day101 Review for finals #2
Due Thursday 5/18/23 (6th period), Friday 5/19/23 (7th period)

As I mentioned last week, we are going to review Python the rest of our classes. Your final exam will be on Python. Today we're going to review list variables, Strings, and the try/except structure.

Remember list variables? Here's an example:

```
myList = []
myList.append("Horse")
myList.append("Cat")
myList.append("Dog")
print(myList)

#prints ['Horse', 'Cat', 'Dog']
```

The variable “myList” is a list. You can access items in a list by number. Remember that the index numbers start at zero.

```
print(myList[0])      #(prints "Horse")
print(myList[2])      #(prints "Dog")

for x in myList:
     print(x)         #prints: Horse, Cat, Dog each on its own line
```

See how I pulled all of the items out of the list, one at a time using a for loop?

You can add items to a list using the .append method like I did above or you can put a bunch of things into a list as you create it like this:

```
myList=["eat","sleep","play"]
```

To sort a list you use the .sort() method like this:

```
myList.sort()
```

If you want to print the list sorted, do the line above, then on a separate line print the list. You cannot say "print(myList.sort())".

(continued on next page)

Strings

You can print string variables using the print command.

You can modify string variables using methods. Here's an example:

```
x = "West County"
print(x.lower())          #prints "west county"
```

It's the .lower() method after the variable name that makes it all lowercase. You can do .upper(), .rjust(20), .ljust(20), .center(20,"$"), .replace(findString,replaceStr), to name a few handy ones. Here they are all used:

```
x = "West County"
print(x.upper())              #prints "WEST COUNTY"
print(x.rjust(20))            #prints "        West County"
print(x.ljust(20,"."))        #prints "West County........"
print(x.center(20,"$"))       #prints "$$$$West County$$$$"
print(x.replace("W","B"))     #prints "Best County"
```

If you want to replace a string with a modified version of itself you have to do this:

```
x = x.upper()
```

Try/Except Structures

Remember these? If not, now is a good time to learn this. A try/except structure allows you to make sure the user enters valid data. If you don't add checks like this, then when you ask the user for a number, if they instead enter text, or leave the entry blank, Python will crash. Here's a simple try/except structure:

```
while True:
    try:
        a=int(input('Please enter an integer: '))
        break
    except ValueError:
        print('Sorry, that is not an integer, please try again.')
```

What happens is you go into a while True loop and then ask the user for an integer. If they enter an integer, Python gets to the break line and leaves the loop. If they do not enter an integer, Python jumps down to the except ValueError block of code, prints the error message, then goes back to the start of the while True loop.

What if you wanted Python to make sure the number is an integer, AND that it is larger than 17? Where would you put the additional check?

Think about it, then see the answer on the next page.

Here is a try except that requests an integer AND makes sure the integer is greater than 17:

```
while True:
    try:
        a=int(input('Please enter an integer over 17: '))
        if a>17:
           break
        print("The number must be over 17, please try again.")
    except ValueError:
        print('Sorry, that is not an integer, please try again.')
```

In the above code, see how I added an if statement to make sure the number was larger than 17? I used a break if that statement is true to leave the loop, and otherwise the loop repeats after giving the error message.

Today's assignment:

1. Create a new Python repl called Review 2.
2. Use a try/except structure to ask the user for a number from 10 to 20 inclusive. Give one error message if they do not enter an integer. Give a different error message if the number is too small. Give a different error message if the number is too large.
3. After you get the right number, print "howdy!" that many times.
4. Create an empty list variable.
5. Create a while True: loop.
6. Ask the user to make a list of something (for example, favorite movies, foods, or podcasters). Each time the user enters something, add it to the list, unless they enter nothing, then break out of the loop.

- After you are out of the loop print a message saying how many items are in the list (use the len() function)
- Print each item from the list on its own line using a string method (for example, upper(), lower(), center(), etc.) using a for loop.
- Report which list item was the longest and then how many characters long it was using the len() function
- Test your code. Try numbers that work and don't work (too high/too low), try not a number, pay attention to whether your code is reporting the correct longest/shortest entries.

Sample output (yours must be a different type of list than my examples):

```
Enter an integer between 10 and 20, please: 5
Please try again, that was too small.
Enter an integer between 10 and 20, please: 22
Please try again, that was too large.
Enter an integer between 10 and 20, please: 15
howdy!howdy!howdy!howdy!howdy!howdy!howdy!howdy!howdy!howdy!howdy!howdy!howdy!howdy!howdy!
Please enter one of your favorite podcasts: Pod Save America
Please enter one of your favorite podcasts: 99% Invisible
Please enter one of your favorite podcasts: Everything is Alive
Please enter one of your favorite podcasts: Reply All
Please enter one of your favorite podcasts:
Your list has 4 items in it.
$$$$$$$Pod Save America$$$$$$$
$$$$$$$$99% Invisible$$$$$$$$
$$$$$Everything is Alive$$$$$$
$$$$$$$$$$Reply All$$$$$$$$$$
The longest item in the list was Everything is Alive at 19 characters long
```