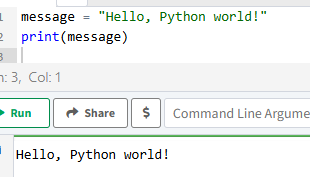
Submitted by: Syed Muhammad Sajjad Haider

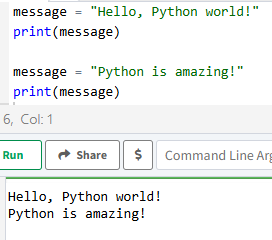
Submitted to: Ma’am Mufeeza

Roll No: FA24-BBA-078

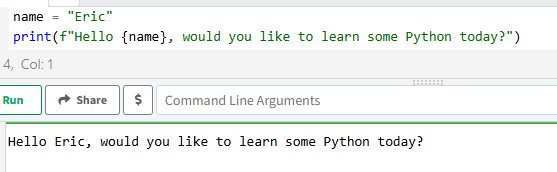
Lab Exercise 1

Q1. Simple Message: Assign a message to a variable, and then print that message.

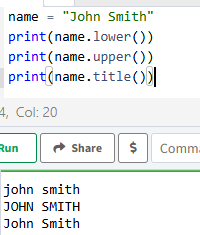


Q2. Simple Messages: Assign a message to a variable, and print that message. Then change the value of the variable to a new message, and print the new message.

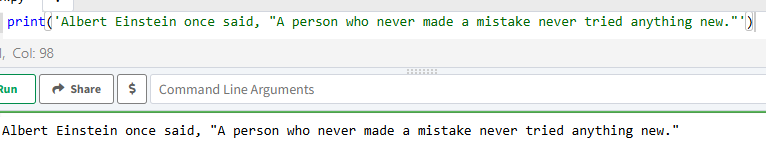
Q3. Personal Message: Use a variable to represent a person’s name, and print a message to that person. Your message should be simple, such as, “Hello Eric, would you like to learn some Python today?”



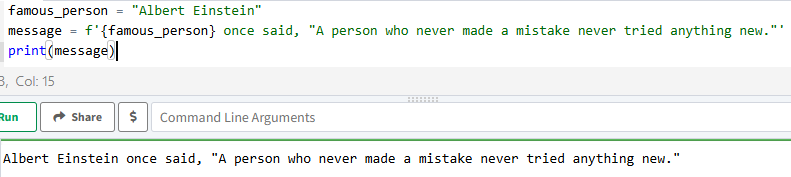
Q4. Name Cases: Use a variable to represent a person’s name, and then print that person’s name in lowercase, uppercase, and title case.



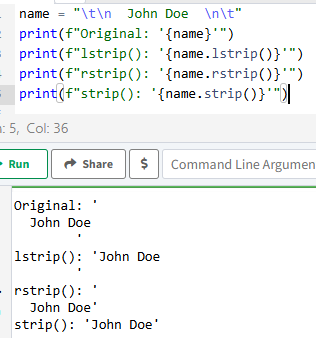
Q5. Famous Quote: Find a quote from a famous person you admire. Print the quote and the name of its author. Your output should look something like the following, including the quotation marks: Albert Einstein once said, “A person who never made a mistake never tried anything new.”



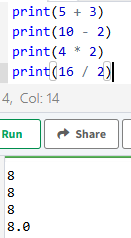
Q6. Famous Quote 2: Repeat Exercise 5, but this time, represent the famous person’s name using a variable called famous\_person. Then compose your message and represent it with a new variable called message. Print your message.



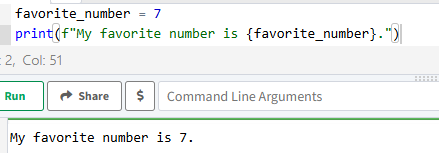
Q7. Stripping Names: Use a variable to represent a person’s name, and include some whitespace characters at the beginning and end of the name. Make sure you use each character combination, "\t" and "\n", at least once. Print the name once, so the whitespace around the name is displayed. Then print the name using each of the three stripping functions, lstrip(), rstrip(), and strip().



