35.1 LAB*: Program: Painting a wall

Program Specifications Write a program to calculate the cost to paint a wall. Amount of required paint is based on the wall area. Total cost includes paint and sales tax.

Note: This program is designed for *incremental development*. Complete each step and submit for grading before starting the next step. Only a portion of tests pass after each step but confirm progress.

Step 1 (2 pts). Read from input wall height, wall width, and cost of one paint can (floats). Calculate and output the wall's area to one decimal place using `print(f"Wall area: {wall_area:.1f} sq ft");`. Submit for grading to confirm 1 test passes.

Ex: If the input is:

```
12.0
15.0
29.95
```

the output is:

```
Wall area: 180.0 sq ft
```

Step 2 (2 pts). Calculate and output the amount of paint needed to three decimal places. One gallon of paint covers 350 square feet. Submit for grading to confirm 2 tests pass.

Ex: If the input is:

```
12.0
15.0
29.95
```

the output is:

```
Wall area: 180.0 sq ft
Paint needed: 0.514 gallons
```


Step 3 (2 pts). Calculate and output the number of 1 gallon cans needed to paint the wall. Extra paint may be left over. Hint: Use ceil() from the math module to round up to the nearest gallon (int). Submit for grading to confirm 4 tests pass.

Ex: If the input is:

```
12.0
```

```
15.0
29.95
```

the output is:

```
Wall area: 180.0 sq ft
Paint needed: 0.5142 gallons
Cans needed: 1 can(s)
```


Step 4 (4 pts). Calculate and output the paint cost, sales tax of 7%, and total cost. Dollar values are output with two decimal places. Submit for grading to confirm all tests pass.

Ex: If the input is:

```
8.0
8.0
49.20
```

the output is:

```
Wall area: 64.0 sq ft
Paint needed: 0.183 gallons
Cans needed: 1 can(s)
Paint cost: $49.20
Sales tax: $3.44
Total cost: $52.64
```

422102.2723990.qx3zqy7

LAB ACTIVITY | 35.1.1: LAB*: Program: Painting a wall | 0 / 10

main.py — Load default template...

```
1 from math import ceil                    # needed in Step #3
2
3 # Type your code here.
```

Develop mode | Submit mode

Run your program as often as you'd like, before submitting for grading. Below, type any needed input values in the first box, then click **Run program** and observe the program's output in the second box.

Enter program input (optional)

If your code requires input values, provide them here.

Run program Input (from above) → **main.py** (Your program) → O

Program output displayed here

Coding trail of your work What is this?

```
History of your effort will appear here once you begin
working on this zyLab.
```

35.2 LAB: Output range with increment of 5

Write a program whose input is two integers. Output the first integer and subsequent increments of 5 as long as the value is less than or equal to the second integer.

Ex: If the input is:

```
-15
10
```


the output is:

```
-15 -10 -5 0 5 10
```

Ex: If the second integer is less than the first as in:

```
20
5
```

the output is:

```
Second integer can't be less than the first.
```


For coding simplicity, output a space after every integer, including the last.

422102.2723990.qx3zqy7

LAB ACTIVITY 35.2.1: LAB: Output range with increment of 5 0 / 10

main.py Load default template...

```
1 ''' Type your code here. '''
```

Develop mode **Submit mode**

Run your program as often as you'd like, before submitting for grading. Below, type any needed input values in the first box, then click **Run program** and observe the program's output in the second box.

Enter program input (optional)

If your code requires input values, provide them here.

Run program Input (from above) → **main.py** (Your program) → O

Program output displayed here

Coding trail of your work What is this?

```
History of your effort will appear here once you begin
working on this zyLab.
```


35.3 LAB: Convert to reverse binary

Write a program that takes in a positive integer as input, and outputs a string of 1's and 0's representing the integer in reverse binary. For an integer x, the algorithm is:

```
As long as x is greater than 0
   Output x modulo 2 (remainder is either 0 or 1)
   Assign x with x divided by 2
```

Note: The above algorithm outputs the 0's and 1's in reverse order.

Ex: If the input is:

```
6
```

the output is:

```
011
```

6 in binary is 110; the algorithm outputs the bits in reverse.

422102.2723990.qx3zqy7

LAB ACTIVITY 35.3.1: LAB: Convert to reverse binary 0 / 10

main.py Load default template...

```
1 ''' Type your code here. '''
2
```

Develop mode | Submit mode

Run your program as often as you'd like, before submitting for grading. Below, type any needed input values in the first box, then click **Run program** and observe the program's output in the second box.

Enter program input (optional)

If your code requires input values, provide them here.

Run program

Input (from above) → **main.py** (Your program) → O

Program output displayed here

Coding trail of your work What is this?

```
History of your effort will appear here once you begin
working on this zyLab.
```

35.4 LAB: Count input length without spaces, periods, exclamation points, or commas

Given a line of text as input, output the number of characters excluding spaces, periods, exclamation points, or commas.

Ex: If the input is:

```
Listen, Mr. Jones, calm down.
```

the output is:

```
21
```

Note: Account for all characters that aren't spaces, periods, exclamation points, or commas (Ex: "r", "2", "?").

