Program Assignments – While Loops. Develop an IPO for each of the problems below. Place the IPO into your repository. Then write code for each problem and place those files (py) into your repository. Paste the link to your repository into the Assignment Completion Link on Blackboard.

1. Display the odd numbers starting at 1 and ending with 25. Use a while loop structure for this problem.

|  |  |  |
| --- | --- | --- |
| Input | Process | Output |
| None | Initilize number to 1.<br>2. While ‘#’ <= 25:<br> a. Print ‘#’ . <br> b. Increment ‘#’ by 2. | Odd numbers from 1 to 25: 1,3,5,7,9,11,13,15,17,19,21,23,25 |

1. Allow the user to enter a start value, stop value and increment value from the keyboard. Display all the numbers from the start value to stop value using the increment value as you proceed. Use a while loop structure for this problem.

|  |  |  |
| --- | --- | --- |
| Input | Process | Output |
| Start Value,  Stop value,  Increment value | 1.Initialize  ‘current\_value’ to  ‘start’. | # from start to stop without increments |
|  | 2.While  ‘current\_value’ <=  ‘stop’: |  |
|  | a.Print  ‘current\_value’. |  |
|  | b. increment  ‘current\_value’ by  ‘increment’. |  |

1. Prompt the user on whether they want to do this program **(just before the while loop**). “Yes” entry means they want to continue. Any other response indicates they will stop the program. This response is the loop control variable. If the user answers “Yes “then go into the while loop.

Once in the while loop. You are to prompt the user for their last name and two exam scores. Compute the average exam score. Display last name and average. After the loop, display a count of the number of students who entered data.

Finally, the **last statements** **within the while loop** will ask the user if they want to do this loop again. In other words the user needs to be prompted again. The reason is that the end of the loop takes execution to the while condition to be evaluated again. It can not take us to the first few lines of code that prompt the user for the first time. That code is out of the loop. Therefore, we need a second prompt at the bottom, inside the loop.

|  |  |  |
| --- | --- | --- |
| Input | Process | Output |
| ‘start\_program’ (User/ input: Yes/No) | Controls the start of the program | Show student’s last name and average exam score |
|  | Intitates or terminates the while loop. | Total # of students who entered data |
| ‘Last\_name’ (user input: String) | Prompt user to enter the student’s last name. |  |
| ‘exam1’ (user input: String) | Prompt user to enter the score of exam 1. |  |
| ‘exam2’ (User input: Float) | Prompt user to enter the score of exam 2. |  |
| ‘continue\_entry’ (User input: Yes/No) | Controls continuation of data entry loop. |  |
|  | Determine whether to ask for data for another student. |  |
|  | Calculate average score: |  |
|  | ‘averege\_score = (exam + exam2) / 2’ |  |
|  | Increment ‘student\_count’ to track number of students. |  |

1. Prompt the user on whether they want to do this program **(just before the while loop**). Yes means they want to continue. Any other response indicates they will stop the program. This response is the loop control variable. If the user answers Yes then go into the while loop.

Once in the while loop. You are to prompt the user for employee last name, hours worked and rate of pay. Compute gross pay. Give the employee time and a half for hours worked over 40. Sum the gross pay and count the number of employees.

For each employee display their last name and gross pay.

After the loop (all data entered) display the sum of all the gross pays, and count of the number of employees. Compute and display the average pay.

Finally, the **last statements** **within the while loop** will ask the user if they want to do this loop again. In other words the user needs to be prompted again. The reason is that the end of the loop takes execution to the while condition to be evaluated again. It can not take us to the first few lines of code that prompt the user for the first time. That code is out of the loop. Therefore, we need a second prompt at the bottom, inside the loop.

|  |  |  |
| --- | --- | --- |
| Input | Process | Output |
| ‘start\_program’ (user input: Yes/no) | Controls program start | Show employee Last name and gross pay |
|  | Intitates or terminates the while loop based on users input | Total Gross pay |
|  |  | Total # of employees |
| ‘last\_name’ (user input: float) | Prompt for employees last name input | Average pay |
| Hours worked’ (user input: float) | Prompt for hours worked input. | Display the total # of employees, total gross pay, and average pay after all data entry is complete |
| Rate\_of\_pay’ (user input:float) | Prompt for rate of pay input |  |
| ‘continue\_entry’ (user input: Yes/No) | Controls continuations of data entry loop. |  |
|  | Determines whether to ask for data for another employee |  |
|  | Calculate the gross income pay based on hours worked and the rate of pay |  |
|  | Apply overtime calculations (time and a half for hours worked over 40) |  |
|  | Accumulate total gross pay |  |
|  | Calculate average pay after data entered |  |

1. Prompt the user on whether they want to do this program **(just before the while loop**). Yes means they want to continue. Any other response indicates they will stop the program. This response is the loop control variable. If the user answers Yes then go into the while loop.

Once in the while loop. You are to prompt the user for quantity and price of an item. Compute extended price (quantity times price of an item. If the extended price is greater than 10000.00 compute a discount of 25%. All other orders get a 10% discount. For each order display extended price, discount amount (extended price x discount percent), total (extended price – discount amount).

For each order sum the discount amount.

After the loop (all data entered) display the sum of all the discounts.

Finally, the **last statements** **within the while loop** will ask the user if they want to do this loop again. In other words the user needs to be prompted again. The reason is that the end of the loop takes execution to the while condition to be evaluated again. It can not take us to the first few lines of code that prompt the user for the first time. That code is out of the loop. Therefore, we need a second prompt at the bottom, inside the loop.

|  |  |  |
| --- | --- | --- |
| Input | process | Output |
| ‘start\_program’  (user input: yes/no) | Controls program start | Show extended price for each order |
|  | Initates or terminates the while loop based on user input | Show discount amount for each order |
|  |  | Show total price for each order |
| ‘quantity’ (user input:float) | Prompt user to enter the quantity of the item |  |
| ‘price\_per\_item’  (user input: float) | Prompt the user to enter the price per item |  |
| ‘continue\_entry’  (user imput: Yes/no) | Controls continuation of data entry loop |  |
|  | Determines whether to ask for data for another order |  |
|  | Calculates extended price:  ‘exented\_price = quantity times price\_per\_item’ |  |
|  | Determine discount rate |  |
|  | ‘if extended price >  10000.00: discount\_price = 0.25 else: discount\_rate  = 0.10’ |  |
|  | Calculate discount amount:  ‘discount\_amount =  Extended\_price times  Discount\_rate’ |  |
|  | Calculate total price:  ‘total\_price =  extended\_price -  discount\_amount |  |
|  | Accumulate total discount amount |  |
|  | Show total amount amount after all data is entered | Total discount amount for all orders |