Name: __________________________________________
Net ID ________________

There are 2 sections, the first section is worth 50 points and the second section is worth 50 points, for a total of 100. Answer all 4 questions in Section 1. Answer 2 out of the 3 questions in Section 2.

It is essential that you PUT YOUR NAME AND NET ID ON ALL TEST MATERIALS. It can be difficult to identify the author of an unsigned test and it would be better to avoid this problem.

There is a GLOSSARY OF TERMS at the end of the test. Please feel free to look up some of the basics in this glossary. I will also answer any reasonable look-up style questions as I am more interested in your ability to reason than your ability to memorize.

1 Answer questions about code.

Worth 50 points

Each example consists of Python code, followed by questions and places to fill in answers. Please read all questions carefully and answer them as precisely as possible. Assume that there are no bugs in the code that will make the system crash, although the code may not solve a problem perfectly. If you find anything that you think is a bug, there is either a typo (and I should fix it for everyone) or you are mistaken.

Sample Question A:

```python
output = '1'+'1'
```

Question: What is the value of the variable `output`?
Answer: `'11`'

Note: Attention to detail is important. The quotes indicate that it is a string. Partial credit is possible. For example, leaving out the quotes would have lost just a little bit, but answering 2, would have resulted in an incorrect answer.
Question 1

import turtle

def turtle_setup():
    global my_screen
    global my_turtle
    my_screen = turtle.Screen()
    my_turtle = turtle.Turtle()

def draw_shape_on_square(sides, length):
    turtle_setup()
    angle = 360/sides
    for square_side in range(4):
        my_turtle.pd()
        for side in range(sides):
            my_turtle.fd(length)
            my_turtle.left(angle)
        my_turtle.pu()
        my_turtle.fd(length)
        my_turtle.right(90)

draw_shape_on_square(3, 50)

Question: draw (approximately) whatever would be drawn by the turtle program above.
Question 2

def two_word_combos(list1, list2, index1s, index2s, length):
    output = []
    for iteration in range(length):
        index1 = index1s.pop(0)
        index2 = index2s.pop(0)
        item1 = list1[index1]
        item2 = list2[index2]
        output.append([item1, item2])
    return(output)

def main():
    word_list1 = ["The cat", "The dog", "The bird", "The monkey", "The tiger"]
    word_list2 = ["booed", "begged", "bobbed", "bugged", "cawed", "chewed", "flew", "jumped", "popped", "ran", "sinned"]
    index1s = [1, 4, 1, 3, 3, 4, 2, 4, 1, 3]
    index2s = [4, 3, 4, 2, 10, 2, 9, 0, 10, 4]
    answer = two_word_combos(word_list1, word_list2, index1s, index2s, 6)
    for pair in answer:
        print(pair[0], pair[1])

main()

Question: What prints out when running the above code?
def add_ing(word):
    length = len(word)
    if (length>2) and (word[-1] in 'aeiou') and (not word[-2] in 'aeiou'):
        ## remove final vowel
        return(word[:-1]+'ing')
    elif (length>2) and (word[-2] in 'aeiou') and (not word[-3] in 'aeiou'):
        ## cases of words of more than 2 letters ending in
        ## a single vowel (not 2 vowels) and a consonant
        if (word[-1] == 'c'):
            ## add 'k' to double c
            return(word+'king')
        elif (word[-1] in 'bdglmnprt'):
            ## really double other consonants
            return(word+word[-1]+'ing')
        else:
            return(word+'ing')
    elif (length == 2) and (word[-1] in 'bdglmnprt'):
        ## 2 letter words with doubled consonants
        return(word+word[-1]+'ing')
    else:
        return(word+'ing')

def main():
    for word in ['up','sound','plume','hum',
                 'fly','frolic','abstain']:
        print(word,add_ing(word))

main()

Question: Write each line that prints out when the function main is executed?
def count_and_sort_strings(input_list):
    output = []
    for word in input_list:
        output.append([len(word), word])
    output.sort()
    return(output)

def main():
    answer = count_and_sort_strings(['up', 'sound', 'plume', 'hum',
                                      'fly', 'frolic', 'abstain'])
    for item in answer:
        print(item)

Question: Write each line that prints out when the function main is executed?
2 Write Code

Worth 50 points
Answer 2 of the 3 questions in section 2 (Questions 5, 6 and 7). For each question that you answer, write a Python program as specified. If you choose to answer all 3 questions, please indicate which ones you would like to count for the test.

