Python Semester 1 Exam Review

Table of Contents

[Instructions 1](#_Toc533586924)

[Open Book Open Computer 1](#_Toc533586925)

[Time 1](#_Toc533586926)

[Upload to Gitee 1](#_Toc533586927)

[Structure of Repository 2](#_Toc533586928)

[Working Code 2](#_Toc533586929)

[Lists 2](#_Toc533586930)

[More Lists 3](#_Toc533586931)

[Conditionals 4](#_Toc533586932)

[Algorithms 4](#_Toc533586933)

[Dictionaries 6](#_Toc533586934)

# Instructions

## Open Book Open Computer

This is an open book, open computer test. You may use any resource except for chatting with another person in or outside of class. All chat apps must be completely (**COMPLETELY!!!!!!!)** closed throughout the entire exam.

## Time

90 minutes

## Upload to Gitee

You must upload all exercises and questions to your gitee account. You will not be allowed to push until the last 5 minutes of the exam. Make sure that you have add and committed all changes prior to the last 5 minutes, so you only need to “push” to turn in the exam.

Again, YOU CANNOT PUSH TO GITEE UNTIL 5 MINUTES PRIOR TO THE END OF THE EXAM

## Structure of Repository

Your repository must be named “final\_exam”. Inside the repository should be files representing each question number. For question 1, you should push a file called 1.py, for question 2, 2.py, and so on.

## Working Code

Your code must be working to get credit. If your code cannot run for any reason you will not be given credit.

# Lists

1. Create a list containing the names of some of your favorite food. Use bracket-notation and position numbers to print each food’s name as well as a message about the food item, one at a time (do not use a for-loop). The text of the message should be the same, with only the name of the food being different.
2. Create a list of songs you like, such as Twinkle Twinkle Little Star and make a list that stores several examples. Use a for-loop to print a series of statements about these songs, such as “I love to sing Twinkle Twinkle little star in the shower.”
3. Make a list of universities you would like to attend. Then use a for-loop to print a message to each university stating your desire to attend the university.
4. Use len() to print a message indicating the number of universities you wish to attend. You should say “I want to apply to 6 universities.”, replacing 6 with the output from len().
5. You just realized that one of the universities you would like to attend does not offer enough scholarships. Start with your program from the previous question and add a print statement at the end of your program stating the name of the university that you cannot attend.
6. Change your list, replacing the name of the university that you cannot attend with the name of a new university you would like to attend.
7. Use a for-loop to print a second set of messages, one for each university in your list, stating your desire to attend the university.
8. You just received more application money, so now you can apply to more universities. Think of three universities that you would like to attend. Add a print statement to the end of your program stating that you have more application money.
9. Use insert() to add one new university to the beginning of your list.
10. Use insert() to add a new university to the middle of your list.
11. Use append to add a new university to the end of your list.
12. Use a for-loop to print a new set of invitation messages, one for each university in your list.
13. You have just been accepted to early admission by UCLA. Unfortunately, their early admission rules limit you to applying to only 2 other universities. Add a new line that prints a message saying that you can only apply to two universities. Then use pop() to remove universities from your list one at a time until only two universities remain in your list . Each time you pop a university from your list, print a message to that university letting them know you’re sorry you can no longer attend.
14. Print a message to each of the two universities still on your list, letting them know you will apply. Then use del to remove the last two universities 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 the program.
15. Movies You Want to See: Think of five movies you want to see. Store the movie in a list. Make sure the list is not in alphabetical order. Use a for-loop to print each movie in its original order. Don’t worry about printing the list neatly, just print it as a raw Python list.
16. Movies You Want to See: Use a for-loop and 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 as well.
17. Movies You Want to See: Use a for-loop and 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.
18. Movies You Want to See: Use reverse() to change the order of your list . Use a for-loop to print to show that its order has changed. Use reverse() to change the order of your list again. Use a for-loop to print the list to show it’s back to its original order.
19. Movies You Want to See: Use sort() to change your list so it’s stored in alphabetical order . Print the list to show that its order has been changed.
20. Movies You Want to See: Use sort() to change your list so it’s stored in reverse alphabetical order. Print the list to show that its order has changed.

