Published

Week 12: Search Sort 5 of 5

12 Week 12 Assignment

Computer SCIENCE

The first two tasks are relatively simple. They are designed to give you experience with list comprehensions and using lists to get an unknown quantity of input values from a user. For task 3, make sure to read the assignment description carefully. The task is comprised of a number of smaller problems. Try to break them down into pieces and create a good design plan for each piece.

Objectives

1. Lists
2. Operator Overloading
3. List Comprehensions
4. Search
5. Sort

Task 1

Starter Code: **unit6_task1_starter.py** (https://usu.instructure.com/courses/681553/files/81729295/download?download_frd=1)

Use the starter code and change the name to task1.py. Fill in the blanks as indicated with <Fill In>. You should only fill in a single line of code. DO NOT change any other code. Note that your code will replace that entire marker including < and >. You will print out three lists. Each list must be based on the baseList.

- list1 will contain all even valued numbers in baseList
- list2 will contain values from baseList that are multiples of 3 and less than 50
- list3 will contain values that are 10 times greater than the numbers from baseList that are multiples of 5 and greater than 50

Sample output. Note the ... is filler because the actual list is long

```
[2, 4, 6, 8, 10, 12, 14, 16, 18, 20, ... , 96, 98, 100]
[3, 6, 9, 12, 15, 18, 21, 24, 27, 30, 33, 36, 39, 42, 45, 48]
```

[550, 600, 650, 700, 750, 800, 850, 900, 950, 1000]

Note: No software dev plan or UML required

Task 2

Call your file task2.py. Write a program that will prompt the user to enter a number. Keep collecting numbers until the user hits enter without entering a number. Once the user hits enter without entering a number, calculate and display the following:

- Number of values entered
- Maximum value. Write your own code. Do not use a built-in function.
- Minimum value. Use a built-in function
- Sum of all values. Use a loop to calculate.
- Average value. Use your calculated sum to calculate.

Note: No software dev plan or UML required

Task 3 - Gronky

Starter Code: **unit6_task3.zip** **(https://usu.instructure.com/courses/681553/files/81729296/download?download_frd=1)**

You will complete a program for a brand-new kind of card game. The game is called *Gronky*. It is similar to a regular deck of cards, with some slight changes. Starter code has been provided. Use the starter code and only fill in required areas. This includes the spots marked <Fill-In> and adding missing functions or methods as indicated. You should not be modifying existing code. Your submission should include five files: task3.py, gronkyutil.py, deck.py, card.py, plan3.txt.

Demo Video

Overview

- You will figure out how many cards you need in the deck to fulfill the requirements
- Each card has three components
  - Id: unique integer for each card (lowest id number is 0, largest is one less than number of cards)
  - Title: Numbers one through ten, Baker, Jester, Page, Scribe, Squire, Armorer, and Marshal
  - Gang: Jets, Pollos, Slugs, Yokels, Keiths, and Elbows
- Each card is unique, meaning there are no cards that match all three components. The Id can be used to determine its title and gang.
- Every possible combination of Title/Gang must be present in the deck of cards

Requirements

- Create a file called plan3.txt
  - Include the Software design for your search and sort functions (three total)
  - No other UML, design, testing, or plan is required.
- Add to a class called Card in card.py
  - Used to create a single card for the deck
  - Fill in partial lines of code for getTitle() and getGang()
  - Add two dunder methods in order to meet other program requirements
- Add to a class called Deck in deck.py
  - Used to create a single deck of cards
  - Fill in three lines/sections as indicated

