REPUBLIC OF THE PHILIPPINES
POLYTECHNIC UNIVERSITY OF THE PHILIPPINES
STA. MESA, MANILA
COLLEGE OF ENGINEERING
ELECTRONICS ENGINEERING DEPARTMENT

CMPE 20022:
COMPUTER PROGRAMMING

INSTRUCTIONAL MATERIAL

MODULE 5 – LAB ACTIVITY

Module 5

Laboratory Activity # 1

Objectives

- improving the student's skills in defining functions;
- using exceptions in order to provide a safe input environment.

Scenario

Your task is to write a **function able to input integer values and to check if they are within a specified range**.

The function should:

- accept three arguments: a prompt, a low acceptable limit, and a high acceptable limit;
- if the user enters a string that is not an integer value, the function should emit the message `Error: wrong input`, and ask the user to input the value again;
- if the user enters a number which falls outside the specified range, the function should emit the message `Error: the value is not within permitted range (min..max)` and ask the user to input the value again;
- if the input value is valid, return it as a result.

```
1  def readint(prompt, min, max):
2  #
3  # put your code here
4  #
5
6  v = readint("Enter a number from -10 to 10: ", -10, 10)
7
8  print("The number is:", v)
```

Test data

Test your code carefully.

This is how the function should react to the user's input:

```
Enter a number from -10 to 10: 100
Error: the value is not within permitted range (-10..10)
Enter a number from -10 to 10: asd
Error: wrong input
Enter number from -10 to 10: 1
The number is: 1
```

Laboratory Activity # 2

Objectives

- improving the student's skills in operating with strings;
- using built-in Python string methods.

Scenario

You already know how `split()` works. Now we want you to prove it.

```
1  def mysplit(strng):
2  #
3  # put your code here
4  #
5
6  print(mysplit("To be or not to be, that is the question"))
7  print(mysplit("To be or not to be,that is the question"))
8  print(mysplit("   "))
9  print(mysplit(" abc "))
10 print(mysplit(""))
```

Your task is to **write your own function, which behaves almost exactly like the original** `split()` **method**, i.e.:

- it should accept exactly one argument - a string;
- it should return a list of words created from the string, divided in the places where the string contains whitespaces;
- if the string is empty, the function should return an empty list;
- its name should be `mysplit()`

Use the template in the editor. Test your code carefully.

Expected output

```
['To', 'be', 'or', 'not', 'to', 'be,', 'that', 'is', 'the', 'questic
['To', 'be', 'or', 'not', 'to', 'be,that', 'is', 'the', 'question']
[]
['abc']
[]
```

Laboratory Activity # 3

Objectives

- improving the student's skills in operating with strings;
- using strings to represent non-text data.

Scenario

You've surely seen a *seven-segment display*.

It's a device (sometimes electronic, sometimes mechanical) designed to present one decimal digit using a subset of seven segments. If you still don't know what it is, refer to the following Wikipedia article.

Your task is to write **a program which is able to simulate the work of a seven-display device**, although you're going to use single LEDs instead of segments.

Each digit is constructed from 13 LEDs (some lit, some dark, of course) - that's how we imagine it:

```
  # ### ### # # ### ### ### ### ### ###
  #   #   # # # #   #     # # # # # # #
  # ### ### ### ### ###   # ### ### # #
  # #     #   #   # # #   # # #   # # #
  # ### ###   # ### ###   # ### ### ###
```

Note: the number 8 shows all the LED lights on.

Your code has to *display* any non-negative integer number entered by the user.

Tip: using a list containing patterns of all ten digits may be very helpful.

Test data

Sample input:

```
123
```

Sample output:

```
  # ### ###
  #   #   #
  # ### ###
  # #     #
  # ### ###
```

Sample input:

```
9081726354
```

Sample output:

```
### ### ###   # ### ### ### ### ### # #
# # # # # #   #   #   # #     # #   # #
### # # ###   #   # ### ### ### ### ###
  # # # # #   #   # #   # #   #   #   #
### ### ###   #   # ### ### ### ###   #
```

Laboratory Activity # 4

Objectives

- improving the student's skills in operating with strings;
- converting characters into ASCII code, and vice versa.

Scenario

You are already familiar with the Caesar cipher, and this is why we want you to improve the code we showed you recently.

The original Caesar cipher shifts each character by one: *a* becomes *b*, *z* becomes *a*, and so on. Let's make it a bit harder, and allow the shifted value to come from the range 1..25 inclusive.

Moreover, let the code preserve the letters' case (lower-case letters will remain lower-case) and all non-alphabetical characters should remain untouched.

Your task is to write a program which:

- asks the user for one line of text to encrypt;
- asks the user for a shift value (an integer number from the range 1..25 - note: you should force the user to enter a valid shift value (don't give up and don't let bad data fool you!)
- prints out the encoded text.

