Lone Star
section 4

final retake quiz

1. (8.2) In each input list, every number repeats at least once, except for two. Write a **function** that takes an array *numbers* and returns the two unique numbers.

   Examples:

   - return_unique([1, 9, 8, 8, 7, 6, 1, 6]) → [9, 7],
   - return_unique([5, 5, 2, 4, 4, 4, 9, 9, 9, 1]) → [2, 1],
   - return_unique([9, 5, 6, 8, 7, 7, 1, 1, 1, 1, 1, 9, 8]) → [5, 6]

2. (8.3) Write a **function** that takes a dictionary, called *employee_salaries*, where the keys are employee names and the values are their salaries. The function should return a list of employees earning above a given salary.

   Examples:

   - high_earners({"Alice": 50000, "Bob": 75000, "Charlie": 100000}, 60000) → ["Bob", "Charlie"]
   - high_earners({"David": 30000, "Emma": 45000, "Frank": 50000}, 40000) → ["Emma", "Frank"]
   - high_earners({"George": 25000, "Hannah": 27000, "Ian": 29000}, 30000) → []

   Write a class for a **Playlist** with the instance variables and methods listed below.
   A Playlist should have a default name of "New Playlist".
   It can be instantiated with initial songs, but it is not required to.
   Create a method called *add_song* which adds a song title (a string) to the Playlist.
   You should be able to combine two Playlists, and print them in a readable way.

   For example:

   - $p_1 =$ ["Song A", "Song B"]
   - $p_2 =$ ["Song C"]
   - $p_1 + p_2 =$ ["Song A", "Song B", "Song C"]

Playlist
name songs (list of strings)
__init__ add_song __add__ __str__

   Your class should support:

   - Creating a playlist with a name and list of songs
   - Adding two playlists (combines song lists)
   - Printing the playlist in a readable way (e.g., list songs)

   Once you have created the class, add code that:

   - Creates two playlists and at least one song to each.
   - Combines the playlists
   - Prints the result

3. (6.1) The **boiling point** of water is 212F in Fahrenheit and 100C in Celsuis. Create a function that determines if the *temp* is considered boiling or not. *temp* will be measured in Fahrenheit and Celsuis. Notice: The F or C will always be the last character in the string.

   Examples:

   - is_boiling("212F") → True,

- is_boiling("100C") → True,
- is_boiling("0F") → False,

4. (9.1) (Game: Odd or Even) Write a **function** that lets the user guess whether a randomly generated number is odd or even. The function randomly generates an integer between 0 and 9 (inclusive) and returns whether the user's guess is correct or incorrect. The argument for the function will be *guess* (the user's guess, either "odd" or "even"), if no argument is provided then the **default** guess should be even.
   Hint: Use the following lines of code to create the function.

   ```
   from random import randint
   value = randint(0,9) #picks a random integer between 0-9 inclusive
   ```

   Examples:

   - guess( ) → "Correct!" (if random value is even) or "Incorrect!" (if random value is odd)
   - guess("odd")→"Correct!" (if random value is odd) or "Incorrect!" (if random value is even)
   - guess("even") → "Correct!" (if random value is even) or "Incorrect!" (if random value is odd)

Dot Matrix
section 5

final retake quiz

1. (4.1) Ask the user for two integers named *larger* and *smaller*. Determine (and output) how many times larger can be halved while still be greater than smaller.

   Examples:

   - if *larger* = 1324 and *smaller* = 98, the result should be 3 since 1324 → 662 → 331 → 165.5
   - if *larger* = 624 and *smaller* = 8, the result should be 6 since
     624 → 312 → 156 → 78 → 39 → 19.5 → 9.75)

2. (15.1) Write a **program** that asks the user to enter the name of a text file and then prints the contents of that file to the screen. The program should handle the following error:

   - If the file does not exist (**FileExistsError**), print "File not found."

   Examples:

   - "Enter file name:" "LunchData.txt" → (prints the content of LunchData.txt)
   - "Enter file name:" "DinnerData.txt" → "File not found."

