CPSC 103 FINAL PROJECT

December 05, 2024

1 Project Final Submission

1.0.1 Step 1a: Planning

Identify the information in the file your program will read Describe (all) the information that is available. Be sure to note any surprising or unusual features. (For example, some information sources have missing data, which may be blank or flagged using values like -99, NaN, or something else.)

The following athlete_event.csv file that I will be using for my project contains the following information:

- ID: Which denotes the ID number of the athlete competing
- Name: This shows the name of the athlete competing - from first name to last name. (Ex. "Yevgeny Sergeyevich Abramenko")
- Sex: The gender of the athlete competing.
- Age: The age of the athlete competing at the time. (Ex. "Killian Albrecht was 36 when he competed at the 2010 Olympic Games')
- Height: The height of the athlete when they were competing. (in cm)
- Weight: The weight of the athlete when they were competing. (in lbs)
- Team: The country that the athletes are representing.
- NOC: (Short for National Olympic Committee) is a worldwide constituent of the worldwide Olympic movement. This denotes what country the athletes are from.
- Year: which represents the year the Olympics took place.
- Season: which represents the season the Olympics took place in (Winter, Spring)
- City: which city that hosted the Olympics that year.
- Sport: which sport that the athletes competed in.
- Event: what event the sport falls under.
- Medal: whether the athlete won a medal or not, with 4 cases. ('NA', 'Bronze', 'Silver', 'Gold')

1.0.2 Step 1b: Planning

Brainstorm ideas for what your program will produce

Select the idea you will build on for subsequent steps You must brainstorm at least three ideas for graphs or charts that your program could produce and choose the one that you'd like to work on. You can choose between a line chart, histogram, bar chart, scatterplot, or pie chart.

If you would like to change your project idea from what was described in the proposal, you will need to get permission from your project TA. This is intended to help ensure that your new project idea

will meet the requirements of the project. Please see the project proposal for things to be aware of when communicating with your project TA.

Idea #1: Using the "year" column, the "sex" column, and the "medal" column, we will compute how many medals that female athletes have won per each year. We will communicate this information by making a line graph with the x-axis being the year and the y-axis being the number of medals that female athletes have won.

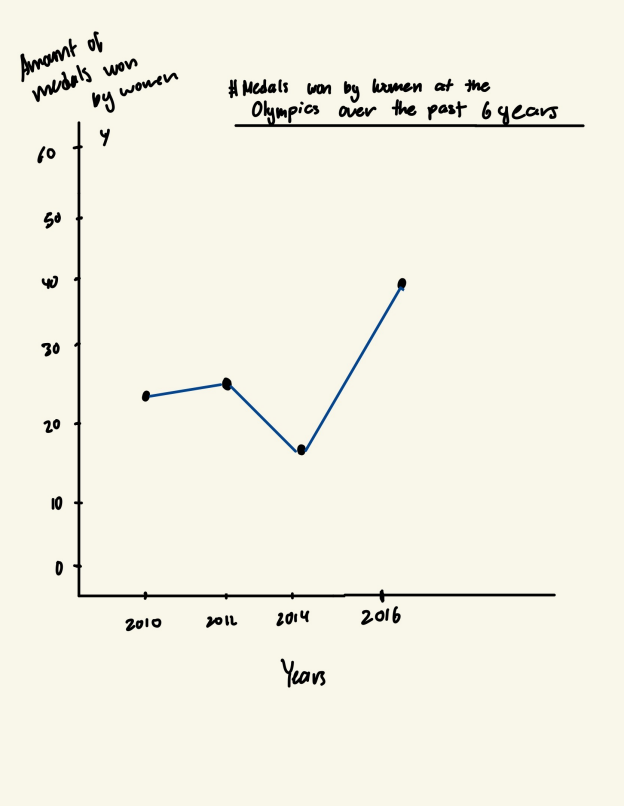
Idea #2: Using the "age" column and the "year" column, we can compute the year that has the oldest average age of Olympic athletes. We could communicate this information by creating a bar chart with the x-axis being the year and the y-axis being the average age of the competitors.

