PROBLEMS - Lesson 5: Functions

Problems in yellow → Assessable problems (evaluable)

The rest of the problems are of intermediate difficulty. If you have any difficulty moving from one recommended problem to the next one, do some of the problems that appear in between for more progressive learning.

Problems in blue → Optional problems (not assessable but available at VPL for autoevaluation).

Problem 1

We can calculate how many different teams a basketball coach can make with 10 players as:

$$\frac{n!}{r!(n-r)!} = \binom{n}{r}$$

n = 10 number of players in the squad
r = 5 number of players you can line up

But now, we want to make a program that can be used for any sport. For this, write a function that calculates the factorial of a number, received as a parameter.

Then make the main program to ask the user the values of **n** (number of players) and **r** (number of players he can line up in the sport). Make the necessary calls to the function that calculates the factorial to calculate the number of combinations of **n** players, selected in groups of **r** that can be made. Finally, the program will display the message: "The number of teams that can be formed is XX" where XX is the number of combinations.

Problem 2

We want to improve the program that asked the user to enter a time in HH:MM:SS format and increment a second. To avoid code repetitions and follow the principles of modular programming, implement two functions:

- twoFigures
- incrementSecond

The *twoFigures* function will receive an integer as a parameter and it will check if it is lower than 10. If so, it will return a string with a zero at the first position, whereas the number passed as a parameter will be at the second position. Otherwise, it will return a string with the number passed as a parameter (without any zero at the beginning).

The *incrementSecond* function will receive three integers as parameters, corresponding to the values of hours, minutes, and seconds (in that order) and will perform the increment of one second, and return the new hour, minute, and second values.

Use these functions in the main program to compute the new time and print it.

Problem 3

Perform a function called *Alert*, that receives and in integer as a parameter and returns another integer. The function must write the message "Alert: There are XX seconds left" where XX is the integer passed as a parameter. In case the parameter is equal to 0, the message must be "Alert: Time is up". The function will always return a 0.

Make the main program where an integer will be requested (number of seconds) and perform a countdown. Each time it decreases a second, the *Alert* function must be called.

Use:

- *from time import sleep*
- *sleep(t)* function, where t is the number of seconds to pause the execution.

Problem 4
Write a function called *factorial* that receives an integer as a parameter and returns the factorial of that number. Then, write a function called *summation* that receives an integer as a parameter and returns the summation of that number. Finally, write a function called *Timer* to know how long it takes to execute a function. The *Timer* function will receive, as parameters, the function name and its parameters.

In the main program, call the *Timer* function to know the execution time of the factorial and summation functions. Use:

- *from time import time*
- *time()* function, which returns the computer time in seconds as a float number

Problem 5
Do a function that adds two numbers. The function will receive two real numbers as parameters, and it will return the result of their summation.

From the main program, ask the user to enter two real numbers and call the function to compute the result of the sum. Finally, it will print the result.

Problem 6
Write a function that receives as parameters a string, corresponding to a person's name, and the current year. The function will ask the year of birth ("Mr. XX, which is your year of birth?", where XX is the person's name), then it will calculate the age and return it.

Make a main program that defines the current year as a constant, asks the user's name, calls the function to calculate the user's age, and displays the person's name and age.

Problem 7
Perform a function that returns the hundreds of a number. The function will receive an integer as a parameter and return the value of the hundreds. (Ex. If the parameter is 21378, the function will return 3).

Then, in the main program, ask the user to enter an integer, and then call the function. Finally, print the result with the message "Hundreds of the number XX are CC" where XX is the number entered by the user and CC is the hundreds of that number.

Problem 8
Write a function that returns an integer read via keyboard, which must be between two limits (lower and upper) that are passed as parameters. The function must ask for numbers via keyboard until the input number is within the limits.

The main program will ask the user to enter the values of the two limits and call the function. The function will return the value between the two limits, and the main program will display: "Value between II and SS: XX" where II and SS are the lower and upper limits, and XX is the value in between the limits returned by the function.

Problem 9
Write a function that computes the power of a real number *x*, where the power *n* must be an integer. The function receives *x* and *n* as parameters and calculates x^n, using only multiplications. Note that the exponent *n* can be negative.

