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. 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.

Section 1: (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**

```python
def plutify(number1,number2):
    output = (number1 / number2) ** number2
    return(output)

answer = plutify(3,2)
```

**Question: What is the value of the global variable `answer` after the above code is executed?**
Question 2

def draw_flower():
    for num in range(5):
        print(' O',end='')
    print()
    for num in range(5):
        print(' V',end='')
    print()
    for num in range(2):
        print('',end='')
        for num in range(4):
            print(' * ',end='')
        print()
    for num in range(2):
        print('',end='')
    for num in range(3):
        print('',end='')
        print(' * ',end='')
    print()
    for num in range(3):
        print('',end='')
    for num in range(2):
        print('',end='')
    print(' * ',end='')
    print()
    for num in range(4):
        for num in range(5):
            print('',end='')
        print(' * ')
    print(' )

draw_flower()

Draw approximately what the above code causes to print out. A 14 by 14 grid is provided, to make this easier.
Question 3

def average (num1,num2):
    average = (num1+num2)/2
    return(average)

def accumulate_average(maximum):
    previous_total = 0
    total = 0
    for number in range(maximum):
        previous_total = total
        total = average(total,number)
        print(previous_total,total)
    return(total)

output = accumulate_average(5)

Question: What is the value of the global variable output after the above code is executed?
def leave_tip():
    price = float(input('How much was the meal, not including tax? '))
    tax = float(input('How much was the tax? '))
    service = float(input('Enter service charge included on the bill. Enter 0 if no charge. '))
    double_tax = input('Do you want to double the tax? Yes or No ')
    if (double_tax == 'yes') or (double_tax == 'Yes '):
        percentage = (tax/price)*2
    else:
        percentage = float(input('Enter total percent tip. '))/100
    percent_service = service/price
    percentage = percentage-percent_service
    if percentage > 0:
        tip = percentage*price
    else:
        tip = 0
    dollars = tip//1
    cents = round(100*(tip-dollars))
    print('You should leave a tip of,',dollars,'dollars and', cents,'cents')
    return(tip)

Question 4a What is printed out and what is returned if: the user paid 10 for the meal, the tax was .88, the user wanted to double the tax and a service charge of 1.80 was already added to the bill.

Question 4b Suppose the meal was 21, the tax was 1.85, the user wanted to pay 20 percent, but a service charge of 1 was already added to the bill. What would print out and what would be returned.
**Section 2** (50 points): Answer 2 of the 3 questions in this section. For each question, you do, write a Python program as specified. If you choose to answer all 3 questions (only do this if you really have enough time), please indicate which ones you would like to count for the test.

**Question 5:** Write a function called `decode_answer` that takes no parameter arguments, but contains an input statement that queries *What is your answer?*. If the first character in the user’s answer is ‘t’, ‘T’, ‘Y’ or ‘y’, the function returns True. If the first character of the answer is ‘f’, ‘F’, ‘n’, or ‘N’, the program returns false. If the first character is a digit character (‘0’, ‘1’, ‘2’, ..., ‘9’), it returns the float of the user’s answer, e.g., it would convert the string ‘57.5’, to the number 57.5. Otherwise, it returns the string that the user entered. Example function calls and returns follow.

```python
>>> decode_answer()
What is your answer? YESSS!!!
True
>>> decode_answer()
What is your answer? False
False
>>> decode_answer()
What is your answer? Yellow chicken
True
>>> decode_answer()
What is your answer? 57.6
57.6
>>> decode_answer()
What is your answer? Banana
'Banana'
```
**Question 6:** Write a function with one parameter, maximum, which we will assume is a positive integer. For all combinations of two numbers from one to that maximum value, the program should print out the two numbers and the result of dividing the first number by the second number by normal (float) division, NOT integer division. An example function call follows.

```python
>>> division_table(5)
1 divided by 1 equals 1.0
1 divided by 2 equals 0.5
1 divided by 3 equals 0.33
1 divided by 4 equals 0.25
1 divided by 5 equals 0.2
2 divided by 1 equals 2.0
2 divided by 2 equals 1.0
2 divided by 3 equals 0.67
2 divided by 4 equals 0.5
2 divided by 5 equals 0.4
3 divided by 1 equals 3.0
3 divided by 2 equals 1.5
3 divided by 3 equals 1.0
3 divided by 4 equals 0.75
3 divided by 5 equals 0.6
4 divided by 1 equals 4.0
4 divided by 2 equals 2.0
4 divided by 3 equals 1.33
4 divided by 4 equals 1.0
4 divided by 5 equals 0.8
5 divided by 1 equals 5.0
5 divided by 2 equals 2.5
5 divided by 3 equals 1.67
5 divided by 4 equals 1.25
5 divided by 5 equals 1.0
```
**Question 7:** Implement a decision tree-based program for telling the user’s fortune. Use the flow chart below to write the program. Use the variables below, rather than writing out all the questions and answers – this should save time. The program asks the user a series of questions (a subset of quest1 through quest7). On the basis of the answers to these questions, the program should print out one of the fortunes provided (fortune8 through fortune12).

Note that the questions and fortunes in the flowchart are numbered, e.g., there is a 1 under the question corresponding to the variable *quest1*. This should help you to quickly associate these boxes with the variable names.

```python
quest1 = "What is your favorite color? Or type "None" if you have no favorite color."'
quest2 = 'Do you have a pet?'
quest3 = 'Do you wear glasses?'
quest4 = 'Do you own blue socks?'
quest5 = 'Is today Monday?'
quest6 = 'Is today Tuesday?'
quest7 = 'Is today Wednesday?'

fortune8 = 'You will see a sign.'
fortune9 = 'You will find something.'
fortune10 = "You will learn something that will change your life a little bit"
fortune11 = 'Your fortune will be told.'
fortune12 = 'You will take a test.'
```
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 multiply two numbers or print a string some number of times.

- The operator ** will represent exponents, e.g., 5**2 == 25

2. Division and Modulus

- 5 // 2 == 2
- 5/2 == 2.5
- 5%2 == 1

3. More Math

- round(4.5) == 4, round(4.6) == 5, round(-4.5) == -4, round(-4.6) == -5
- round(2/3,2) == .67 ## The second argument of round indicates number of decimal places. The default is to round to the nearest whole number.

- math.floor(4.9) == 4, math.floor(-4.9) == 5
- math.ceil(4.9) == 5, math.ceil(-4.9) == -4
- math.trunc(4.9) == 4, math.trunc(-4.9) == -4
- math.pi == 3.141592653589793 – a variable for the value of pi

4. 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)

- 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', '!']

5. print

- *sep* – separator between items
- *end* – printed at the end of print statement

6. 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.

7. 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.

8. logical operators

- *X and Y* returns True only if both *X* and *Y* are True
- *X or Y* returns True only if *X* is True, *Y* is True or both are True
- *X in Sequence* returns True if *X* is in a member of a sequence, e.g., *'a' in 'abcdefg'* would return True
- *X == Y* returns True if *X* and *Y* are the same
- *X != Y* returns True if *X* and *Y* are the different
- *X < Y* returns True if *X* is less than *Y*
- *X > Y* returns True if *X* is greater than *Y*
- *X <= Y* returns True if *X* is less than or equal to *Y*
- *X >= Y* returns True if *X* is greater than or equal to *Y*
- *Not X* returns True if *X* is False