Assignment 7

Lists and Tuples

LILLEBAELT ACADEMY OF
PROFESSIONAL HIGHER EDUCATION

Author
Martin Grønholdt
mart80c7@edu.eal.dk

Sunday 15 January 2017

Table of Contents

Introduction

The programs in this hand-in is about list and their use.

All files for this hand in are available at:
https://github.com/deadbok/eal_programming/tree/master/Assignment 7

Error handling

All programs that requests user input, handle bad input by asking the user, to use only the correct data type, where after it exits.

```
Enter the amount of a purchase: 2hjjhg

Please use only numbers.
```

Example output of a program when the user enters an incorrect value

All programs using file I/O will show an error message if something goes wrong during file access.

1. Total Sales

This program compute the total sale for a week, after the user hans entered the sales figures.

prog1.py

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
# The above lines tell the shell to use python as interpreter when the
# script is called directly, and that this file uses utf-8 encoding,
# because of the country specific letter in my surname.
'''
Name: Program 1 "Total Sales"
Author: Martin Bo Kristensen Grønholdt.
Version: 1.0 (2017-01-15)

Compute the total sale for a week.
'''


def get_sales(weekdays):
    """
    Get the sales figure for each day from the user.

    :param weekdays: Weekdays to get the sales figures for.
    :return: A list of the sales figures for the week.
    """
    # Create the sales list, holding sale figures.
    sales = list()

    try:
        # Ask for the sales figure for each day.
        for day in weekdays:
            sales.append(float(input('Input sales for {:10}: '.format(day))))
    except ValueError:
        # Complain when something unexpected was entered.
        print('\nPlease use only numbers.')
        exit(1)

    return (sales)
```

```python
def print_sales(weekdays, sales):
    """
    Print the sales and the acumulated total.

    :param weekdays: Weekdays to print the sales for.
    :return: Nothing.
    """
    # Index into the lists
    i = 0
    # Sum of sales
    sum = 0
    # Print the header
    print(' {:10}|{:^9} | {:9}'.format('Week day', 'Sale', 'Total'))
    print('-' * 33)
    # Print row for each day
    for i in range(len(weekdays)):
        sum += sales[i]
        print(' {:10}|{:9.2f} |{:9.2f}'.format(weekdays[i], sales[i], sum))


def main():
    '''
    Main entry point.
    '''
    # List of the days for that are calculated.
    weekdays = ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday',
                'Saturday', 'Sunday']
    # Get number of feet.
    sales = get_sales(weekdays)
    print()
    # Output result.
    print_sales(weekdays, sales)


# Run this when invoked directly
if __name__ == '__main__':
    main()
```

Result

```
Input sales for Monday    : 45
Input sales for Tuesday   : 7
Input sales for Wednesday : 8000
Input sales for Thursday  : 456
Input sales for Friday    : 56
Input sales for Saturday  : 8
Input sales for Sunday    : 4

 Week day  |  Sale    | Total
---------------------------------
 Monday    |    45.00 |    45.00
 Tuesday   |     7.00 |    52.00
 Wednesday |  8000.00 |  8052.00
 Thursday  |   456.00 |  8508.00
 Friday    |    56.00 |  8564.00
 Saturday  |     8.00 |  8572.00
 Sunday    |     4.00 |  8576.00
```

Output of the program when run from the command line.

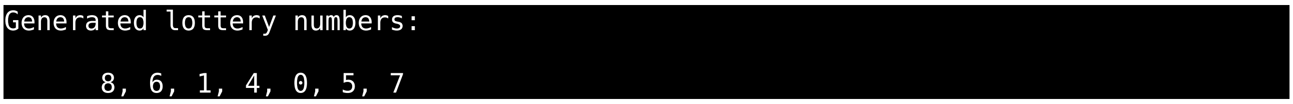
2. Lottery Number Generator

This program generates seven random lottery numbers. This program uses list comprehensions where the book suggests loops.

prog2.py

```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
# The above lines tell the shell to use python as interpreter when the
# script is called directly, and that this file uses utf-8 encoding,
# because of the country specific letter in my surname.
'''
Name: Program 2 "Lottery Number Generator"
Author: Martin Bo Kristensen Grønholdt.
Version: 1.0 (2017-01-15)

A program that generates seven lottery numbers.
'''
import random


def get_lottery_numbers():
    """
    Get the result from the user.

