevich, and Gagarin.

The function `split` can be invoked on a string, and it will generate a list whose elements are the space-separated words in the String.

```
aList = firstManInSpace.split()
print(aList)
print(aList[2])
```
Now that we’ve taken a close(er) look at lists, there is a function, **split** that is very useful when you need to “break up” a String.

```python
firstManInSpace = "Yuri Alekseyevich Gagarin"
```

```python
print(aList)  # ['Yuri', 'Alekseyevich', 'Gagarin']
print(aList[2])  # Gagarin
```

The function *split* can be invoked on a string, and it will generate a list whose elements are the space-separated words in the String.
Now that we’ve taken a close(er) look at lists, there is a function, \texttt{split} that is very useful when you need to “break up” a String.

\texttt{firstManInSpace} = “Yuri Alekseyevich Gagarin”

When the function \texttt{split} is invoked with a single argument that is a String, then the String parameter specifies the character(s) that separate(s) “words” in the String that is being processed.
Now that we’ve taken a close(er) look at lists, there is a function, `split` that is very useful when you need to “break up” a String.

```python
firstManInSpace = "Yuri Alekseyevich Gagarin"
```

When the function `split` is invoked with a single argument that is a String, then the String parameter specifies the character(s) that separate(s) “words” in the String that is being processed.

```python
tooManyLetters = "csci322 wwu comp sci"
aList = tooManyLetters.split("s")
for item in aList:
    print(item)
```

Q: What does the code on the left print to the screen?
Now that we’ve taken a close(er) look at lists, there is a function, `split` that is very useful when you need to “break up” a String.

```
firstManInSpace = "Yuri Alekseyevich Gagarin"
```

When the function `split` is invoked with a single argument that is a String, then the String parameter specifies the character(s) that separate(s) “words” in the String that is being processed.

```
tooManyLetters = "csci322 wwu comp sci"
aList = tooManyLetters.split("s")
for item in aList:
    print(item)
```

The function `split` is being invoked on the String `tooManyLetters`, and the argument is the String `s`. 
Now that we’ve taken a close(er) look at lists, there is a function, `split` that is very useful when you need to “break up” a String.

```
firstManInSpace = “Yuri Alekseyevich Gagarin”
```

When the function split is invoked with a single argument that is a String, then the String parameter specifies the character(s) that separate(s) “words” in the String that is being processed.

The `split` function makes a list made up of elements that are ‘words” in the String `tooManyLetters` separated by the substrring `s`.

```
tooManyLetters = "csci322 wwu comp sci"
aList = tooManyLetters.split("s")
for item in aList:
    print(item)
```

As a first step, identify all of the Strings s, and everything else is a “word”
Now that we’ve taken a close(er) look at lists, there is a function, `split` that is very useful when you need to “break up” a String.

```
firstManInSpace = "Yuri Alekseyevich Gagarin"
```

When the function `split` is invoked with a single argument that is a String, then the String parameter specifies the character(s) that separate(s) “words” in the String that is being processed.

The `split` function returns a List, and iterating over that list will print to the screen:

```
c
ci322 wwu comp
ci
```
Because we want to write programs that either process the contents of a file or write contents to a file, we need functionality for accessing or creating a file.
Because we want to write programs that either process the contents of a file or write contents to a file, we need functionality for accessing or creating a file.

<table>
<thead>
<tr>
<th>kennelPooches.txt</th>
</tr>
</thead>
<tbody>
<tr>
<td>Emma female 3</td>
</tr>
<tr>
<td>Frost male 7</td>
</tr>
<tr>
<td>Sparky female 3</td>
</tr>
<tr>
<td>Kujo male 8</td>
</tr>
</tbody>
</table>

Assume you have a file, *kennelPooches.txt*, that includes the names, gender, and ages of all dogs currently being housed at the kennel.

**Task:** Write a program that accesses the file, and calculates the average age of the dogs currently at the kennel.
Because we want to write programs that either process the contents of a file or write contents to a file, we need functionality for accessing or creating a file.

The Python function `open` allows you to access the contents of a file for writing or reading (or both).

