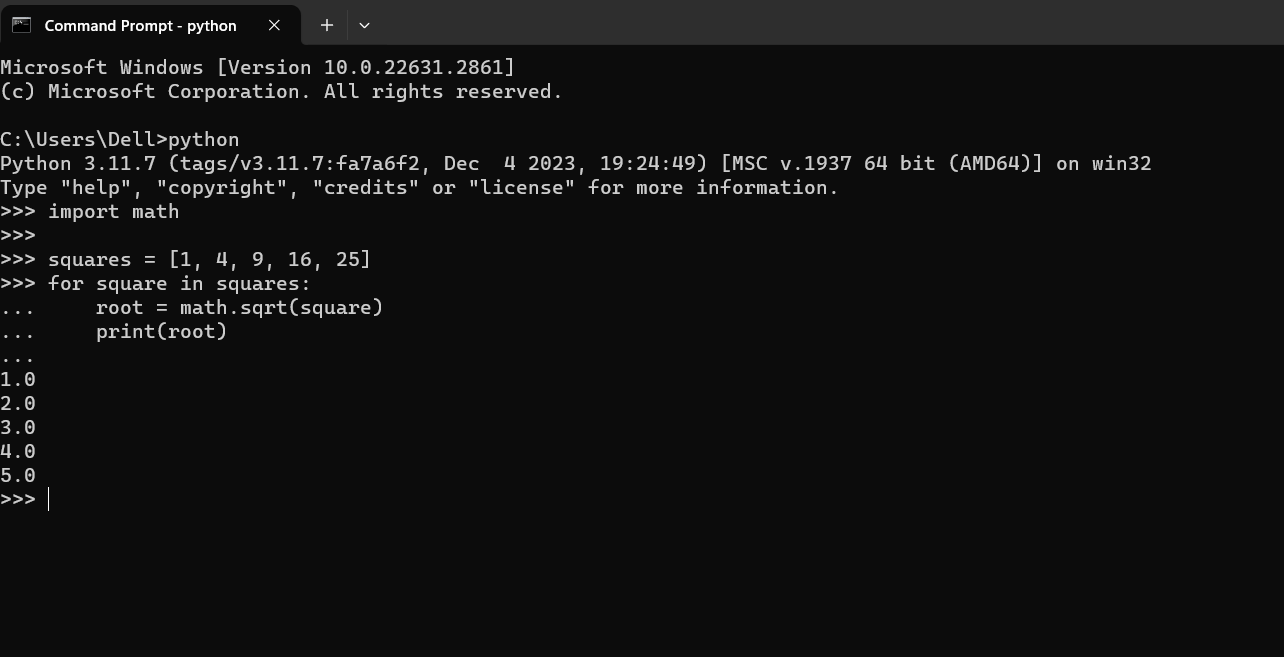
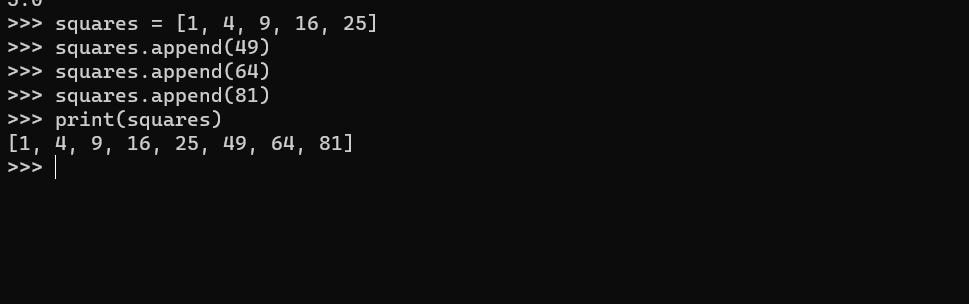
Prithu Adhikari

Week 6

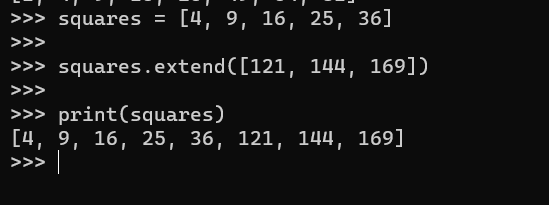
TASK: Write a for..in loop that iterates over all the elements of the squares list and prints the square root of each to the screen. Hint: you may want to import a function from the math module to help achieve this.



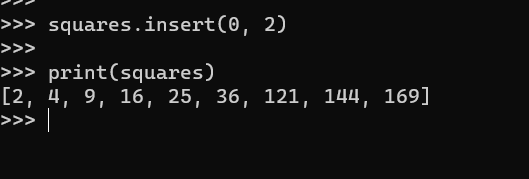
**TASK**: Write some code that uses the append() method to add the next three square values (49, 64, 81) to the end of the squares list.