This function must be called from a main program, which will first ask the user to enter the values of *x* and *n*, then, it will call the function, and it will display the result with the message "The result of XX to the power of NN is YY", where XX and NN will be the number x and the exponent n, and YY will be the value of x^n.

Problem 10
Make a function that reads a maximum of N numbers via keyboard and checks if they are prime. The value of N will be passed to the function as a parameter. The data entry must stop either when a prime number is detected or when the N numbers have been entered. The function will return a 1 if there is any prime number, 0 otherwise.

Make a main program that reads the value N and calls the function. The result must be displayed with "The function has detected a prime number" if the function returns a 1 or "The function has not detected any prime number" if the function has returned a 0.

Problem 11
Write a function to calculate the division (quotient only) between two integers, using only the + (addition) and - (subtraction) functions. You must consider the sign of the operands.

Make a main program read two integers (dividend and divisor). If the divisor is nonzero, call the function and display the quotient with a message "XX: YY = QQ" where XX is the dividend, YY is the divisor and QQ is the quotient calculated by the function. If the divisor is zero, the program will display an error message: "Error: Division by zero."

Note: Consider using the *abs(X)* function to calculate the absolute value of a number X.

Problem 12
Make a function that receives an integer N as a parameter and checks if N is an exact power of 2 (for example, 32 is power of 2, but not 24). If the value is an exact power of 2 the function will return 1 and otherwise it will return -1.

Then make a main program that reads the value of N and calls the function. If the function returns that it is an exact power of 2, the message "NN is an exact power of 2" should be displayed, where NN is the number entered by the user. Otherwise, the message "NN is not an exact power of 2" should be displayed.

Problem 13
Perform a function to calculate the term *n* of the Fibonacci series, defined by:

$$a_0 = 0$$
$$a_1 = 1$$
$$a_n = a_{n-1} + a_{n-2}$$

The function will receive *n* as a parameter and return the value of the term a_n.

Then make the main program ask which term of the Fibonacci series must calculate. If the user enters a negative term, it will print "Error: Number cannot be negative", and it will ask the user to enter the term again. This process will be repeated until a non-negative integer is introduced. Then, the Fibonacci function will be called. The result will be displayed with a message: "The term NN of the Fibonacci series is XX" where NN is the number n entered by the user and XX is the term a_n computed by the function.

Problem 14
Make a function that receives a value N as a parameter. The function will read N positive values via keyboard and return how many values greater than 10 have been entered. Each time a value is entered it will check that the value is positive value, otherwise, it will print "Error: Negative value" and it will ask the user to enter the value again.

In the main program, the user will introduce the value N, and the function will be called. The output of the function will be printed with a message: "The user has entered XX values above 10" where XX will be the value returned by the function.

Problem 15
Retrieve the code from the multi-function calculator from previous lessons. Write a procedure to print the menu. Then, from the main program, this function will be called to print the menu when appropriate. The rest of the calculator code will be exactly the same as before.

Problem 16
Perform a procedure that prints all the divisors (when dividing, the remainder is 0) of a number that is passed as a parameter. The divisors will be displayed within the same line, separated by a space between them. Also make the main program that asks the user to enter an integer value and call this function by passing this value as a parameter.

Problem 17
Perform a procedure that receives two integer values as parameters and writes all the powers of the first number that are lower than the second number. When printing, these values must be separated by a space. For example, if the function receives 3 and 90 as parameters, the procedure should display the values: 3 9 27 81.

The main program should ask the user to enter two integer values, N and L, and should call the above procedure to print those powers of N that are lower than the value L. Check that N is greater than 1 and that L is greater than N. Only if these two conditions are met, the function will be called. Otherwise, print the error message: "Error: Incorrect values".

Problem 18
Do a procedure that given an integer *n* other than 1, write all integers between 1 and *n* that are multiples of 3 or 7 but not 2 at once. The case where the number *n* is negative should be considered, and it should work equally well. For example, if the number -10 is entered, the program should display the following result: -3, -6. -7 and -9.