422102.2723990.qx3zqy7

LAB ACTIVITY | 35.4.1: LAB: Count input length without spaces, periods, exclamation points, or commas | 0 / 10

main.py — Load default template...

```
1 user_text = input()
2
3 ''' Type your code here. '''
4
```

Develop mode | **Submit mode**

Run your program as often as you'd like, before submitting for grading. Below, type any needed input values in the first box, then click **Run program** and observe the program's output in the second box.

Enter program input (optional)

If your code requires input values, provide them here.

Run program Input (from above) → **main.py** (Your program) → O

Program output displayed here

Coding trail of your work What is this?

```
History of your effort will appear here once you begin
working on this zyLab.
```


35.5 LAB: Password modifier

Many user-created passwords are simple and easy to guess. Write a program that takes a simple password and makes it stronger by replacing characters using the key below, and by appending "!" to the end of the input string.

- i becomes 1
- a becomes @
- m becomes M
- B becomes 8
- s becomes $

Ex: If the input is:

```
mypassword
```

the output is:

```
Myp@$$word!
```

Hint: Python strings are immutable, but support string concatenation. Store and build the stronger password in the given ***`password`*** *variable.*

422102.2723990.qx3zqy7

LAB ACTIVITY 35.5.1: LAB: Password modifier 0 / 10

main.py Load default template...

```
word = input()
password = ''

''' Type your code here. '''
```

Develop mode | Submit mode

Run your program as often as you'd like, before submitting for grading. Below, type any needed input values in the first box, then click **Run program** and observe the program's output in the second box.

Enter program input (optional)

If your code requires input values, provide them here.

Run program

Input (from above) → **main.py** (Your program) → O

Program output displayed here

Coding trail of your work What is this?

```
History of your effort will appear here once you begin
working on this zyLab.
```

35.6 LAB: Warm up: Drawing a right triangle

This program will output a right triangle based on user specified height triangle_height and symbol triangle_char.

(1) The given program outputs a fixed-height triangle using a * character. Modify the given program to output a right triangle that instead uses the user-specified triangle_char character. (1 pt)

(2) Modify the program to use a loop to output a right triangle of height triangle_height. The first line will have one user-specified character, such as % or *. Each subsequent line will have one additional user-specified character until the number in the triangle's base reaches triangle_height. Output a space after each user-specified character, including a line's last user-specified character. (2 pts)

Example output for triangle_char = % and triangle_height = 5:

```
Enter a character:
%
Enter triangle height:
5

%
% %
% % %
% % % %
% % % % %
```

422102.2723990.qx3zqy7

LAB ACTIVITY 35.6.1: LAB: Warm up: Drawing a right triangle 0 / 3

main.py Load default template...

```
1 triangle_char = input('Enter a character:\n')
2 triangle_height = int(input('Enter triangle height:\n'))
3 print('')
4
5 print ('* ')
6 print ('* * ')
7 print ('* * * ')
8
```


Develop mode **Submit mode** Run your program as often as you'd like, before submitting for grading. Below, type any needed input

values in the first box, then click **Run program** and observe the program's output in the second box.

Enter program input (optional)

If your code requires input values, provide them here.

Run program Input (from above) → **main.py** (Your program) → O

Program output displayed here

Coding trail of your work What is this?

```
History of your effort will appear here once you begin
working on this zyLab.
```

35.7 LAB: Countdown until matching digits

Write a program that takes in an integer in the range 11-100 as input. The output is a countdown starting from the integer, and stopping when both output digits are identical.

Ex: If the input is:

```
93
```

the output is:

```
93
92
91
90
89
88
```


Ex: If the input is:

```
11
```

the output is:

```
11
```

Ex: If the input is:

```
9
```


or any value not between 11 and 100 (inclusive), the output is:

```
Input must be 11-100
```

Use a while loop. Compare the digits; do not write a large if-else for all possible same-digit numbers (11, 22, 33, ..., 99), as that approach would be cumbersome for larger ranges.

422102.2723990.qx3zqy7

LAB ACTIVITY | 35.7.1: LAB: Countdown until matching digits | 0 / 10

main.py — Load default template...

```
1 ''' Type your code here. '''
2
```


Develop mode | **Submit mode**

Run your program as often as you'd like, before submitting for grading. Below, type any needed input values in the first box, then click **Run program** and observe the program's output in the second box.

Enter program input (optional)

If your code requires input values, provide them here.

Run program

Input (from above) → main.py (Your program) → O

Program output displayed here

Coding trail of your work What is this?

```
History of your effort will appear here once you begin
working on this zyLab.
```

35.8 LAB: Print string in reverse

Write a program that takes in a line of text as input, and outputs that line of text in reverse. The program repeats, ending when the user enters "Done", "done", or "d" for the line of text.

Ex: If the input is:

```
Hello there
Hey
done
```

then the output is:

```
ereht olleH
yeH
```

422102.2723990.qx3zqy7

LAB ACTIVITY | 35.8.1: LAB: Print string in reverse 0 / 10

main.py Load default template...

```
1 ''' Type your code here. '''
2
```

Develop mode | Submit mode

Run your program as often as you'd like, before submitting for grading. Below, type any needed input values in the first box, then click **Run program** and observe the program's output in the second box.