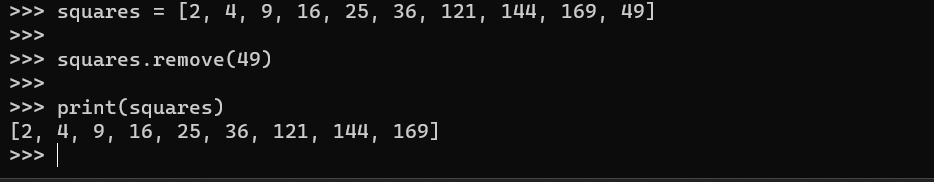
**TASK**: Write some code that uses the extend() method to add the next three square values, starting at 121 (11 x 11), to the end of the squares list.



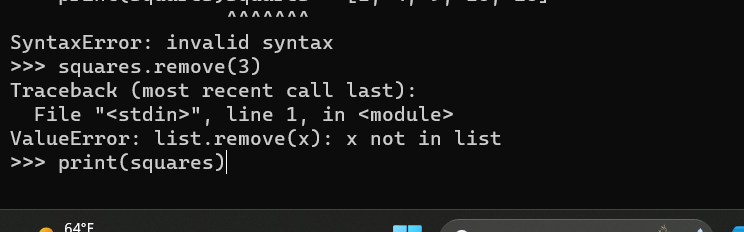
**TASK**: Write some code that uses the insert() method to insert the value 2, to the very beginning of the squares list.



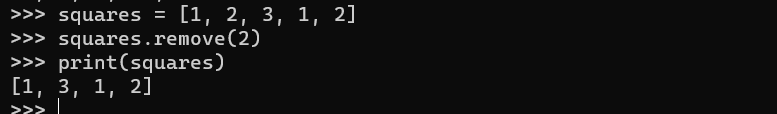
**TASK**: Write some code that uses the remove() method to remove the value 49 from the squares list. Print the list afterwards to ensure the value has indeed been removed.



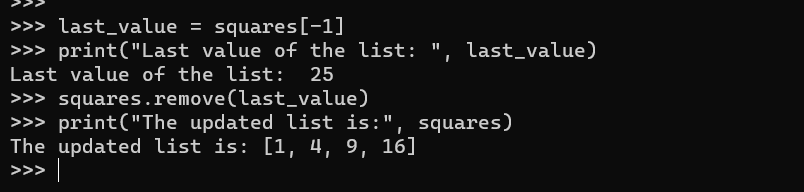
**TASK**: Write some code that uses the remove() method to remove the value 3 from the squares list. Notice how an error is generated since the given value was not present.



**TASK**: Create a simple list that contains the values [1, 2, 3, 1, 2] and then use the remove () method to remove the value 2. Which value is removed?

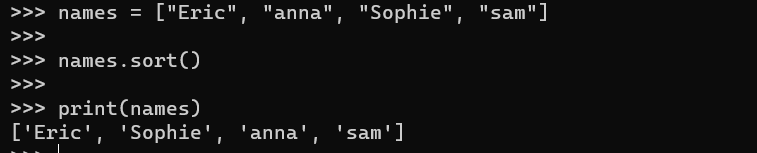


**TASK**: Write some code that uses the pop() method to remove and display the first value of the squares list. Print the list afterwards to ensure the value has been removed.

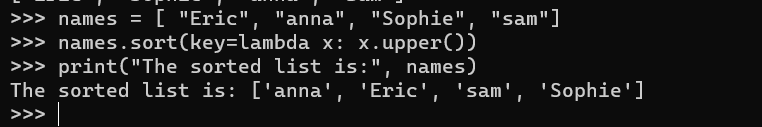


**TASK**: Write some code that uses the sort() method with no arguments, to alphabetically sort the exact list of names shown below. Display the list after the sort has been called.

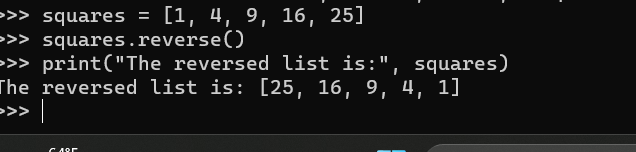
names = [ "Eric", "anna", "Sophie", "sam" ]



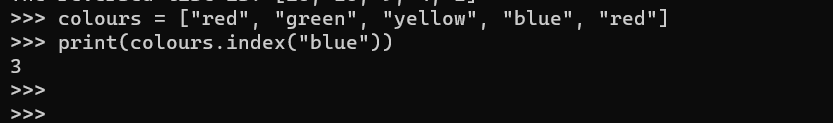
**TASK**: Improve your previous solution so that the list is sorted correctly, ignoring the case used to write the names. To achieve this you will have to include a key argument in the form of a *lambda expression* that returns each string as uppercase letters only. Hint: you can use the str.upper() method to convert a name to uppercase letters.



