PYTHON EXERCISE I

Q1. Assume that two variables, `varA` and `varB`, are assigned values, either numbers or strings. Write a piece of Python code that evaluates `varA` and `varB`, and then prints out one of the following messages:

- `"string involved"` if either `varA` or `varB` are strings.
- `"bigger"` if `varA` is larger than `varB`.
- `"equal"` if `varA` is equal to `varB`.
- `"smaller"` if `varA` is smaller than `varB`.

Q2. Convert the following into code that uses a while loop.

prints: `Hello!`
prints: `10`
prints: `8`
prints: `6`
prints: `4`
prints: `2`

Q3. Write a while loop that sums the values 1 through `end`, inclusive. `end` is a variable that we define for you. So, for example, if we define `end` to be 6, your code should print out the result: `21`

which is 1 + 2 + 3 + 4 + 5 + 6.

Q4. Repeat Q2 and Q3 using for loop instead of while loop.

Q5. Below are some short Python programs. For each program, determine what the code prints out. Try to answer the questions without running the code.

This question is going to ask you what some simple loops print out. If a given loop will not terminate, write the phrase 'infinite loop'. write what it prints out, separating what appears on a new line by a comma and a space.

A)

```
num = 10
while True:
    if num < 7:
        print('Breaking out of loop')
        break
    print(num)
    num -= 1
print('Outside of loop')
```

B)

```
num = 100
while not False:
    if num < 0:
        break
print('num is: ' + str(num))
```

c)

```
for variable in range(20):
    if variable % 4 == 0:
        print(variable)
    if variable % 16 == 0:
                   print('Foo!')
```

D)

```
count = 0
for letter in 'Snow!':
    print('Letter # ' + str(count) + ' is ' + str(letter))
    count += 1
    break
print(count)
```

E)

```
greeting = 'Hello!'
count = 0
for letter in greeting:
    count += 1
    if count % 2 == 0:
        print(letter)
    print(letter)
print('done')
```

- How many times does `H` print out?
- How many times does `e` print out? Disregard the letters in the word done.
- How many times does `l` print out?
- How many times does `o` print out? Disregard the letters in the word done.
- How many times does `!` print out?
- How many times does `done` print out?

Q6.

Code Sample

```
iteration = 0
count = 0
while iteration < 5:
    for letter in "hello, world":
        count += 1
    print("Iteration " + str(iteration) + "; count is: " +
str(count))
    iteration += 1
```

We wish to re-write the above code, but instead of nesting a `for` loop inside a `while` loop, we want to nest a `while` loop inside a `for` loop. Which of the following loops gives the same output as the **Code Sample**?

Try to answer the following questions by just reading the code. Reading code is a very good skill to have (and will help you both in your programming career and on your exams!). It is okay to check your answers that you obtain from just reading the code, then in your interpreter run the code for the ones you got wrong on your first attempt.

For each of the following codes, determine whether it gives the same output as the code sample or not.

Test 1

```
for iteration in range(5):
    count = 0
    while True:
        for letter in "hello, world":
            count += 1
        print("Iteration " + str(iteration) + "; count is:
" + str(count)
```

Test 2

```
for iteration in range(5):
    count = 0
    while True:
        for letter in "hello, world":
            count += 1
        print("Iteration " + str(iteration) + "; count is:
" + str(count))
        break
```

Test 3

```
count = 0
phrase = "hello, world"
for iteration in range(5):
    index = 0
    while index < len(phrase):
        count += 1
        index += 1
    print("Iteration " + str(iteration) + "; count is: " +
str(count))
```

Test 4

```
count = 0
phrase = "hello, world"
for iteration in range(5):
    while True:
        count += len(phrase)
        break
    print("Iteration " + str(iteration) + "; count is: " +
str(count))
```

Test 5

```
count = 0
phrase = "hello, world"
for iteration in range(5):
    count += len(phrase)
    print("Iteration " + str(iteration) + "; count is: " +
str(count))
```

Q7. Assume `s` is a string of lower case characters. Write a program that prints the number of times the string `'bob'` occurs in `s`. For example, if `s = 'azcbobobegghakl'`, then your program should print:

Number of times bob occurs is: 2

Q8. Try to answer the questions without running the code.

```
iteration = 0
while iteration < 5:
    count = 0
    for letter in "hello, world":
        count += 1
        if iteration % 2 == 0:
            break
    print("Iteration " + str(iteration) + "; count is: " +
str(count))
    iteration += 1
```

How many times will the `print` statement be executed?
What is the largest value of the variable `iteration` that will be printed out (at the `print` statement)?
What is the largest value of the variable `count` that will be printed out (at the `print` statement)?
What is the smallest value of the variable `count` that will be printed out (at the `print` statement)?

Q9. In this problem, you'll create a program that guesses a secret number!

