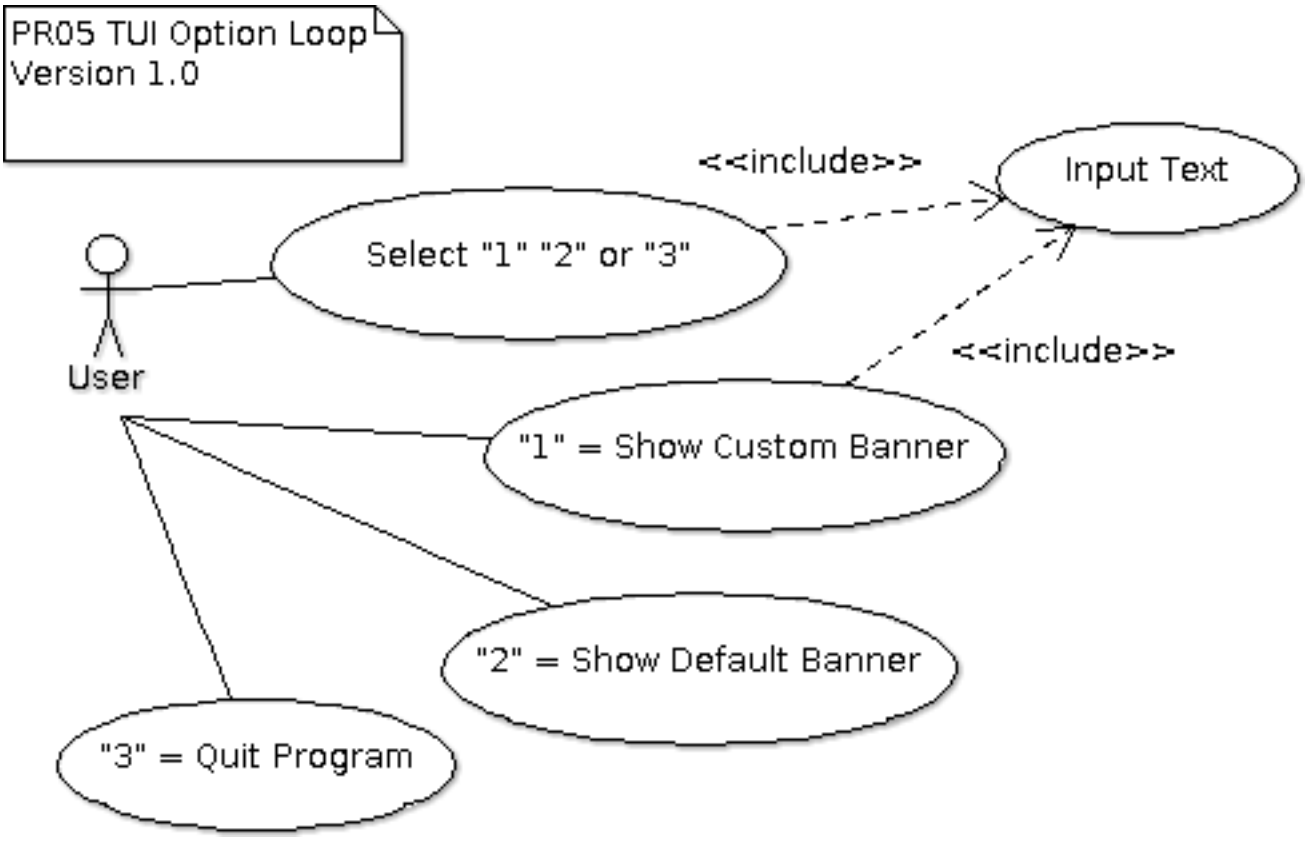
Problem Domain:	Textual User Interface (“TUI”)
Mission Summary:	Create an “Input Option Loop”
Prerequisite:	Completion of “Python 1000” (available at Udemy.com)
Your Script Name:	PR05_MenuChoices.py
Solution Name:	PR05S_MenuChoices.py
Version:	1.0 [Step By Step Edition]

Synopsis

Once activated, Textual User Interfaces can allow users to choose between several options.

