

Theater Ticket System Project in Python

A program that allows users to select and purchase theater tickets at the “Uptown Theater” based on price, location, and availability, while tracking sales for the theater staff.

To start the program each day the manager must login. The main (manager) window will appear with a login, create password, and cancel button. A password must exist (see file later) for the login button to be enabled. The password is created in a separate window and must be 9 characters or more, and it must have at least one digit, uppercase and lowercase letter. The program will continue to show error messages and prompt for a password until a valid password is created. The valid password will be stored in a file for validation on login.

When a valid password has been created or on subsequent program runs, the main window “Create Account” button for the manager will be disabled. When the login button is clicked, the main window will change (morph) to add a login entry box.

When the correct password has been entered, a price selection window for the manager will appear requesting a “Pricing Selection” and a choice of Matinée or Evening pricing. The selection will determine which file is read in to load the program pricing for seating in the theater. Files are used to allow for text edit of pricing for seasonal and special event pricing.

When the pricing selection has been made, the Theater Kiosk window will appear with the theater seating and the interface for customers. The design must accommodate a user friendly seat selection and ticket purchasing interface (see the theater seating image later in this document), and the current time and next “show time”.

When a seat has been selected for purchase, the user must be able to unselect that seat or choose another. When a seat has been purchased (we’ll assume the payment is made), a ticket will be printed (displayed in a separate window), and the manager screen will be updated to reflect the change.

Sold out sections must be disabled and handled in some user-friendly way, as well as reported on the management display. When all of the seats have been sold in the theater, a banner should appear on the theater display and manager screen with the text “This showing has been sold out.”, and the interface should be disabled.

The program is launched from a computer by the manager, and runs on a kiosk in the theater lobby. The manager monitors seat sales by section in another window or web page on her computer, which also displays total ticket sales in number of seats and dollars. The data is also displayed graphically, and is near-real time.

A **Design Document** is required and will be presented as a part of project demonstrations and submitted as a final submission. It will include step-by-step implementation screen captures, descriptions, and explanations of functionality in the program (see the sample file).

Presentations are required during the semester and a final **Presentation/demonstration**. The Design document will be presented first during each presentation, and then the program operation will be demonstrated.

The screen captures in this document are for reference only. The design and interface of your program do not have to mirror the examples in this document in terms of appearance and widgets used, but must handle the operations.

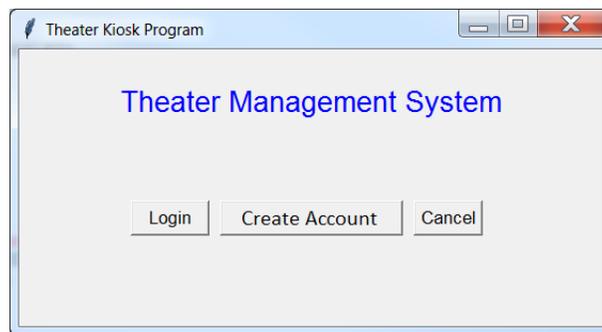
Assignment

Get Python with IDLE and create a working program using the “Getting Started in Python” exercise. The latest IDLE comes with Python 3.5.1 and above from python.org. Determine the installation location. IDLE will run fine on a flash drive. Create and run a “Hello World”-like program. Create a design/development status document that you will update as you complete the project. This will be reviewed along with the code at the Milestones. See the example design documentation.

Project Milestone #1

Begin the **Design/development** documentation with a generalized pseudo-code description of the order of operations for the program. Use this document for ideas and issues as well to document progress.

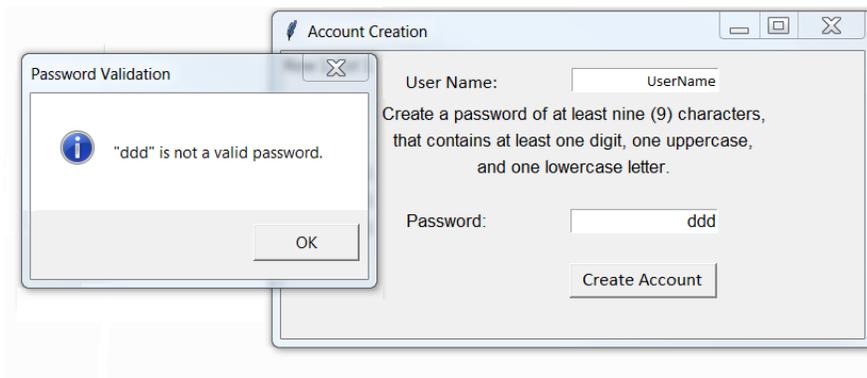
Implement the main loop for the program and initial main window using a grid and button widgets.



