Warm up: Given the following variables, what will print when the single lines of code below are executed? If a line crashes, provide a description of the error.

```
a_list = [10, 2, 5, 2, [0,1], [ [30,40], 30] ]
a_string = "More practice problems!"
a_dictionary = {"cat":4, "dog":[2,5], "foobar":"bar", "!":-1}

print ( a_list[2] )
5
print ( a_list.index(2) )
1
print ( a_list[a_list[2]] )
[[30, 40], 30]
print ( a_string[9:13] )
tice
print ( a_string.split("a") )
['More pr', 'ctice problems!']
print ( a_dictionary["!"] )
-1
print ( "bar" in a_dictionary )
False
print ( a_list[a_dictionary["cat"]] )
[0, 1]
print ( a_dictionary.keys() )
dict_keys(['cat', 'dog', 'foobar', '!'])
print ( a_list[ a_dictionary[ a_string[-1] ] ][0][1] )
40
```

1. Trace the output of the following program:

```
word = "xyz123!@#"
for i in range(len(word)-1, -2, -1):

     print (i, word[i], end='')

     if word[i].isdigit():
          for j in range(int(word[i])):
          print ("*", end='')
     print()
```

```
8 #
7 @
6 !
5 3***
4 2**
3 1*
2 z
1 y
0 x
-1 #
```

2. Trace the output of the following program:

```
def foo(a):
     if a.isdigit():
          print (a)

data1 = "5,10,15"
for item in data1:
     foo(item)

print ("----")

data2 = data1.split(",")

for item in data2:
     foo(item)
```

```
5
1
0
1
5
----
5
10
15
```

3. Trace the output of the following program

```
code = { 6:["t", "g"], 14:["p","q"], 15:["c", "d"], 9:["o", "a"] }
data = "5,0:3,1:2,0"

splitdata = data.split(":")

for item in splitdata:
      splititem = item.split(",")
      print ( code[ int(splititem[0])*3 ][ int(splititem[1]) ], end="")
```

cat

4. Write a FUNCTION called “valid_n_number” that determines if a NYU Student ID is valid. For the purpose of this question a valid NYU Student ID is defined as follows:

- exactly 9 characters long
- begins with the uppercase character ‘N’
- all characters beside the beginning ‘N’ character must be numbers

Your function should accept a test “N number” as an ARGUMENT (String) and RETURN a status value (Boolean). Here’s a sample program that uses your function:

```
test1 = valid_n_number("N123")
test2 = valid_n_number("N1234567890")
test3 = valid_n_number("P12345678")
test4 = valid_n_number("NXYZ!5678")
test5 = valid_n_number("N12345678")

print (test1, test2, test3, test4, test5)
```

And here’s the expected output:

```
False False False False True
```

```
def valid_n_number(n):
    if len(n) == 9 and n[0] == "N" and n[1:len(n)].isdigit():
        return True
    else:
        return False
```

5. Write a program that asks the user to enter in 5 NYU Student ID numbers. Ensure that the ID numbers that the user enters are valid (use your function from question #4) and that they do not enter duplicate IDs. Let the user know if they make a mistake. When you have collected 5 IDs print them out in ascending order. Here's a sample running of this program. Note that shaded items represent data that the user has typed into the program:

```
Enter a N Number: N12345678
Enter a N Number: N123
Invalid N Number
Enter a N Number: P12345678
Invalid N Number
Enter a N Number: N12345678
Duplicate N Number
Enter a N Number: N00000003
Enter a N Number: N00000001
Enter a N Number: N00000002
Enter a N Number: N00000000

Your N-Numbers:

N00000000
N00000001
N00000002
N00000003
N12345678
```

```
ns = []
for i in range(5):
    while True:
        n = input("Enter a N Number: ")
        if n in ns:
            print("Duplicate N Number")
        elif valid_n_number(n):
            ns.append(n)
            break
        else:
            print("Invalid N Number")

print()
print("Your N-Numbers\n")
ns.sort()
for n in ns:
    print(n)
```

6. Trace the output of the following program. Then rewrite the program using a "while" loop instead of a "for" loop.

```
colors = ['Red', 'Green', 'Blue', 'Yellow', 'Pink']

for i in range(0,len(colors), 2):
     print (i, colors[i])
```

```
0 Red
2 Blue
4 Pink
```

```
while i < len(colors) - 1:
    print (i, colors[i])
    i += 2
```

7. Justin Bieber needs your help! He is having a tough time writing songs that aren't completely repetitive, and he is turning to you to write a program to help him clean up his act.

Justin has asked you to write a program that analyzes a single line of text from one of his songs. A line of text contains a mixture of letters and spaces, but it does not contain any punctuation. Here's an example of what this line could look like.

```
lyrics = "like Baby baby baby ohhh Baby baby Like nooo"
```

Using this String, extract one unique copy of each word that you find and then print back the "unique" lyrics back to the user. Your program should be case insensitive (i.e. "Baby" is the same as "baby"). For example, the lyrics above would reduce down to the following:

```
like baby ohhh nooo
```

Note that for this question you can simply "hard code" these lyrics into your program (i.e. no user input required)

```
lyrics = "like Baby baby baby ohhh Baby baby Like nooo"

words = lyrics.split(" ")
reduced = []

for w in words:
    if w.lower() not in reduced:
        reduced.append(w.lower())

for w in reduced:
    print(w, end= " ")
```

8. Write a function called “line” that accepts a string as an argument. Your function should generate a string pattern based on the argument provided. Here are the rules:

(1) if the string provided is a number, and that number is even, generate a pattern of hash signs. For example, calling line(“4”) will generate the following:

####

and calling line(“10”) will generate the following:

##########

(2) if the string provided is a number, and that number is odd, generate a pattern of stars. For example, calling line(“3”) will generate the following:

***

and calling line(“9”) will generate the following:

****

(3) if the string provided is not a number you can simply generate an empty string.

Once you have generated your pattern you should return the result when completed. You cannot assume anything about the supplied string, and your function should avoid raising any exceptions that will cause your program to crash. If an exception is raised you can return an empty string.

```
def line(num):
    try:
        multiplier = int(num)
    except:
        return ""
    else:
        if multiplier % 2 == 1:
            char = "*"
        else:
            char = "#"
```

```
        return char * multiplier

print(line("4"))                ####
print(line("10"))               ##########
print(line("3"))                ***
print(line("asdfjhadlksfj"))
```