In this exercise, we will create a program that will “loop” until a specific option-string has been entered.



Please note that pure string options are to be both collected, as well as used throughout this application[i].

Requirements

1) Create a script named PR05_MenuChoices.py
2) Review the Activity Diagram located in the “**Related Diagrams**” section below
3) Match the following terms & requirements (or “states”) to those in the Activity Diagram:
4) GREETING:
   A) Display “WELCOME to PR05!”
   B) Display three (3) empty lines

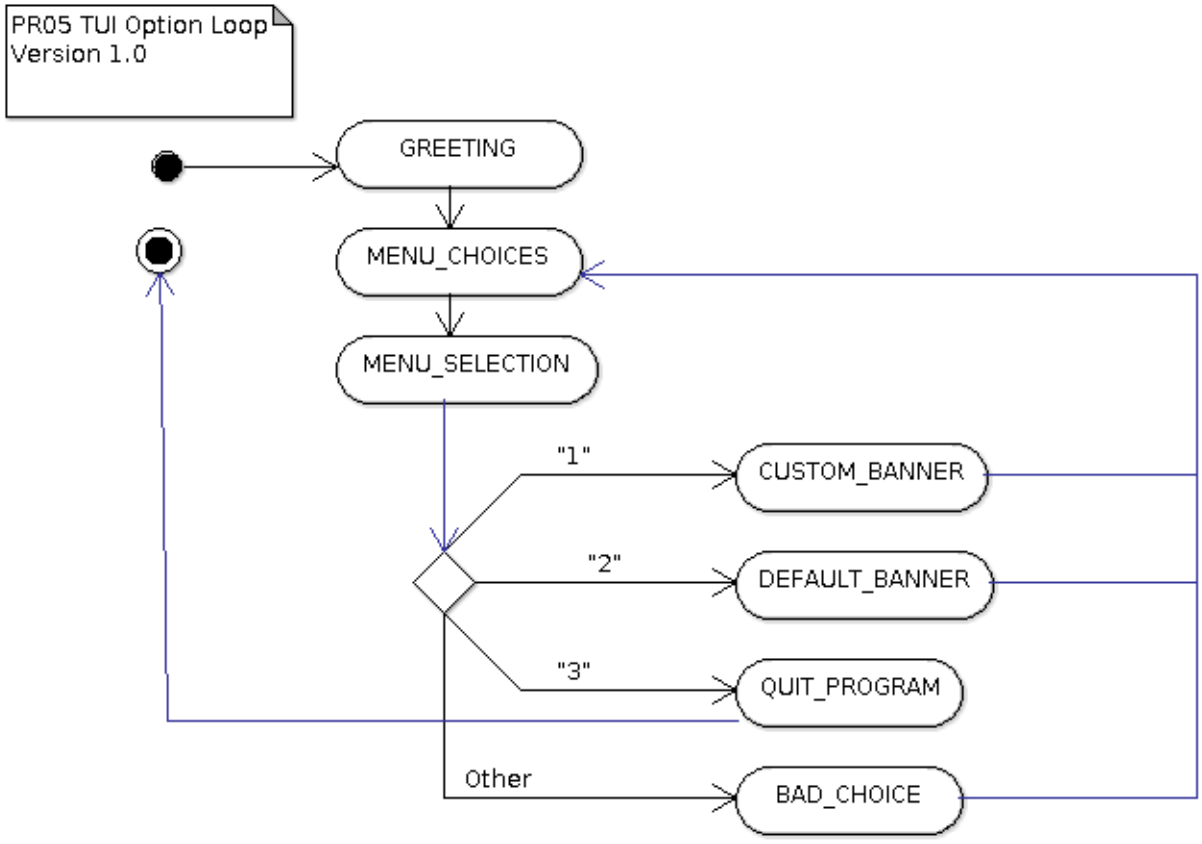
5) MENU_CHOICES:
   A) Option “CUSTOM_BANNER”
      1) Present as option “1.) Display Custom Banner”
      2) Allows user to input a custom message to be displayed as a banner
   B) Option “DEFAULT_BANNER”
      1) Present as option “2.) Display Default Banner”
      2) Allows user to display “Hello User!”
   C) Option “QUIT_PROGRAM”
      1) Present as option “3.) Quit Program”
      2) Allows user to quit the program / break the main loop
   D) Condition “BAD_CHOICE”
      1) The entry of any other text shall display the message “Bad Choice”
      2) The BAD_CHOICE message shall be displayed upon a single line
      3) After the BAD_CHOICE message has been displayed, the program will re-display (loop to / resume) the MENU_CHOICES operation