Idea #3: Using the "sex" column, the "sport" column, and the "year" column, we could find which sport has the greatest percentage of female athletes in a given year. We could communicate this information by creating a pie chart with each sport being a section of the chart where the size of each section relates to the relative percentage of female athletes.

Conclusion: After much careful consideration, we have decided to work on idea #1, finding the number of medals that female athletes have won per year.

1.0.3 Step 1c: Planning

Write or draw examples of what your program will produce You must include an image that shows what your chart or plot will look like. You can insert an image using the Insert Image command near the bottom of the Edit menu.



1.0.4 Step 2a: Building

Document which information you will represent in your data definitions Before you design data definitions in the code cell below, you must explicitly document here which information in the file you chose to represent and why that information is crucial to the chart or graph that you'll produce when you complete step 2c.

We are going to represent the number of olympic medals that female athletes have won over the past 10 years. To do this, we will have to first filter for the number of female athletes that competed in the Olympics over the past 10 years, then we will filter for how many of those female athletes won medals over the past 10 years. Lastly we will find what year each of those medals was won in. These pieces of data that we have chosen here are explicitly important to our analyze function because they denote the gender, year, which season the medal was won in, and whether a medal was won or not. The other parts of the csv file are not necessary for this information. This information is crucial to the line graph that we are aiming to produce as we cannot plot the number of medals won by female athletes vs. the year if we do not know how many medals were won by female athletes in each of those years.

Design Data Definitions

[5]:
```
from cs103 import *
from typing import NamedTuple, List
import csv
import matplotlib.pyplot as pyplot

##################
# Data Definitions

OlympicData = NamedTuple('MedalsWon', [('id', int), # in range [0, ...]
                                       ('name', str),
                                       ('sex', str),
                                       ('age', int), # in range [0, ...]
                                       ('height', int), # in range [0, ...]
                                       ('weight', int), # in range [0, ...]
                                       ('team', str),
                                       ('noc', str),
                                       ('games', str),
                                       ('year', int), # in range [2010, 2016]
                                       ('season', str),
                                       ('city', str),
                                       ('sport', str),
                                       ('event', str),
                                       ('medals_won', str)])
# interp. Olympic data with the athlete ID number, name of the athlete, sex of␣
 ↪the athlete (male or female),
# the age of the athlete at the time of the competition, the height of the␣
 ↪athlete (in cm), the weight of the athlete (in kg), the country that the␣
 ↪athlete is competing for,
```

```
# the three-letter code of the county (ex. CAN), the Olympic games that the␣
 ↪athlete competed in (ex. 2010 Winter), the year the Olympics took place (2010,␣
 ↪2012, 2014, or 2016),
# the season that the athlete competed in (winter or summer), the city that the␣
 ↪athlete competed in (ex. Vancouver), the sport that the athlete competed in,␣
 ↪the event that the athlete competed in,
# and the type of medal that the athlete won (gold, silver, bronze, NA)

OD1 = OlympicData(736, 'Brigitte Action', 'F', 24, 168, 63, 'Canada', 'CAN',␣
 ↪'2010 Winter', 2010, 'Winter', 'Vancouver', 'Alpine Skiing', "Alpine Skiing␣
 ↪Womens Slalom", 'NA')
OD2 = OlympicData(956, "Richard Adjei", 'M', 27, 190, 110, "Germamy-2", "GER",␣
 ↪"2010 Winter", 2010, 'Winter', "Vancouver", 'Bobsleigh', 'Bobsleigh Mens Two',␣
 ↪'Silver')
OD3 = OlympicData(78271, 'Rta Meilutyt', 'F', 15, 176, 75, 'Lithuania', 'LTU',␣
 ↪'2012 Summer', 2012, 'Summer', 'London', 'Swimming', 'Swimming Womens 100␣
 ↪metres Freestyle', 'NA')
OD4 = OlympicData(78271, 'Rta Meilutyt', 'F', 15, 176, 75, 'Lithuania', 'LTU',␣
 ↪'2012 Summer', 2012, 'Summer', 'London', 'Swimming', 'Swimming Womens 100␣
 ↪metres Breaststroke', 'Gold')


@typecheck
def fn_for_olympic_data(od: OlympicData) -> ...: #template based on compound
  return ...(od.id,
             od.name,
             od.sex,
             od.age,
             od.height,
             od.weight,
             od.team,
             od.noc,
             od.games,
             od.year,
             od.season,
             od.city,
             od.sport,
             od.event,
             od.medals_won)

#List[OlympicData]
#interp. a list of OlympicData

LOOD0 = []
LOOD1 = [OD1, OD2]
LOOD2 = [OD1, OD2, OD3]
```

```
LOOD3 = [OD1, OD2, OD3, OD4]



@typecheck
def fn_for_lood(lood: List[OlympicData]) -> ...: #template based on␣
 ↪arbitrary-sized and the reference rule
    #description of the acc
    acc = ... #type:...
    for od in lood:
      acc = ...(fn_for_olympic_data(od), acc)
    return ... (acc)
```