Make a program that uses this feature. The user must first be prompted to enter an integer value other than 1, 0, and -1. It is necessary to check that the integer entered by the user is correct and if it is not, it is necessary to display an error message and ask for the number again. This process must be repeated until the user enters a correct value. The program will then call the function with the value provided by the user.

Problem 19
Write a function called *GetSecretNumber* that does the same as the code in **bold**. Modify the program to call this function, so that we can remove the code in bold.

```
y = 5  # a secret number that the user does not know

x = int (input ("Enter a number between 1 and 10:"))
while (x <1) or (x> 10):
   x = int (input ("Enter a number between 1 and 10:"))

if x == y:
   print ("You discovered the secret number!!!")
```

Then use the *randint(a,b)* function to provide a random integer between a and b (both inclusive). In this way, we can generate a new random value y between 1 and 10 for each execution of the program. ***Note***: To use the *randint* function, include this line at the beginning of your program: *from random import randint*

Finally, modify the game as follows:

- The user can play as many times as he/she wants
- In each new game the user defines the number of chances to guess the number.
- The secret number can be between 1 and k, where k is defined by the user for each new game.

Problem 20

Write a function that calculates the area of a square. The function will receive the length of the square side as a parameter, and it will check that the value is positive. In case the condition is met it will calculate the result and the function will return 0. Otherwise, the operation will not be calculated, and the function will return 1.

Make the main program that asks the user to enter the value of the square side. The program will then call the function. If the area could be calculated, it will display the message "The area of a square of side NN is RR" where NN will be the number entered by the user and RR will be the result calculated by the function. If the result could not be calculated, the message "Error: The square of the side is negative" will be displayed.

Problem 21

Make a function that calculates the division (quotient and remainder) between two integers, using only the functions + (addition) and - (subtraction). Consider the sign of the operands. If the divisor is zero, the function must return 1 and, if not, return 0.

Make a main program that reads two integers, calls the function, and outputs the result with the message: “Result - Quotient: QQ and Residue: RR” where QQ and RR are the values calculated by the function. If the function returned an error value (1), the program will output “Error: Division by Zero”, so the division could not be calculated.

Note: Consider using the *abs(X)* function to calculate the absolute value of a number X. Remember that 'None' is used to return indefinite values.

Problem 22

Write a function that calculates the area and perimeter of a circle. The radius (real number) is passed as parameter, and the area and perimeter are returned as real numbers. The function must check that the value of the radius is greater than zero. If so, it will calculate and return the area and perimeter, and also return a value of 0. Otherwise, no calculation will be made, and the function will return a 1.

Make a main program that reads the radius of the circle and calls the function. If the function returned the area and perimeter, it will display them on the screen. If the function returned 1, the error message "Error: Radius must be a positive value" will be displayed.

Note: Remember that 'None' is used to return indefinite values.

Problem 23

Perform a function that reads positive and negative integers via keyboard until a zero is entered or until 10 values are read. The function must calculate and return the minimum and maximum (in that order) of all read values. If the first number is zero, the function should return a 1 as the error code. If one or more non-zero values are entered (the maximum and minimum can be calculated) the function will return a 0 as the return value.

Then make the main program that uses this function to calculate the minimum and maximum of the sequence of numbers. If the function could calculate the minimum and maximum values, these will be displayed with a message: "The minimum of the sequence is XX and the maximum is YY" where XX and YY will be the values calculated by the function. If the function could not calculate the minimum and maximum because the user entered only one zero, the message: "Error: empty sequence" will be displayed. Note: 'None' is used to return indefinite values.

Problem 24
Write a function called "polars" that receive two float values, corresponding to the x and y coordinates of a point, and return the modulus and angle (in that order) of the point.

Make the main program that reads the coordinates of the point (x, y), calls the function, and prints the result obtained.

Note: module = $\sqrt{x^2 + y^2}$ (use *sqrt()*) and angle = $\tan^{-1}\frac{y}{x}$ (use *atan()*)

Problem 25
Write a function that computer and returns the slope (a) and the independent term (b) of a linear function $y = ax + b$ that receives, as parameters, the two coordinates of the line (x_1, x_2, y_1, y_2).

