kents06-22704037_13937_rerun

PROJECT CODE

File: common_words

```
  1  #!/bin/bash
  2
  3  #Author: Samuel Kent 22704037
  4
  5  #common_words program that finds all the words in a text and from that the most common
 word
  6  #Can have two optional args: -nth or -w that are used seperately
  7
  8  # -nth option is followed by an integer N. then program is to report the word that is
the Nth most common for the
  9  #largest number of files in the specified directory. Your program must also report the
 number of files for which that
 10  #word is Nth most common
 11
 12  # -w option is followed by a word. and then determines the frequency rank for that wor
d in each text file and reports   #the highest rank
 13
 14
 15  #variables to be used throughout
 16
 17
 18  export intgroup='^[1-9][0-9]*$' #int group regular expression, must be an int greater
than 0
 19  export wordgroup='^[a-zA-Z]+$' #for this assignment a word is a string of letters and
case is preserved
 20
 21
 22  export N_of_textfiles=0
 23  export N_of_empty=0
 24
 25  #variables that become true if a certain arg given
 26  export w_on
 27  export nth_on
 28  export most_common_on
 29
 30
 31  export highest_freq
 32  export highest_fname
 33  export curr_freq
 34
 35
 36  export provided_word
 37  export provided_N
 38  export dir
 39
 40  #usage
 41  #appears when no args are given or if more than 3 args are given
 42  if [ $# -eq 0 ] || [ $# -gt 3 ]
 43  then
 44     echo "Usage: common_words [-w word | -nth N] <directory of text files>"
 45     exit 1
 46  fi
 47
 48
 49  #process the given arguments and ensure correct usage using case, shift and while
 50  while [[ $# -gt 0 ]]
 51  do
 52     case $1 in
 53
 54              -w)     if [[ -d $2 ]] #check that an operand for -w has been given, if dir
is $2 than it cannot have been given
 55                      then
 56                      echo "Operand for -w not provided"
 57                      exit 1
```