1.0.5 Step 2b and 2c: Building

Design a function to read the information and store it as data in your program

Design functions to analyze the data Complete these steps in the code cell below. You will likely want to rename the analyze function so that the function name describes what your analysis function does.

Unless approved by your project TA, you **cannot** use libraries such as `numpy` or `pandas`. The project is meant as a way for you to demonstrate your knowledge of the learning goals in this course. While it is convinent to use external libraries, it will do all the work and will not help us gauge your mastery of the concepts.

You also cannot use built in list functions (e.g., `sum` or `average`) when writing code to do your substantial computation. Normally we encourage you to make use of what is already available but in this case, the final project involves demonstrating skills from class (e.g., how to work with a list). Using pre-built functions for this does not enable you to demonstrate what you know.

If you wish to change your project idea, you must **first** obtain permission from your TA. When contacting your TA, please provide a valid reason for why you want to change your project. Each time you change your topic idea, your TA will have to evaluate it to see if it will meet all of the project requirements. This is non-trivial task during one of the busiest times of the semester. As such, the deadline for project idea changes will be 3 business days before the deadline. Note that the deliverable deadline will not be extended and there is no compensation for the time you spent on the previous idea.

[10]:
```
###########
# Functions

@typecheck
def main(filename: str) -> None:
    """
    Reads the file from given filename, analyzes the data, returns the result
    """
    # return None # stub
    # Template from HtDAP, based on function composition
```

```
    return show_line_chart(read(filename))



@typecheck
def read(filename: str) -> List[OlympicData]:
    """
    Reads information from the specified file and returns a list of olympic data
    """
    # return []  #stub
    # Template from HtDAP
    # lood contains the result so far
    lood = [] # type: List[OlympicData]

    with open(filename) as csvfile:

        reader = csv.reader(csvfile)
        next(reader) # skip header line

        for row in reader:
            od = OlympicData(parse_int(row[0]), row[1], row[2],␣
 ↪parse_int(row[3]), parse_int(row[4]), parse_int(row[5]), row[6], row[7],␣
 ↪row[8], parse_int(row[9]), row[10], row[11], row[12], row[13], row[14])
            lood.append(od)

    return lood


@typecheck
def show_line_chart(lood: List[OlympicData]) -> None:
    """
    Show a line chart of the Olympic medals won by a woman over the
    span of 6 years, with the years being on the x-axis and the number
    of medals on the y-axis.

    Assumes that the years provided on the csv file are in increasing order
    """
    # return None # stub
    # template based on visualization

    # set the x-axis label, y-axis label, and the plot title
    pyplot.xlabel('Years')
    pyplot.ylabel('Amount of Medals Won By Women')
    pyplot.title('Number of Olympic Medals Won by Women from 2010 - 2016')

    # set the range for the axes
    # [x-min, x-max, y-min, y-max]
```

```
    pyplot.axis ([2010, 2016, 0, 20])

    # plot the graph
    p1 = pyplot.plot(list_of_years(lood),␣
↪count_medals_per_year(list_of_years(lood), lood))

    # set some properties for the line (color to red, line width to 2, and␣
↪marker to a small circle)
    pyplot.setp(p1, color='r', linewidth=2.0, marker="o", label='Medals Won By␣
↪Woman')

    # set the location for the legend
    pyplot.legend(bbox_to_anchor = (1.05, 1), loc = 2)