    :return: User result.
    """
    # Create a list of 7 random numbers in the range 0-9
    return ([random.randrange(9) for _ in range(7)])


def print_lottery_numbers(numbers):
    """
    Print an addition puzzle.

    :return: The result of the addition.
    """
    # Print the list by turning all entries into strings and adding ', '
    print('\t{}'.format(', '.join([str(number) for number in numbers])))


def main():
    '''
    Program main entry point.
    '''
    # Welcome message.
    print('Generated lottery numbers:')
    # Generate the numbers
    numbers = get_lottery_numbers()
    print()
    # Print them
    print_lottery_numbers(numbers)


# Run this when invoked directly
if __name__ == '__main__':
    main()
```

Result



```
Generated lottery numbers:

      8, 6, 1, 4, 0, 5, 7
```

Console output of the program.

4. Number Analysis Program

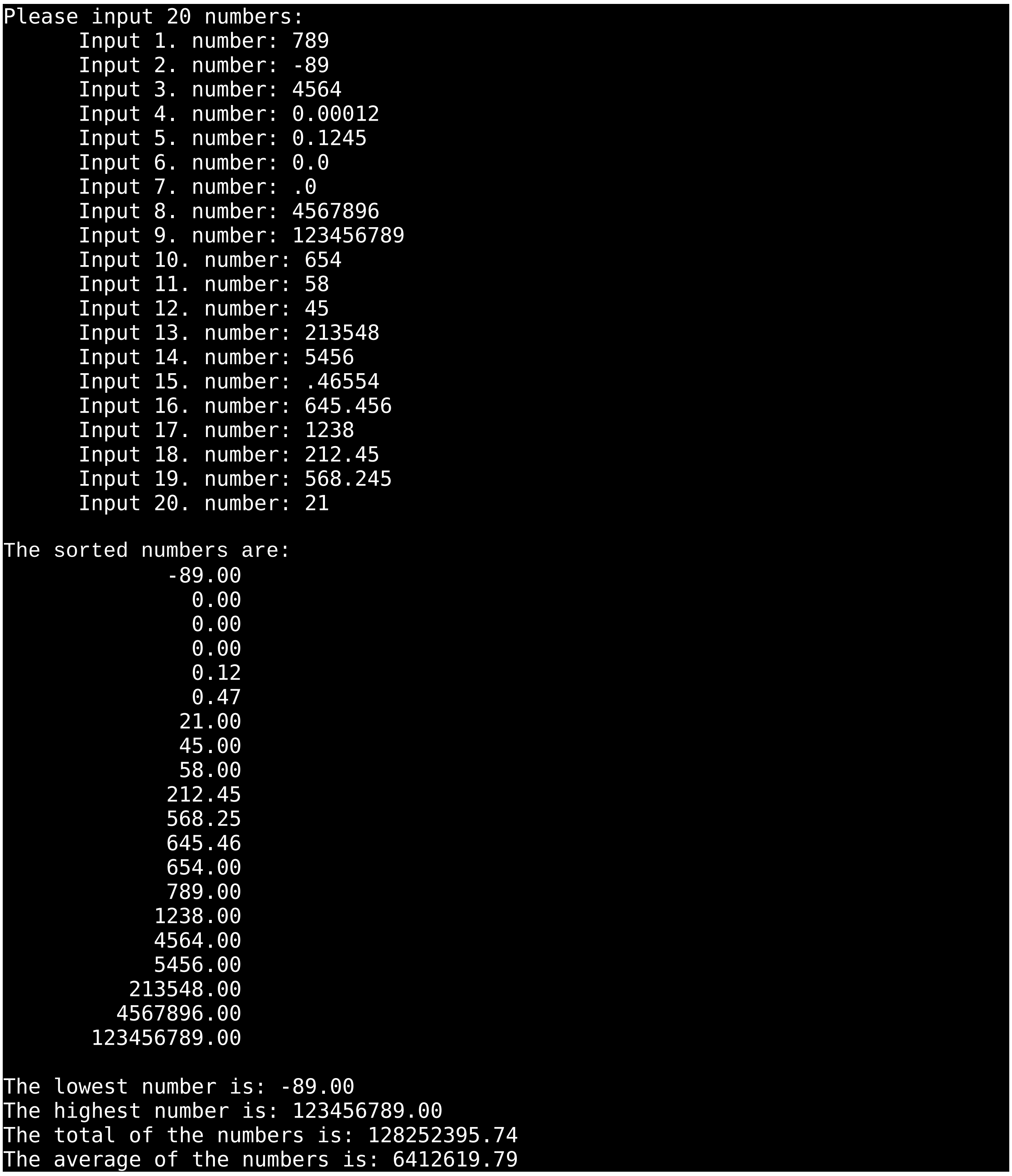
Get 20 numbers from the user and show the lowest, highest, total, and average number. Very KISS use of list, not necessarily the fastest.

prog4.py

