1. **(25 Points)** Write a complete, working Python 3 program called euidA.py (where “euid” is your EUID) that does the following:

* *Background*: The internal angles in a regular polygon are all the same and the angles in degrees can be calculated using the formula: , where is the number of sides and the 180 value is to present the angle in degrees. For example, each of the internal angles of a regular pentagon with 5 sides is degrees. Note that this formula is not valid for number of sides .
* Write your EUID in comments at the top of the file.
* Define a custom exception class called SideException using Exception as its parameter. You may simply have the pass statement inside the body.
* Define a new class called RegPoly with the \_\_init\_\_() constructor, \_\_str\_\_() special method, and two instance attributes, sides and angle, as follows:
  + Define the \_\_init\_\_() constructor that accepts a single integer parameter called numSides (in addition to the default self parameter) that will initialize the sides and angle instance attributes as follows:
    - The sides instance attribute will be initialized to the value of the passed in argument numSides.
    - The angle instance attribute will be computed using the internal angles formula (from above) with the sides instance attribute for the number of sides.
  + Define the \_\_str\_\_() special method that just accepts the default self parameter and returns a string with details about the number of sides and internal angle degrees (see SAMPLE OUTPUT).
* Initialize a loop control variable and then use a while loop to loop through the number of command-line arguments passed to the program (excluding the name of the Python 3 program itself). Inside the loop:
  + Use a try-except-else-finally block as follows:
    - In the try block, if the integral value of the command-line argument is less than 3, raise a SideException, passing a suitable error message that includes the integer causing the exception to be raised.
    - In the except block, handle the SideException by printing out 'SideException:' with its accompanying error message.
    - In the else block, instantiate a RegPoly object called my\_poly, passing in the command-line argument as an integer. Then, print the my\_poly object.
    - In the finally block, increment the control variable (for your loop) so that you iterate through all of the command-line arguments.
* You may assume the user passes integers to the program when executing it, though the number and values of the integers may vary. Due to time constraints, no further comments are required.

Here is some sample output to help you write the code. The items in bold are entered by the user. Note that when the user passes no command-line arguments, there is no output since there are no command-line arguments to loop through.

$ **python3 mat0299A.py**

$ **python3 mat0299A.py 7 8 1 5**

sides=7, internal angles=128.57142857142858

sides=8, internal angles=135.0

SideException: Invalid number of sides for regular polygon: 1

sides=5, internal angles=108.0

1. **(25 Points)** Write a complete, working Python 3 program called euidB.py (where “euid” is your EUID) that does the following:

* Outside of the euidB.py program that you will create for this portion of the lab exam, create another Python program called euidBMOD.py that does the following:
  + Define a user-defined function called home\_prices() that accepts one argument, an integer called number for the size of the list to create. Inside the function:
    - Create an empty list called price\_list.
    - Initialize (i.e., seed) the random number generator.
    - Using a loop, randomly generate the number of integral home prices between 100,000 and 500,000, inclusively, and append each to the price\_list list.
    - Return the price\_list list.
  + Outside of the function (in the main part), prompt the user to enter the number of homes being put on the market and assign that value to a variable called number.
  + Call the home\_prices() function, passing number as an argument, and assign the result to a new list called my\_list.
  + Print the my\_list list.
  + Since this Python module will be imported into your euidB.py program, add the necessary statements so that the prompt, function call, and print are not executed if this Python module is imported by another Python program.
* Write your EUID in comments at the top of the file.
* Import your euidBMOD.py module to this Python program.
* Create a dictionary called real\_estate and initialize it with the following keys and values:

'Park Avenue': 325999

'Lawther Drive': 692848

* Print 'Current homes on the market:' and then iterate through real\_estate to print all the keys (i.e., the street names of the homes on the market).
* Prompt the user for the number of new homes to go on the market and assign that value to a variable called number.
* Call the home\_prices() function from the other Python module, passing number as an argument, and assign the result to a new list called price\_list.
* Print the price\_list list.
* Using a loop, prompt for and read in the number of street names (i.e., keys) of new homes on the market to add, and then add them to real\_estate with the corresponding value (i.e., price) from price\_list. For example, the first street name entered would be initialized to the first value from price\_list, the second street name entered would be initialized to the second value from price\_list, and so on.
* Iterate through your real\_estate dictionary to print the street names (i.e., keys) and prices formatted with the "$" and two decimal places as shown in SAMPLE OUTPUT.
* Prompt for the street name of the home you wish to purchase and assign that value to a variable called purchase.
* If that street name exists in the real\_estate dictionary, print a message that the home is being removed from the market and then delete that key-value pair from the dictionary. If that street name is not found in the real\_estate dictionary, print a message that the home is not available.
* Finally, print the real\_estate dictionary.
* You may assume the user enters all elements using the appropriate data type. Due to time constraints, no further comments are required.

Here is a sample output to help you write the code. The items in bold are entered by the user.

$ **python3 mat0299B.py**

Current homes on the market: Park Avenue Lawther Drive

Enter number of new homes to go on the market: **3**

[302772, 400996, 219547]

Enter street name of new home on market: **Swiss Avenue**

Enter street name of new home on market: **Elm Street**

Enter street name of new home on market: **College Avenue**

Park Avenue : $325999.00

Lawther Drive : $692848.00

Swiss Avenue : $302772.00

Elm Street : $400996.00

College Avenue : $219547.00

Enter the street name of the home you wish to buy: **Elm Street**

Elm Street home removed from market

{'Park Avenue': 325999, 'Lawther Drive': 692848, 'Swiss Avenue': 302772, 'College Avenue': 219547}

$ **python3 mat0299B.py**

Current homes on the market: Park Avenue Lawther Drive

Enter number of new homes to go on the market: **2**

[105409, 416113]

Enter street name of new home on market: **Main Street**

Enter street name of new home on market: **Valley Road**

Park Avenue : $325999.00

Lawther Drive : $692848.00

Main Street : $105409.00

Valley Road : $416113.00

Enter the street name of the home you wish to buy: **Elm Street**

Elm Street home not available

{'Park Avenue': 325999, 'Lawther Drive': 692848, 'Main Street': 105409, 'Valley Road': 416113}