EAST WEST UNIVERSITY

Mini Project

CSE-366

Sec-02

Submitted to:

Md Al-Imran
Lecturer
Department of Computer Science & Engineering

Submitted by:

Name: Hasib Ar Rafiul Fahim
ID: 2019-1-60-036

1. Income tax calculator
Code

```
def normalCitizen(income):
    if 1 <= income <= 300000:
        print("Tax: ", (income*0))
    elif 300001 <= income <= 400000:
        print("Tax: ", (income*0.05))
    elif 400001 <= income <= 700000:
        print("Tax: ", (income*0.1))
    elif 700001 <= income <= 1100000:
        print("Tax: ", (income*0.15))
    elif 1100001 <= income <= 1600000:
        print("Tax: ", (income*0.2))
    else:
        print("Tax: ", (income*0.25))


def womenAndCitizenWithAgeGreaterThen65(income):
    if 1 <= income <= 350000:
        print("Tax: ", (income*0))
    elif 350001 <= income <= 450000:
        print("Tax: ", (income*0.05))
    elif 450001 <= income <= 750000:
        print("Tax: ", (income*0.1))
    elif 750001 <= income <= 1150000:
        print("Tax: ", (income*0.15))
    elif 1150001 <= income <= 1650000:
        print("Tax: ", (income*0.2))
    else:
        print("Tax: ", (income*0.25))


def Disabled(income):
    if 1 <= income <= 450000:
        print("Tax: ", (income*0))
    elif 450001 <= income <= 550000:
        print("Tax: ", (income*0.05))
    elif 550001 <= income <= 850000:
        print("Tax: ", (income*0.1))
    elif 850001 <= income <= 1250000:
        print("Tax: ", (income*0.15))
    elif 1250001 <= income <= 1750000:
        print("Tax: ", (income*0.2))
    else:
        print("Tax: ", (income*0.25))


def parentsOfDisabled(income):
    if 1 <= income <= 350000:
        print("Tax: ", (income*0.0))
    elif 350001 <= income <= 450000:
        print("Tax: ", (income*0.05))
    elif 450001 <= income <= 750000:
        print("Tax: ", (income*0.1))
    elif 750001 <= income <= 1150000:
        print("Tax: ", (income*0.15))
    elif 1150001 <= income <= 1650000:
        print("Tax: ", (income*0.20))
```

```
    else:
        print("Tax: ", (income*0.25))


def woundedFreedomFighters(income):
    if 1 <= income <= 475000:
        print("Tax: ", (income*0.0))
    elif 475001 <= income <= 575000:
        print("Tax: ", (income*0.05))
    elif 575001 <= income <= 875000:
        print("Tax: ", (income*0.1))
    elif 875001 <= income <= 1275000:
        print("Tax: ", (income*0.15))
    elif 1275001 <= income <= 1775000:
        print("Tax: ", (income*0.20))
    else:
        print("Tax: ", (income*0.25))


def main():
    while True:
        print("Enter your age: ")
        try:
            age = int(input())
        except ValueError:
            print("Please enter correct age")
            break

        print("Enter your Gender: ")
        print("1. Man: ")
        print("2. Women: ")
        try:
            gender = int(input())
        except ValueError:
            print("Please enter correct Gender")
            continue
        if gender != 1 and gender != 2:
            print("Please Enter 1 or 2")
            break

        print("Enter your Income: ")
        try:
            income = float(input())
        except ValueError:
            print("Please enter correct income")
            break

        print("Choose any criteria: ")
        print("1. Normal Citizen: ")
        print("2. Disabled: ")
        print("3. Parent of Disabled: ")
        print("4. Wounded Freedom Fighters: ")
        try:
            criteria = int(input())
        except ValueError:
            print("Please enter correct criteria")
            break

        if age > 65 or gender == 2:
            womenAndCitizenWithAgeGreaterThen65(income)
        else:
```

```
            if criteria == 1:
                normalCitizen(income)

            elif criteria == 2:
                Disabled(income)

            elif criteria == 3:
                parentsOfDisabled(income)

            elif criteria == 4:
                woundedFreedomFighters(income)

            else:
                print("Wrong Input")
        break


main()
```

