Grading & Deliverables

***Grading*:** a problem is correct if all our tests pass

Grade by #correct: Each question worth 20 points

You need to submit *hw3.ipynb* and *independent_completion_form* (without this form your homework will not be graded)

Mandatory Functions

The following functions should be implemented and the return statements should be modified with the correct credentials. Do not forget to call the functions

```
def myName():
    return "James Bond"
def myBlazerID():
    return "jbon12"

# Call these functions
print("My Name is =", myName(), " and my BlazerId is =",myBlazerID())
```

HW3 problems

Please use exactly the same names for the functions and the input parameters. Otherwise the auto grader will not work on your submissions. Also, make sure your script file name is hw3.ipynb Failing to do so will be penalized.

`listBuilder(l1, length)`

Write the function **`listBuilder(l1, length)`** that takes a list of strings called **`l1`** and an int called **`length`** and returns another list that contains all the strings in **`l1`** that have a length of at least **`length`**.

Sample Inputs	**Sample Outputs**
`l1 = ["Hello", "I", "am", "a", "list"]` `length = 4`	`["Hello", list"]`
`l1 = ["Homework", "is", "fun"]` `length = 8`	`["Homework"]`
`l1 = ["CS103", "is", "the", "best"}` `length = 8`	`[]`

`lettersOnly(s)`

Write the function **`lettersOnly(s)`** that takes in a string called **`s`**, and returns a string containing only the alphabetic characters of **`s`**. (For more information on how this can be done look at the Python documentation on String Methods found here: https://docs.python.org/3/library/stdtypes.html#string-methods

Sample Inputs	**Sample Outputs**
`s = "CS103 is kind of fun."`	`"CSiskindoffun"`
`s = "Spaces and punctuation don't count!"`	`"Spacesandpunctuationdontcount"`
`s = "This is Hw3"`	`"ThisisHw"`

`elementLengthIsIndex(l2)`

Write the function **`elementLengthIsIndex(l2)`** that takes in a list of strings called **`l2`**, and returns another list of strings that contains each element of the original list where the length of the element matches its index within **`l2`**.

Sample Inputs	**Sample Outputs**
`l2 = ["I'm", "a", "tricky", "one"]`	`["a", "one"]`
`l2 = ["", "cyan", "gold", "red", "blue"]`	`["", "red", "blue"]`
`l2 = ["0", "1", "22", "333", "4"]`	`["1", "22", "333"]`

`typeMatters(par)`

Write the function **`typeMatters(par)`** that takes in an arbitrary parameter called **`par`** that is of an arbitrary type and returns a different a different effect depending on the ***type of par***.
If `par` is a string: return the string repeated s number of times equal to the length of par.
If `par` is an integer: return a string that is the representation of each integer 1 through par followed by a space.
If `par` is a float: return a string that says *"I am a float of value: "* followed by par.
If `par` is a list: return par sorted and reversed.
If `par` is none of the types listed: return a string that says "*I am not a string, int, float, or list."*

Sample Inputs	**Sample Outputs**	**Hints**
`par = "hello"`	`"hellohellohellohellohello"`	`5 times hello`
`par = 8`	`1 2 3 4 5 6 7 8`	`8 is an int`
`par = 5.634`	`"I am a float of value: 5.634"`	`5.634 is a float`
`par = [23, 8, 4, 16, 15, 42]`	`[43, 23, 16, 15, 8, 4]`	`Reverse sorted list`
`par = (3, 4)`	`"I am not a string, int, float, or list."`	`It is a tuple`

reverseNum(n)

Write the function "**`reverseNum`**" that takes an integer **n** and returns another **integer**. The function will reverse the order of the digits and return the new value. Assume the input will contain the positive integers only

Sample Input:	Expected Output:
n = 1234	4321
n = 29	92
n = 10001	10001