Make the main program that reads the coordinates (x_1, x_2, y_1, y_2), calls the function, and displays the result obtained.

Note: $a = \frac{y_2-y_1}{x_2-x_1}$ and $b = y_1 - \frac{y_2-y_1}{x_2-x_1}x_1$

Problem 26

Write a function called "equacio" to solve quadratic equations: $ax^2 + bx + c = 0$

The function has three parameters: the coefficients *a, b, c* (integers) and it will return a numeric code to indicate the number of roots of the equation and the two possible roots (real numbers):

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

The numeric code with the number of roots may have values:

- 0 if b^2-$4ac < 0$
- 1 if b^2-$4ac = 0$
- 2 if b^2-$4ac > 0$

Make a main program that reads the values a,b,c, calls the function, and prints the result.

Note: To calculate the square root use the *sqrt()* function of the math library. For doing so, add: *from math import sqrt*. Remember that 'None' is used to return indefinite values.

Problem 27
The following functions calculate the area of a square, a rectangle, and a triangle:

```
def square_area (side):
    if side> 0:
        area = side * side
        return 0, area
    else:
        return 1, None

def area_rectangle (base, height):
    if (base> 0) and (height> 0):
        area = base * height
        return 0, area
    else:
        return 1, None
```

```
def area_triangle (base, height):
     if (base> 0) and (height> 0):
          area = base * height / 2
          return 0, area
     else:
          return 1, None
```

The following procedure prints the selection menu that allows you to choose between the different options to calculate the area of different geometric figures:

```
def menu_selection ():
    print ("----- MENU ----- \ n")
    print ("1.-Area of Square")
    print ("2.-Area of Rectangle")
    print ("3.-Area of Triangle")
    print ("4.-Exit \ n")
    print ("Choose one of the options:")
```

Using the above functions and procedure, make a main program that shows the selection menu and asks the user to choose one of the options. Depending on the option chosen, the program must ask the user the data (base, height, etc.) that is needed to calculate the area. Then, the corresponding function will be called. With the value returned by the function, the message “Area: AA” should be displayed, where AA is the area of the selected figure. If the area could be calculated, print the message “Error: Incorrect dimensions”.

The process must be repeated until the user chooses the exit option. In case of selecting a non-existent option, the program should display the message "Error: Incorrect option".

Problem 28
Define a function that allows to modify the position in the plane of a point (x, y) in a given direction, but within a box limited by the coordinates (x_min, x_max, y_min, y_max). The coordinates of the point, the direction of the motion, and the boundaries of the box are integer values that are passed as parameters to the function. We will pass the direction of movement as an integer: 1 = move to the left, 2 = move to the right, 3 = move down, 4 = move up. The function must check that the movement to be made does not place the point outside the square. In such a case, the coordinates of the point will not change.

Use this function in a main program that initially asks the user for the boundaries of a square (x_min, x_max, y_min, y_max) and checks that the boundaries of the square are correct, which means that (x_min <x_max and y_min <y_max). If it is not the case, the program will ask for the limits until they are correct.

The program will then ask the user to enter the coordinates of a point (x, y) located inside the square. It will be necessary to check that the point is within the limits of the square. Otherwise, the program will ask for the coordinates of the point again, until they are correct. Then the program will read the direction of movement (without showing any message) and only if it is a valid value (1, 2, 3, 4), it will call the function to compute the new position of the point.

Once the function has been executed, the program will print the new point coordinates with the format "(X, Y)".

The program should read directions of movement, calling the function and printing the new point until the user enters the value 5 as the direction of movement, which means Exit.

Problem 29. Given this algorithm:

```
def factorial(n):
     total=1
     while n>1:
          total=total*n
          n-=1
     return total

def summation(n):
    total=0
    while n>=1:
        total=total+n
        n-=1
    return total

def calculate(n):
    fac=factorial(n)
    sum=summation(n)
    return fac, sum

n=int(input())
rfac,rsum=calculate(n)
print(n,rfac,rsum)
```

Indicate, using pen and paper, the result that a main algorithm writes on the screen if the user enters a value of n = 5. **This problem does not need any coding, just "tracking" the execution.**

Problem 30
Perform two functions to process lists of integers:

- **ReadList**: This function receives the number of items to read and returns a full list with integer values entered by keyboard.
- **AddLists**: This function receives two lists and returns another list where, in each position, it contains the sum of the elements of the two input lists in that position. This means that `l[i] = l1[i]+l2[i].` Assume that the two lists are of the same size.