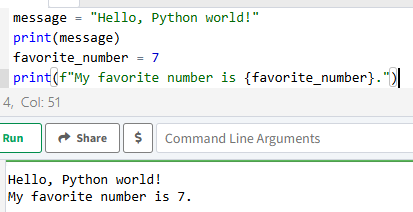
Q8. Number Eight: Write addition, subtraction, multiplication, and division operations that each result in the number 8. Be sure to enclose your operations in print() calls to see the results. You should create four lines that look like this: print (5+3) Your output should simply be four lines with the number 8 appearing once on each line.



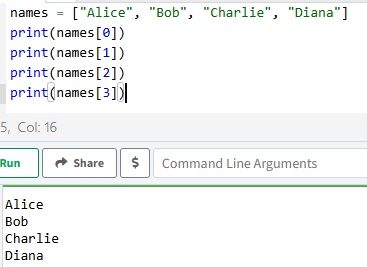
Q9. Favorite Number: Use a variable to represent your favorite number. Then, using that variable, create a message that reveals your favorite number. Print that message.



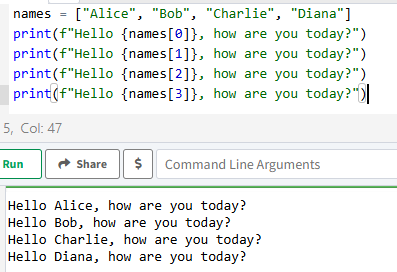
Q10. Adding Comments: Choose two of the programs you’ve written, and add at least one comment to each. If you don’t have anything specific to write because your programs are too simple at this point, just add your name and the current date at the top of each program file. Then write one sentence describing what the program does.

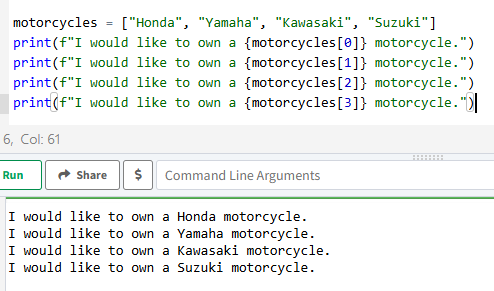


Q11. Store the names of a few of your friends in a list called names. Print each person’s name by accessing each element in the list, one at a time.

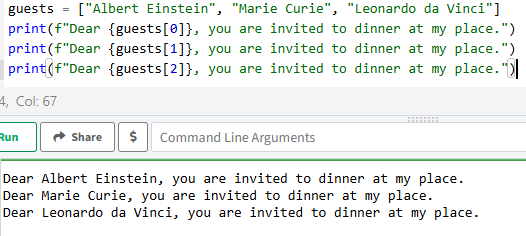
s

Q12. Start with the list you used in 11, but instead of just printing each person’s name, print a message to them. The text of each message should be the same, but each message should be personalized with the person’s name.



Q13. Your Own List: Think of your favorite mode of transportation, such as a motorcycle or a car, and make a list that stores several examples. Use your list to print a series of statements about these items, such as “I would like to own a Honda motorcycle.”

Q14. If you could invite anyone, living or deceased, to dinner, who would you invite? Make a list that includes at least three people you’d like to invite to dinner. Then use your list to print a message to each person, inviting them to dinner.



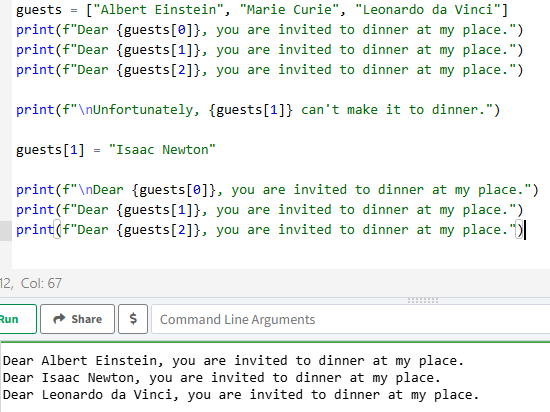
15. Changing Guest List: You just heard that one of your guests can’t make the dinner, so you need to send out a new set of invitations. You’ll have to think of someone else to invite.

• Start with your program from Exercise 14.