```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
# The above lines tell the shell to use python as interpreter when the
# script is called directly, and that this file uses utf-8 encoding,
# because of the country specific letter in my surname.
"""
Name: Program 4 "Number Analysis Program"
Author: Martin Bo Kristensen Grønholdt.
Version: 1.0 (2017-01-15)

Get 20 numbers from the user put them in a list and show:

 * The lowest number in the list
 * The highest number in the list
 * The total of the numbers in the list
 * The average of the numbers in the list.
"""


def get_numbers(numbers):
    """
    Get some numbers from the user.

    :param numbers: Number of numbers to generate.
    :return: A list of random numbers.
    """
    # Create the list.
    number_list = list()
    # Ask nicely.
    print('Please input {} numbers:'.format(numbers))
    try:
        # Counter for the output.
        i = 0
        # Ask for each number.
        for number in range(numbers):
            i += 1
            number_list.append(float(input('\tInput {}. number: '.format(i))))
    except ValueError:
        # Complain when something unexpected was entered.
        print('\nPlease use only numbers.')
        exit(1)

    return (number_list)
```

```python
def print_numbers_info(numbers):
    """
    Print info about a list of numbers

    :param numbers: The list of numbers.
    """
    # The numbers, sorted.
    sorted_numbers = sorted(numbers)

    # Print them
    print('The sorted numbers are:\n\t{}'.format(
        '\n\t'.join(['{:13.2f}'.format(number) for number in sorted_numbers])))
    print()
    # Print the rest of the info.
    print('The lowest number is: {:.2f}'.format(sorted_numbers[0]))
    print('The highest number is: {:.2f}'.format(sorted_numbers[-1]))
    print('The total of the numbers is: {:.2f}'.format(sum(numbers)))
    print('The average of the numbers is: {:.2f}'.format(
        sum(numbers) / len(numbers)))


def main():
    """
    Program main entry point.
    """
    # Handle the input in a list.
    numbers = list()
    numbers = get_numbers(20)

    print()

    print_numbers_info(numbers)


# Run this when invoked directly
if __name__ == '__main__':
    main()
```

Result



```
Please input 20 numbers:
      Input 1. number: 789
      Input 2. number: -89
      Input 3. number: 4564
      Input 4. number: 0.00012
      Input 5. number: 0.1245
      Input 6. number: 0.0
      Input 7. number: .0
      Input 8. number: 4567896
      Input 9. number: 123456789
      Input 10. number: 654
      Input 11. number: 58
      Input 12. number: 45
      Input 13. number: 213548
      Input 14. number: 5456
      Input 15. number: .46554
      Input 16. number: 645.456
      Input 17. number: 1238
      Input 18. number: 212.45
      Input 19. number: 568.245
      Input 20. number: 21