Then make a main program that asks the user to enter dimension N of the lists and use the above functions to read two lists of length N, calculate their sum, and display the list with the result.

Problem 31

Fourth-year students want to do a program to manage voting on the destination of the end-of-career trip. The voting options are identified by an integer, and are: 1-Cancun, 2-Cuba, 3-Crete, 4-Croatia, 5-Canarian Islands. The vote (a number from 1 to 5) of each student will be saved in a list of integers. In the fourth year, there are 156 students.

First, do a procedure called *ReadVotes* that reads and stores the votes of all students into the data structure. The procedure will receive a list as a parameter, in which votes will be stored, and it will read each student's vote with the message "Enter student vote n". The procedure must verify that the vote is valid (between 1 and 5, both inclusive), otherwise, the following error message "Invalid vote" will be shown, and the student's vote will be requested again.

Then make a main program where a list is declared to store the votes. Then, the *ReadVotes* procedure is called. Next, the winning option should be determined by searching the maximum value in the voting list. Finally, this option should be displayed on the screen.

Problem 32

Make a small library of generic operations for lists of integers. To do this, you program the functions listed below in a single "lists.py" file:

- The **ReadList** function receives as a parameter the number of items to read and returns a list. This function creates an empty list, asks the user to enter the value of each position in the list via keyboard and stores it in the corresponding position in the list.
- The **InitializeList** function has two parameters: the number of items in the list (its length) and the value that will be used to initialize the list. This function creates a list with this length and initializes each position in the list with the value of the second parameter.
- The **AverageList** function receives as a parameter a list of integers and returns a real value, corresponding to the average of the items in the list.
- The **MaxList** function receives as a parameter a list of integers and returns an integer value, which corresponds to the **position** of the maximum value in the list.
- The **MinList** function receives as a parameter a list of integers and returns an integer value, which corresponds to the **position** of the minimum value in the list.
- The **MinListNonZero** function receives as a parameter a list of integers and returns an integer value, corresponding to the **position** of the minimum value of the items in the list, but disregarding the values that are zero. If the list contains only zeros, it returns -1.

Some of these operations (or very similar ones) have already been codified in previous problems, so you can reuse some code. Once you have created this library, create a *main.py* file with the main program, where these functions are imported and used to check that they properly work.

Note: To import a function from a function library, you must include at the beginning of the program the statement *'from lists import function_name'* where *function_name* is the name of the function we want to import. Once the import is done, we can use the function defined in *'lists.py'* in the program in *'principal.py'*.

Problem 33

Implement problem 27 of the previous lesson (statistics on the average temperatures of the 12 months) following the principles of modular programming, putting each task in a function. For this, use the functions and procedures in Problem 32 (*lists.py*), and replace the code that performs some of the operations in the library with the call to the corresponding function or procedure.

Note: To import a function from a function library, you must include at the beginning of the program the statement '*from lists import function_name*' where *function_name* is the name of the function we want to import.

Problem 34
Make a program that reads and stores in a data structure the maximum temperatures of each day for two weeks (14 days) and lets you know what is the maximum daily temperature that has been repeated more times during this time period. We can assume that the temperature will never be lower than -10º or higher than 50º (integer values).

The program will ask the user the maximum temperature of all the days of the week and will have to construct a table (histogram of temperatures) in which, for each possible value of temperature (from -10 to 50), how many days had that specific temperature throughout the week.

Then the program must calculate which temperature has been repeated more times and which one less times. The program will print the two messages on two different lines:

"Most repeated temperature: XX"
"Less repeated temperature: YY"

where XX and YY will be two integer values between -10 and 50, corresponding to the temperatures with the largest and smallest value (non-zero) of the histogram of temperatures. If there is more than one temperature that is the most/least repeated, take the first temperature found.