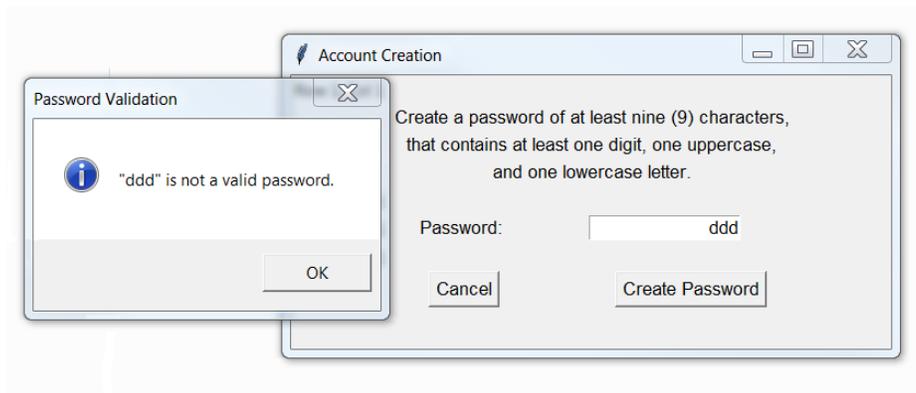
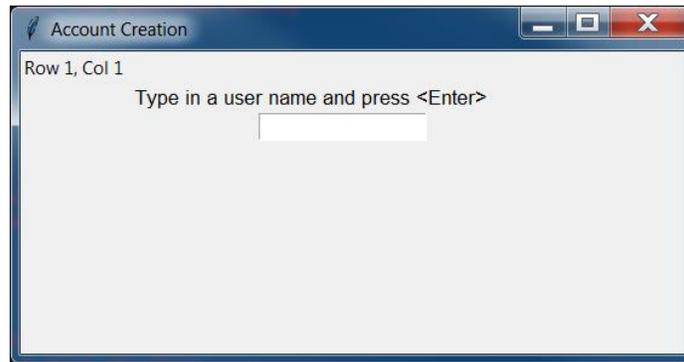
Create the “Account Creation / Login” interface and the functions using a class for the GUI in a separate module (file) and begin the design for user name and password file handling. The window is launched when the “Create Account” button is clicked. Use the <enter> key to obtain the user entry.

Design and develop the functionality for changing the window or creating a new one to obtain a user name and password including input validation. Create an error dialog that echoes the password, and create the password validation function which must test for length (9 characters or more), an uppercase and lowercase letter, and a digit. The window must contain instructions and operating “Cancel” and “Create Account” buttons.

This can be done in a single window as shown here, or as a two (2) step process.



Two step process:



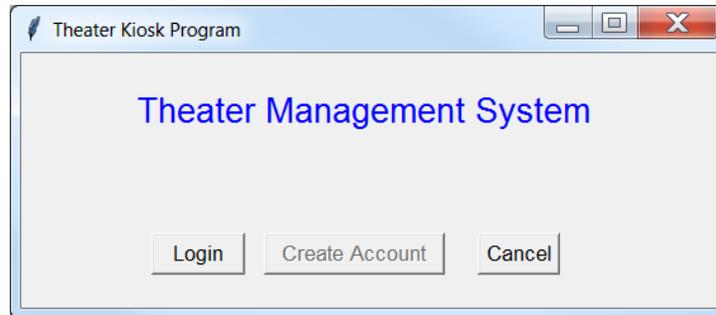
When the “OK” button is clicked, the previous password should be cleared.