    # show the plot
    pyplot.show()

    return None

@typecheck
def count_medals_per_year(list_of_years: List[int], lood: List[OlympicData]) ->␣
↪List[int]:
    '''
    Returns a list of medals won by women from 'lood' broken down by year in␣
↪'list_of_years'.

    Assume 'years' is not empty.
    '''
    # return [] # stub
    # template from List[OlympicData] with extra paramater 'list_of_years'
    # medal_count is the number of medals won by woman by year in the instances␣
↪seen so far
    medal_count = [] #type: List[int]

    for year in list_of_years:
        medals_in_year = medals_won_by_woman_per_year(lood, year)
        medal_count.append(medals_in_year)

    return medal_count


@typecheck
def medals_won_by_woman_per_year(lood: List[OlympicData], year: int) -> int:
    '''
    Returns the number of medals won by woman in a given year.
    '''
```

```
    # return 0 # stub
    # template from List[OlympicData] with extra parameter 'year'
    # num_medals is the number of medals won by woman in the given year in the␣
 ↪instances seen so far
    num_medals = 0 #type: int
    for od in lood:
        if if_woman(od) and if_medal_won(od) and medals_per_year(od, year):
            num_medals = num_medals + 1
    return num_medals


@typecheck
def filter_for_medals_per_year(lood: List[OlympicData], year: int) ->␣
 ↪List[OlympicData]:
  '''
  This function returns a list of Olympic Data where a medal was won in each od,␣
 ↪and
  in which the year of each od matches the given year.
  '''
  # return [] #stub
  # template based on List[OlympicData] with extra parameter 'year'
  # list_medals_per_year is a list of Olympic Data where a medal was won in each␣
 ↪od
  # and in which the year of each od matches the given year in the instances␣
 ↪seen so far

  list_medals_per_year = [] # type: List[OlympicData]

  for od in lood:
        if medals_per_year(od, year) and if_medal_won(od):
            list_medals_per_year.append(od)
  return list_medals_per_year


@typecheck
def medals_per_year(od: OlympicData, year: int ) -> bool:
  '''
  This function returns True if the year in od is equal to the year given,
  returns False otherwise.
  '''
  # return False  # stub
  # template from OlympicData with extra parameter 'year'

  if od.year == year:
    return True
  else:
    return False
```

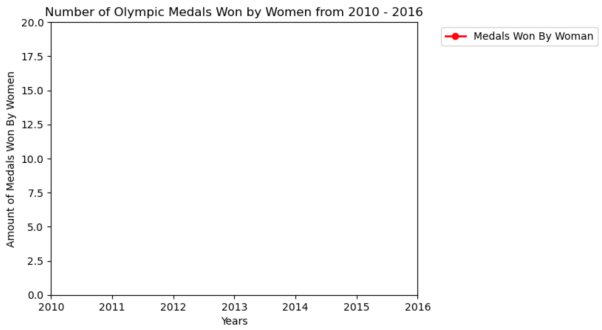
```
@typecheck
def filter_for_medals(lood: List[OlympicData]) -> List[OlympicData]:
    '''
    Returns a list of olympic data in which a medal was won in each od
    '''
    # return [] # stub
    # template from List[OlympicData]
    # list_of_medals_won is a list of olympic data in which a medal was won
    # in each od in instances seen so far
    list_of_medals_won = [] #type: List[OlympicData]
    for od in lood:
      if if_medal_won(od):
        list_of_medals_won.append(od)
    return list_of_medals_won


@typecheck
def if_medal_won(od: OlympicData) -> bool:
    '''
    Returns True if the athlete in od has won a medal,
    returns False otherwise.
    '''
    # return False # stub
    # template from OlympicData
    if od.medals_won == 'Gold':
        return True
    elif od.medals_won == 'Silver':
        return True
    elif od.medals_won == 'Bronze':
        return True
    else:
        return False


@typecheck
def filter_for_woman(lood: List[OlympicData]) -> List[OlympicData]:
    '''
    Returns a list of Olympic data in which the sex of the athlete is female
    in each od.
    '''
    # return [] # stub
    # template from List[OlympicData]
    # list_female is a list of olympic data in which the sex of the
    # athlete is female in each od in the instances seen so far
    list_female = [] # type: List[OlympicData]
```