**Question 5:** Write a program that takes a list of lists of numbers as input and produces a single list of numbers as output. Each of the input number lists should be the same length. The output list should be derived by averaging the corresponding positions in the input lists. Thus the first number in the output list is the average of all the first numbers in the input lists. The second number in the output list is an average of all the second numbers in the input lists, and so on. Consider the following example:

```python
>>> average_list([[1, 2, 3, 4], [3, 3, 3, 5], [10, 1, 2, 17]])
[4.67, 2.0, 2.67, 8.67]
```

4.67 is the average of 1, 3 and 10; 2.0 is the average of 2, 3, and 1; 2.67 is the average of 3, 3 and 2; and 8.67 is the average of 4, 5 and 17. It is optional to round to two decimal places as I did. If you want to do this, use `round`, with the number as the first argument and 2 (number of decimal places) for the second, e.g.,

```python
>>> round(1234.19111, 2)
1234.19
```

Remember, your program should work for any length list and any length list of lists. It should not be limited to lists of four numbers. It should not be limited to three input lists.
Question 6: Write a program using the turtle library that draws a circle on top of an equilateral triangle, as in the following picture. By definition, the sides of an equilateral triangle are equal. The diameter of the circle should be the same as the length of a side of the triangle. The circle should be drawn parallel to the base of the triangle.

Remember that the turtle starts off facing directly rightward. When facing rightward, `my_turtle.circle(R)` will draw a circle with a radius of $R$ above it and parallel to the direction it is facing. Similarly, if a turtle is facing leftwards, it will draw the circle below the line, etc. (Note that 2 times the radius is the diameter.)
Question 7: Write a program for playing a dice game called wombat. The object of the game is to get the highest score. First, the player and the computer each roll 6 dice. These dice are sorted from highest to lowest (remember list.sort() sorts the items from lowest to highest and list.reverse() reverses a list). Next, the player must re-roll 6 times, one die at a time. Each time, the player can choose which die to re-roll. The program should re-sort the list of rolls, after each re-roll. Finally, the computer re-rolls 6 times, 1 die at a time. The computer will always re-roll the lowest scoring die.

The score for a set of 6 dice is computed as the sum of the face values of the dice plus bonus amounts for multiple instances of the same number. If the same number occurs 6 times, there is a 20 point bonus; if the same number occurs 5 times there is a 16 point bonus; 4 times yields a 12 point bonus; 3 times yields an 8 point bonus; and a pair of the same number yields a 4 point bonus (i.e., the bonus for N-tuple = (N-1) X 4).

Here is an example game being played (note that the dice are re-sorted after each re-roll):

```python
>>> wombat()
Computer’s hand is [6, 5, 5, 4, 4, 4]
Your current roll is [6, 3, 3, 2, 2, 1]
Which item should be rerolled?
Choose a number 1 to 6. 1 is the first die, 2 is the second die, ...6
Your current roll is [6, 4, 3, 3, 2, 2]
Which item should be rerolled?
Choose a number 1 to 6. 1 is the first die, 2 is the second die, ...2
Your current roll is [6, 3, 3, 3, 2, 2]
Which item should be rerolled?
Choose a number 1 to 6. 1 is the first die, 2 is the second die, ...6
Your current roll is [6, 3, 3, 3, 2, 1]
Which item should be rerolled?
Choose a number 1 to 6. 1 is the first die, 2 is the second die, ...6
Your current roll is [6, 3, 3, 3, 2, 1] Computer hand: [6, 6, 5, 5, 5, 4]
You Lose! 43 to 32
```

In this example, the final score of the player’s hand [6, 5, 4, 3, 3, 3] consists of: (a) 6+5+4+3+3+3=24 and (b) 8 bonus points for having 3 3s, for a total of 32. The computer’s score consists of (a) 6+6+5+5+5+4=31, (b) 4 bonus points for the pair of 6s and (c) 8 bonus points for getting 3 5s, for a total of 43.

Some useful functions you may want to use to write this program include: `sum(list)`, which will create a sum from a list of numbers, and `list.count(item)`, which counts the number of instances of a particular item. Examples follow:

```python
>>> sum([6, 6, 5, 5, 5, 4])
31
>>> [6, 6, 5, 5, 5, 4].count(5)
3
```

Extra Credit for Question 7: Only do this part if you have time. The question 7 program represents a single round of wombat. Modify the previous program so each round returns 'win', 'lose' or 'tie'. Write a program that uses these rounds as part of a betting game. The player starts out with $1000 and can bet any amount per round. In the case of a tie, no money is won or lost. A ‘lose’ results in losing the amount bet and a ‘win’ results in gaining the amount bet. The player must stop playing if he/she runs out of money, but can choose to stop playing at any time. If the player either runs out of money or stops with less than $500, print that they 'lost bigtime'. If the player stops with less than $1000, print that they 'lost some money'. If they stop with $1500 or more, print that they 'won big time'. If they have exactly $1000 when the game ends then print that they 'broke even'.
Basic Stuff to Look Up for the Test

1. Some Basics

- **return(X)** causes the current function to exit and cause the expression represented by the function call to evaluate as X. For example given the following steps, the value of output would be 5:

  ```python
def add(num1, num2):
    return(num1+num2)
output = add(2, 3)
```

- **print(X)** prints X to the screen. This is only for the benefit of the user. It is not useful for having programs interact.