`password_entry.delete(0,END)` # clear the text



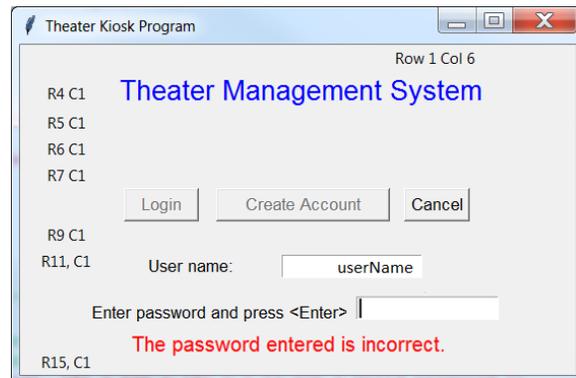
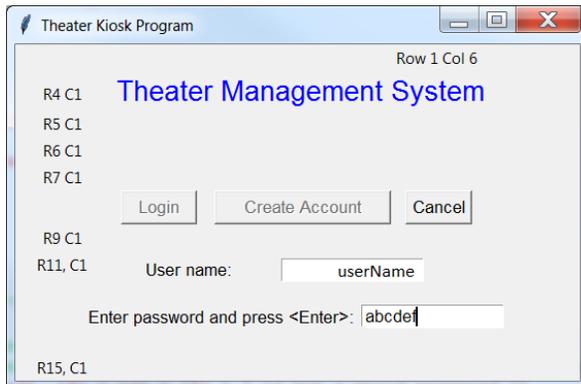
When the “Ok” button in the “Password Accepted” dialog is clicked the Account Creation dialog should be destroyed and the main window will now have the “Create Account” button disabled. Store the user name and password for future checking. The user should login next.

Hint: The call to the login function may not be able to be handled directly through the “command=” option of the button. A call to a function within the main GUI (outside the main loop) may need to be called which then calls the login function in the separate module.



Add the code to the login function to accept the password when the <Enter> key is pressed by binding the entry widget to a function call that tests for a match with the stored password by calling a “verify” function. The entry widget should accept input left aligned in the text box. If the entered password and stored password don’t match, alert the user and the text box should be cleared like the account creation example- `entry.delete(0, END)`.

Login operations can be done on the main window as shown here or in a separate window. Consider how the user and manager would like to interface with your program.



Note: It may be easier to locate widgets by using labels for unoccupied rows and columns to show where they are in the grid. Use `rowspan`, `columnspan`, and `padx` to align widgets in the grid columns.

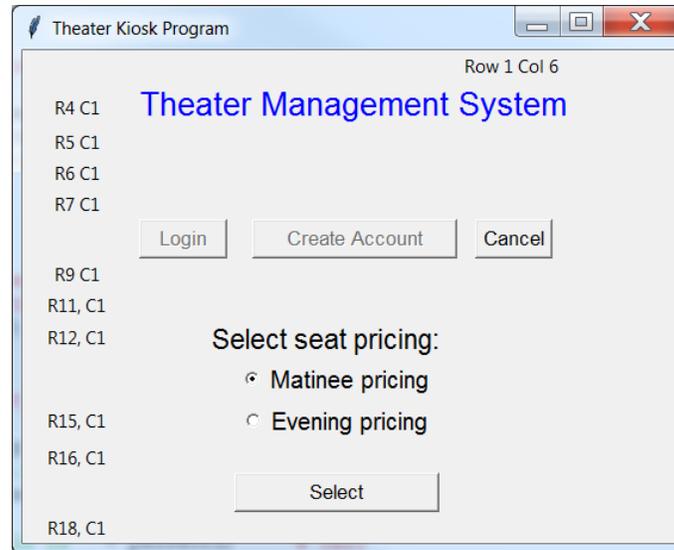
STATUS: At this point, there should be a main loop for the program with the main interface. There should be a separate file with a class that handles the creation of a user account that contains a function for validating the password and storing a valid password. There should be a function in main that creates an instance of the account creation object and enables/disables buttons as appropriate. There should be a login function in the separate module.

Design/development documentation – update the document to include screen captures of all working functionality with explanations and the code. Include all code at the end of the document.

Project Milestone #2

Once the login is successful, the manager will select the seat pricing depending on Matinee or Evening prices. The main GUI should change to allow selection. . There is no reason to create another window for this functionality. This pricing choice will select the pricing input file for the program.