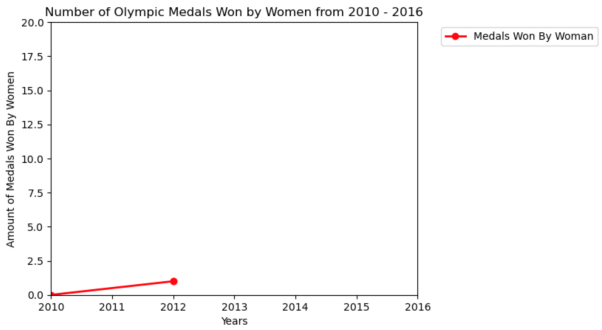
```
    for od in lood:
        if if_woman(od):
            list_female.append(od)
    return list_female


@typecheck
def if_woman(od: OlympicData) -> bool:
  '''
  Returns True if the sex of the athlete in
  od is female, returns False otherwise.
  '''
  # return False # stub
  # template from OlympicData
  if od.sex == 'F':
    return True
  else:
    return False


@typecheck
def list_of_years(lood: List[OlympicData]) -> List[int]:
  '''
  Return a list of all the years in lood
  '''
  # return [] # stub
  # template from List[OlympicData]
  # list_of_years is a list of the years in lood in the instances seen so far
  list_of_years = [] # type: List[int]
  for od in lood:
    list_of_years.append(get_year(od))
  return list_of_years


@typecheck
def get_year(od: OlympicData) -> int:
    '''
    Returns the year that the Olympics took place in, in the given od
    '''
    # return 0 # stub
    # template from OlympicData

    return od.year


start_testing()
```