The program works as follows: you (the user) thinks of an integer between 0 (inclusive) and 100 (not inclusive). The computer makes guesses, and you give it input - is its guess too high or too low? Using bisection search, the computer will guess the user's secret number!

Here is a transcript of an example session (i.e. your program should print the following- if your number is 83):

```
Please think of a number between 0 and 100!
Is your secret number 50?
Enter 'h' to indicate the guess is too high. Enter 'l' to
indicate the guess is too low. Enter 'c' to indicate I
guessed correctly. l
Is your secret number 75?
Enter 'h' to indicate the guess is too high. Enter 'l' to
indicate the guess is too low. Enter 'c' to indicate I
guessed correctly. l
Is your secret number 87?
Enter 'h' to indicate the guess is too high. Enter 'l' to
indicate the guess is too low. Enter 'c' to indicate I
guessed correctly. h
Is your secret number 81?
Enter 'h' to indicate the guess is too high. Enter 'l' to
indicate the guess is too low. Enter 'c' to indicate I
guessed correctly. l
Is your secret number 84?
Enter 'h' to indicate the guess is too high. Enter 'l' to
indicate the guess is too low. Enter 'c' to indicate I
guessed correctly. h
Is your secret number 82?
Enter 'h' to indicate the guess is too high. Enter 'l' to
indicate the guess is too low. Enter 'c' to indicate I
guessed correctly. l
Is your secret number 83?
```

```
Enter 'h' to indicate the guess is too high. Enter 'l' to
indicate the guess is too low. Enter 'c' to indicate I
guessed correctly. c
Game over. Your secret number was: 83
```

Q10. Below are some short Python programs. For each program, determine what the code prints out. Try to answer the questions without running the code. Check your answers, then run the code for the ones you get wrong.

A)

```
a = 10
def f(x):
  return x + a
a = 3
f(1)
```

B)

```
x = 12
def g(x):
  x = x + 1
  def h(y):
      return x + y
  return h(6)
g(x)
```

Q11.

The greatest common divisor of two positive integers is the largest integer that divides each of them without remainder. For example,

- gcd(2, 12) = 2
- gcd(6, 12) = 6
- gcd(9, 12) = 3
- gcd(17, 12) = 1

Write a function, gcdIter(a, b), that implements this idea. One easy way to do this is to begin with a test value equal to the smaller of the two input arguments, and iteratively reduce this test value by 1 until you either reach a case where the test divides both a and b without remainder, or you reach 1.

Q12. A regular polygon has n number of sides. Each side has length s.

- The area of a regular polygon is: $\frac{0.25*n*s^2}{tan(\pi/n)}$
- The perimeter of a polygon is: length of the boundary of the polygon

Write a function called polysum that takes 2 arguments, n and s. This function should sum the area and square of the perimeter of the regular polygon. The function returns the sum, rounded to 4 decimal places.

Hint: Which library should you be importing at the beginning of your code in order to call the tan function and to get the pi constant?

Q13. Write a Python program that includes the following functions:

A) A function that finds the repeated items of a tuple.

Sample tuple/Input: `(2, 4, 5, 6, 2, 3, 4, 4, 7)`
Function returns**:** `3`

B) A function that find the second largest number in a list.

Sample List/Input**:** `[1, 1, 1, 0, 0, 0, 2, -2, -2]`
Output: `1`
Sample List/Input: `[2,2]`
Output: `None`

C) A function that removes duplicates from a list.

Sample List/Input: `[10,20,30,20,10,50,60,40,80,50,40]`
Output: `[40, 10, 80, 50, 20, 60, 30]`

D) A function that counts the number of elements in a list within a specified range.

Sample List/Input: `[10,20,30,40,40,40,70,80,99],range= 40,100`
Output: `6`

Q14. Write a Python codes using NumPy that:

A) Form the 2-D array (without typing it in explicitly), and generate a new array containing its 2nd and 4th rows.

B) Create an array of the first 10 odd integer. Then convert it to 2D array with 2 rows.

C) Create a 4x3 array and compute: the sum of all the entries, the sum of the rows and the sum of the columns.

D) Create a random 6x6 array. Then sort the rows according to the second column. Try combining array slicing + argsort + indexing to do this.

E) Write a Python program to normalize a 3x3 random matrix.

```
Original Array:
[[ 0.89372503  0.99865458  0.77120044]
 [ 0.67632984  0.99990084  0.64110391]
 [ 0.34845794  0.66557903  0.29031742]]

After normalization:
[[ 0.85036881  0.99824367  0.67769765]
 [ 0.54399864  1.  0.49435553]
 [ 0.08193614  0.52884777  0. ]]
```

F) Create a 2d array with 1 on the border and 0 inside.

G) Create a 4x5 array, then remove the first column and store it in another array, multiply the modified array by the Identity matrix then stack it after with the 4x1 sliced array to reform a 4x5 array.