The sorted numbers are:
             -89.00
               0.00
               0.00
               0.00
               0.12
               0.47
              21.00
              45.00
              58.00
             212.45
             568.25
             645.46
             654.00
             789.00
            1238.00
            4564.00
            5456.00
          213548.00
         4567896.00
       123456789.00

The lowest number is: -89.00
The highest number is: 123456789.00
The total of the numbers is: 128252395.74
The average of the numbers is: 6412619.79
```

Output of the program.

9. World Series Champions

Program that reads World Series Champions from a data file, and outputs info about nuber of winds and years that the team has won. I use "`return`" a number of times in the function " `get_team`", I really think this is the cleanest solution, but I know it could have been done using a variable, and a single return statement.

Prog9.py

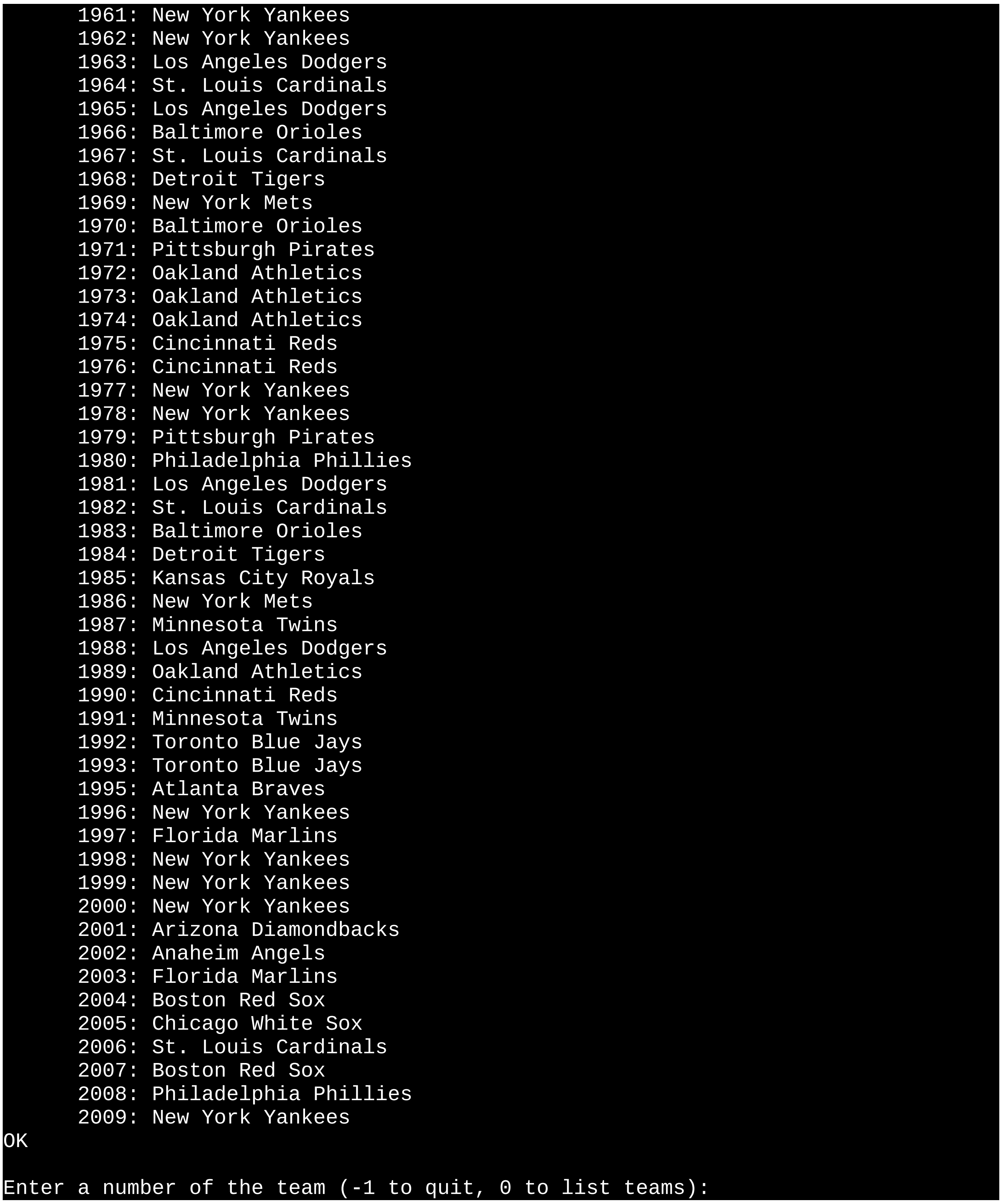
```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
# The above lines tell the shell to use python as interpreter when the
# script is called directly, and that this file uses utf-8 encoding,
# because of the country specific letter in my surname.
"""
Name: Program 9 "Rock, Paper, Scissors Game"
Author: Martin Bo Kristensen Grønholdt.
Version: 1.0 (2017-01-15)