• Add a print() call at the end of your program stating the name of the guest who can’t make it.

• Modify your list, replacing the name of the guest who can’t make it with the name of the new person you are inviting.

• Print a second set of invitation messages, one for each person who is still in your list.



16. More Guests: You just found a bigger dinner table, so now more space is available. Think of three more guests to invite to dinner.

• Start with your program from Exercise 14 or Exercise 15.

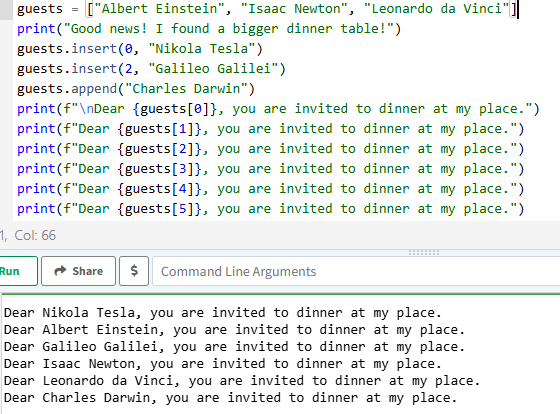
• Add a print() call to the end of your program informing people that you found a bigger dinner table.

• Use insert() to add one new guest to the beginning of your list.

• Use insert() to add one new guest to the middle of your list.

• Use append() to add one new guest to the end of your list.

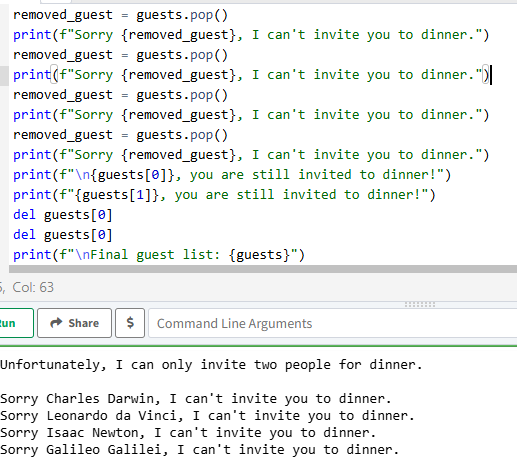
• Print a new set of invitation messages, one for each person in your list.



17. You just found out that your new dinner table won’t arrive in time for the dinner, and you have space for only two guests.

• Start with your program from Exercise 16. Add a new line that prints a message saying that you can invite only two people for dinner.

• Use pop() to remove guests from your list one at a time until only two names remain in your list. Each time you pop a name from your list, print a message to that person letting them know you’re sorry you can’t invite them to dinner.

• Print a message to each of the two people still on your list, letting them know they’re still invited. Use del to remove the last two names from your list, so you have an empty list. Print your list to make sure you actually have an empty list at the end of your program.

18. Think of at least five places in the world you’d like to visit.

• Store the locations in a list. Make sure the list is not in alphabetical order.

• Print your list in its original order. Don’t worry about printing the list neatly, just print it as a raw Python list.

• Use sorted() to print your list in alphabetical order without modifying the actual list.

• Show that your list is still in its original order by printing it.

• Use sorted() to print your list in reverse alphabetical order without changing the order of the original list.

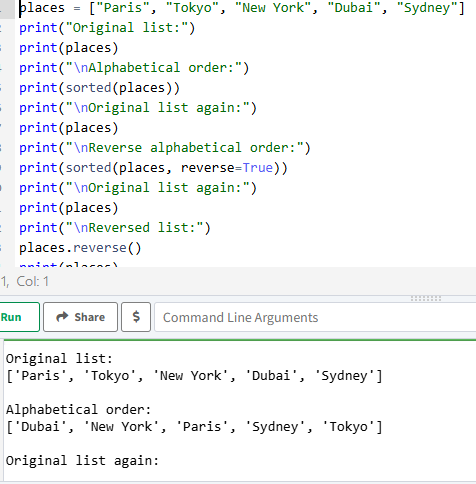
• Show that your list is still in its original order by printing it again.

• Use reverse() to change the order of your list. Print the list to show that its order has changed.

