

# Lab #5 Grading Rubric and Instructions

This is the fifth lab for the 'Introduction to Computer Programming' course.

Please see the syllabus for information about when the work in this lab is due.

## Lab Objective

The purpose of this lab is to give you practice using loops (iteration) to solve problems. By the end, you should be able to ascertain whether to use a single loop or a nested loop structure.

## Instructions/ Deliverables

**NOTE:** These tasks can be completed in any order you like. See the **Grading Items** section below for the point distribution.

**CITATION:** Some of the exercises found here are completely original to the instructor.

- The abbreviation 'MH' will be used to indicate these exercises. This citation will be placed next to the exercise title.
  - ex: [MH]

### Lecture Notes/ Supplemental Readings:

- 'Check off' your notes in your Engineering Notebook for the following material with the TA/ Instructor. These notes should already be done before the start of the lab period.
  - Runestone chapter 8
    - **NOTE:** You do not need to complete any of the exercises at the end of the Runestone chapters. However, it would be helpful to you in the long term if you were to do so.
    - **NOTE:** You only need to take notes on the Runestone readings if you did *not* take lecture notes in class. The contents of these chapters are what was covered in the lectures.
  - Nested Loops in Python - by: Vishal Hule
    - Available: <https://pynative.com/python-nested-loops/>

### `studentLoanAmortization.py`: [MH]

- Read the following article, and make sure you understand how the concept of loan amortization works:
  - <https://www.investopedia.com/terms/a/amortization.asp>
- For this task, you will program a Python script which calculates the monthly payment on a loan, repeatedly subtracts that monthly payment from the remaining balance, and which then prints out a table similar to the one at the bottom of the article.
- Take input from the user in the form of two floats - `principal` and `yearlyInterestRate`, and one integer `numberOfYears`.
  - Here, the `yearlyInterestRate` should be expressed as a decimal value (i.e. 5% = 0.05), and the `numberOfYears` will eventually be multiplied by 12 to get the number of months needed to pay off the loan.
- Write a function that takes these values as input.
- This function will be called `studentLoanAmortization()`, and it will print out the amortization schedule of a loan with columns labeled: `Period`, `Total Payment Due`, `Computed Interest Due`, `Principal Due`, and `Principal Balance`.
  - Please see the "Example Output" section below for what this should generally look like. Please note that you do not need to match the formatting exactly.
- For this file, use a `main()` function, like the one seen in section 6.8 of the Runestone textbook.

- Use the `if __name__ == "__main__":` version as seen in activity 6.8.2.
- Call `main()` inside the `if __name__ == "__main__":` statement.
- Collect your input inside the `main()` function.
- Call the new function you created for the file and pass your input values into the new function which should print out the table.
  - Ex: `studentLoanAmortization(principal, yearlyInterestRate, numberOfYears)`
- **HINT:** To format the output, consider using formatted string literals (<https://docs.python.org/3/tutorial/inputoutput.html#formatted-string-literals>).
- Save your code, including your name, code creation date, lab number, and brief description of what your code does, to a file called `studentLoanAmortization.py`.

### `multiplicationTable.py: [MH]`

- Take input from the user in the form of two integers - `lowNum` and `highNum`.
- Write a function that takes these integers as input.
- This function will be called `multiplicationTable()`, and it will print out a multiplication table like the example seen below, where `lowNum` is 1 and `highNum` is 10:

```

1  2  3  4  5  6  7  8  9  10
2  4  6  8  10 12 14 16 18 20
3  6  9  12 15 18 21 24 27 30
4  8  12 16 20 24 28 32 36 40
5  10 15 20 25 30 35 40 45 50
6  12 18 24 30 36 42 48 54 60
7  14 21 28 35 42 49 56 63 70
8  16 24 32 40 48 56 64 72 80
9  18 27 36 45 54 63 72 81 90
10 20 30 40 50 60 70 80 90 100

```

- For this file, use a `main()` function, like the one seen in section 6.8 of the Runestone textbook.
  - Use the `if __name__ == "__main__":` version as seen in activity 6.8.2.
- Call `main()` inside the `if __name__ == "__main__":` statement.
- Collect your input inside the `main()` function.
- Call the new function you created for the file and pass your input values into the new function which should print out the table.
  - Ex: `multiplicationTable(lowNum, highNum)`
- **HINT:** There are two 'sets' of numbers you are multiplying together. How, for each thing in the *first* group of things, would you make the computer do something for each thing in the *other* group of things?
- **HINT:** To format the output, consider using the `.rjust()` method on a string containing each answer (<https://docs.python.org/3/tutorial/inputoutput.html#manual-string-formatting>).
- **HINT:** To print something without a 'newline,' investigate the `end` parameter of the `print()` function (<https://docs.python.org/3/library/functions.html#print>).
- Save your code, including your name, code creation date, lab number, and brief description of what your code does, to a file called `multiplicationTable.py`.