Important: Use functions from Problem 32 (lists.py) to perform the necessary operations.

Problem 35
Make the *Detect1000* function that receives a list of integers as a parameter. The function returns 1 if any value in the list is greater than or equal to 1000, and 0 if all values are less than 1000.

Make the main program ask for a list of integers from the user, call the *Detect1000* function and display a message with the returned value to check that the function is properly working.

Problem 36
Write the *DescendingSorting* function that receives a list of integers as a parameter. The function returns 1 if the list is a sequence of numbers, sorted in descending order, and 0 otherwise.

Make the main program to read a list of integers, call the *DescendingSorting* function and display a message with the returned value of the function, to check that the function is properly working.

Problem 37
Perform the *AscendingPairs* function that receives a list of integers as a parameter. The function returns 1 if the list is an increasing sequence of consecutive even numbers, and 0 otherwise.

Make the main program ask the user for a list of integers, call the *AscendingPairs* function, and display a message with the value returned to verify that the function is properly working.

Problem 38
Make the *FindPosition* function that receives as parameters a list of integers sorted in descending order, and an integer value *x*. The function looks at which position in the list the value *x* should be located, that is, the first position in the list that contains a value lower than *x*. The position where *x* should be inserted will be the return value of the function. If the end of the list is reached and no element is lower than the value x, the function will return -1.

Then, make a program that reads a list of 5 integers, where the numbers are sorted in descending order. Then use the *DescendingSorting* function of Problem 36 to verify that the list is correctly sorted. If so, the program will prompt the user to enter a value x and it will call the *FindPosition* function. If the function returns -1 the program will print "All values in the list are greater than XX" where XX will be the value x entered by the user. Otherwise, it will display the message “The value must go at position PP” where PP is the value returned by the function. If the list is not sorted in descending order, the message "The list is not correctly sorted" will be displayed.

Problem 39
Make the *ReverseList* function that receives two lists of integers (with the same length) as parameters. The function returns 1 if the two lists are the same, one read from left to right and the other from right to left, and 0 if they are different.

Also make the main program read a list of integers from the user, call the *ReverseList* function and display a message with the returned value to check that the function is properly working.

Problem 40
Complete the generic operations library with lists of integers (made in Problem 32), with the following functions and procedures:

- **ReadListStop** function has two parameters: a list of integers and a stop value. It asks the user to enter by keyboard the value of each position in the list and stores it. When the stop value is entered, the data entry ends.
- **PositiveList** function has an integer list as a parameter, and returns an integer value. This function counts and returns the number of positive values in the list.
- **NegativesList** function has an integer list as a parameter, and returns an integer value. This function counts and returns the number of negative values in the list.
- **StdevList** function has a list of integers as a parameter, and returns a real value. This function calculates the standard deviation of the items in the list and returns the value as a real number.

Problem 41
Make a program that reads and stores a series of up to 25 integers, ending in 0. That is, the reading process stops when a 0 is entered, or when 25 values have been read. After reading and storing the series, the maximum and minimum, the number of positive and negative values of the series, the mean and the standard deviation must be calculated and displayed on the screen.

For this problem, use the list operations library made in Problems 32 and 40.

Problem 42
Make the *SortList* procedure that takes a list of integers as a parameter and sorts the items in the list from lower to higher.

Also make a program that reads 8 integers (which can be in any order) via keyboard and stores them in a list. Then use the *SortList* procedure to sort the list and display the sorted list on the screen in the format "Sorted list: [X, Y, Z, ...]" where X, Y, Z, etc. will be the values in the list.

Note: To do this, you can use the BubbleSort method:

```
For i between 0 and length(list)
        For j between 0 and length(list)-i-1
             If list[j] > list[j+1]
                  swap list[j] and list[j+1]
```

Problem 43
Perform the *DeleteRepeated* procedure that takes a list of integers (different from zero) as a parameter and removes all repeated items from the list. The first occurrence of each repeated item must be kept in the final list. All operations must be done on the original list.

Also make the main program to read a list of integers, call the *DeleteRepeated* procedure and display the list after deleting the repetitions to check that the procedure correctly works.