CIS 106 – Loops Part 2

For each problem prepare an IPO chart. Then write the code for each. Save the IPO within this document and upload to your repository. After code is complete upload the files (.py) to your repository. Paste the link to your repository into the assignment completion link in Blackboard.

1. Allow the user to enter a principle amount and interest rate repeatedly (need a loop to control the program execution). Compute the annual interest (principle x rate). Compute ending balance to be principle (beginning balance + interest). Display year, beginning balance and ending balance for each of the 5 years. Display the accumulated interest for the 5 years. Note: the new balance by year (this will be the principle for the following year. Format the output.

Example:

Enter principle amount: 10000.00

Enter interest rate: 0.10

Year Beginning Ending

Balance Balance

1. $10,000.00 $11,000.00
2. $11,000.00 $12,100.00
3. $12,100.00 $13,310.00
4. $13,310.00 $14,641.00
5. $14,641.00 $16,105.00

Total interest earned: $6,156.00

|  |  |  |
| --- | --- | --- |
| Principal | interest = principle \* rate | Year (in loop) |
| rate | Ending balance = principle + interest | Principle (in loop) |
|  | Principal = ending balance | Ending balance (in loop) |
|  | Total interest = total interest + interest | Accumulated interest for the 5 years (format output) (Out of loop/end) |
|  |  |  |

1. Fibonacci sequence is a sequence of natural order. The sequence is:

1, 1, 2, 3, 5, 8 etc

Use of for loop compute and display first 20 numbers in the sequence. Hint: start with 1 , 1.

|  |  |  |
| --- | --- | --- |
| A= 1 | Runs 20 times | Print C |
| B= 1 | A+B=C |  |
|  | A = b |  |
|  | B = c |  |
|  |  |  |

1. Create a text file that contains employee last name and salary. Read in this data. Determine the bonus rate based on the chart below. Use that rate to compute bonus. For each line display the employee last name, salary and bonus. After the loop display the sum of all bonuses paid out.

Salary Bonus Rate

100,000.00 and up 20%

50,000.00 15%

All other salaries 10%

Example file (create your own data with at least 5 lines:

Adams

50000.00

Baker

75000.00

Smith

45000.00

Etc

|  |  |  |
| --- | --- | --- |
| Input = datafile | If salary >= 100000:  bonus\_rate = .2  elif salary 50000:  bonus\_rate = .15  else:  bonus\_rate = .1 | employee last name, salary and bonus |
| Salary (from file) | bonus\_amount = salary \* bonus\_rate | Sum of all bonuses |
| Name (from file) | Sum of all bonuses = 0  in loop sum of all bonuses = bonus + sum of all bonuses |  |
|  |  |  |
|  |  |  |

1. Create a text file with item, quantity and price. Read through the file one line at a time. Compute the extended price (quantity x price). For each line display the item, quantity, price and extended price. After the loop display the sum of all the extended prices, the count of the number of orders and the average order.

Example Data File

Widget

10

50

Hammer

2

10

Saw

4

8

Etc

|  |  |  |
| --- | --- | --- |
| datafile | extendedprice=qty\*price | Item (in loop) |
| Item = from file | Sum\_ext\_prices = extendedprice+sum\_ext\_prices | Qty (in loop) |
| Qty = from file | Average of extended prices = numberoforder/sum\_ext\_prices | price(in loop) |
| Price = from file |  | Extendedprice (in loop) |
| sum\_ext\_prices = 0 |  | Numberoforders |
| Average of extended prices = 0 |  | sum\_ext\_prices |
|  |  | Average of extended prices |

1. Create a text file with student last name, district code (I or O) and number of credits taken. Compute tuition owed (credits taken x cost per credit). Cost per credit for in district students (district code I) is 250.00. Out of district students pay 500.00 per credit. For each line display student last name, credits taken and tuition owed. After the loop display sum of all tuition owed and the number of students.

Example file

Jones

I

12

Adams

I

10

Baker

O

12

Smith

O

16

|  |  |  |
| --- | --- | --- |
| file | Tuition = credits \* cost\_per\_credit |  |
| From file – Last name | If dist\_code == I:  cost\_per\_credit = 250  elif dist\_code == O:  cost\_per\_credit = 500  else:  print(“No valid district code entered”) | In loop output:  last name  credits  tuition owed |
| From file - dist\_code |  | Out of loop:  sum\_tuition\_owed |
| From file - credits |  | student\_count |
| Student count = 0  Sum\_tuition\_owed = 0 |  |  |

### [Discussion Post 8](https://harper.blackboard.com/webapps/blackboard/content/launchLink.jsp?course_id=_163827_1&content_id=_9283120_1&mode=view)

How does a for loop work? What types of problems are suited for using it?

\_\_\_\_\_\_\_\_\_\_\_\_\_

A **for loop** is good for when you know how many times something needs to be run/executed. **For loops** can run a range, string, sequence, and more a certain number of times. Lets say you want to run something to count from 1 – 10. This can be done in a **while loop**, but also a **for loop**.

>for x in range(1, 11):

print (x)

in a while loop you would have to do something like this. It will repeat until a conditional value is met.

>x=0

>while x < 11:

x = x+1

print (x)

1-10 will be printed. The 1 in the statement above is where you start the range and the 11 is the end. It’s where it stops so in this case it is 10+1 for it to count to 10.

If you wanted to print by 2’s you would add an increment in the 3rd space. This will run the action in increments of 2. You want a 2 in the starting point of the range if you don’t want to start at zero so you have all even numbers.

>for x in range (2,11,2):

print(x)

In a **while** loop you would do the following

>x=0

>while x < 11

x = x+2

print (x)

There are other ways to use **for loops** such as printing a list.

>fruits = [“orange”, “water melon”, “pear”}

>for x in fruits:

print(x)

This will print the fruits in a list like format.

Right now numbers are easier for me to understand than lists, but you can also nest loops or if statements to add more complexity to the loops.