1. Trace the output of the following programs:

code = {"#":"foo", "@":"bar", ":":"hello", ":":"world"}

for i in range(33, 40):
    if chr(i) in code:
        print ( i, chr(i), code[ chr(i) ] )
    else:
        print ( i, chr(i) )

33 ! hello
34 "
35 # foo
36 $
37 %
38 & world
39 '

code = { "a":"o", "b":"e", "c":"p", "d":"t", "e":"r" }
data = "2tzczy6a8i_d!@55qd*88bopse"

for d in data:
    if d in code:
        print (code[d], end="")

potter
2. Given the following program:

```python
myfile = open("foobar.txt", "w")
for i in range(1, 5):
    myfile.write("#" * i + 

myfile.close()

myfile2 = open("foobar.txt", "a")

x = 0
while x < 3:
    myfile2.write(".")
    x += 1

myfile2.close()
```

What will be stored in the file “foobar.txt” at the end of this program? Note that “foobar.txt” does not exist at the beginning of this program.

```
#
#
##
###
####
...
3. Write a program that lets user enter in a potentially unlimited series of price values. Ensure that the numbers entered are greater than 0. You cannot assume that the user will enter an integer or a float. If they input a negative number (or zero) you should continually prompt them to enter a valid number until they do so. When the user enters a price with a value of 0 you can stop collecting prices.

Then use their input to generate a summary report that includes the cost of all items purchased, the average cost of each item, highest priced item, lowest priced item, number of items above & below the average. Below is a sample running of this program. Don’t worry about formatting your numbers for this problem.

```
Enter a price, 0 to end: apple
That's not a number!

Enter a price, 0 to end: -5
Prices must be positive!

Enter a price, 0 to end: 10
Enter a price, 0 to end: 20
Enter a price, 0 to end: 30
Enter a price, 0 to end: 40
Enter a price, 0 to end: 50
Enter a price, 0 to end: 0

Total cost: 150.0
Average cost: 30.0
Highest price: 50.0
Lowest price: 10.0
# of prices >= the average: 3
# of prices < the average: 2
```
#Question 3

total = 0

while True:
    try:
        price = float(input("Enter a price, 0 to end: "))
        if price < 0:
            print("Prices must be positive!")
        elif price == 0:
            break
        else:
            items.append(price)
            total += price
    except:
        print("That's not a number!")

average = total/len(items)
aboveAvg = 0
belowAvg = 0

for i in items:
    if i >= average:
        aboveAvg += 1
    else:
        belowAvg += 1

print('Total cost:', total)
print('Average cost:', average)
print('Highest price:', max(items))
print('Lowest price:', min(items))
print('# of prices >= the average:', aboveAvg)
print('# of prices < the average:', belowAvg)
4. Write a FUNCTION called “valid_url” that determines if URL (website address) is valid. Here is a sample valid URL:

```
http://www.nyu.edu
```

For the purpose of this question a valid URL is defined as follows:

- Begins with the protocol String “http://”
- After the protocol the URL must contain a domain name. The domain name consists of a series of alphabetic Strings separated by periods. A domain name must contain at least two Strings separated by a single period, but it can contain more than two. For example:

```
http://www.nyu.edu # valid
http://nyu.edu     # valid
http://nyu         # invalid
http://new.classes.nyu.edu # valid
```

Your function should accept a test URL as an ARGUMENT (String) and RETURN a status value (Boolean). Your function should not ask the user to supply any input. Here’s a sample program that uses your function:

```
print ( valid_url("http://www.nyu.edu") ) # True
print ( valid_url("http://files.nyu.edu") ) # True
print ( valid_url("http://www.myweb.nyu.edu") ) # True
print ( valid_url("http://nyu.edu") ) # True
print ( valid_url("http://nyu") ) # False
print ( valid_url("http://") ) # False
print ( valid_url("foobar") ) # False
```
#Question 4

'''Unlike I did in class, I think this one is best served by just returning False if it's no longer a valid URL, otherwise return true'''

def valid_url(string):
    # make sure it's even long enough to bother checking
    if len(string) > 9:
        # If the first 7 positions are not the http://
        if string[0:7] != "http://":
            return False
        # If it doesn't even have a period
        if string.find('.') == -1:
            return False
        # make a substring of everything after the http://
        subString = string[7:]
        # Split at the period(s)
        subList = subString.split('.')
        # For each value between periods make sure it's all alpha
        for i in subList:
            if i.isalpha() == False:
                return False
        # If it gets here and nothing has been returned, then it's valid
        return True
    # If it was never long enough to be a possible URL
    else:
        return False
5. Britney Spears needs your help! She was recently accused of being overly repetitive in her song lyrics and she wants you to analyze her songs to find out which words she is over-using.

Britney has asked you to write a program that analyzes a single line of text from one of her songs. A line of text contains a mixture of letters, and spaces and punctuation. Here’s an example of what this line could look like.

```
lyrics = "Oh baby,baby Oh baby,baby Ah,yeah,yeah Oh baby,baby How was I supposed to know"
```

Using this String as a sample, determine the word that appears most frequently in the song. Your program should be case insensitive (i.e. "Ah" is the same as “ah”). For example, the lyrics above would generate the following output:

```
“baby” was used 6 times
```

Note that for this question you can simply “hard code” these lyrics into your program (i.e. no user input required). This doesn’t mean you can only look for these words in your code.
#Question 5

```python
lyrics = "Oh baby,baby Oh baby,baby Ah,yeah,yeah Oh baby,baby How was I supposed to know"

# Need to split and the subsplit again when there's commas. Start with splitting at the space
lyricsSplit = lyrics.split(' ')
lyricsDict = {}

for lyric in lyricsSplit:
    # If there is a comma in a list item
    if lyric.find(',') != -1:
        # Split at the comma, now there's a list within the list
        subSplit = lyric.split(',' ')
        # For each value in that sub list
        for word in subSplit:
            # If it's in the dictionary add a value, otherwise start one
            if word.lower() in lyricsDict:
                lyricsDict[word.lower()] += 1
            else:
                lyricsDict[word.lower()] = 1
    else:
        # If there is no comma in the list item do same thing with thesaurus
        if lyric.lower() in lyricsDict:
            lyricsDict[lyric.lower()] += 1
        else:
            lyricsDict[lyric.lower()] = 1

# To count max, have a start maximum and blank string for that word
maximum = 0
maxLyric = ''

# Cycle through the key and value in that dictionary
for key, value in lyricsDict.items():
    # If the current value is greater than the previous max
    if value > maximum:
        # Set the max to the new value
        maximum = value
        # And the lyric to the corresponding key
        maxLyric = key

# Now print off the lyrics and max value
print('"' + maxLyric + '"', "was used", maximum, "times")
```