"""

Created on Sat May 18 16:14:34 2019

Last Edited on 6/11/2019

@author: Daniel Alderman

Purpose: basic web scaper which takes all cubs players and returns sorted list

of all players, user can choose what batting statistic to use for sorting

"""

from urllib.request import urlopen as uReq

from bs4 import BeautifulSoup as soup

#website to scrape

url = 'https://www.baseball-reference.com/teams/CHC/2019.shtml'

#opening page

uClient = uReq(url)

#storing text on page

page\_text = uClient.read()

uClient.close()

#html parsing

page\_clean = soup(page\_text, "html.parser")

#grabs every player and their homeruns

page\_batting = page\_clean.find("div",{"id":"div\_team\_batting"})

players = page\_batting.find\_all("td",{"data-stat":"player"})

homeruns = page\_batting.find\_all("td",{"data-stat":"HR"})

hits = page\_batting.find\_all("td",{"data-stat":"H"})

avgs = page\_batting.find\_all("td",{"data-stat":"batting\_avg"})

opss = page\_batting.find\_all("td",{"data-stat":"onbase\_plus\_slugging"})

rbis = page\_batting.find\_all("td",{"data-stat":"RBI"})

rank = page\_batting.find\_all("th",{"data-stat":"ranker"})

#sets length of array to the number of players, works for all teams

num = len(rank)

length = num - 8

#used to parse out decimals for avg and ops

import re

#goes through for each player stored

for i in range(length):

#clear all data except for number of home runs and reassign it

#try statement catches blank boxes when a player has not at bats

try:

temp = int(''.join(filter(str.isdigit, homeruns[i])))

except Exception:

temp = 0

homeruns[i] = temp

#same thing but for hits

try:

temp = int(''.join(filter(str.isdigit, hits[i])))

except Exception:

temp = 0

hits[i] = temp

#same but for rbi

try:

temp = int(''.join(filter(str.isdigit, rbis[i])))

except Exception:

temp = 0

rbis[i] = temp

#some players have no data for avg so this catches that error

try:

temp = re.findall(r'\d', str(avgs[i]))

temp = float(''.join(filter(str.isdigit, temp))) /1000

except Exception:

temp = -1

avgs[i] = temp

#same thing as above

try:

temp = re.findall(r'\d', str(opss[i]))

temp = float(''.join(filter(str.isdigit, temp))) /1000

except Exception:

temp = -1

opss[i] = temp

#strip extraneous letters from player name and reassign

temp = players[i].text.strip()

players[i] = temp

#creating class of ballplayers where name and stats will be stored for sorting

class Person:

def \_\_init\_\_(self, name, hr, h, avg, ops, rbi):

self.name = name

self.hr = hr

self.h = h

self.rbi = rbi

self.avg = avg

self.ops = ops

#list of person classes

team = []

#creates a person from each player and adds it to team list

for i in range(length):

team.append(Person(players[i], homeruns[i], hits[i], avgs[i], opss[i], rbis[i]))

sorted = False

#goes until user gives proper input to sort with

while(not sorted):

#print("How do you want to sort the players? By hits(h), homeruns(hr), runs batted in (rbi), batting average(avg), or on-base plus slugging(ops)?", end = " ")

sort = input("Sort by: ")

if sort == "h" or sort == "H" or sort == "hits" or sort == "Hits":

team.sort(key=lambda x: x.h, reverse=True)

sorted = True

elif sort == "hr" or sort == "HR" or sort == "Homeruns" or sort == "homeruns":

team.sort(key=lambda x: x.hr, reverse=True)

sorted = True

elif sort == "rbi" or sort == "RBI" or sort == "runs batted in" or sort == "Runs Batted In":

team.sort(key=lambda x: x.rbi, reverse=True)

sorted = True

elif sort == "avg" or sort == "AVG" or sort == "batting average" or sort == "Batting Average":

team.sort(key=lambda x: x.avg, reverse=True)

sorted = True

elif sort == "ops" or sort == "OPS" or sort == "on-base plus slugging" or sort == "On-Base Plus Slugging":

team.sort(key=lambda x: x.ops, reverse=True)

sorted = True

else:

print("Sorry that is not a valid input, please input one of the suggested statistics to sort with.")

#name of new file that will store the data

filename = "baseball-stats.csv"

#titles of the columns in the csv sheet

headers = ["Player Name", "Home runs", "Hits", "Runs Batted In", "Batting Average", "On-base plus Slugging"]

#alerts user to what the symbols after a players name means

explanations = "KEY: \* = bats left-hande; # = bats both; else = bats right; ? = unknown\n"

rows = []

import csv

for i in range(length):

tname = str(team[i].name)

thr = str(team[i].hr)

th = str(team[i].h)

trbi = str(team[i].rbi)

#accounts for no atbats, which I input as -1 for sorting, converting it back

if team[i].avg == -1:

tavg = "No at bats."

else:

tavg = str(team[i].avg)

#same as above but for ops

if team[i].ops == -1:

tops = "No at bats."

else:

tops = str(team[i].ops)

trow = [tname, thr, th, trbi, tavg, tops]

print(trow)

rows.append(trow)

#opens file and then writes the headers, explanation, and rows to it

with open(filename, 'w', newline = '') as csvfile:

csvwriter = csv.writer(csvfile)

csvwriter.writerow([explanations])

csvwriter.writerow(headers)

csvwriter.writerows(rows)