This booklet contains:

- The questions for Section 2: Please answer these questions in the dotted pages provided at the end of Booklet 1.
- A Glossary of Terms

**Question 5:** Write a function that takes a string as an argument. Loop through the characters in the string and store the vowels and the nonvowels as separate strings. Then print out those strings. The following is an example execution of such a program. Note that vowels include: a, e, i, o and u. All other characters (numbers, consonants, spaces, punctuation, etc.) are nonvowels.

```python
>>> divide_string_into_vowels_and_nonvowels('This is a sentence')
The vowels are: iiaeee
The nonvowels are: Ths s sntnc
```

**Question 6:** Write a program that prints a slanted shape, as in the example below. Lines can contain some number of spaces, followed by some number of Xs. The first line should contain no spaces and one X, the second line should contain one space followed by 2 Xs, the third line should contain 2 spaces and 3 Xs and so on, up to a maximum number of Xs. The function should take the maximum number of Xs as an argument. In the example below, the function is called with the parameter set to 10.

```python
>>> draw_double_slant(10)
x
xx
   XXX
      XXXX
             XXXXX
                    XXXXXX
                           XXXXXXXX
                                  XXXXXXXXXXX
                                         XXXXXXXXXXXX
```

```python
>>> draw_double_slant(10)
```
Question 7: Write an implementation of the decision tree program in the flow chart on the next page. You can assume that the following global variables have already been defined. So you can refer to the variable names rather than the strings that they represent.

```python
lap_question1 = '''What will be the primary purpose of your computer?:
1) surfing the internet; 2) making a fashion statement; 3) running standard programs;
4) ease of use; 5) writing software?; 6) playing games; 7) None of the Above'''

lap_question2 = 'Is it worth a few hundred dollars to double battery life?'

lap_question3 = 'Is running standard programs more important than ease of use?'

lap_question4 = 'Is the availability of open source software very important?'

lap_question5 = 'Is low price very important?'

lap_question6 = 'Is it OK, if you can only run programs in a web browser?'
```

This is a sample of what it would look like to run the program.

```python
>>> buy_a_laptop()
What will be the primary purpose of your computer?:
1) surfing the internet; 2) making a fashion statement; 3) running standard programs;
4) ease of use; 5) writing software?; 6) playing games; 7) None of the Above?
Is it worth a few hundred dollars to double battery life?
Yes or No? no
Is running standard programs more important than ease of use?
Yes or No? yes
Is the availability of open source software very important?
Yes or No? no
Is low price very important?
Yes or No? yes
Is it OK, if you can only run programs in a web browser?
Yes or No? no
You should buy a Windows computer
```
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.
- 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.
   - Indices: an index from 0 to (sequence length) represent positions before and after items in a sequence. For example, in the string duck, the subscripts indicate indexed positions: 0d1u2c3d4. In square brackets, and index can be used to identify the character that follows that idnex, e.g., the expression duck[2] would evaluate to be c, the character after index position 2.
   - 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', '!']
     - string.lower() will return an all-lowercase version of a string, e.g., 'CaT'.lower() is equivalent to 'cat'. In the same way, string.upper() converts to uppercase, e.g., 'CaT'.upper() is equivalent to CAT.

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 break. After a break, the statement after the loop executes.
     - 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. while loops
   - First line While (BOOLEAN-EXPRESSION):
     - The loop keeps executing the indented body until BOOLEAN-EXPRESSION evaluates as False.
     - If BOOLEAN-EXPRESSION is always True, the loop is endless.
     - Typically BOOLEAN-EXPRESSION contains one or more variable(s), such that some values of these variables (or this variable) cause BOOLEAN-EXPRESSION to evaluate as True and other values cause it to evaluate as False.
     - The body of the loop can change these variables. The loop starts when BOOLEAN-EXPRESSION is True. Then, under most circumstances, BOOLEAN-EXPRESSION eventually evaluates as False and the loop halts. For example, if the BOOLEAN-EXPRESSION is stop == False, setting stop to True inside the loop, will cause the loop to finish.
     - It is common to use accumulator variables in a similar manner as with for loops.
     - break and return behave the same way for while loops as they do with for loops.
8. **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.

9. **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
- **Not X** returns True if X is False