### `starRightTriangle.py: [MH]`

- Take input from the user in the form of one integers - `num`.
- Write a function that takes this integer as input.
- This function will be called `starRightTriangle()`, and it will print out a pattern like the example seen below, where `num` is 5:

```

*
**
***
****
*****

```

- For this file, use a `main()` function, like the one seen in section 6.8 of the Runestone textbook.
  - Use the `if __name__ == "__main__":` version as seen in activity 6.8.2.
- Call `main()` inside the `if __name__ == "__main__":` statement.
- Collect your input inside the `main()` function.
- Call the new function you created for the file and pass your input values into the new function which should print out the triangle.
  - Ex: `starRightTriangle(num)`
- **HINT:** To print something without a 'newline,' investigate the `end` parameter of the `print()` function (<https://docs.python.org/3/library/functions.html#print>).
- Save your code, including your name, code creation date, lab number, and brief description of what your code does, to a file called `starRightTriangle.py`.

### `numberPyramid.py: [MH]`

- Take input from the user in the form of one integers - `num`.
- Write a function that takes this integer as input.
- This function will be called `numberPyramid()`, and it will print out a pattern like the example seen below, where `num` is 5:

```

1
1 2
1 2 3
1 2 3 4
1 2 3 4 5

```

- For this file, use a `main()` function, like the one seen in section 6.8 of the Runestone textbook.
  - Use the `if __name__ == "__main__":` version as seen in activity 6.8.2.
- Call `main()` inside the `if __name__ == "__main__":` statement.
- Collect your input inside the `main()` function.
- Call the new function you created for the file and pass your input values into the new function which should print out the pyramid.
  - Ex: `numberPyramid(num)`
- **HINT:** Notice that the pyramid is 'centered' around its top 'point.' How many spaces would you have to print out in front of this 'point' to make it 'line up' with the bottom row? Meaning - what is the relationship between the number of spaces printed out at the beginning of each row, and the total number of rows?
- **HINT:** To print something without a 'newline,' investigate the `end` parameter of the `print()` function (<https://docs.python.org/3/library/functions.html#print>).
- Save your code, including your name, code creation date, lab number, and brief description of what your code does, to a file called `numberPyramid.py`.

### `numberDiamond.py: [MH]`

- Take input from the user in the form of one integers - `num`.
- Write a function that takes this integer as input.
- This function will be called `numberDiamond()`, and it will print out a pattern like the example seen below, where `num` is 5:

```

1
1 2
1 2 3
1 2 3 4
1 2 3 4 5
1 2 3 4
1 2 3
1 2
1

```

- For this file, use a `main()` function, like the one seen in section 6.8 of the Runestone textbook.
  - Use the `if __name__ == "__main__":` version as seen in activity 6.8.2.
- Call `main()` inside the `if __name__ == "__main__":` statement.
- Collect your input inside the `main()` function.
- Call the new function you created for the file and pass your input values into the new function which should print out the diamond.
  - Ex: `numberDiamond(num)`
- **HINT:** Is it possible to do this exercise in just one 'pass?' Or, would you have to do it in multiple stages?
- **HINT:** To print something without a 'newline,' investigate the `end` parameter of the `print()` function (<https://docs.python.org/3/library/functions.html#print> ).
- Save your code, including your name, code creation date, lab number, and brief description of what your code does, to a file called `numberDiamond.py`.