6) MENU_SELECTION:
   A) The user need only input “1”, “2”, or “3” followed by the “enter” key
   B) The use of the enter key is required

7) BONUS
   A) Re-use one of the BANNER functions created for Python 1000
   B) Change CUSTOM_BANNER to display the current date & time as a banner

Related Diagrams

The main Activity Diagram depicts a graphical summary of the requirements, as well as a reasonably demonstrative operational overview:



BONUS SECTION: Step By Step Instructions

We encourage more advanced students to attempt to complete this activity using the above requirements documentation. The following instructions are provided for those who need a little more experience in programming from requirements.

Incremental Approach

Detailed designs often leave new students feeling overwhelmed. Yet the best way to walk a mile is to take one step at a time!

Let's break the above requirements into a series of smaller challenges:

1.) While Challenger
    a. Use the "while" keyword
    b. Use "while" to "loop over"
        i. An indented "block"
        ii. Asking for input from the user
    c. Break out of the "loop" when "3" is input
    d. Test your solution. Verify that:
        i. Inputting "3" exits the "while" loop
        ii. Inputting anything else will not

2.) Display Challenger
    a. Update the above to display the list of MENU_OPTIONS
    b. Add support for BAD_CHOICE, as defined above
    c. Test your solution. Verify that:
        i. Each option is displayed
        ii. Your "while loop" works as required

3.) Control Challenger
    a. Update the above to process MENU_SELECTIONS
    b. Ignoring the banner requirement for now:
        i. Use print() to display the default message
        ii. Use print() to display your input
        iii. Use print() to display an appropriate exit message
    c. Test your solution. Verify that:
        i. The "while loop" works as before
        ii. The CUSTOM_BANNER, DEFAULT_BANNER, and QUIT_PROGRAM messages will display (not banner!) their messages, as required

4.) Banner Challenger
    a. Use what we learned in Python 1000 to create a "Banner" function
    b. Replace the above BANNER print() displays with calls to the "Banner" function
    c. Verify that:
        i. The "while loop" works as before
        ii. The CUSTOM_BANNER, DEFAULT_BANNER, and QUIT_PROGRAM now BANNER the menu choices, as required

5.) Menu-List Challenger (optional)
    a. If you have not already done so, re-factor your solution to:
        i. Keep all options in a list
        ii. Display each option from the list
    b. Verify that your program works as expected

6.) List of Tuple Challenger (optional)

a. Update your list of menu-option strings to be a list of tuples
b. In addition to the menu-option string, include and display the menu item’s number
c. Update your code to use the menu-items number for input comparison
d. Verify that your program works as expected

7.) State Machine Challenge (optional)
a. Define functions to manage CUSTOM_BANNER, DEFAULT_BANNER, and QUIT_PROGRAM
b. Update your menu-tuple to include each function definition
c. Call each functions in response to the proper input
d. Verify that your program works as expected

8.) Discussion
a. What is the advantage / disadvantage of using a state machine?

ℹ In order to avoid type-conversion errors / exceptions, the conversion from textual input-strings to other types (int, etc.) is NOT to be used. The use of Exceptions will be covered in Python 2000.