Implement the function in a separate file that will again modify the window to allow the manager to select the pricing for this particular showing. The choices are “Matinee” and “Evening”, and the selection can be made using radio buttons (which are mutually exclusive) or any appropriate widget. Again, your design and operation do not have to mirror those shown in this document.



Once one of the pricing selections is made, and the “Select” button is clicked, the theater display and user interface should begin.

Hint: Use the config and hide attributes, and the destroy function as appropriate to modify the window.

Example: `self.select_button.destroy()` # destroy the select button

Begin writing the function that will display the theater seating and allow users to select a seat to purchase. This function should again modify the main window to display the entire theater (GIF provided) and provide the user interface. The function can be in another module. Begin the interface design and how users will interact with the program. Decide on a design for handling the various sections and pricing.

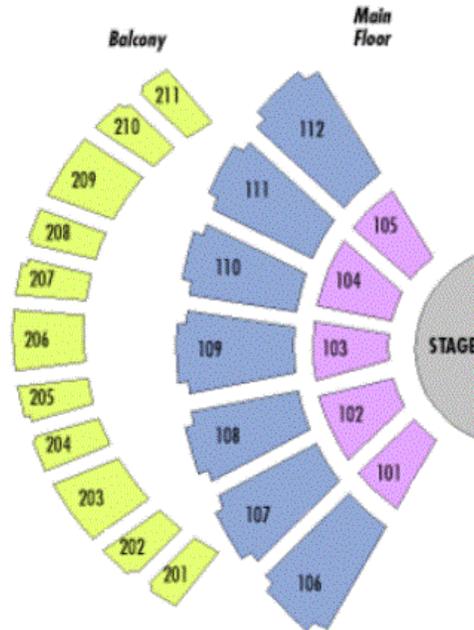
Also, begin cleaning up the appearance of the program including the various displays and window handling. Windows should appear centered on the desktop, and widgets should be for the most part aligned or centered.

Design/development documentation – update the document to include screen captures of all working functionality with explanations and the code. Include all code at the end of the document.

Project Milestone #3

Determine the design for the theater display and create the pricing files. Pricing files are used to allow for changes to seat pricing for seasonal adjustments or special events. The file data should be read in after the pricing selection is made by the manager. It shall not be hard-coded in the program. Pricing should also be displayed in some way to customers.

Theater Specifics: The theater seats are sold by section and prices vary by section. Seating in each section is first come, first served until a section is sold out. Show times are: Matinee 2:00, Evening 8:00.



Seat pricing and section capacity (750 seats total) are as follows:

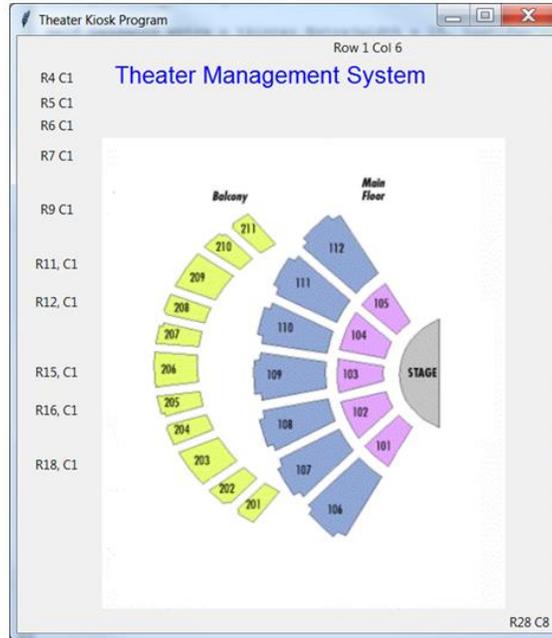
<u>Sections</u>	<u>Evening</u>	<u>Matinée</u>	<u>Capacity</u>
101 – 105	\$80.00	\$50.00	150 seats total
108 – 110	\$60.00	\$40.00	120 seats total
106, 107 & 111, 112	\$50.00	\$30.00	200 seats total
Balcony			
205, 206, 207	\$40.00	\$20.00	80 seats total
203, 204 & 208, 209	\$30.00	\$15.00	120 seats total
201, 202 & 210, 211	\$20.00	\$10.00	80 seats total

Hint: For testing sold-out algorithms, reduce the number of seats in a section or sections.