3. (3.1) Write a program that asks the user for three numbers, and then determines (and outputs) which of the numbers is the largest. Do not use the built-in function *max*().
   For example,

```
Pick a number: 10
Pick another number: 9
Pick another number: 15
The largest number is 15.
```

```
Pick a number: 21
Pick another number: 3
Pick another number: 17
The largest number is 21.
```

4. (15.2) You are helping a teacher update students' scores after a quiz. The teacher wants to add points for extra credit and needs your program to do the math safely.

   A dictionary stores the number of points each student has earned, shown below:

   - {''Alice'': 90, ''Bob'': 75, ''Charlie'': 60}

   Write a **program** that asks the user to enter a student's name and a number to add to their score. The program should print the new number of points.

   The program should handle the following errors:

   - If the name is not found in the dictionary (**KeyError**), print "Student not found."
   - If the number entered is not valid (e.g., not a number) (**ValueError**), print "Invalid number."

   Examples:

   - "Enter student name:" "Bob"
     "Enter number to add:" 10 → 85

- "Enter student name:" "David" → "Student not found."
- "Enter student name:" "Alice"
  "Enter number to add:" "ten" → "Invalid number."

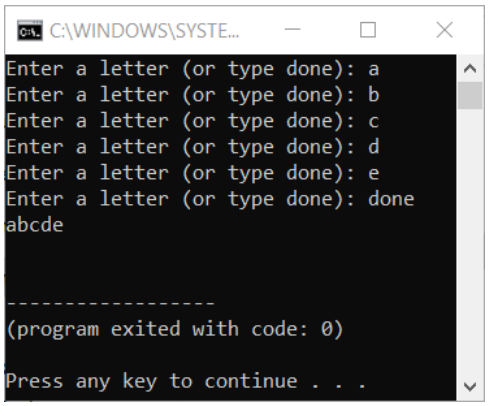
5. (3.2) Write a program that prompts the user to enter three integers and displays the integers in decreasing order (largest to smallest). You may not use the built-in functions *max*(), *min*(), *sort*() or *sorted*().

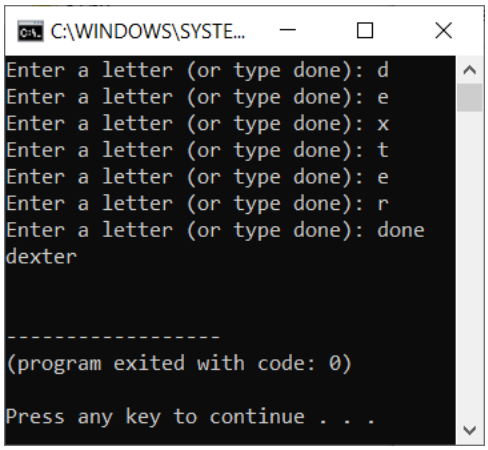
Dark Helmet
section 4

final retake quiz

1. (4.2) Write a program to create a word one letter at a time. You should prompt the user to enter a single letter one at a time until they type *done*. Once they type done, output their newly created word.

   For example,





2. (15.3) You've been asked to help build part of a travel booking system. One of the features lets users type in a country code (like "US") and shows them the full country name to confirm their destination.

   A dictionary stores some country codes, shown below:

   ```
   {''US'':''United States'', ''FR'':''France'', ''JP'':''Japan'', ''BR'':''Brazil''}
   ```

   Write a **program** that asks the user to enter a country code (like "US") and then prints the full country name. If the user enters an invalid code, they should be asked to try again until a valid code is entered.

   The program should handle the following errors:

   - If the code is not found in the dictionary (**KeyError**), print "Code not found. Try again."