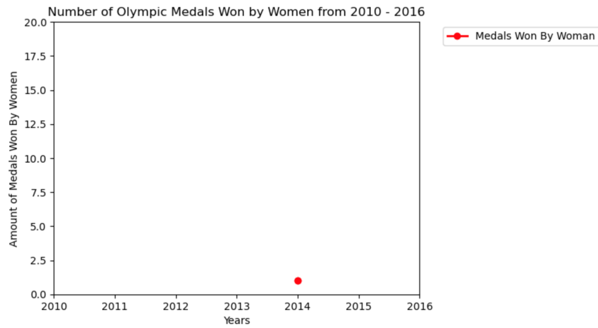
```
# examples and tests for main
expect(main("empty_file.csv"), None)
expect(main("athlete_events_test1.csv"), None)
expect(main("athlete_events-test2.csv"), None)

# examples and tests for read
expect(read("athlete_events_test1.csv"), [OlympicData(736, 'Brigitte Acton␣
↪(-Smith)', 'F', 24, 168, 63, 'Canada', 'CAN', '2010 Winter', 2010,
                                                    'Winter', 'Vancouver',␣
↪'Alpine Skiing', "Alpine Skiing Womens Slalom", 'NA'),
                                                OlympicData(956, 'Richard "Richy" ␣
↪Adjei', 'M', 27, 190, 110, "Germany-2", "GER", "2010 Winter", 2010,
                                                    'Winter', "Vancouver",␣
↪'Bobsleigh', 'Bobsleigh Mens Two', 'Silver'),
                                                OlympicData(78271, 'Rta Meilutyt',␣
↪'F', 15, 176, 75, 'Lithuania', 'LTU', '2012 Summer', 2012,
                                                    'Summer', 'London',␣
↪'Swimming', "Swimming Womens 100 metres Freestyle", 'NA'),
                                                OlympicData(78271, 'Rta Meilutyt',␣
↪'F', 15, 176, 75, 'Lithuania', 'LTU', '2012 Summer', 2012,
                                                    'Summer', 'London',␣
↪'Swimming', "Swimming Womens 100 metres Breaststroke", 'Gold')])
expect(read("athlete_events-test2.csv"),
       [OlympicData(96520, 'Valeriya Leonidovna Potyomkina-Reznik', 'F', 28,␣
↪167, 63, 'Russia', 'RUS', '2014 Winter', 2014, 'Winter',
                    'Sochi', 'Short Track Speed Skating', "Short Track Speed␣
↪Skating Women's 3,000 metres Relay", 'NA'),
        OlympicData(96539, 'Marie-Philip Poulin', 'F', 22, 169, 72, 'Canada',␣
↪'CAN', '2014 Winter', 2014, 'Winter', 'Sochi', 'Ice Hockey',
                    "Ice Hockey Women's Ice Hockey", 'Gold')])

# examples and tests for show_line_chart
expect(show_line_chart(LOOD0), None)
expect(show_line_chart(LOOD3), None)

# examples and tests for count_medals_per_year
expect(count_medals_per_year([], []), [])
expect(count_medals_per_year([2010, 2012, 2014, 2016], LOOD0), [0, 0, 0, 0])
expect(count_medals_per_year([2010, 2012, 2014, 2016], LOOD1), [0, 0, 0, 0])
expect(count_medals_per_year([2010, 2012, 2014, 2016], [OD1, OD3]), [0, 0, 0, 0])
expect(count_medals_per_year([2010, 2012, 2014, 2016], LOOD3), [0, 1, 0, 0])
expect(count_medals_per_year([2010, 2012, 2014, 2016], [OD1, OD2, OD3, OD4,␣
↪OlympicData(1181, 'Meghan Christina Agosta (-Marciano)', 'F', 23,
                                                                        ␣
↪       168, 67, 'Canada', 'CAN', '2010 Winter', 2010, 'Winter',
```

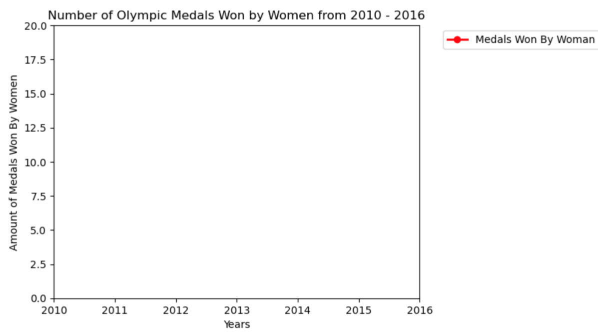
```
                                                                          ␣
↪       'Vancouver', 'Ice Hockey', "Ice Hockey Women's Ice Hockey", 'Gold'),
                                                    OlympicData(70941,␣
↪'Yekaterina Aleksandrovna Lobysheva', 'F', 28, 178, 69, 'Russia', 'RUS', '2014␣
↪Winter',
                                                                2014,␣
↪'Winter', 'Sochi', 'Speed Skating', "Speed Skating Women's Team Pursuit (6␣
↪laps)",
                                                                'Bronze')]),␣
↪[1, 1, 1, 0])
expect(count_medals_per_year([2010, 2012, 2014, 2016], [OD4, OlympicData(9460,␣
↪'Elizabeth Lyon Beisel', 'F', 19, 169, 68, 'United States',
                                                                    'USA',␣
↪'2012 Summer', 2012, 'Summer', 'London', 'Swimming', "Swimming Women's 200␣
↪metres Backstroke",
                                                                    ␣
↪'Bronze'),
                                                    OlympicData(4724,␣
↪'Emilia Elisabeth Appelqvist', 'F', 26, 168, 65, 'Sweden', 'SWE',
                                                                '2016␣
↪Summer', 2016, 'Summer', 'Rio de Janeiro', 'Football', "Football Women's␣
↪Football",
                                                                'Silver')]),␣
↪[0, 2, 0, 1])

# examples and tests for medals_won_by_woman_per_year
expect(medals_won_by_woman_per_year(LOOD0, 2010), 0)
expect(medals_won_by_woman_per_year([OD1, OD3], 2012), 0)
expect(medals_won_by_woman_per_year(LOOD3, 2010), 0)
expect(medals_won_by_woman_per_year(LOOD3, 2012), 1)

# examples and tests for filter_for_medals_per_year
expect(filter_for_medals_per_year(LOOD0, 2016), [])
expect(filter_for_medals_per_year(LOOD1, 2014), [])
expect(filter_for_medals_per_year(LOOD3, 2012), [OD4])

# examples and tests for medals_per_year
expect(medals_per_year(OD1, 2010), True)
expect(medals_per_year(OD4, 2014), False)

# examples and tests for filter_for_medals
expect(filter_for_medals(LOOD0), [])
expect(filter_for_medals(LOOD1), [OD2])
expect(filter_for_medals([OD1, OD3]), [])

# examples and tests for if_medal_won
```