Once the login is successful, and the seat pricing has been selected, the theater seating should be displayed. This requires again morphing the main GUI. The widgets that are no longer needed should be destroyed.

The theater seating image is a GIF which is supported by the language, but a reference must be retained (shown in the code below).

```
# get the image
photo = PhotoImage(file= "Theater.gif")
self.labelGIF = tkinter.Label(image= photo)
self.labelGIF.image = photo      # retain a reference
self.labelGIF.grid(row = 7, column = 3, columnspan=5, rowspan=20)
```

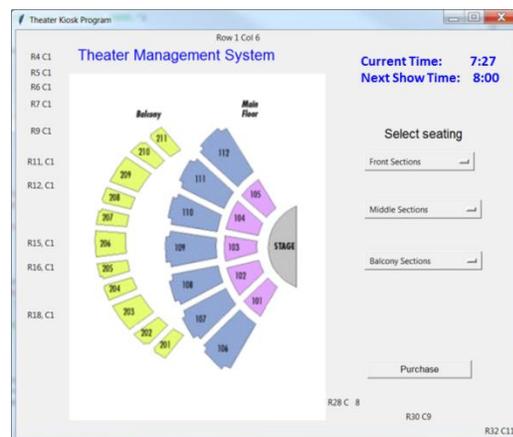
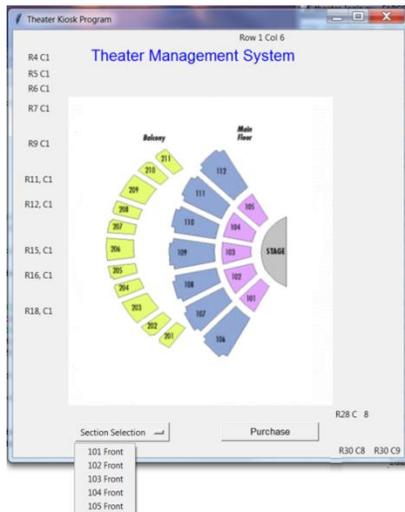


Also, the current time and next show time needs to be displayed (as shown below right).
Design the algorithm for reading and storing the seat price data from the file.

Design/development documentation – update the document to include screen captures of all working functionality with explanations and the code. Include all code at the end of the document.

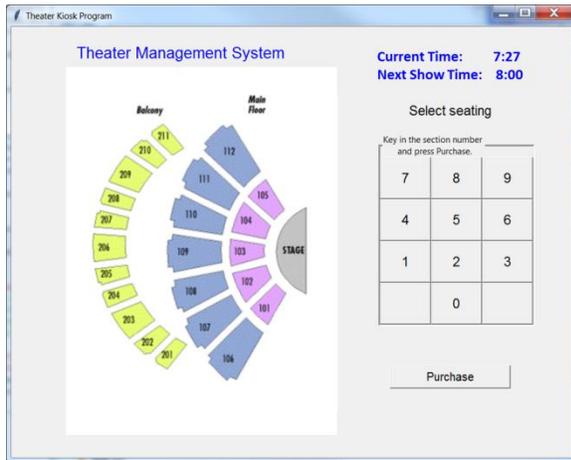
Project Milestone #4

Design the algorithm for seat selection and maintaining the number of seats purchased for each section. Consider how this should be handled from a user perspective.



Above left, an option list is used and a purchase button. Above right, multiple option lists are used.

A keypad could also be used in the interface to allow users to key in the section number or select them directly. This would need to display pricing for the user. Consider how you would want the program to run if you were a customer. When a section is sold out, the selection should no longer be available for purchase by a customer. Note that the window on the left shows the time and Next Show Time. The window on the right has a new title since this is the purchase window.



Design/development documentation – update the document to include screen captures of all working functionality with explanations and the code. Include all code at the end of the document.

Project Milestone #5

Begin the design for the manager window or program that will display the ticket sales status. Keep in mind that the display must indicate sales by section and alert the manager to sections that are sold out, and display total seats sold and total sales figures.

The manager display will show the number of tickets sold by section in text and graphically, and the total sales in dollars.