Enter program input (optional)

If your code requires input values, provide them here.

Run program

Input (from above) → **main.py** (Your program) → O

Program output displayed here

Coding trail of your work What is this?

```
History of your effort will appear here once you begin
working on this zyLab.
```

35.9 LAB: Brute force equation solver

Numerous engineering and scientific applications require finding solutions to a set of equations. Ex: 8x + 7y = 38 and 3x - 5y = -1 have a solution x = 3, y = 2. Given integer coefficients of two linear equations with variables x and y, use brute force to find an integer solution for x and y in the range -10 to 10.

Ex: If the input is:

```
8
7
38
3
-5
-1
```


Then the output is:

```
x = 3 , y = 2
```

Use this brute force approach:

```
For every value of x from -10 to 10
   For every value of y from -10 to 10
      Check if the current x and y satisfy both equations. If so,
output the solution, and finish.
```

Ex: If no solution is found, output:

```
There is no solution
```

Assume the two input equations have no more than one solution.

Note: Elegant mathematical techniques exist to solve such linear equations. However, for other kinds of equations or situations, brute force can be handy.

422102.2723990.qx3zqy7

LAB ACTIVITY | 35.9.1: LAB: Brute force equation solver | 0 / 10

main.py — Load default template...

```
''' Read in first equation, ax + by = c '''
a = int(input())
b = int(input())
c = int(input())

''' Read in second equation, dx + ey = f '''
d = int(input())
e = int(input())
f = int(input())

''' Type your code here. '''

```

Develop mode | Submit mode

Run your program as often as you'd like, before submitting for grading. Below, type any needed input values in the first box, then click **Run program** and observe the program's output in the second box.

Enter program input (optional)

If your code requires input values, provide them here.

Run program Input (from above) → **main.py** (Your program) → O

Program output displayed here

Coding trail of your work What is this?

```
History of your effort will appear here once you begin
working on this zyLab.
```

35.10 LAB: Smallest and largest numbers in a list

Write a program that reads a list of integers into a list as long as the integers are greater than zero, then outputs the smallest and largest integers in the list.

Ex: If the input is:


```
10
5
3
21
2
-6
```

the output is:

```
2 and 21
```

You can assume that the list of integers will have at least 2 values.

422102.2723990.qx3zqy7

LAB ACTIVITY 35.10.1: LAB: Smallest and largest numbers in a list 0 / 10

main.py Load default template...

```
1 ''' Type your code here. '''
2
```

Develop mode | **Submit mode**

Run your program as often as you'd like, before submitting for grading. Below, type any needed input values in the first box, then click **Run program** and observe the program's output in the second box.

Enter program input (optional)

If your code requires input values, provide them here.

Run program Input (from above) → **main.py** (Your program) → O

Program output displayed here

Coding trail of your work What is this?

```
History of your effort will appear here once you begin
working on this zyLab.
```


35.11 LAB: Output values in a list below a user defined amount

Write a program that first gets a list of integers from input. The input begins with an integer indicating the number of integers that follow. Then, get the last value from the input, which indicates a threshold. Output all integers less than or equal to that last threshold value.

Ex: If the input is:

```
5
50
60
140
200
75
100
```

the output is:

```
50,60,75,
```

The 5 indicates that there are five integers in the list, namely 50, 60, 140, 200, and 75. The 100 indicates that the program should output all integers less than or equal to 100, so the program outputs 50, 60, and 75.

For coding simplicity, follow every output value by a comma, including the last one.

Such functionality is common on sites like Amazon, where a user can filter results.

422102.2723990.qx3zqy7

LAB ACTIVITY | 35.11.1: LAB: Output values in a list below a user defined amount | 0 / 10

main.py Load default template...

Develop mode | Submit mode

Run your program as often as you'd like, before submitting for grading. Below, type any needed input values in the first box, then click **Run program** and observe the program's output in the second box.

Enter program input (optional)

If your code requires input values, provide them here.

Run program Input (from above) → **main.py** (Your program) → O

Program output displayed here

Coding trail of your work What is this?

```
History of your effort will appear here once you begin
working on this zyLab.
```

©zyBooks 12/15/22 01:02 1361995
John Farrell
COLOSTATECS220SeaboltFall2022

35.12 LAB: Adjust values in a list by normalizing

When analyzing data sets, such as data for human heights or for human weights, a common step is to adjust the data. This adjustment can be done by normalizing to values between 0 and 1, or throwing away outliers.

For this program, adjust the values by dividing all values by the largest value. The input begins with an integer indicating the number of floating-point values that follow.

Output each floating-point value with two digits after the decimal point, which can be achieved as follows:

```
print(f'{your_value:.2f}')
```


Ex: If the input is:

```
5
30.0
50.0
10.0
100.0
65.0
```

the output is:

```
0.30
0.50
0.10
1.00
0.65
```

The 5 indicates that there are five floating-point values in the list, namely 30.0, 50.0, 10.0, 100.0, and 65.0. 100.0 is the largest value in the list, so each value is divided by 100.0.