**It takes a single or two arguments.** The first argument is the name of the file, and the second (optional) argument specifies if the file should be read from “r”, or written to “w”. If you omit the second argument, you can read from AND write to a file.

```
fileToProcess = open("kennelPooches.txt", "r")
```

This will create a reference variable `fileToProcess` that points “to” the file `kennelPooches.txt`. 

---

<table>
<thead>
<tr>
<th>kennelPooches.txt</th>
</tr>
</thead>
<tbody>
<tr>
<td>Emma female 3</td>
</tr>
<tr>
<td>Frost male 7</td>
</tr>
<tr>
<td>Sparky female 3</td>
</tr>
<tr>
<td>Kujo male 8</td>
</tr>
</tbody>
</table>
Because we want to write programs that either process the contents of a file or write contents to a file, we need functionality for accessing or creating a file.

```python
kennelPooches.txt
Emma female 3
Frost male 7
Sparky female 3
Kujo male 8
```

`fileToProcess = open("kennelPooches.txt", "r")`

Next: in order to calculate the average, you need to keep track of the number of dogs, and the sum of their ages.

This will create a reference variable `fileToProcess` that points “to” the file `kennelPooches.txt`. 
Because we want to write programs that either process the contents of a file or write contents to a file, we need functionality for accessing or creating a file.

The variables `sumOfAges` and `countOfEntries` will be updated for EACH line in the input file.

Next: How do we inspect each line of a file?
Because we want to write programs that either process the contents of a file or write contents to a file, we need functionality for accessing or creating a file.

```
fileToProcess = open("kennelPooches.txt", "r")
sumOfAges = 0
countOfEntries = 0
for aLine in fileToProcess:
```

The `for` functionality is also used to access a file’s lines. The variable `aLine` will take on successive values (Strings) of the lines in the file.

Next up: What do we want to do with each line in the file?
Because we want to write programs that either process the contents of a file or write contents to a file, we need functionality for accessing or creating a file.

- Use `split` to create a list from each line
- Keep track of the number of lines that have been processed
- The 0 entry is the name, the 1 entry is male or female, and the 3 entry for each list is the age.

Because we want to calculate average age, retrieve the age for each line, and convert it to an integer.

```python
fileToProcess = open("kennelPooches.txt", "r")
sumOfAges = 0
countOfEntries = 0
for aLine in fileToProcess:
    entries = aLine.split()
    countOfEntries = countOfEntries + 1
    sumOfAges = sumOfAges + int(entries[2])
```
Because we want to write programs that either process the contents of a file or write contents to a file, we need functionality for accessing or creating a file.

Files

Lastly, compute the average age and print it to the screen.

```python
kennelPooches.txt

Emma female 3  
Frost male 7   
Sparky female 3  
Kujo male 8

fileToProcess = open("kennelPooches.txt", "r")
sumOfAges = 0
countOfEntries = 0
for aLine in fileToProcess:
    entries = aLine.split() 
    countOfEntries = countOfEntries + 1
    sumOfAges = sumOfAges + int(entries[2])
print("Avg. age "+ str(sumOfAges/countOfEntries))
```
Because we want to write programs that either process the contents of a file or write contents to a file, we need functionality for accessing or creating a file.

| kennelPooches.txt | Emma female 3  
| Frost male 7  
| Sparky female 3  
| Kujo male 8 |

It is a good idea to close a file after you are done with it, because keeping too many active pointers to files consumes computer resources.

```python
fileToProcess = open("kennelPooches.txt", "r")
sumOfAges = 0
countOfEntries = 0
for aLine in fileToProcess:
    entries = aLine.split()
    countOfEntries = countOfEntries + 1
    sumOfAges = sumOfAges + int(entries[2])
print("Avg. age "+ str(sumOfAges/countOfEntries))
fileToProcess.close()
```
Writing to a file

Lastly, in addition to reading from a file, you can write to a file. Use the write method to write contents to a file.

```python
outfile = open("aNewFile.txt", "w")
for x in range(0,5):
    outfile.write(str(x) + '\n')
outfile.close()
```

On the board explanation and walk through.
Up Next

Objects and References