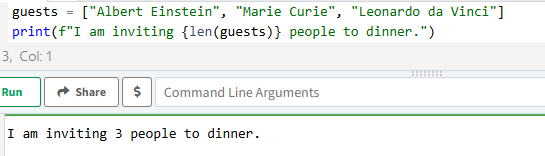
• Use reverse() to change the order of your list again. Print the list to show it’s back to its original order.

• Use sort() to change your list so it’s stored in alphabetical order. Print the list to show that its order has been changed.

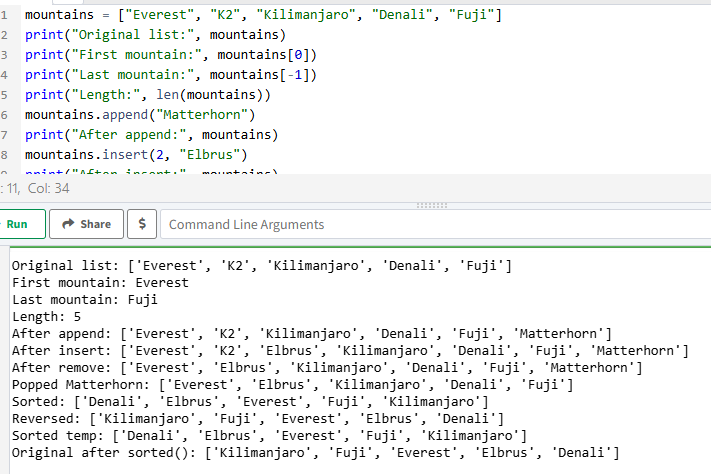
• Use sort() to change your list so it’s stored in reverse alphabetical order. Print the list to show that its order has changed.



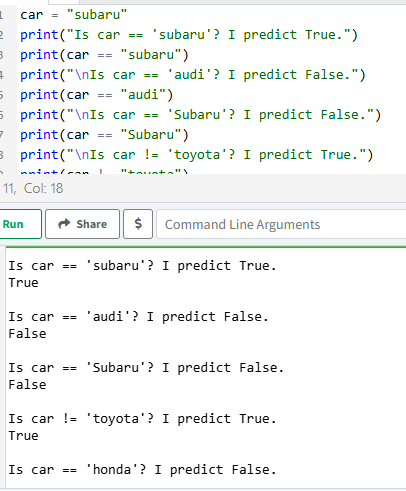
19. Dinner Guests: Working with one of the programs from Exercises 14 through 17, use len() to print a message indicating the number of people you are inviting to dinner. 20. Every Function: Think of something you could store in a list. For example, you could make a list of mountains, rivers, countries, cities, languages, or anything else you’d like. Write a program that creates a list containing these items and then uses each function introduced in this chapter at least once.



20. Every Function: Think of something you could store in a list. For example, you could make a list of mountains, rivers, countries, cities, languages, or anything else you’d like. Write a program that creates a list containing these items and then uses each function introduced in this chapter at least once.



21. Conditional Tests: Write a series of conditional tests. Print a statement describing each test and your prediction for the results of each test. Your code should look something like this: car = 'subaru' print("Is car == 'subaru'? I predict True.") print(car == 'subaru') print("\nIs car == 'audi'? I predict False.") print(car == 'audi') • Look closely at your results, and make sure you understand why each line evaluates to True or False. • Create at least ten tests. Have at least five tests evaluate to True and another five tests evaluate to False.



22. More Conditional Tests: You don’t have to limit the number of tests you create to ten. If you want to try more comparisons, write more tests and add them to conditional\_tests.py. Have at least one True and one False result for each of the following:

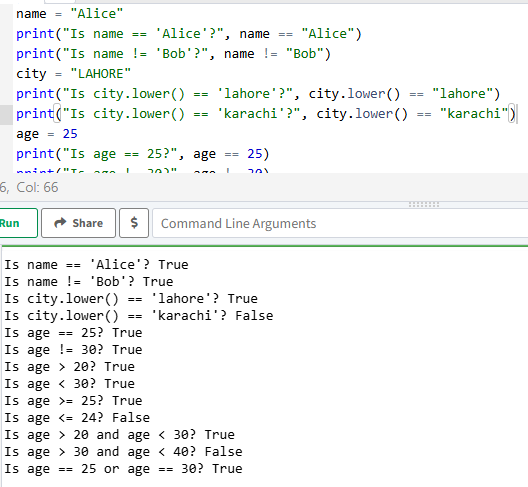
• Tests for equality and inequality with strings