422102.2723990.qx3zqy7

LAB ACTIVITY | 35.12.1: LAB: Adjust values in a list by normalizing | 0 / 10

main.py — Load default template...


```
1 ''' Type your code here. '''
```

Develop mode | Submit mode

Run your program as often as you'd like, before submitting for grading. Below, type any needed input values in the first box, then click **Run program** and observe the program's output in the second box.

Enter program input (optional)

If your code requires input values, provide them here.

Run program Input (from above) → **main.py** (Your program) → O

Program output displayed here

Coding trail of your work What is this?

```
History of your effort will appear here once you begin
working on this zyLab.
```

35.13 LAB*: Program: Drawing a half arrow

Program Specifications Write a program that outputs a downwards facing arrow composed of a rectangle and a right triangle. Arrow dimensions are defined by user specified arrow base height, arrow base width, and arrow head width.

Note: this program is designed for *incremental development*. Complete each step and submit for grading before starting the next step. Only a portion of tests pass after each step but confirm progress.

Step 1 (3 pts). Input the arrow base height (int) and width (int). Draw a rectangle using asterisks (height x width). Hint: use a nested loop in which the inner loop draws one row of *s, and the outer loop iterates a number of times equal to the height. Submit for grading to confirm two tests pass.

Ex: If input is:

```
6
4
```

Sample output is:

```
****
****
****
****
****
****
```


Step 2 (3 pts). Input the arrow head width and draw a right triangle. Hint: use a nested loop. Submit for grading to confirm four tests pass.

Ex: If input is:

```
4
3
4
```

Sample output is:

```
***
***
***
***
****
***
**
*
```

Step 3 (4 pts). Modify the program to only accept an arrow head width that is larger than the arrow base width. Use a loop to continue inputting the arrow head width until the value is larger than the arrow base width. Submit for grading to confirm all tests pass.

Ex: If input is:

```
4
3
3
2
```

4

Sample output is:

```
***
***
***
***
****
***
**
*
```


422102.2723990.qx3zqy7

LAB ACTIVITY | 35.13.1: LAB*: Program: Drawing a half arrow | 0 / 10

main.py Load default template...

```
1 # Type your code here.
2
```

Develop mode | **Submit mode**

Run your program as often as you'd like, before submitting for grading. Below, type any needed input values in the first box, then click **Run program** and observe the program's output in the second box.

Enter program input (optional)

If your code requires input values, provide them here.

Run program

Input (from above) → main.py (Your program) → O

Program output displayed here

Coding trail of your work What is this?

```
History of your effort will appear here once you begin
working on this zyLab.
```

35.14 LAB: Input and formatted output: Right-facing arrow

Given input characters for an arrowhead and arrow body, print a right-facing arrow.

Ex: If the input is:

```
*
#
```

Then the output is:

```
      #
******##
******###
******##
      #
```

422102.2723990.qx3zqy7

LAB ACTIVITY 35.14.1: LAB: Input and formatted output: Right-facing arrow 0 / 10

main.py Load default template...

```
1 base_char = input()
2 head_char = input()
```

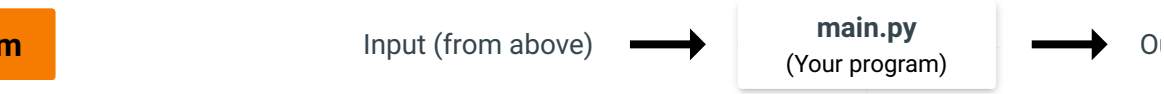
Develop mode | Submit mode

Run your program as often as you'd like, before submitting for grading. Below, type any needed input values in the first box, then click **Run program** and observe the program's output in the second box.

Enter program input (optional)

If your code requires input values, provide them here.

Run program



Program output displayed here

Coding trail of your work What is this?

```
History of your effort will appear here once you begin
working on this zyLab.
```


35.15 LAB: Divide input integers

Write a program that reads integers `user_num` and `div_num` as input, and outputs `user_num` divided by `div_num` three times using floor divisions.

Ex: If the input is:

```
2000
2
```

the output is:

```
1000 500 250
```


Note: In Python 3, floor division discards fractions. Ex: 6 // 4 is 1 (the 0.5 is discarded).

422102.2723990.qx3zqy7

LAB ACTIVITY | 35.15.1: LAB: Divide input integers | 0 / 10

main.py Load default template...

```
1 ''' Type your code here. '''
2
```

Develop mode | **Submit mode**

Run your program as often as you'd like, before submitting for grading. Below, type any needed input values in the first box, then click **Run program** and observe the program's output in the second box.

Enter program input (optional)

If your code requires input values, provide them here.

Run program Input (from above) → **main.py** (Your program) → O

Program output displayed here

Coding trail of your work What is this?

```
History of your effort will appear here once you begin
working on this zyLab.
```


35.16 LAB: Using math functions

Given three floating-point numbers x, y, and z, output x to the power of z, x to the power of (y to the power of z), the absolute value of (x minus y), and the square root of (x to the power of z).