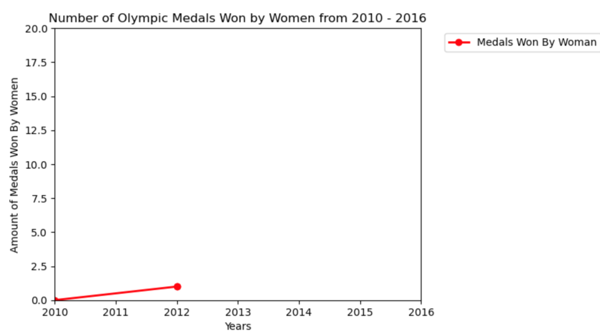
```
expect(if_medal_won(OD1), False)
expect(if_medal_won(OD2), True)

# examples and tests for filter_for_woman
expect(filter_for_woman(LOOD0), [])
expect(filter_for_woman(LOOD1), [OD1])
expect(filter_for_woman(LOOD3), [OD1, OD3, OD4])

# examples and tests for if_woman
expect(if_woman(OD1), True)
expect(if_woman(OD2), False)

# examples and tests for list_of_years
expect(list_of_years(LOOD0), [])
expect(list_of_years(LOOD3), [2010, 2010, 2012, 2012])

# examples and tests for get_year
expect(get_year(OD1), 2010)
expect(get_year(OD3), 2012)

summary()
```









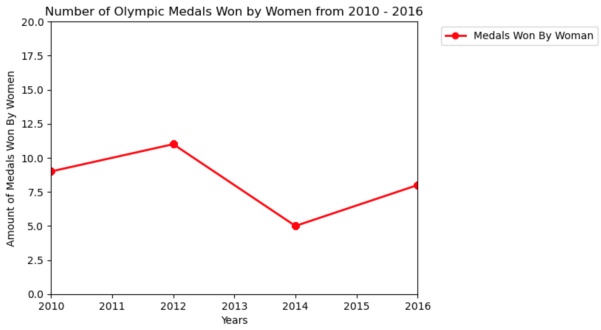



```
37 of 37 tests passed
```

1.0.6 Final Graph/Chart

Now that everything is working, you **must** call `main` on the intended information source in order to display the final graph/chart:

[11]:
```
main("athlete_events_edited.csv")
```


Number of Olympic Medals Won by Women from 2010 - 2016
Medals Won By Woman
Amount of Medals Won By Women
Years
20.0
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
2010
2011
2012
2013
2014
2015
2016