A program that lets the user enter the name of a team and then displays the
number of times that team has won the World Series in the time period from 1903
through 2009.
"""
import random


def load_winners(filename='WorldSeriesWinners.txt'):
    """
    Load the World Series winnin data from a file.

    :return: Dictionary of winners and years.
    """
    print('Loading winner data...')
    winners = dict()
    try:
        # Keep track of the year.
        year = 1903
        # Open 'numbers.txt' for reading.
        with open(filename, 'r') as data_file:
            # Run until 2010
            while (year < 2010):
                # Skip 1904 and 1994
                if (year != 1904) and (year != 1994):
                    # Get the team from the file remove new lines and convert to
                    # lower case to facilitate later matching.
                    team = data_file.readline().strip().lower()
                    print('\t{}:\t{}'.format(year, team.title()))
                    if team in winners.keys():
                        # The team is here update the list with current year.
                        winners[team].append(year)
                    else:
                        # Team is not here create a new list for the winning
                        # years
                        winners[team] = [year]
                year += 1

    except IOError as ex:
        # Complain when something goes wrong with the file access.
        print('Exception: {}'.format(str(ex)))
        print('Error loading winners.')
        exit(1)
    print('OK\n')
    return (winners)
```

```
def get_team(winners):
    """
    Get the the team to get the wins for.

    :return: The name that the user selected.
    """

    def list_teams():
        """
        List all team names in the winners dictionary.
        """
        # Counter for the output
        i = 1
        # Print eac team
        for team in teams:
            print(
                '{}:\t{} ({})'.format(i, team.title(), len(winners[team])))
            i += 1

    # List of all team names
    teams = [winner for winner in winners]
    try:
        # Ask for the sales figure for each day.
        team = int(input(
            '\nEnter a number of the team (-1 to quit, 0 to list teams): '))
        # Give the user a chance to get out.
        if team == -1:
            # Signal that we want out.
            return (None)
        # List team names.
        if team == 0:
            list_teams()
            # Select team.
            return get_team(winners)

        # Adjust for indexing the list
        team -= 1
        print('\t{} selected.'.format(teams[team].title()))
    except ValueError:
        # Complain when something unexpected was entered.
        print('\nPlease use only numbers.\n')
        # Select team.
        return get_team(winners)
    except IndexError:
        # The index was not in the list
        print('\nTeam not found.\n')
        list_teams()
        # Select team.
        return get_team(winners)

    return (teams[team])
```

```python
def print_team_info(team, years):
    """
    Print the number of wins and the winning years of a team.