Output:

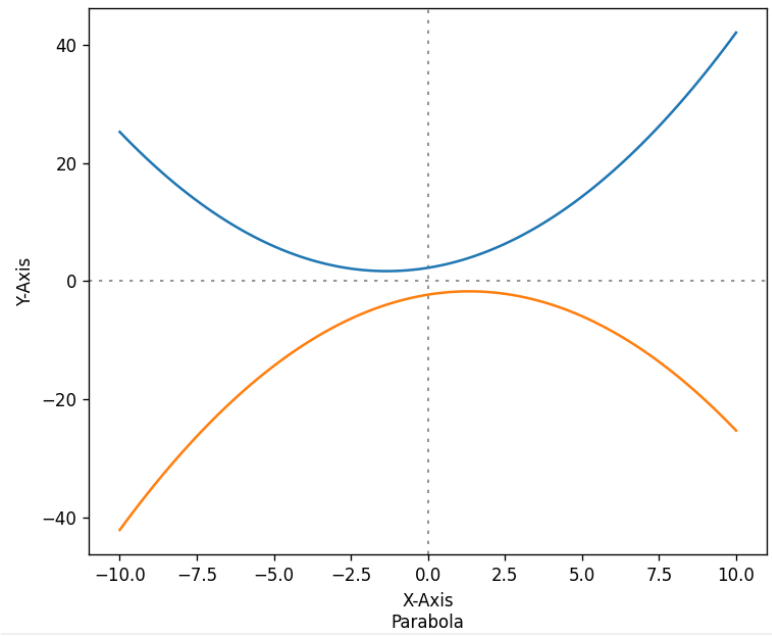
```
Enter your age:
66
Enter your Gender:
1. Man:
2. Women:
1
Enter your Income:
1600000
Choose any criteria:
1. Normal Citizen:
2. Disabled:
3. Parent of Disabled:
4. Wounded Freedom Fighters:
1
Tax:  320000.0
```

```
Enter your age:
aaa
Please enter correct age
```

2. Plotting
Code with Explanation

```
import matplotlib.pyplot as plt
import numpy as np
# some imports to plot an equation


def f(x):
    return (0.3139 * x ** 2) + (0.8402 * x) + 2.2701
    # This is the equation I got after plotting the given first curve


def f2(x):
    return (-(0.3139 * x ** 2)) + (0.8402 * x) - 2.2701
    # This is the equation I got after plotting the given second
        curve which is opposite of the first one


xlist = np.linspace(-10, 10, num=100)    # list of x values
xlist2 = np.linspace(-10, 10, num=100)    # list of x values for
second curve
ylist = f(xlist)    # list of y values after calculating from the
equation
ylist2 = f2(xlist2)    # list of y values for second curve after
calculating from the equation


plt.figure(num=0, dpi=120)    # this determine the figure number and
display pixels
plt.axhline(0, color='black', alpha=0.5, dashes=[2, 4], linewidth=1)
plt.axvline(0, color='black', alpha=0.5, dashes=[2, 4], linewidth=1)
plt.xlabel('X-Axis' + '\n' + 'Parabola')    # axis level
plt.ylabel('Y-Axis')
plt.plot(xlist, ylist)    # plotting the first curve
plt.plot(xlist2, ylist2)    # plotting the second curve
plt.show()    # showing the plot on pyCharm
```

Plot:



3. Maze Solver
Code

```
# this function will print the solved map
def print_map(map):
    for row in map:
        for item in row:
            print(item, end='')
        print()


# this function will find the start point
def find_start(map):
    for row in range(len(map)):
        for col in range(len(map[0])):
            if map[row][col] == 'o':
                return row, col



# this function will find if the given position is on the map or not
def is_valid_position(map, pos_r, pos_c):
    if pos_r < 0 or pos_c < 0:
        return False
    if pos_r >= len(map) or pos_c >= len(map[0]):
        return False
    if map[pos_r][pos_c] in ' x':
        return True
    return False


# this is the method which will solve the map
def solve_map(map, start):
    stack = []
    stack.append(start)
    while len(stack) > 0:
        pos_r, pos_c = stack.pop()

        if map[pos_r][pos_c] == 'x':
            print("Goal: (%d,%d)" % (pos_r, pos_r))
            print("Goal Found")
            if print_map(map) is not None:
                print(print_map(map))
            return True

        if map[pos_r][pos_c] == '.':
            continue
        map[pos_r][pos_c] = '.'
        if is_valid_position(map, pos_r - 1, pos_c):
            stack.append((pos_r - 1, pos_c))
        if is_valid_position(map, pos_r + 1, pos_c):
            stack.append((pos_r + 1, pos_c))
        if is_valid_position(map, pos_r, pos_c - 1):
            stack.append((pos_r, pos_c - 1))
        if is_valid_position(map, pos_r, pos_c + 1):
            stack.append((pos_r, pos_c + 1))
    return False
```

```
#given map
map = "##############################\n"\
       "#          #               #    #\n"\
       "# ####     ########        #    #\n"\
       "#   o #     #               #    #\n"\
       "#     ###      #####  ######    #\n"\
       "#       #    ###    #            #\n"\
       "#       #      #    #   #   #     #\n"\
       "#      #####     #     #   # x    #\n"\
       "#                #         #     #\n"\
       "##############################"

# it will convert the map and put the values in a list
converted_map = []
lines = map.splitlines()
for line in lines:
    converted_map.append(list(line))
map = converted_map
start = find_start(map)
print("Starting Point:", start)
solve_map(map, start)
```

Output

```
Starting Point: (3, 3)
Goal: (7,7)
Goal Found
##############################
#         #..............#    #
# ####    ########.......#    #
#   ..#    # ............#    #
#....###   ...#####..######    #
#......#...###    #...........#
#......#.....#    #   #   #.....#
#.....#####....#     #   #.x    #
#     ..........#        #      #
##############################
```

4. Logic Program

Code

```
from logpy import *
from logpy.core import lall

people = var()
# we import rules given on the questions
rules = lall(
    (eq, (var(), var(), var(), var()), people),
    (membero, ('Steve', var(), 'blue', var()), people),
    (membero, (var(), 'cat', var(), 'Canada'), people),
    (membero, ('Matthew', var(), var(), 'USA'), people),
    (membero, (var(), var(), 'black', 'Australia'), people),
    (membero, ('Jack', 'cat', var(), var()), people),
    (membero, ('Alfred', var(), var(), 'Australia'), people),
    (membero, (var(), 'dog', var(), 'France'), people),
    (membero, (var(), 'rabbit', var(), var()), people)
)

output = 0
# we find the solution by passing the values ro run() function
solutions = run(0, people, rules)
for house in solutions[0]:
    if 'rabbit' in house:
        output = house[0]

# now we print the output result
print('\n' + output + ' is the owner of the rabbit')
# now we print all the details that were retrieved from logic
programming
print('\nHere are all the details:')
attributes = ['Name', 'Pet', 'Color', 'Country']
print('\n' + '\t\t'.join(attributes))
print('=' * 50)
for item in solutions[0]:
    print('')
    print('\t\t'.join([str(x) for x in item]))
```

Output

```
Matthew is the owner of the rabbit

Here are all the details:

Name        Pet     Color       Country
==================================================

Steve       dog     blue        France

Jack        cat     ~_9     Canada

Matthew     rabbit      ~_11        USA

Alfred      ~_13        black       Australia
```