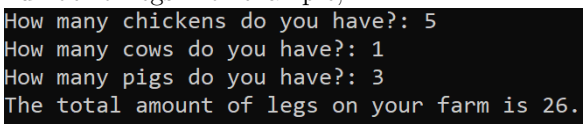
   Examples:

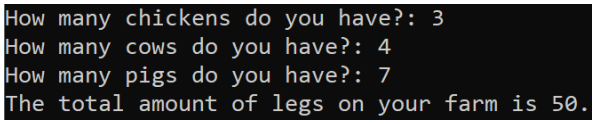
   - "Enter a country code:" "JP" → "Japan"
   - "Enter a country code:" "XYZ" → "Code not found. Try again."
   - (after retry) "Enter a country code:" "BR" → "Brazil"

3. (2.1) A farmer is asking you to tell him how many legs can be counted among all his animals. The farmer breeds three species:

   - chickens, which have **2** legs
   - cows, which have **4** legs
   - pigs, which have **4** legs

   Write a program that asks the farmer how many of each animal he has, and then outputs the total number of legs. For example,





4. (15.1) Write a **program** that asks the user to enter the name of a text file and then prints the contents of that file to the screen. The program should handle the following error:

   - If the file does not exist (**FileExistsError**), print "File not found."

   Examples:

   - "Enter file name:" "LunchData.txt" → (prints the content of LunchData.txt)
   - "Enter file name:" "DinnerData.txt" → "File not found."

5. (15.2) You are helping a teacher update students' scores after a quiz. The teacher wants to add points for extra credit and needs your program to do the math safely.

   A dictionary stores the number of points each student has earned, shown below:

   - `{''Alice'': 90, ''Bob'': 75, ''Charlie'': 60}`

   Write a **program** that asks the user to enter a student's name and a number to add to their score. The program should print the new number of points.

   The program should handle the following errors:

   - If the name is not found in the dictionary (**KeyError**), print "Student not found."
   - If the number entered is not valid (e.g., not a number) (**ValueError**), print "Invalid number."

   Examples:

   - "Enter student name:" "Bob"
     "Enter number to add:" 10 → 85
   - "Enter student name:" "David" → "Student not found."
   - "Enter student name:" "Alice"
     "Enter number to add:" "ten" → "Invalid number."

President Skroob final retake quiz
section 1

1. (15.3) You've been asked to help build part of a travel booking system. One of the features lets users type in a country code (like "US") and shows them the full country name to confirm their destination.

   A dictionary stores some country codes, shown below:

   ```
   {''US'':''United States'', ''FR'':''France'', ''JP'':''Japan'', ''BR'':''Brazil''}
   ```

   Write a **program** that asks the user to enter a country code (like "US") and then prints the full country name. If the user enters an invalid code, they should be asked to try again until a valid code is entered.

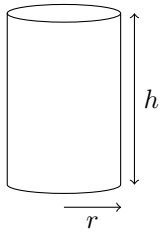
   The program should handle the following errors:

   - If the code is not found in the dictionary (**KeyError**), print "Code not found. Try again."

   Examples:

   - "Enter a country code:" "JP" → "Japan"
   - "Enter a country code:" "XYZ" → "Code not found. Try again."
   - (after retry) "Enter a country code:" "BR" → "Brazil"

2. (2.3) Write a program that calculates the volume of a cylinder. The user should be able to pick the height and radius. Use the value of π from the math module in your calculation.
   Hint: $V = \pi r^2 h$



3. (4.2) Write a program that repeatedly asks the user for integers until a negative integer is given. The program should keep track of the sum of the numbers and print the sum at the end (not including the negative number).

   For example,

```
Enter an integer: 7
Enter an integer: 10
Enter an integer: 3
Enter an integer: -4
20
```

```
Enter an integer: 1
Enter an integer: 2
Enter an integer: 3
Enter an integer: 4
Enter an integer: 5
Enter an integer: -1
15
```

4. (4.3) You are the newest rug fashion designer on the scene, but you're running out of ideas. Write a program that will help you design rugs. The program should ask for a width, a length, and pattern, and then create a rug consisting of that pattern and dimensions.

For example,

```
Enter a width: 3
Enter a length: 5
Enter a pattern: $

Your rug is:
$$$
$$$
$$$
$$$
$$$
```

```
Enter a width: 16
Enter a length: 5
Enter a pattern: @

Your rug is:
@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@@@@
```

5. (2.2) Write a program that calculates then outputs the volume of a right square pyramid. The user should be able to pick b (the base edge) and h (the height).

   Hint: $V = \dfrac{b^2 h}{3}$