# More Lists

1. Make a list of the numbers from one to five million, and then use min() and max() to make sure your list actually starts at one and ends at five million . Also, use the sum() function to see how quickly Python can add a million numbers .
2. Use the third argument of the range() function to make a list of the even numbers from 30 to 40 . Use a for loop to print each number.
3. Make a list of the multiples of 4 from 1 to 40 . Use a for loop to print the numbers in your list .
4. Cubes: A number raised to the second power is called a square. For example, the square of 2 is written as 5\*\*2 in Python. Use a for-loop to make a list of the first 10 squares (that is, the square of each integer from 1 through 10) and print out the value of each square.
5. Use a list comprehension to generate a list of the first 10 squares.
6. Create a list of the last 5 stores you shopped at.
   * Print the message, “The first three items in the list are: ”. Then use a slice and a for-loop to print the first three items from the list.
   * Print the message, “Three items from the middle of the list are: . Then use a slice and a for-loop to print three items from the middle of the list
   * Print the message, “The last three items from the end of the list are: ”. Then use a slice and a for-loop to print the last three items in the list.
7. Start with your program from the first question. Make a copy of the list of food and call it friend\_food. Then do the following:
   * Add a new food item to the original list
   * Add a different food item to the list friend\_food.
   * Prove that you have two separate lists.
     1. Print the message, “My favorite types of food are: “ and use a for-loop to print the first list.
     2. Print the message, my friend’s favorite types of food are: and then use a for loop to print the second list.
     3. Make sure each new food item is stored in the appropriate list.
8. Create at least 10 tests. Have at least 5 tests evaluate to True and another 5 tests evaluate to False . A test should look like this:
   * print(29+3 == 32, “This will be True”)

Or like this

* + city = “Chongqing, China”
  + print(“Chongqing, china” == city, “This will be False”)

# Conditionals

1. Write 6 conditional tests. These conditional tests should:
   * Test using the lower() function
   * Compare numbers with inequality, greater than, less than, greater than or equal to, and less than or equal to
2. Write 4 more conditional tests. These conditional tests should:
   * Test 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

# Algorithms

1. Write a program that converts a list of gpas to the appropriate percentage scale. Then convert the following list [1.3, 3.4, 3.9, 3.3, 2.8, 2.2, 3.7] into the appropriate percentage grade. You should output the gpas in the same order as the letter grades listed above.



1. Bubble Sort: Write a program that implements bubble sort on the list [38, 4, 84, 19, 54]. Your output should be the list sorted from smallest to biggest. Except for len(), you cannot use any built-in functions/methods like sort or sorted() or .sort().
2. Bubble Sort: Calculate the worst-case scenario number of basic calculations made when your program is ran with the list above.
3. Bubble Sort: Calculate the worst-case scenario number of basic calculations made when your program is ran with a list containing 4 numbers.
4. Bubble Sort: Calculate the worst-case scenario number of basic calculations made when your program is ran with a list containing n numbers.
5. Bubble Sort: Create a word document that uses summations to describe the worst-case scenario number of basic calculations made in your program.
6. Largest Difference: Write a program that implements the largest difference algorithm. Run the program with the list [77, -10, 177, 83, 68]. Except for len(), you cannot use any built-in functions or methods. Your output should be the largest difference between any two integers.
7. Largest Difference: Calculate the worst-case scenario number of basic calculations made when your program ran with the list above. Your output should be the list sorted from smallest to biggest. Except for len(), you cannot use any built-in functions/methods like sort or sorted() or .sort().
8. Largest Difference: Calculate the worst-case scenario number of basic calculations made when your program is ran with a list containing 4 numbers.
9. Largest Difference: Calculate the worst-case scenario number of basic calculations made when your program ran with a list containing n numbers.
10. Largest Difference: Create a word document that uses summations to describe the worst-case scenario number of basic calculations made in your program.

# Dictionaries

1. Create a dictionary called players. Use the names of three NBA players as the keys. Create dictionaries containing information about each of the three players such as their height, weight, and points per game. The keys for each player’s dictionary should be something like “points\_per\_game”, “height”, and “weight”.
2. Create a dictionary called friends. Use the names of three friends as the keys. Create dictionaries containing information about each of the three friends such as their age, which school they go to, and their hobbies. The keys for each friend’s dictionary should be something like “age”, “hobbies”, and “school”.