    :param team: The name of the team.
    :param years: A list of ears the team has won.
    """
    # Create a comma seperated list of the years in a string
    year_str = ','.join(['{:5}'.format(year) for year in years])
    # Print the numbers.
    if len(years) > 1:
        print('\n{} has {} victories in the years:{}'.format(team.title(),
                                                             len(years),
                                                             year_str))

    else:
        print('\n{} has {} victory in: {}'.format(team.title(), len(years),
                                                  year_str))


def main():
    '''
    Program main entry point.
    '''
    # Load the data
    winners = load_winners()
    # Select a team.
    team = get_team(winners)
    # Keep going until the user says stop.
    while team is not None:
        # Print team info
        print_team_info(team, winners[team])
        # Select team
        team = get_team(winners)

    print('\nBye.')


# Run this when invoked directly
if __name__ == '__main__':
    main()
```

Result

```
Loading winner data...
      1903: Boston Americans
      1905: New York Giants
      1906: Chicago White Sox
      1907: Chicago Cubs
      1908: Chicago Cubs
      1909: Pittsburgh Pirates
      1910: Philadelphia Athletics
      1911: Philadelphia Athletics
      1912: Boston Red Sox
      1913: Philadelphia Athletics
      1914: Boston Braves
      1915: Boston Red Sox
      1916: Boston Red Sox
      1917: Chicago White Sox
      1918: Boston Red Sox
      1919: Cincinnati Reds
      1920: Cleveland Indians
      1921: New York Giants
      1922: New York Giants
      1923: New York Yankees
      1924: Washington Senators
      1925: Pittsburgh Pirates
      1926: St. Louis Cardinals
      1927: New York Yankees
      1928: New York Yankees
      1929: Philadelphia Athletics
      1930: Philadelphia Athletics
      1931: St. Louis Cardinals
      1932: New York Yankees
      1933: New York Giants
      1934: St. Louis Cardinals
      1935: Detroit Tigers
      1936: New York Yankees
      1937: New York Yankees
      1938: New York Yankees
      1939: New York Yankees
      1940: Cincinnati Reds
      1941: New York Yankees
      1942: St. Louis Cardinals
      1943: New York Yankees
      1944: St. Louis Cardinals
      1945: Detroit Tigers
      1946: St. Louis Cardinals
      1947: New York Yankees
      1948: Cleveland Indians
      1949: New York Yankees
      1950: New York Yankees
      1951: New York Yankees
      1952: New York Yankees
      1953: New York Yankees
      1954: New York Giants
      1955: Brooklyn Dodgers
      1956: New York Yankees
      1957: Milwaukee Braves
      1958: New York Yankees
      1959: Los Angeles Dodgers
      1960: Pittsburgh Pirates
```



```
      1961: New York Yankees
      1962: New York Yankees
      1963: Los Angeles Dodgers
      1964: St. Louis Cardinals
      1965: Los Angeles Dodgers
      1966: Baltimore Orioles
      1967: St. Louis Cardinals
      1968: Detroit Tigers
      1969: New York Mets
      1970: Baltimore Orioles
      1971: Pittsburgh Pirates
      1972: Oakland Athletics
      1973: Oakland Athletics
      1974: Oakland Athletics
      1975: Cincinnati Reds
      1976: Cincinnati Reds
      1977: New York Yankees
      1978: New York Yankees
      1979: Pittsburgh Pirates
      1980: Philadelphia Phillies
      1981: Los Angeles Dodgers
      1982: St. Louis Cardinals
      1983: Baltimore Orioles
      1984: Detroit Tigers
      1985: Kansas City Royals
      1986: New York Mets
      1987: Minnesota Twins
      1988: Los Angeles Dodgers
      1989: Oakland Athletics
      1990: Cincinnati Reds
      1991: Minnesota Twins
      1992: Toronto Blue Jays
      1993: Toronto Blue Jays
      1995: Atlanta Braves
      1996: New York Yankees
      1997: Florida Marlins
      1998: New York Yankees
      1999: New York Yankees
      2000: New York Yankees
      2001: Arizona Diamondbacks
      2002: Anaheim Angels
      2003: Florida Marlins
      2004: Boston Red Sox
      2005: Chicago White Sox
      2006: St. Louis Cardinals
      2007: Boston Red Sox
      2008: Philadelphia Phillies
      2009: New York Yankees
OK

Enter a number of the team (-1 to quit, 0 to list teams):
```

Output when the program starts

```
Loading winner data...
Exception: [Errno 2] No such file or directory: 'WorldSeriesWinners.txt'
Error loading winners.
```

Output when the programs fails redaing the data file.