### **sameNumberTriangle.py: [MH]**

- Take input from the user in the form of one integers - `num`.
- Write a function that takes this integer as input.
- This function will be called `sameNumberTriangle()`, and it will print out a pattern like the example seen below, where `num` is 5:

```

1
2 2
3 3 3
4 4 4 4
5 5 5 5

```

- For this file, use a `main()` function, like the one seen in section 6.8 of the Runestone textbook.
  - Use the `if __name__ == "__main__":` version as seen in activity 6.8.2.
- Call `main()` inside the `if __name__ == "__main__":` statement.
- Collect your input inside the `main()` function.
- Call the new function you created for the file and pass your input values into the new function which should print out the diamond.
  - Ex: `sameNumberTriangle(num)`
- **HINT:** Is it possible to do this exercise in just one 'pass?' Or, would you have to do it in multiple stages?
- **HINT:** To print something without a 'newline,' investigate the `end` parameter of the `print()` function (<https://docs.python.org/3/library/functions.html#print> ).
- Save your code, including your name, code creation date, lab number, and brief description of what your code does, to a file called `sameNumberTriangle.py`.

### **Attendance:**

- If you have completed all of your tasks for the lab, you may work on any of the 'Additional Resources for Study' found in the Canvas announcement of the same name.
  - **NOTE:** If you leave early, you will not receive the 'attendance points' for the lab.

## **Optional Readings**

**NOTE:** These readings are not required. However, they may provide a bit of interest/ insight into the broader world of Computer Science. Please complete the rest of your lab tasks before doing these readings. You do not need to take notes on these in your Engineering Notebook.

### **Ways to avoid social engineering attacks - by: Kaspersky**

- Available: <https://usa.kaspersky.com/resource-center/threats/how-to-avoid-social-engineering-attacks>

## **Generative AI to Become a \$1.3 Trillion Market by 2032, Research Finds - by: Bloomberg**

- Available: <https://www.bloomberg.com/company/press/generative-ai-to-become-a-1-3-trillion-market-by-2032-research-finds/>

## **Landing a Job in Artificial Intelligence - by: Will Capella**

- Available: <https://www.computerscience.org/resources/landing-a-job-in-artificial-intelligence/>

## **Lost Sacred Gems: The State of Indie Games in 2022 and Beyond - by: Andrew Johnston**

- Available: <https://medium.com/super-jump/lost-sacred-gems-the-state-of-indie-games-in-2022-and-beyond-98332fd86d56>

## **How to Become an Indie Game Developer in 2022 - by: Rokoko**

- Available: <https://www.rokoko.com/insights/how-to-become-indie-game-developer>

## **Files Provided**

**None**

## **Example Script**

**None**

## **Example Output**

**Running:** `python studentLoanAmortization.py`

```
Please Input the Principal: 500
Please Input the Yearly Interest: 0.05
Please Input the Number of Years: 1
```

Period	Total Payment Due	Computed Interest	Principal Due	Principal Balance
1	42.80	2.08	40.72	459.28
2	42.80	1.91	40.89	418.39
3	42.80	1.74	41.06	377.33
4	42.80	1.57	41.23	336.10
5	42.80	1.40	41.40	294.69
6	42.80	1.23	41.58	253.12
7	42.80	1.05	41.75	211.37
8	42.80	0.88	41.92	169.45
9	42.80	0.71	42.10	127.35
10	42.80	0.53	42.27	85.08
11	42.80	0.35	42.45	42.63
12	42.80	0.18	42.63	0.00

## **Grading Items**

- **(Lecture Notes/ Supplemental Readings)** Has the student taken notes on the listed material, and shown their notes in their Engineering Notebook to the TA/ Instructor?: \_\_\_\_\_ / 20
- **(studentLoanAmortization.py)** Has the student completed the task above, and saved their work to a file called `studentLoanAmortization.py`? \_\_\_\_\_ / 20
- **(multiplicationTable.py)** Has the student completed the task above, and saved their work to a file called `multiplicationTable.py`? \_\_\_\_\_ / 10

- (**starRightTriangle.py**) Has the student completed the task above, and saved their work to a file called starRightTriangle.py?: \_\_\_\_\_ / 10
- (**numberPyramid.py**) Has the student completed the task above, and saved their work to a file called numberPyramid.py?: \_\_\_\_\_ / 10
- (**numberDiamond.py**) Has the student completed the task above, and saved their work to a file called numberDiamond.py?: \_\_\_\_\_ / 10
- (**sameNumberTriangle.py**) Has the student completed the task above, and saved their work to a file called sameNumberTriangle.py?: \_\_\_\_\_ / 10
- (**Attendance**) Did the student attend the full lab meeting in person?: \_\_\_\_\_ / 10

**TOTAL** \_\_\_\_\_ / 100