Output each floating-point value with two digits after the decimal point, which can be achieved as follows:
`print(f'{your_value1:.2f} {your_value2:.2f} {your_value3:.2f} {your_value4:.2f}')`

Ex: If the input is:

```
5.0
1.5
3.2
```

Then the output is:

```
172.47 361.66 3.50 13.13
```

422102.2723990.qx3zqy7

LAB ACTIVITY 35.16.1: LAB: Using math functions 0 / 10

main.py Load default template...


```
1 import math
2 ''' Type your code here. '''
```

Develop mode | Submit mode

Run your program as often as you'd like, before submitting for grading. Below, type any needed input values in the first box, then click **Run program** and observe the program's output in the second box.

Enter program input (optional)

If your code requires input values, provide them here.

Run program

Input (from above) → **main.py** (Your program) → O

Program output displayed here

Coding trail of your work What is this?

```
History of your effort will appear here once you begin
working on this zyLab.
```

35.17 LAB: Driving costs

Driving is expensive. Write a program with a car's gas mileage (miles/gallon) and the cost of gas (dollars/gallon) as floating-point input, and output the gas cost for 20 miles, 75 miles, and 500 miles.

Output each floating-point value with two digits after the decimal point, which can be achieved as follows:

```
print(f'{your_value1:.2f} {your_value2:.2f} {your_value3:.2f}')
```

Ex: If the input is:

```
20.0
3.1599
```

where the gas mileage is 20.0 miles/gallon and the cost of gas is $3.1599/gallon, the output is:

```
3.16 11.85 79.00
```


Note: Real per-mile cost would also include maintenance and depreciation.

422102.2723990.qx3zqy7

LAB ACTIVITY 35.17.1: LAB: Driving costs 0 / 10

main.py Load default template...

```
1 ''' Type your code here. '''
2
```

Develop mode | **Submit mode**

Run your program as often as you'd like, before submitting for grading. Below, type any needed input values in the first box, then click **Run program** and observe the program's output in the second box.

Enter program input (optional)

If your code requires input values, provide them here.

Run program Input (from above) → **main.py** (Your program) → O

Program output displayed here

Coding trail of your work What is this?

```
History of your effort will appear here once you begin
working on this zyLab.
```


35.18 LAB: Expression for calories burned during workout

The following equation estimates the average calories burned for a person when exercising, which is based on a scientific journal article (source):

Calories = ((Age x 0.2757) + (Weight x 0.03295) + (Heart Rate x 1.0781) — 75.4991) x Time / 8.368

Write a program using inputs age (years), weight (pounds), heart rate (beats per minute), and time (minutes), respectively. Output the average calories burned for a person.

Output each floating-point value with two digits after the decimal point, which can be achieved as follows:
`print(f'Calories: {calories:.2f} calories')`

Ex: If the input is:

```
49
155
148
60
```

then the output is:

```
Calories: 736.21 calories
```


422102.2723990.qx3zqy7

LAB ACTIVITY 35.18.1: LAB: Expression for calories burned during workout 0 / 10

main.py Load default template...

```
1 ''' Calories = ((Age x 0.2757) + (Weight x 0.03295) + (Heart Rate x 1.0781) – 75.4991) x
```

Develop mode | Submit mode

Run your program as often as you'd like, before submitting for grading. Below, type any needed input values in the first box, then click **Run program** and observe the program's output in the second box.

Enter program input (optional)

If your code requires input values, provide them here.

Run program Input (from above) → **main.py** (Your program) → O

Program output displayed here

Coding trail of your work What is this?

```
History of your effort will appear here once you begin
working on this zyLab.
```


35.19 LAB: Musical note frequencies

On a piano, a key has a frequency, say f0. Each higher key (black or white) has a frequency of f0 * r^n, where n is the distance (number of keys) from that key, and r is $2^{(1/12)}$. Given an initial key frequency, output that frequency and the next 4 higher key frequencies.

Output each floating-point value with two digits after the decimal point, which can be achieved as follows:

```
print(f'{your_value1:.2f} {your_value2:.2f} {your_value3:.2f}
{your_value4:.2f} {your_value5:.2f}')
```

Ex: If the input is:

```
440
```


(which is the A key near the middle of a piano keyboard), the output is:

```
440.00 466.16 493.88 523.25 554.37
```

Note: Use one statement to compute r = $2^{(1/12)}$ using the pow function (remember to import the math module). Then use that r in subsequent statements that use the formula fn = f0 * r^n with n being 1, 2, 3, and finally 4.

422102.2723990.qx3zqy7

LAB ACTIVITY | 35.19.1: LAB: Musical note frequencies | 0 / 10

main.py — Load default template...

```
1 ''' Type your code here. '''
2
```


Develop mode | Submit mode

Run your program as often as you'd like, before submitting for grading. Below, type any needed input values in the first box, then click **Run program** and observe the program's output in the second box.

Enter program input (optional)

If your code requires input values, provide them here.

Run program

Input (from above) → main.py (Your program) → O

Program output displayed here

Coding trail of your work What is this?

```
History of your effort will appear here once you begin
working on this zyLab.
```

35.20 LAB: Warm up: Variables, input, and type conversion

(1) Prompt the user to input an integer between 32 and 126, a float, a character, and a string, storing each into separate variables. Then, output those four values on a single line separated by a space. (Submit for 2 points).