- Use the module called gronkyutil.py
  - This stores the map for TITLE and GANG
  - It contains a function you can use to check to see if your id to title/gang conversion is working properly. It may also come in handy somewhere else (hint hint).
  - Fill in areas as indicated
- Change the unit6_task3_starter.py file name to task3.py.
- Main menu
    - Each menu option should result in a function call, except for option 5
    - The function signature for option 1 is included and should help you with the others
    - Display
      - Display each card in the hand on a single line
      - Use the print() with the card object as the parameter
        - Ex. print(card1)
        - This means you should not call a method on the card object to determine what to print
      - Each card should be displayed as "Title of Gang"
        - Ex. Baker of Slugs
    - Sort by Title
      - Perform a sort using a selection, insertion, or bubble sort.
      - A message should indicate which sort is used
      - Use an existing function in the code to make it appear as if the program is processing
      - Do not display the list of cards. The user should choose to display it after the sort is complete
    - Sort by Gang
      - Perform a sort using a selection, insertion, or bubble sort.
      - Do not use the same sort technique as the Sort by Title
      - A message should indicate which sort is used
      - Use an existing function in the code to make it appear as if the program is processing
      - Do not display the list of cards. The user should choose to display it after the sort is complete
    - Search for Card
      - Display submenus to allow the user to select a Title and Gang for the card to search for
      - Use an existing sort algorithm to sort the hand
      - Perform a binary search to determine if the card is in the hand
      - Display a message indicating whether or not the card is in the hand

Points 100

Submitting a text entry box or a file upload

Due	For	Available from	Until
Dec 3, 2021	Everyone	-	-

Unit 6

Criteria	Ratings		Pts
Task 1: <--Fill in sections--> completed properly and based on baseList	**4 pts Full Marks**	**0 pts No Marks**	4 pts
Task 1: Proper output for all lists	**3 pts Full Marks**	**0 pts No Marks**	3 pts
Task 2: Proper use of list to collect input	**5 pts Full Marks**	**0 pts No Marks**	5 pts
Task 2: Complete input loop properly when a user hits enter without input	**3 pts Full Marks**	**0 pts No Marks**	3 pts
Task 2: Properly process numbers where the user adds spaces before or after the number	**3 pts Full Marks**	**0 pts No Marks**	3 pts
Task 2: Proper calculation and output of all required values	**3 pts Full Marks**	**0 pts No Marks**	3 pts
Task 3: Software Design Only required for search and sort functions	**3 pts Full Marks**	**0 pts No Marks**	3 pts
Task 3: task3.py completed properly Fill-in sections completed properly as indicated in file. Appropriate functions are implemented.	**8 pts Full Marks**	**0 pts No Marks**	8 pts
Task 3: Card class completed properly As indicated in file, fill in two spots, not using numeric literals, and define two dunder methods. No other methods are allowed.	**7 pts Full Marks**	**0 pts No Marks**	7 pts

Criteria	Ratings		Pts
Task 3: Deck class completed properly Fill in three areas as indicated in starter file. Only single lines of code or partial lines are allowed. Do not use a numeric literal as indicated.	**7 pts Full Marks**	**0 pts No Marks**	7 pts
Task 3: Binary search implemented properly Use the appropriate sort method first. Then perform a binary search on the list of cards. Properly indicate if the card is in the hand or not. thinking() function used properly to indicate processing.	**13 pts Full Marks**	**0 pts No Marks**	13 pts
Task 3: Sort by Title Implement a selection, insertion, or bubble sort based on the card title. Use the thinking() function to indicate the processing is happening. Display an appropriate message indicating the sort algorithm used.	**13 pts Full Marks**	**0 pts No Marks**	13 pts
Task 3: Sort by Gang Implement a selection, insertion, or bubble sort based on the card gang. Use the thinking() function to indicate the processing is happening. Display an appropriate message indicating the sort algorithm used. Zero points if the same sort algorithm as sort by title is used.	**13 pts Full Marks**	**0 pts No Marks**	13 pts
Task 3: Hand printed properly You must print a hand by printing card objects: print(cardObject). You may not print using a string that is returned from a getter type method.	**6 pts Full Marks**	**0 pts No Marks**	6 pts
Task 3: Proper use of gronkyutil.py Use in appropriate places for creating cards, and menus (as listed in description)	**6 pts Full Marks**	**0 pts No Marks**	6 pts
Task 3: Quit exits program properly	**3 pts Full Marks**	**0 pts No Marks**	3 pts
			Total Points: 100