**Program for personal finances**

***Modifications to Instructions:***

* ***Original Program instructions:***
  + Program cannot show red syntax error code at any time
    - Rather than loop or continue if user messes up, the program will ask user to restart again. It is recommended to use sys.exit() or except the statements to end the program
    - Should only have one and only one try-except in your program to prevent syntax errors from crashing the program
  + ***Modification:*** 
    - I have used multiple while statements where errors could occur. Instead of restarting the program again and filling out all information repeatedly whenever there is an error, I believe it is good practice to just loop until desired input for a particular prompt is achieved.

**Key Points:**

* Should ask the user for a series of inputs
* Validate user’s entries to prevent bad data from entering or crashing the program
* At least 3 functions plus main()
  + At minimum:
    - These functions might validate input is numeric or alphabetic
    - Calculate an equation and return result
    - And etc
    - Only main() is allowed to ask the user for input
      * Other functions may not ask for input, but the main() may pass data to other functions
      * Use functions to avoid duplication/redundant code
* Should only have one and only one try-except in your program to prevent syntax errors from crashing the program
* Program cannot show red syntax error code at any time
  + Rather than loop or continue if user messes up, the program will ask user to restart again. It is recommended to use sys.exit() or except the statements to end the program

**Data Validation Scenarios:**

* Dollar amounts may be entered with $, comma and/or period like: $45,000.00 or simply 45000 or any combination thereof
* Program will check that numbers entered are numbers
* Name cannot contain numbers or characters
* DOB must contain numbers within a range of months, days and years (1900 through the current year)
  + Although it is not required to check that the exact date entered is on the calendar (for instance, program does not check if February 30 is invalid because 2 and 30 are within normal range for month and day respectively
  + Program tells the user which number was invalid (dd/mm/yy) but only the first invalid number
    - If input is invalid for both month and date, it will only tell them the month was invalid number because it is typed first
* If user enters salary less than $1000, the program will ask if they have entered monthly by mistake
  + If yes: display error message and ask to restart the program
  + Is no: allow user to continue

**Output:**

* All numbers are formatted with a $, thousands separator and cents
* If user forgets to capitalize their first/last name, program will fix it in the output
* Based on user inputs:
  + Program will either tell them they spend too much or they have extra money to save or they break even
    - Debt is not shown as negative number
* User can enter yes, y, no, n for some questions within the program
  + if user leaves it as blank, program will automatically assume it as no
* Although user enters 0 within DOB, output will not show it (ex: 1/8/2020)

**Pseudocode/Process flow**

1. Prompt the user to enter first and last name
   1. If input is valid-only letters and space, prompt user to enter annual salary (also, input can be entered as $45,000.00 or 45000)
   2. If input is invalid:
      1. Display:
         1. Response can only contain letters
         2. Loop until desired input is achieved
2. Prompt the user to enter annual salary
   1. If input is valid-entered number is more than 1000 and it contains number, prompt the user to enter date of birth
   2. If input is not valid-2 scenarios:
      1. Invalid value was entered, display:
         1. Response must be numeric
         2. Loop until desired input is achieved
      2. Value was correct but is less than $1000:
         1. Prompt user to answer following question:
            1. Did you enter annual salary (yes or no?)

If yes or y, display next prompt for DOB

If no or n or (blank), display error messages:

Annual salary should be at least $1,000

Restart program to try again (system exits)

1. Prompt the user to enter DOB
   1. If input is valid-in mm/dd/yyyy format and valid date range was entered
      1. Additional data validation:
         1. It is not required to check that the exact date is on calendar
            1. Thus, program will check for the max 31 days for all months
         2. Year should be: 1900 through 2022
         3. Mm and dd should be within valid range (max 31 dd and 12 mm)
   2. If input is invalid-2 scenarios:
      1. Format is not valid
      2. Format is valid but date range is not valid:
         1. MM or DD or YYYY is not within the valid range ((max 31 dd and 12 mm)
         2. Program display message about which one was wrong (if mm and dd are both wrong, mm will be mentioned only)
      3. In both scenarios:
         1. Display reason of error
         2. Loop until desired input is achieved
2. Prompt the user to enter hours worked per week on average
   1. If input is valid-only numbers were entered, calculate and display hourly wage per week
      1. Calculate
         1. Income per week = salary /52
         2. Income per hour = income per week / hours per week
         3. Display approximate Hourly wage per hour
   2. If input is not valid:
      1. Display error message
      2. Loop until desired input is achieved
      3. After getting desired input, execute (if input is valid scenario)
3. Prompt the user to enter tax rate
   1. If input is valid-only decimal number (within 0 and 1), calculate and display estimated taxes per year and income per hour after taxes
      1. Calculate:
         1. Tax = tax rate \* salary
         2. Net salary = salary – tax
         3. Display estimated taxes per year and income per hour after taxes
   2. If input is invalid-2 scenarios:
      1. Input is not decimal:
         1. Display error message
         2. Loop until desired input is achieved
      2. Input is decimal but not within the range of 0 and 1:
         1. Display error message
         2. Loop until desired input is achieved
      3. After getting desired input, execute (if input is valid scenario)
4. Prompt the user to enter expenses total
   1. If input is valid number—multiple scenarios:
      1. Calculate: saving= net salary – expense \*12
         1. Saving equals to 0, display break even message
         2. Saving is more than 0, display extra available amount to be saved
         3. Saving is less than 0, display debt (extra amount that was spent)
   2. If input is not valid number:
      1. Display error message
      2. Loop until desired input is achieved
      3. After getting desired input, execute (if input is valid scenario)
5. Prompt user to enter yes or no for printing responses out question:
   1. If yes:
      1. Display name
      2. Display DOB, display and calculate how old is user
      3. Display net salary
   2. If no:
      1. Terminate program
   3. Else:
      1. Display error message
      2. Loop until desired input is achieved
      3. After getting desired input, execute (if input is valid scenario)