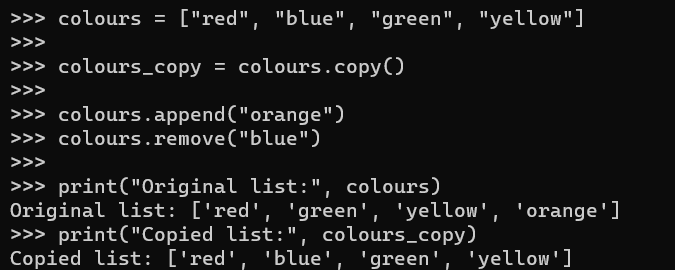
**TASK**: Write some code that uses the reverse() method to reverse the values of the squares list. Print the list afterwards to ensure the values have been reversed.



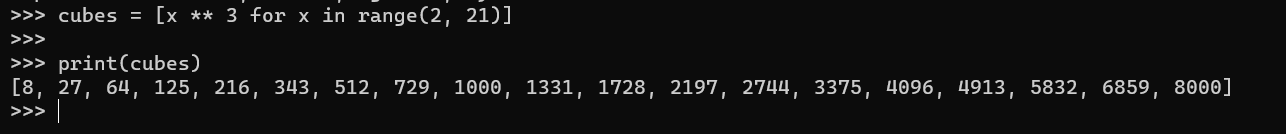
**TASK**: Write some code that finds the index of the colour blue.



**TASK**: Write some code that makes a copy of the colours using the copy() method. Then make some changes to the original list. Print the contents of the copied list to ensure these changes have not affected the copy.



**TASK**: Write some code that uses a list *comprehension* to create a list called cubes that contains the cubed values (x \* x \* x) of the numbers from 2 to 20 inclusive



some\_users = [u for u in all\_users if len(u) < 8]

**TASK**: Examine the above code and work out which user names will be placed in the some\_users list. What is the condition that has to be met for inclusion?

The some\_users list will contain user names from the all\_users list where the length of each user name (len(u)) is less than 8. In other words, it includes user names that have fewer than 8 characters. The condition for inclusion is that the length of the user name must be less than 8 characters.

**TASK:** Create a tuple called address that includes your own “house number”, “street” and, “postcode” as three different values.

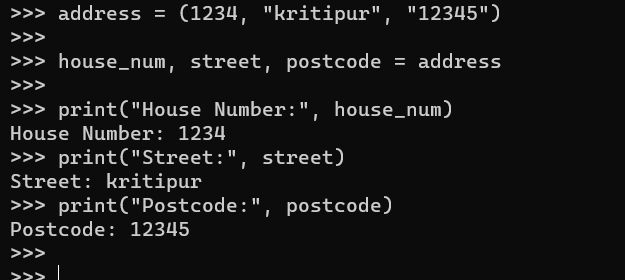
address = (1234, "kritipur", "12345")

**TASK:** Try entering the above examples to create single element tuples. Then use the type() function to examine the data-type of the created variables.

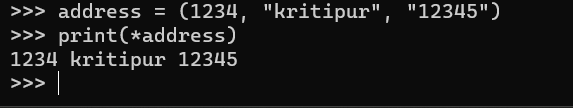
Hint: For this reason it is common to include a trailing comma even when one is not needed. For example:

Student = ("Griffin, P.", 2, 38.2,)

**TASK:** Use *sequence unpacking* to extract the values you stored within the address tuple earlier. Unpack the tuple into variables named house\_num, street and postcode.



**TASK:** Write some code that calls the print() function to output the contents of the address tuple you created earlier. Ensure you use the ‘\*’ prefix so that the elements are extracted before being passed to the function. Compare this with a version of the same code that calls the print() function without using the ‘\*’ prefix,



**TASK**: Look at each of the phrases below and ensure you understand what each of these means. For any that you do not understand, do a little research to find a definition of each term. This research may involve looking back over these notes, or the associated lecture notes. It may also involve searching for these terms on the Internet.

* Method
* List comprehension
* Tuple
* Tuple Packing
* Sequence Unpacking

Method: In Python, a method is a function that is associated with an object. It is called on an object and can access and modify the data within that object.

List Comprehension: List comprehension is a concise way to create lists in Python. It consists of an expression followed by at least one for clause and zero or more if clauses. It provides a more readable and compact syntax for creating lists.

Tuple: A tuple is an ordered and immutable sequence of elements. Tuples are defined using parentheses and can contain elements of different data types.

Tuple Packing: Tuple packing refers to the process of combining multiple values into a single tuple. This happens automatically when you separate values with commas.

Sequence Unpacking: Sequence unpacking is the process of extracting values from a sequence (like a tuple or a list) and assigning them to individual variables. It's often used with multiple assignment statements.