Note: This zyLab outputs a newline after each user-input prompt. For convenience in the examples below, the user's input value is shown on the next line, but such values don't actually appear as output when the program runs.

```
Enter integer (32 - 126):
99
Enter float:
3.77
Enter character:
z
Enter string:
Howdy
99 3.77 z Howdy
```


(2) Extend to also output in reverse. (Submit for 1 point, so 3 points total).

```
Enter integer (32 - 126):
99
Enter float:
3.77
Enter character:
z
Enter string:
Howdy
99 3.77 z Howdy
Howdy z 3.77 99
```


(3) Extend to convert the integer to a character by using the 'chr()' function, and output that character. (Submit for 2 points, so 5 points total).

```
Enter integer (32 - 126):
99
Enter float:
3.77
Enter character:
z
Enter string:
Howdy
99 3.77 z Howdy
Howdy z 3.77 99
99 converted to a character is c
```

422102.2723990.qx3zqy7

LAB ACTIVITY 35.20.1: LAB: Warm up: Variables, input, and type conversion 0 / 5

main.py Load default template...

```
1 user_int = int(input('Enter integer (32 - 126):\n'))
2
3 # FIXME (1): Finish reading other items into variables, then output the four values on a si
4
5 # FIXME (2): Output the four values in reverse
6
7 # FIXME (3): Convert the integer to a character, and output that character
```

Develop mode | Submit mode

Run your program as often as you'd like, before submitting for grading. Below, type any needed input values in the first box, then click **Run program** and observe the program's output in the second box.

Enter program input (optional)

If your code requires input values, provide them here.

Run program

Input (from above) → main.py (Your program) → O

Program output displayed here

Coding trail of your work What is this?

```
History of your effort will appear here once you begin
working on this zyLab.
```

35.21 LAB: Phone number breakdown

Given an integer representing a 10-digit phone number, output the area code, prefix, and line number using the format (800) 555-1212.

Ex: If the input is:


```
8005551212
```

the output is:

```
(800) 555-1212
```

Hint: Use % to get the desired rightmost digits. Ex: The rightmost 2 digits of 572 is gotten by 572 %

100, which is 72.

Hint: Use // to shift right by the desired amount. Ex: Shifting 572 right by 2 digits is done by 572 // 100, which yields 5. (Recall integer division discards the fraction).

For simplicity, assume any part starts with a non-zero digit. So 0119998888 is not allowed.

422102.2723990.qx3zqy7

LAB ACTIVITY | 35.21.1: LAB: Phone number breakdown | 0 / 10

main.py — Load default template...

```
1 phone_number = int(input())
2
3 ''' Type your code here. '''
4
5
```

Develop mode | **Submit mode**

Run your program as often as you'd like, before submitting for grading. Below, type any needed input values in the first box, then click **Run program** and observe the program's output in the second box.

Enter program input (optional)

If your code requires input values, provide them here.

Run program

Input (from above) → **main.py** (Your program) → O

Program output displayed here

Coding trail of your work What is this?

```
History of your effort will appear here once you begin
working on this zyLab.
```


35.22 LAB*: Program: Cooking measurement converter

Output each floating-point value with two digits after the decimal point, which can be achieved as follows:
`print(f'{your_value:.2f}')`

(1) Prompt the user for the number of cups of lemon juice, water, and agave nectar needed to make lemonade. Prompt the user to specify the number of servings the recipe yields. Output the ingredients and serving size. (Submit for 2 points).

Note: This zyLab outputs a newline after each user-input prompt. For convenience in the examples below, the user's input value is shown on the next line, but such values don't actually appear as output when the program runs.

```
Enter amount of lemon juice (in cups):
2
Enter amount of water (in cups):
16
Enter amount of agave nectar (in cups):
2.5
How many servings does this make?
6

Lemonade ingredients - yields 6.00 servings
2.00 cup(s) lemon juice
16.00 cup(s) water
2.50 cup(s) agave nectar
```


(2) Prompt the user to specify the desired number of servings. Adjust the amounts of each ingredient accordingly, and then output the ingredients and serving size. (Submit for 4 points, so 6 points total).

```
How many servings would you like to make?
48

Lemonade ingredients - yields 48.00 servings
16.00 cup(s) lemon juice
128.00 cup(s) water
20.00 cup(s) agave nectar
```


(3) Convert the ingredient measurements from (2) to gallons. Output the ingredients and serving size. Note: There are 16 cups in a gallon. (Submit for 2 points, so 8 points total).

```
Lemonade ingredients - yields 48.00 servings
1.00 gallon(s) lemon juice
8.00 gallon(s) water
1.25 gallon(s) agave nectar
```

422102.2723990.qx3zqy7

LAB ACTIVITY 35.22.1: LAB*: Program: Cooking measurement converter 0 / 8

main.py Load default template...

```
1 lemon_juice_cups = float(input('Enter amount of lemon juice (in cups):\n'))
2
3 # FIXME (1): Finish reading other items into variables, then output the three ingredients
4
5 # FIXME (2): Prompt user for desired number of servings. Convert and output the ingredients
6
7 # FIXME (3): Convert and output the ingredients from (2) to gallons
8
9
```


Develop mode | **Submit mode**

Run your program as often as you'd like, before submitting for grading. Below, type any needed input values in the first box, then click **Run program** and observe the program's output in the second box.

