CSCI 141
Computer Programming I

Filip Jagodzinski
Announcements

• HW 5 (second to last homework)

• Project
  • Remember to submit your 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.
Announcements

HW 5 (second to last homework)

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)?
Announcements

HW 5 (second to last homework)

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

HW 5 (second to last homework)

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

HW 5 (second to last homework)

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.

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

Main
Announcements

HW 5 (second to last homework)

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.

```python
print(generateDJName(userInput))
```
Announcements

Could you complete the programming task without using functions? Yes. But then the program is more difficult to debug if something goes wrong. And, 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.

isInputValid

generateDJName
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.

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

What do these lists “look” like?
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
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.

The correct way to think of this is the following:
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
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:
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 ...

(On the board explanation)
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)
```
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).
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)
```

When the + operator is used, their 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 this line of code so that the print statement completes?
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)
```

When the + operator is used, their 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 this line of code so that the print statement completes?

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

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 value of x is 0
The value of x is 1
The value of x is 2

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.
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 generate?
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 on if the string that is the left operand is a substring in the string that is the right operand.
Warmup

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 generate?
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)
```

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...

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...
Today

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**

Task: Be able to recall from memory (your memory) what each of the bolded terms mean
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")
```

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

```
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")
```
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?
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"
presnt(aSent[4])
if "a" in aSent:
    print("woohoo 'a' is in aSent")
aNewSentence = aSent + "!!!"
presnt(aSent * 3)
presnt(aSent[0:6] + aSent[17:25] + "s")
```

Q: What does this print to the screen?  

We're the wizas
Lists

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

Q: How should you “visualize” this myList object?
Lists

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

Now you can perform the following:

```
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
Lists

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 is the difference between these two lines of code?

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

\[
\text{myList} = ["hello", 2.0, 5, [10, 20]]
\]

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.

\[
\text{myList}[0] = "Fred"
\text{myList}.append("Susan")
\]
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?

```
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:

```python
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']
```
Objects and References
Files