- The parameters of a function are the local variables inside of the parentheses in the function definition. They are useful when you have functions call functions.

- **input(prompt)** is used to ask a human being a question so that a program can interact with a human being. This is useful when you want a human being to enter information interactively. input statements should be used only when human interaction is appropriate. input statements return a string corresponding to what the user typed in. It may be necessary to convert this string to some other data type, e.g., an integer (with int) or a float (with float).

- The operator + will add two numbers or concatenate two strings

- The operator * will multiple two numbers or print a string some number of times.

2. sequences

- object made up of other objects in an order

- the function len(sequence) returns the number of items in the sequence

- the operator in tests for membership in sequence, e.g., ('a' in 'abc') would have the value True.

- sequences are used in for loops (see below)

- indices and slices
  - Indices in a sequence are numbers from zero to the length of the string. Zero refers to the position before the first item in the string and the length of the string refers to the position following the last item. Thus each item in the sequence are between two consecutive indices. For example, the subscripted numbers indicate indices for the string The book: 'aT1h2e3 b5o6o7k8. Similarly, the indices in [0 'The',1 'book',2 is,3 'there',4] indicate positions in the list ['The', 'book', 'is','here'].
  - sequence[num] indicates an element in a sequence beginning at num (a number from zero to one less than the length of the string), e.g., 'The book'[4] evaluates to 'b'; ['The', 'book', 'is','here'][0] evaluates to 'The'.
  - sequence[num1:num2] indicates a subsequence beginning at position num1 and ending at num2,e.g., 'The book'[4:6] evaluates to 'bo'; ['The', 'book', 'is','here'][0:2] evaluates to ['The', 'book'].
  - Leaving out the number before the colon suggests that a subsequence begins at the beginning of the sequence and leaving out the number after the colon suggests that the subsequence ends at the end of the list. Thus 'The book'[:3] evaluates as 'The' and ['The', 'book', 'is','here'][2:] evaluates as ['is','here'].

- ranges
  - **range(5)** is approximately equivalent to [0,1,2,3,4]
  - **range(1,5)** is approximately equivalent to [1,2,3,4]

- Strings
  - an empty string has zero characters "
strings are sequences of characters, e.g., 'Hello World!' consists of the items ['H', 'e', 'l', 'l', 'o', ',', 'W', 'o', 'r', 'l', 'd', ', ']' string1.strip(string2) – removes instances of string2 from beginning and end of string. For example, '***Hello World***'.strip('*') will return 'Hello World'.

string.split(character) – creates a list by dividing a string at each instances of character. For example, 'Hello World'.split(' ') will return the list ['Hello', 'World'].

string.lower() converts string to lower case

• Lists
  – A list is represented by square brackets surrounding a list of objects, divided by commas, e.g., ['A', 'List', 'of', 'Strings']
  – Lists are mutable.
  – You can add an object to the end of a list using the append method. For example, suppose my_list = ['a', 'b', 'c']. Then my_list.append('d') will add 'd' to the end of my_list, setting it to ['a', 'b', 'c', 'd'].

3. Division and Modulus
  • 5 // 2 == 2
  • 5/2 == 2.5
  • 5%2 == 1

4. print
  • sep – separator between items
  • end – printed at the end of print statement

5. for loops
  • First Line: for VARIABLE in SEQUENCE:
  • VARIABLE is set to each item in the sequence one at a time
  • The Indented body repeats once for each item in sequence (for each setting of VARIABLE).
  • It is common to exit a loop of any kind by using a return to exit the function.
  • It is common to initialize a variable outside a loop (called an accumulator) that then gets incremented inside the loop.

6. if statements
  • the first line of an if statement consists of if BOOLEAN-EXPRESSION:
  • the body of text indented under the first line is executed if the BOOLEAN-EXPRESSION evaluates to True
  • the if statement can be followed by optional elif statements of the same form, except that the first line begins with elif. Each elif statement is only evaluated if the BOOLEAN expressions in the if and elif statements leading up to this one are False.
  • The block of if and optional elif statements can end with an optional else statement. The first line is simply else:. The body of text under else executes if the Boolean expressions for all previous if and elif statements in the sequence evaluate to False.

7. Turtles
  • Screen and Turtle objects are created using the commands turtle.Screen() and turtle.Turtle().
  • The turtle is initially in the center of the screen facing rightward.
  • my_turtle.left(degrees) – rotates the my_turtle degrees left (from its perspective).
• `my_turtle.fd(distance)` – moves the `my_turtle distance` units forward.
• `my_turtle.pu()` – picks the pen up
• `my_turtle.pd()` – puts the pen down (ready to write)
• `my_turtle.circle(radius)` – creates a circle with radius `radius`. The circle will be above the direction the turtle was facing when it started drawing. The turtle will move left and up in a circle and end up in the same place as before.

8. `time.sleep(sec)` – pauses for `sec` seconds (requires the module `sleep` to be imported)