Enter program input (optional)

If your code requires input values, provide them here.

Run program

Input (from above) → main.py (Your program) → O

Program output displayed here

Coding trail of your work What is this?

```
History of your effort will appear here once you begin
working on this zyLab.
```

35.23 LAB*: Program: Food receipt

Note: When accuracy is essential, floats are not used to represent currency due to rounding and accumulation errors. Python provides several primitives specifically developed to implement financial applications. However, these topics are beyond the scope of this lab.

Output each floating-point value with two digits after the decimal point, which can be achieved as follows:
`print(f'{your_value:.2f}')`

(1) Prompt the user to input a food item name, price, and quantity. Output an itemized receipt. (Submit for 2 points)

Note: This zyLab outputs a newline after each user-input prompt. For convenience in the examples below, the user's input value is shown on the next line, but such values don't actually appear as output when the program runs.


```
Enter food item name:
hot dog
Enter item price:
2.00
Enter item quantity:
5
```

```
RECEIPT
5 hot dog @ $2.00 = $10.00
Total cost: $10.00
```

(2) Extend the program to prompt the user for a second item. Output an itemized receipt. (Submit for 2 points, so 4 points total)


```
Enter food item name:
hot dog
Enter item price:
2.00
Enter item quantity:
5

RECEIPT
5 hot dog @ $2.00 = $10.00
Total cost: $10.00


Enter second food item name:
ice cream
Enter item price:
2.50
Enter item quantity:
4

RECEIPT
5 hot dog @ $2.00 = $10.00
4 ice cream @ $2.50 = $10.00
Total cost: $20.00
```

(3) Extend again to output a third receipt that adds a mandatory 15% gratuity to the total cost. Output the total cost, the cost of gratuity, and the grand total. (Submit for 3 points, so 7 points total)


```
Enter food item name:
hot dog
Enter item price:
2.00
Enter item quantity:
5

RECEIPT
5 hot dog @ $2.00 = $10.00
```

```
Total cost: $10.00


Enter second food item name:
ice cream
Enter item price:
2.50
Enter item quantity:
4

RECEIPT
5 hot dog @ $2.00 = $10.00
4 ice cream @ $2.50 = $10.00
Total cost: $20.00

15% gratuity: $3.00
Total with tip: $23.00
```


422102.2723990.qx3zqy7

LAB ACTIVITY | 35.23.1: LAB*: Program: Food receipt | 0 / 7

main.py — Load default template...

```
1 item_name = input('Enter food item name:\n')
2
3 # FIXME (1): Finish reading item price and quantity, then output a receipt
4
5 # FIXME (2): Read in a second food item name, price, and quantity, then output a second rece
6
7 # FIXME (3): Add a gratuity and total with tip to the second receipt
```


Develop mode | Submit mode

Run your program as often as you'd like, before submitting for grading. Below, type any needed input values in the first box, then click **Run program** and observe the program's output in the second box.

Enter program input (optional)

If your code requires input values, provide them here.

Run program

Program output displayed here

Coding trail of your work What is this?

```
History of your effort will appear here once you begin
working on this zyLab.
```

35.24 LAB: Input and formatted output: Caffeine levels

A half-life is the amount of time it takes for a substance or entity to fall to half its original value. Caffeine has a half-life of about 6 hours in humans. Given caffeine amount (in mg) as input, output the caffeine level after 6, 12, and 24 hours. Use a string formatting expression with conversion specifiers to output the caffeine amount as floating-point numbers.

Output each floating-point value with two digits after the decimal point, which can be achieved as follows:
`print(f'{your_value:.2f}')`

Ex: If the input is:

```
100
```


the output is:

```
After 6 hours: 50.00 mg
After 12 hours: 25.00 mg
After 24 hours: 6.25 mg
```

Note: A cup of coffee has about 100 mg. A soda has about 40 mg. An "energy" drink (a misnomer)

has between 100 mg and 200 mg.

422102.2723990.qx3zqy7

LAB ACTIVITY | 35.24.1: LAB: Input and formatted output: Caffeine levels | 0 / 10

main.py Load default template...

```
1 caffeine_mg = float(input())
2
3 ''' Type your code here. '''
4
5
```

Develop mode | Submit mode

Run your program as often as you'd like, before submitting for grading. Below, type any needed input values in the first box, then click **Run program** and observe the program's output in the second box.

Enter program input (optional)

If your code requires input values, provide them here.

Run program Input (from above) → **main.py** (Your program) → O

Program output displayed here

Coding trail of your work What is this?

```
History of your effort will appear here once you begin
working on this zyLab.
```

35.25 LAB: Input and formatted output: House real estate summary

Sites like Zillow get input about house prices from a database and provide nice summaries for readers. Write a program with two inputs, current price and last month's price (both integers). Then, output a summary listing the price, the change since last month, and the estimated monthly mortgage computed as (current_price * 0.051) / 12.

