Practical Exercises

1. Create a script called MyHouse.py. This script should display the following pattern (or similar) to screen:

```
  X
 XXX
XXXXX
|   |
|   |
-----
```

2. Create a script called Output4Values.py. Ask the user to input 4 different string values. Your program needs to store the values in 4 different variables. Output these values to screen using a single print statement. The print statement needs to do the following:

   - Separate the values using a semi-colon.
   - Automatically add the statement ". Output complete!" to the end of the print() output.
   - Not move the cursor to the following line after performing the output.

3. Create a script called CalculateVolume.py. An engineer wants to calculate the volume of a rectangular tank, using the formula length *width* height. The program needs to request these 3 values as input from the user and store them in 3 different variables. Using these 3 variable, calculate the volume of the tank and store it in a fourth variable. The engineer has learned from experience that measurements and calculations are always a bit short of the volume the tank can store, because of the way the plastic tank expands when filled, so he requires that the script increase the calculated volume of the tank by 1% before displaying the resulting volume to the user.

4. Create a script called TimeCalculation.py. A manager has requested that you write a script that allows him to enter a value representing the number of minutes one of his employees has worked in a month. He wants the script to use the number of minutes to calculate the number of days worked in the month, the number of hours left over (not adding up to a full working day) and the number of minutes left over (not adding up to a full hour). For the sake of this calculation, a working day consists of 8 hours. Once calculated, display the resulting calculation in the following format: working days:hours:minutes.

5. Create a script called PizzaShop.py. The owner of the pizza shop has asked you to write a script that allows a customer to calculate the cost of a pizza. He has asked you to make the following options available to the customer:

   - **Pizza base (Customer must choose 1):** Thick (Kr 10) or Thin (Kr 5)
   - **Pizza size (Custumer must choose 1):** Small (Kr 30), Medium (Kr 40) or Large (Kr 50)
   - **Basic sauce (Customer must choose 1):** Tomato (Kr 5) or Barbecue (Kr 10)
   - **Toppings (Customer may choose any or none):** Cheese (Kr 5), Mushrooms (Kr 3), Avocado (Kr 7), Pineapple (Kr 5), Bacon (Kr 7), Chicken (Kr 7) or Fish (Kr 6).

If a specific item is selected display the item and its price on the screen. At the bottom (the last line) of the display present the customer with the total of the purchase.

6. Create a script called ListTest.py. The script is required to do the following:
    a. Define a list
    b. Request 10 names of friends from the user and save them in the list.
    c. Sort the list.
    d. Ask the user for a name to search for.
    e. If the name exists in the list, display its index otherwise display the message “The name does not exist”.
    f. Reverse the list.
    g. Create a slice of the list consisting of the first half of the list entries.
    h. Display the contents of the slice.
    i. Ask the user for one more name.
    j. Append the name to the end of the original list.
    k. Display the contents of the original list.
7. Create a script called SetTest.py. The script is required to do the following:
    a. Define a set
    b. Populate the set with the names of your 5 favourite fast foods (no user input required).
    c. Define a second set and populate it with the names of a friend’s 5 favourite fast foods (no user input required).
    d. Add another type of fast food to your set of favourite foods.
    e. Determine which favourite fast foods are shared by you and your friend and display these overlapped foods.
8. Create a script called MyPhonebook.py The script is required to do the following:
    a. Define a dictionary consisting of 5 business names and their associated phone numbers. The business name should be used as the key.
    b. Request the name and number of another business from the user and add it to the dictionary.
    c. Ask the user to type in the name of a business. If the business exists, display it’s phone number; otherwise display the message “The business is not in the phonebook.”.
    d. Display the entire dictionary (names and phone numbers).

e. Display only the business names (no phone numbers).

9. Create a script called CalculateTotal.py. The script should continuously ask a user to enter an integer value. This should continue until the user has entered the value -1. Once the loop has ended print the total of all the values entered by the user (excluding -1).

10. Create a script called PrintPerfect.py. Create a script that requests a maximum integer from the user. The script should then calculate every perfect number from 1 to the maximum value entered by the user. A number is perfect if it is equal to the sum of all the smaller integers that divide equally into it. 6 is a perfect number, because 1, 2 and 3 all divide equally into it and 1 + 2 + 3 = 6.

11. Create a script called PostalCodes.py. Create a data structure able to store 10 postal codes with their associated residential areas. Prompt the user to enter a postal code. If the postal code exists, output the residential area linked to the postal code.

12. Create a script called SalesCalculator.py. The script should request 10 sales values from the user using a loop. If the value entered by the user is less than 5000 the rest of the current loop iteration should not be performed. If the value is higher than 5000 and less than 10001 the current sales total and sales average of the user should be printed to screen. If the value is higher than 10000, the loop should stop entirely and a message should be displayed that a possible data entry error has occurred.

13. Create a script called MetricConverter.py. In the script, create a function called metric_converter, which receives a single numeric argument. The purpose of this method is to receive a value in inches and then convert it to centimetres. The function should display on screen what the number of inches were that it received, as well as the number of centimetres after conversion. The function should not return any values. Demonstrate the use of the function in the main section of your script.

14. Create a script called SummingMachine.py. In the script, create a function called summing_machine. The function should receive no argument. It should continously ask the user to enter a number until they enter a value to stop, e.g. 's'. The values entered must be added together and the final result returned to the calling code. Demonstrate the use of the function in the main section of your script.

15. Create a script called CalculateAreaOfShape.py. In the script, create a function called calculate_area_of_shape. The function should receive 5 arguments. The first argument should define what shape's area needs to be calculated, e.g. "Square". The other 4 arguments should be used to send measurements for length, height, width and radius. These 4 arguments should default to the value 0 if not used. Your function should be able to calculate the area of at least 3 different shapes, e.g. square, cube and circle. In the main section of your script, demonstrate the calculation of the area of 3 shapes. Your function calls should make use of named arguments.

16. Create a script called GenerateLetter.py. In the script, create a function called generate_letter which is to serve as a generator function. The generator should return a letter in the range a to e. If the generator returns the letter *e*, it should return the letter *a* the next time the generator function object is used. Demonstrate the use of the generator in the main portion of your script by display the letters a to e twice.