```
Enter a number of the team (-1 to quit, 0 to list teams): 0
1:     Anaheim Angels (1)
2:     Boston Americans (1)
3:     Minnesota Twins (2)
4:     St. Louis Cardinals (10)
5:     Cleveland Indians (2)
6:     Washington Senators (1)
7:     Cincinnati Reds (5)
8:     Brooklyn Dodgers (1)
9:     Boston Braves (1)
10:    Kansas City Royals (1)
11:    Arizona Diamondbacks (1)
12:    Baltimore Orioles (3)
13:    Chicago Cubs (2)
14:    Chicago White Sox (3)
15:    Toronto Blue Jays (2)
16:    New York Yankees (27)
17:    Milwaukee Braves (1)
18:    Boston Red Sox (6)
19:    Detroit Tigers (4)
20:    Atlanta Braves (1)
21:    New York Giants (5)
22:    New York Mets (2)
23:    Oakland Athletics (4)
24:    Philadelphia Phillies (2)
25:    Pittsburgh Pirates (5)
26:    Los Angeles Dodgers (5)
27:    Philadelphia Athletics (5)
28:    Florida Marlins (2)

Enter a number of the team (-1 to quit, 0 to list teams):
```

Input when user requests a list of all teams

```
Enter a number of the team (-1 to quit, 0 to list teams): 45

Team not found.

1:    Anaheim Angels (1)
2:    Boston Americans (1)
3:    Minnesota Twins (2)
4:    St. Louis Cardinals (10)
5:    Cleveland Indians (2)
6:    Washington Senators (1)
7:    Cincinnati Reds (5)
8:    Brooklyn Dodgers (1)
9:    Boston Braves (1)
10:   Kansas City Royals (1)
11:   Arizona Diamondbacks (1)
12:   Baltimore Orioles (3)
13:   Chicago Cubs (2)
14:   Chicago White Sox (3)
15:   Toronto Blue Jays (2)
16:   New York Yankees (27)
17:   Milwaukee Braves (1)
18:   Boston Red Sox (6)
19:   Detroit Tigers (4)
20:   Atlanta Braves (1)
21:   New York Giants (5)
22:   New York Mets (2)
23:   Oakland Athletics (4)
24:   Philadelphia Phillies (2)
25:   Pittsburgh Pirates (5)
26:   Los Angeles Dodgers (5)
27:   Philadelphia Athletics (5)
28:   Florida Marlins (2)

Enter a number of the team (-1 to quit, 0 to list teams):
```

Program listing teams when the user select an invalid one.

```
Enter a number of the team (-1 to quit, 0 to list teams): -1

Bye.
```

Output when the user quits the program.

```
Enter a number of the team (-1 to quit, 0 to list teams): 7
      Cincinnati Reds selected.

Cincinnati Reds has 5 victories in the years: 1919, 1940, 1975, 1976, 1990

Enter a number of the team (-1 to quit, 0 to list teams): 5
      Cleveland Indians selected.

Cleveland Indians has 2 victories in the years: 1920, 1948

Enter a number of the team (-1 to quit, 0 to list teams): 12
      Baltimore Orioles selected.

Baltimore Orioles has 3 victories in the years: 1966, 1970, 1983

Enter a number of the team (-1 to quit, 0 to list teams):
```

Output when the user select a valid team.

Conclusion

List is where Python starts getting really fun and can help creating compact, yet clear code.