Output each floating-point value with two digits after the decimal point, which can be achieved as follows:
`print(f'{your_value:.2f}')`

Ex: If the input is:

```
200000
210000
```

the output is:

```
This house is $200000. The change is $-10000 since last month.
The estimated monthly mortgage is $850.00.
```

Note: Getting the precise spacing, punctuation, and newlines *exactly* right is a key point of this assignment. Such precision is an important part of programming.

422102.2723990.qx3zqy7

LAB ACTIVITY 35.25.1: LAB: Input and formatted output: House real estate summary 0 / 10

main.py Load default template...

```
1 current_price = int(input())
2 last_months_price = int(input())
3
4 ''' Type your code here. '''
5
```

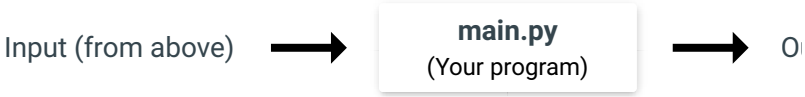
Develop mode | Submit mode

Run your program as often as you'd like, before submitting for grading. Below, type any needed input values in the first box, then click **Run program** and observe the program's output in the second box.

Enter program input (optional)

If your code requires input values, provide them here.

Run program



Program output displayed here

Coding trail of your work What is this?

```
History of your effort will appear here once you begin
working on this zyLab.
```

35.26 LAB: Simple statistics

Given 4 floating-point numbers. Use a string formatting expression with conversion specifiers to output their product and their average as integers (rounded), then as floating-point numbers.

Output each rounded integer using the following:
`print(f'{your_value:.0f}')`

Output each floating-point value with three digits after the decimal point, which can be achieved as

follows:
`print(f'{your_value:.3f}')`

Ex: If the input is:

```
8.3
10.4
5.0
4.8
```


the output is:

```
2072 7
2071.680 7.125
```

422102.2723990.qx3zqy7

LAB ACTIVITY 35.26.1: LAB: Simple statistics 0 / 10

main.py Load default template...

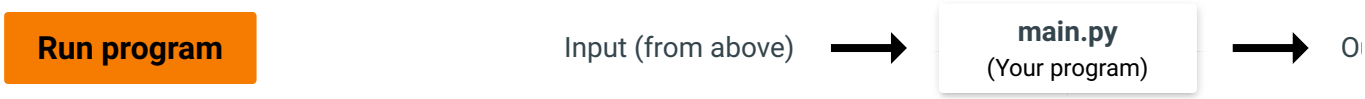
```
num1 = float(input())
num2 = float(input())
num3 = float(input())
num4 = float(input())

''' Type your code here. '''
```


Develop mode **Submit mode**

Run your program as often as you'd like, before submitting for grading. Below, type any needed input values in the first box, then click **Run program** and observe the program's output in the second box.

Enter program input (optional)

If your code requires input values, provide them here.



Program output displayed here

Coding trail of your work What is this?

```
History of your effort will appear here once you begin
working on this zyLab.
```

35.27 LAB: Warm up: Creating passwords

(1) Prompt the user to enter two words and a number, storing each into separate variables. Then, output those three values on a single line separated by a space. (Submit for 1 point)

Ex: If the input is:

```
yellow
Daisy
6
```

the output after the prompts is:

```
You entered: yellow Daisy 6
```

Note: User input is not part of the program output.

(2) Output two passwords using a combination of the user input. Format the passwords as shown below. (Submit for 2 points, so 3 points total).

Ex: If the input is:

```
yellow
Daisy
6
```

the output after the prompts is:

```
You entered: yellow Daisy 6

First password: yellow_Daisy
Second password: 6yellow6
```

(3) Output the length of each password (the number of characters in the strings). (Submit for 2 points, so 5 points total).

Ex: If the input is:

```
yellow
Daisy
6
```

the output after the prompts is:

```
You entered: yellow Daisy 6

First password: yellow_Daisy
Second password: 6yellow6

Number of characters in yellow_Daisy: 12
Number of characters in 6yellow6: 8
```

422102.2723990.qx3zqy7

LAB ACTIVITY 35.27.1: LAB: Warm up: Creating passwords 0 / 5

main.py Load default template...

```
1  # FIXME (1): Finish reading another word and an integer into variables.
2  # Output all the values on a single line
3  favorite_color = input('Enter favorite color:\n')
4
5
6  # FIXME (2): Output two password options
7  password1 = favorite_color
8  print('\nFirst password:')
9
10
11 # FIXME (3): Output the length of the two password options
12
```

Develop mode | Submit mode

Run your program as often as you'd like, before submitting for grading. Below, type any needed input values in the first box, then click **Run program** and observe the program's output in the second box.

Enter program input (optional)

If your code requires input values, provide them here.

Run program

Input (from above) → **main.py** (Your program) → O

Program output displayed here

Coding trail of your work What is this?

```
History of your effort will appear here once you begin
working on this zyLab.
```

35.28 LAB: Convert to dollars

This section's content is not available for print.