```
"""
exception_handling_lab.py
@author: ITP 150 Student
Date Created: October 7, 2024
mac users. open terminal go to directory and chmod 555 hello_world.py
We are examining exception handling for files, values, data types, indexes
and dividing by 0.
"""


import csv


def main():
    menu_string = 'Please choose from the following menu: \
            \nEnter 1 to print the pop list.\
            \nEnter 2 to analyze pop statistics.\
            \nEnter 3 to save the statistics.\
            \nEnter 99 to Quit.'
    valid_choices = [1, 2, 3, 99]
    pop_list = []

    print('Task 1. Check if a file exists and process a file.')
    process_the_file()

    print('Task 2. Open and process a file with a context handler')
    pop_list = read_the_file(pop_list)
    print(pop_list)

    print('Task 3. Check input for ValueError in datatype and range')
    choice = input_menu_choice(menu_string, valid_choices)
    print(choice)

    print('Task 4. Check for IndexError, raise TypeError, print the list.')
    display_list(pop_list)

    # pop_list = []
    print('Task 5. Calculate an average. Check for ZeroDivisionError.')
    calc_average(pop_list)


# Exceptions related to accessing and processing files.
def process_the_file():
    found_file = False  # comment out to see unreachable code
    try:
        # 1st run, use filename of hello_world..py to generate file not found
        # 2nd run, use filename of hello_world.py and mode of w for permissions
        # 3rd run, comment the PermissionError to see the IOError exception
        # 4th run, comment found_file = False to see unreachable code
        file = open('hello_world.py', 'r')
        file.write('Writing in the file')
        found_file = True  # code that can not be reached if exception throws
    except FileNotFoundError:  # prefer to use specific before broad
        print("The file was not found.")
    # except PermissionError as err:  # prefer to use specific before broad
    #     print('A permission error occurred', err)
    # except IOError as err:  # changed in Python 3.3. Still works
    #     print('The process can not take place on the file.', err)
    except Exception as err:  # broad exception that will catch anything else
        print("An error occurred and it was.", err)
    finally:  # finally blocks always execute and they can throw an error
        # so consider if you need a finally block. Be careful with it.
        if found_file:
            file.close()


def read_the_file(pop_list):
    try:
        # 1st run, IOError catch a FileNotFound by name to us_population_csv
        # 2nd run, let IndexError catch an index not found meaning
        # you are trying to access an item from the list that does not exist by
        # changing len(pop_list) to len(pop_list) + 1
        with open('us_population.csv', newline='') as pop_file:
            pop_reader = csv.reader(pop_file, delimiter=',')  # \t tab
            pop_list = [row for row in pop_reader]
        for row in range(1, len(pop_list)):
            pop_list[row][0] = int(pop_list[row][0])
            pop_list[row][1] = int(pop_list[row][1])
        return pop_list  # This will return None if we can't process the list
    except IOError:
        print('An error occurred trying to read from the file.')
    except IndexError:
        print('An index error occurred.')
    except Exception:
        print('An error occurred.')


def input_menu_choice(menu_string, valid_choices):
    while True:
        # 1st run enter a to raise ValueError
        # 2nd run enter 0 to raise broad exception
        try:
            print('-'*50)
            print(menu_string)
            print('-'*50)
            # Get the user's choice
            choice = int(input())
            if choice in valid_choices:
                return choice
            else:
                raise Exception
        except ValueError:  # throws on wrong datatype
            print('Invalid value. Please enter 1, 2, 3, or 99: \U0001F600')
        except Exception:  # throws on anything else
            print('Error on input. Please try again.')


def display_list(pop_list):
    # run 1. normal
    # run 2. change len(pop_list) to str(len(pop_list)) range for TypeError
    print('='*35)
    print(f'{"As Of Date":12s}{"pop":>20s}')
    try:
        for row in range(1, len(pop_list)):
            print(f'{pop_list[row][0]:12}{pop_list[row][1]:>20,d}')
        print('='*35)
    except IndexError:
        print('Index is out of range.')
    except TypeError:
        print('Wrong datatype for operation.')
    except Exception:
        print('An error occurred.')


def calc_average(pop_list):
    sum_pop = 0
    try:
        # 1st run be sure pop_list = [] in main method to generate divide by 0
        # 2nd run comment or delete pop_list = [] to generate broad exception
        # 3rd run subtract 1 from len(pop_list)  for mathematical accuracy
        for row in range(1, len(pop_list)):  # start with 1 please
            sum_pop = sum_pop + pop_list[row][1]
        average_pop = sum_pop / len(pop_list)  # subtract 1 for accuracy after
        print(average_pop)
        return average_pop
    except IndexError:
        print('Index is out of range')
    except ZeroDivisionError:
        print('A Zero Division Error occurred.')
    except Exception:
        print('An error occurred.')


if __name__ == '__main__':
    main()

```