Test your code using the data we've provided.

Test data

Sample input:

```
abcxyzABCxyz 123
2
```

Sample output:

```
cdezabCDEzab 123
```

Sample input:

```
The die is cast
25
```

Sample output:

```
Sgd chd hr bzrs
```

Laboratory Activity # 5

Objectives

- improving the student's skills in operating with strings;
- encouraging the student to look for non-obvious solutions.

Scenario

Do you know what a palindrome is?

It's a word which look the same when read forward and backward. For example, "kayak" is a palindrome, while "loyal" is not.

Your task is to write a program which:

- asks the user for some text;
- checks whether the entered text is a palindrome, and prints result.

Note:

- assume that an empty string isn't a palindrome;
- treat upper- and lower-case letters as equal;
- spaces are not taken into account during the check - treat them as non-existent;
- there are more than a few correct solutions - try to find more than one.

Test your code using the data we've provided.

Test data

Sample input:

```
Ten animals I slam in a net
```

It's a palindrome

Sample input:

```
Eleven animals I slam in a net
```

It's not a palindrome

Laboratory Activity # 6

Objectives

- improving the student's skills in operating with strings;
- converting strings into lists, and vice versa.

Scenario

An anagram is a new word formed by rearranging the letters of a word, using all the original letters exactly once. For example, the phrases "rail safety" and "fairy tales" are anagrams, while "I am" and "You are" are not.

Your task is to write a program which:

- asks the user for two separate texts;
- checks whether, the entered texts are anagrams and prints the result.

Note:

- assume that two empty strings are not anagrams;
- treat upper- and lower-case letters as equal;
- spaces are not taken into account during the check - treat them as non-existent

Test your code using the data we've provided.

Test data

Sample input:

```
Listen
Silent
```

Anagrams

Sample input:

```
modern
norman
```

Not anagrams

Laboratory Activity # 7

Objectives

- improving the student's skills in operating with strings;
- converting integers into strings, and vice versa.

Scenario

Some say that the *Digit of Life* is a digit evaluated using somebody's birthday. It's simple - you just need to sum all the digits of the date. If the result contains more than one digit, you have to repeat the addition until you get exactly one digit. For example:

- 1 January 2017 = 2017 01 01
- 2 + 0 + 1 + 7 + 0 + 1 + 0 + 1 = 12
- 1 + 2 = 3

3 is the digit we searched for and found.

Your task is to write a program which:

- asks the user her/his birthday (in the format YYYYMMDD, or YYYYDDMM, or MMDDYYYY - actually, the order of the digits doesn't matter)
- outputs the *Digit of Life* for the date.

Test your code using the data we've provided.

Test data

Sample input:

```
19991229
```

Sample output:

```
6
```

Sample input:

```
20000101
```

Sample output:

```
4
```

Laboratory Activity # 8

Objectives

- improving the student's skills in operating with strings;
- using the `find()` method for searching strings.

Scenario

Let's play a game. We will give you two strings: one being a word (e.g., "dog") and the second being a combination of any characters.

Your task is to write a program which answers the following question: **are the characters comprising the first string hidden inside the second string?**

For example:

- if the second string is given as `"vcxzxduybfdsobywuefgas"`, the answer is `yes`;
- if the second string is `"vcxzxdcybfdstbywuefsas"`, the answer is `no` (as there are neither the letters "d", "o", or "g", in this order)

Hints:

- you should use the two-argument variants of the `pos()` functions inside your code;
- don't worry about case sensitivity.

Test your code using the data we've provided.

Test data

Sample input:

```
donor
Nabucodonosor
```

Sample output:

```
Yes
```

Sample input:

```
donut
Nabucodonosor
```

Sample output:

```
No
```

Laboratory Activity # 9

Objectives

- improving the student's skills in operating with strings and lists;
- converting strings into lists.

Scenario

As you probably know, *Sudoku* is a number-placing puzzle played on a 9x9 board. The player has to fill the board in a very specific way:

- each row of the board must contain all digits from 0 to 9 (the order doesn't matter)
- each column of the board must contain all digits from 0 to 9 (again, the order doesn't matter)
- each of the nine 3x3 "tiles" (we will name them "sub-squares") of the table must contain all digits from 0 to 9.

Your task is to write a program which:

- reads 9 rows of the Sudoku, each containing 9 digits (check carefully if the data entered are valid)
- outputs `Yes` if the Sudoku is valid, and `No` otherwise.

Test your code using the data we've provided.

Test data

Sample input:

```
295743861
431865927
876192543
387459216
612387495
549216738
763524189
928671354
154938672
```

Sample output:

```
Yes
```