• Tests using the lower() method

• Numerical tests involving equality and inequality, greater than and less than, greater than or equal to, and less than or equal to

• Tests using the and keyword and the or keyword

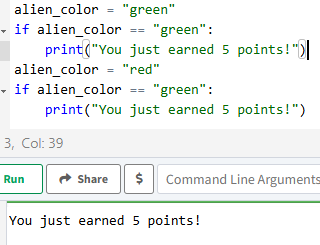
• Test whether an item is in a list

• Test whether an item is not in a list

23. Alien Colors #1: Imagine an alien was just shot down in a game. Create a variable called alien\_color and assign it a value of 'green', 'yellow', or 'red'.

• Write an if statement to test whether the alien’s color is green. If it is, print a message that the player just earned 5 points.

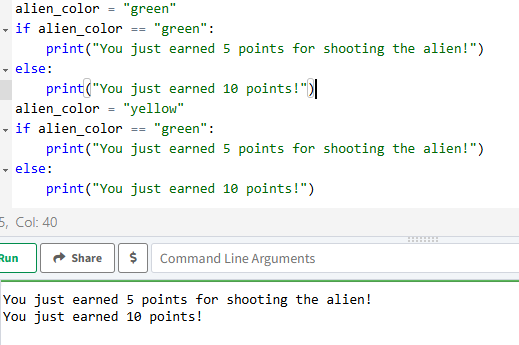
• Write one version of this program that passes the if test and another that fails. (The version that fails will have no output.)



24. Alien Colors #2: Choose a color for an alien as you did in Exercise 23, and write an if-else chain.

• If the alien’s color is green, print a statement that the player just earned 5 points for shooting the alien.

• If the alien’s color isn’t green, print a statement that the player just earned 10 points.

• Write one version of this program that runs the if block and another that runs the else block.

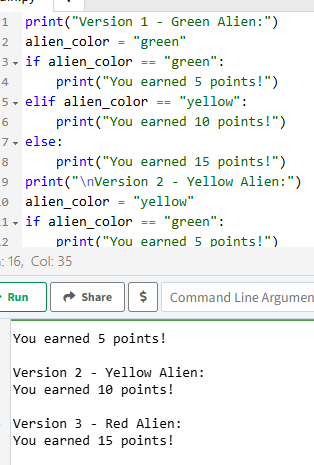
25. . Alien Colors #3: Turn your if-else chain from Exercise 24 into an if-elifelse chain.

• If the alien is green, print a message that the player earned 5 points.

• If the alien is yellow, print a message that the player earned 10 points.

• If the alien is red, print a message that the player earned 15 points.

• Write three versions of this program, making sure each message is printed for the appropriate color alien.



26. Stages of Life: Write an if-elif-else chain that determines a person’s stage of life. Set a value for the variable age, and then:

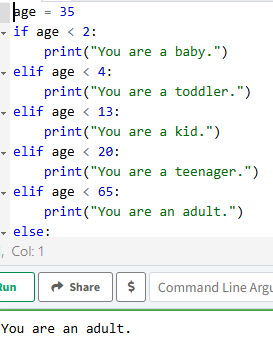
• If the person is less than 2 years old, print a message that the person is a baby.

• If the person is at least 2 years old but less than 4, print a message that the person is a toddler.

• If the person is at least 4 years old but less than 13, print a message that the person is a kid.

• If the person is at least 13 years old but less than 20, print a message that the person is a teenager. If the person is at least 20 years old but less than 65, print a message that the person is an adult.

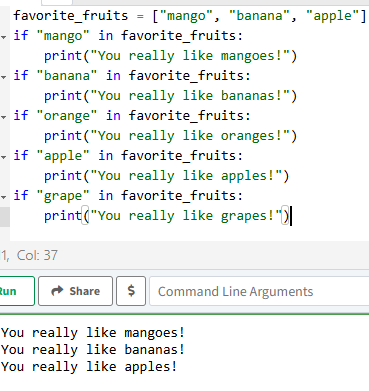
• If the person is age 65 or older, print a message that the person is an elder.



