CSE4IP – Lab 3

Q1 – Maximum of Two Numbers

Write a program that reads two integers and prints out the maximum without using `min` or `max` function.

Traditionally, this is one of the basic exercises for if statement – especially for programming languages which do not have functions `min`, `max` *readily available. So let us do it as a basic exercise.*

Q2 – Maximum of Three Numbers

Write a program that reads three integers and prints out the maximum of the three without using `min` or `max` function.

Q3 – Displaying Numbers in Order

Write a program that reads three integers and prints them out in increasing order.

Q4 – Counting Digits

Write a program that reads a positive integer and prints out how many digits it has, by checking whether the number is >= 10, >=100, and so on. Assume that the number is less than 1 million.

Strictly speaking, the number can be entered with a plus sign. But we will not be concerned about this, which can be fixed if we want to.

Q5 – Counting Digits Again

Repeat the previous question without using if statements of any kind.

You can use the math function `log10`*. Try it. You can also use a function that you should be quite familiar with. Try this as well.*

Q6 – Nearest Integer Square

Let us define an "integer square" to be an integer which is equal to the square of another integer. For example, 9 is an integer square, but 10 is not.

Write a program that reads a positive integer `n` and prints out `m`, where `m` is the integer square nearest to `n`.

For example, if $n = 10$ then $m = 9$. If $n = 18$, $m = 16$. If $n = 25$, $m = 25$.

Q7 – Number Relations

Write a program that reads three numbers and prints "all the same" if they are all the same, "all different" if they are all different, and "neither" otherwise.

Once you have got a solution, you may want to consider this: Can you use a feature rather unique to Python regarding comparison operators to improve the readability of your program (if you have not already done so)?

Q8 – Are numbers Increasing?

Write a program that reads three numbers and prints "increasing" if they are in increasing order, "decreasing" if they are in decreasing, and "neither" otherwise. We will take "increasing" to mean "strictly increasing", with each value larger than its predecessor. The sequence `3 4 4` would not be consider increasing.

Q9 – Three Sides of a Triangle

Write a program that prompts the user for three numbers and then determines if the numbers can be the three sides of a triangle. For three positive numbers to be the sides of a triangle, none should be bigger than or equal to the sum of the other two.

Q10 – Floating-Point Number String

Write a program that prompts the user for a floating-point number in decimal number format. The program then determines if the string entered by the user actually represents a float value or not.

Q11 – Egg Classification

Eggs that weigh less than 55 grams are graded as small. Eggs in the range 55-65 are graded medium. Eggs over 65 grams are graded large.

Write a program that asks a user to enter the weight of an egg and prints out its grade.

What will happen to your program if the user enters a weight that is 0 or less? This is the issue of input validation. It is an interesting issue in general. We will not worry about it here. We will consider it in a later question.

Note: Actually, the modern standard egg classification in Australia has the following sizes: King-size, Jumbo, Extra-Large, Large and Medium.

Q12 – Employee's Payment

Write a program to calculate weekly payment for an employee working at a store.

Information about the employee includes:

- the employee's name (a string)
- the hourly pay rate (a float)
- the number of hours the employee worked in the week (a float)
- the total sales for the week (a float).

An employee is expected to sell at least $100 worth of goods in each hour they work. If they sell on average more than $100 an hour they receive a bonus. The pay is calculated as the number of hours they worked multiplied by their hourly pay rate plus the bonus if they are entitled to one. The bonus is 10% of the sales amount they made above the expected sales amount based on $100 an hour.

Q13 – Ordering Books

A university bookstore wants a program to determine the number of copies it should order for a book. Experience has shown that sales depend largely on whether the book is required as a prescribed textbook or merely recommended, and whether or not it has been used before. A new, required textbook will sell 90% of prospective enrollment, but if it has been used before 65% will buy. Similarly, 40% will buy a newly recommended books, but just half that many buy a book that has been used before.

Write a program that asks the user for all the information required for a book, including the prospective class enrollment, and prints out how many copies the bookstore should order (assuming there is no left-over stock for the book).

It is useful to approach the problem like this:

1. *Identify the inputs*
2. *Identify the output*
3. *Write down the computation rules*

It is also useful to recognize this as a problem about ***decision tables****. If you know about decision tables, construct one for this problem. Also, you can search the Web about it, if interested.*

Q14 – Input Validation

Consider the example that reads a mark and print out the grade A. B. C, D, etc. Suppose for a mark to be valid it has to be in the range from 0 to 100, inclusive.

One option is to write a decision structure like this:

```
if mark > 100:
   print("Error: The mark must not exceed 100")
elif mark >= 80
   print("Grade: ", A)
...
elif mark >= 0:
   print("Grade: ", N)
else
   print("Error: mark must not be negative")
```

The problem of this style is that it mixes two kinds of logic: the input validation logic (to check if the mark is in the valid range) and the business logic (to classify the grade).

The problem caused by the mixing would rapidly get worse if we have more input values to validate.

A more rational approach is to separate the two tasks. A very clear way to do this is to do the validation as shown below:

```
1  # Check the validation condition first
2  # If input is not valid, terminate the program
3  pre = 0 <= mark <= 100
4  if not pre:
5     print("Error: mark is outside the valid range
       0-100"')
6     import sys
7     sys.exit()
8
9  # At this point, the input is valid
10 # Put statements for business logic here
11 ...
```

A few remarks are in order:

1. On line 3, variable **pre** stands for "precondition". In rigorous approach to programming, precondition is the condition that must be satisfied for the actions to go ahead.

2. Also, on line 3, we compute the precondition by specifying precisely what it is that we need. Hence, it is very clear from the code that the precondition is the mark has to be between 0 and 100, inclusive.

3. Then we have an if statement which says literally *if the precondition is not satisfied, do this and this.*

   And what we choose to do when the precondition is not satisfied is to terminate the program.

   * * *

   As a quick exercise, modify your egg classification program, assuming that the valid range is from 40 grams to 80 grams, inclusive.

■

The choice to terminate the program is a very sensible choice. There is no point in going ahead! (unless we want to provide a loop to allow the user three attempts, say).

It is also very consistent with the standard practice of exception handling, which you will study later. The standard practice is that whenever we detect an abnormal condition, we raise what is known as an exception and we do not continue with our normal flow of actions.

(There are people who disagrees with the approach we take here. Usually these are people who have a rigid view of what is known as structured programming. This rigid view has been shown by research and actual practice to be invalid.)