27. Favorite Fruit: Make a list of your favorite fruits, and then write a series of independent if statements that check for certain fruits in your list.

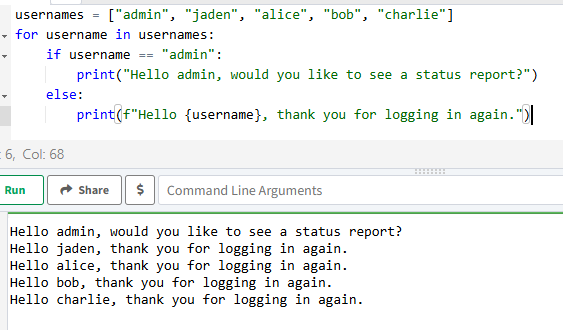
• Make a list of your three favorite fruits and call it favorite\_fruits.

• Write five if statements. Each should check whether a certain kind of fruit is in your list. If the fruit is in your list, the if block should print a statement, such as You really like bananas!



28. Hello Admin: Make a list of five or more usernames, including the name 'admin'. Imagine you are writing code that will print a greeting to each user after they log in to a website. Loop through the list, and print a greeting to each user:

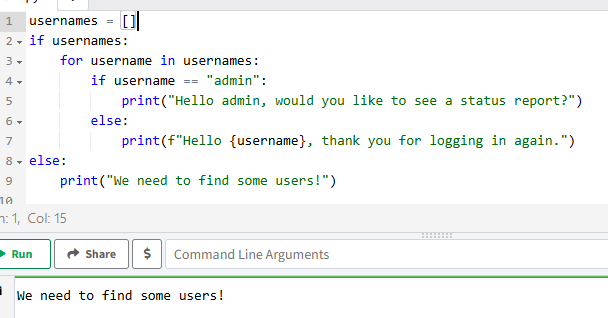
• If the username is 'admin', print a special greeting, such as Hello admin, would you like to see a status report?

• Otherwise, print a generic greeting, such as Hello Jaden, thank you for logging in again.

29. No Users: Add an if test to hello\_admin.py to make sure the list of users is not empty.

• If the list is empty, print the message We need to find some users!

• Remove all of the usernames from your list, and make sure the correct message is printed.

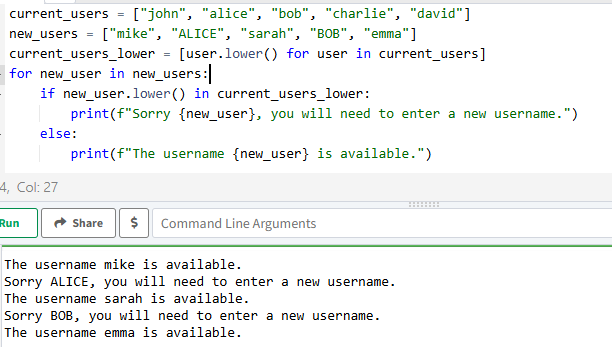


30. Checking Usernames: Do the following to create a program that simulates how websites ensure that everyone has a unique username.

• Make a list of five or more usernames called current\_users.

• Make another list of five usernames called new\_users. Make sure one or two of the new usernames are also in the current\_users list.

• Loop through the new\_users list to see if each new username has already been used. If it has, print a message that the person will need to enter a new username. If a username has not been used, print a message saying that the username is available.

• Make sure your comparison is case insensitive. If 'John' has been used, 'JOHN' should not be accepted. (To do this, you’ll need to make a copy of current\_users containing the lowercase versions of all existing users.)

31. Ordinal Numbers: Ordinal numbers indicate their position in a list, such as 1st or 2nd. Most ordinal numbers end in th, except 1, 2, and 3.

• Store the numbers 1 through 9 in a list.

• Loop through the list.

• Use an if-elif-else chain inside the loop to print the proper ordinal ending for each number. Your output should read "1st 2nd 3rd 4th 5th 6th 7th 8th 9th", and each result should be on a separate line.