kents06-22704037_13937_rerun

```
 58
 59                              elif ! [[ $2 =~ $wordgroup ]] #check that -w operand is a v
alid word (cannot be word-word )
 60                              then
 61                              echo "Operand for -w is not a valid word"
 62                              exit 1
 63
 64                                      elif ! [[ -d $3 ]] #check that a directory given af
ter the -w operand (ie at $3)
 65                                      then
 66                                      echo "A directory must be provided"
 67                                      exit 1
 68                                      fi
 69
 70                      #set the neccasary variables for the -w arg after it's been checked
 to be valid
 71                      provided_word=$2
 72                      dir=$3
 73                      w_on="true"
 74                      shift;; #shift such that $2 becomes $1 etc..
 75
 76              -nth)   if [[ -d $2 ]]
 77                           then
 78                           echo "Operand for -nth not provided"
 79                           exit 1
 80
 81                              elif ! [[ $2 =~ $intgroup ]] #operand belong to intgroup, i
e is an int greater than 0
 82                                   then
 83                                   echo "Operand for -nth argument must be an Integer and
 be greater than 0"
 84                                   exit 1
 85
 86                                      elif ! [[ -d $3 ]]
 87                                           then
 88                                           echo "A directory must be provided"
 89                                           exit 1
 90                                           fi
 91
 92                      provided_N=$2
 93                      dir=$3
 94                      nth_on="true"
 95                      shift;;
 96
 97              -*)     echo "Unknown argument $1" #if anything else following "-" is given
 is not a valid option
 98                           exit 1;;
 99
100               *)  if ! [[ -d $1 ]] #if only one arg given check it is a directory
101                           then
102                           echo "A directory must be provided"
103                           exit 1
104                           fi
105
106                      #if neither -nth or -w has been used and only a dir is given, than
find the 1st most common
107                      if [ -z $nth_on ] && [ -z $w_on ]
108                      then
109                           dir=$1
110                      most_common_on="true"
111                      fi
112
113                           shift;;
114
115              esac
116
117              shift
118
```

```
119     done
120
121
122
123
124   #checking that the Directory is Usable:
125   #check there is atleast 1 non empty textfile in the directory
126   if ! [ "$(ls -A $dir)" ] #returns number of files in a dir
127   then
128             echo "Given directory $dir is empty"
129           exit 1
130   fi
131
132   #for loop to find the number of textfiles in the directory, to be used later
133   for f in $dir/*
134   do
135      N_of_textfiles=$(( $N_of_textfiles + 1 )) #iterate when a new txt file found
136
137
138      if [[ "$( wc -w $f | cut -d" " -f1 )" == "0" ]] #checks to see if the curretn text
file is empty
139      then
140              N_of_empty=$(( $N_of_empty + 1)) #iterate if empty file found
141      fi
142
143   done
144
145   #if the number of textfiles is equal to the N of empty textfiles than the dir has no t
extfiles that are not empty
146   if [[ "$N_of_textfiles" == "$N_of_empty" ]]
147   then
148      echo "must be atleast 1 non-empty textfile in the directory"
149      exit 1
150   fi
151
152
153
154
155
156
157
158   #process for if -w used
159   if [[ "$w_on" == "true" ]] #turned on earlier when -w <word> provided
160   then
161
162      #loop through each file in the dir
163      for f in $dir/*
164      do
165
166              #get everyword in the file, remove duplicates, sort by order of how often t
hey appear (highest on top), find the provided_word (ie the -w operand) and take the line n
umber only (frequency)
167              grab_freq=$(tr -cs '[A-Za-z]' '\012' < $f | sort | uniq -c | sort -k 1nr |
grep -nw "$provided_word" | cut -d':' -f1)
168
169              if ! [[ -z $grab_freq ]] #if $grab_freq is not null ie exists in the curren
t file, set it as the curr_frequency
170              then
171                      curr_freq=$grab_freq
172              fi
173
174              if [[ -z "$highest_freq" ]] #first time highest_freq is set, is when a word
 is found of any frequency ie is null
175              then
176              highest_freq=$curr_freq
177              highest_fname=$f
178
179                      elif [[ "$curr_freq" -lt "$highest_freq" ]] #if curr_freq is lower,
```

```
 ie higher rank and more frequent. set it as the new highest_freq
180                     then
181                     highest_freq=$curr_freq
182                     highest_fname=$f
183                     fi
184
185     done
186
187             #if highest_freq is never set then the word was not found in any files
188             if [[ -z "$highest_freq" ]]
189             then
190                     echo "Word Not found in any files of this directory"
191                     exit 0 #exit success the program has still worked as intended
192             fi
193
194  #Output the result to the user, use sed to cut the directory name ie leave only the na
me of the file.txt
195  echo "The most significant rank for the word $provided_word is $highest_freq in file $
( sed 's#.*/##' <<< "$highest_fname") "
196
197  fi
198
199
200
201
202
203
204
205  #process for when -nth is used
206  if [[ "$nth_on" == "true" || "$most_common_on" == "true" ]] #used if either -nth or no
 arg
207  then
208     if [ "$most_common_on" == "true" ] && [ -z $nth_on ] #if no arg given set provided_
N to 1 and act as if -nth 1 given
209     then
210             provided_N="1"
211     fi
212
213
214     #iterate through each file, outer loop
215     for f in $dir/*
216     do
217
218     #order each word in current file ($f), arrange by frequency, search for line number
 N and take the name of the word at that frequency (line number) using awk
219     curr_word=$(tr -cs '[A-Za-z]' '\012' < $f | sort | uniq -c | sort -k 1nr | sed -n "
$provided_N"p | awk '{s=$2} END {print s}')
220
221     #since a word has been found in the first file set the count to 1, if no word is th
ere (N too large for that file) and curr_word is null will be checked later
222     #ie reset the count as a new word is now being compared against the other files

223     curr_count=1
224
225             #inner loop that compares the word found in $f at Nth rank to the words in
the same rank in every other file
226             for i in $dir/*
227             do
228                     #find the word at the same N (frequency rank) but for $i
229                     curr_word_i=$(tr -cs '[A-Za-z]' '\012' < $i | sort | uniq -c | sort
 -k 1nr | sed -n "$provided_N"p | awk '{s=$2} END {print s}')
230
231                     #compare the word found in the outer loop to word in the same pos f
or all other files, must be a different file to outer loop, and the word must not be null
232                     #if they are the same word than iterate the count, the word has bee
n found at the same rank for another file
233                     if [ "$i" != "$f" ] && [ "$curr_word_i" == "$curr_word" ] && [ ! -z
 $curr_word_i ]
```

kents06-22704037_13937_rerun

```
234                     then
235
236                     curr_count=$(( $curr_count + 1 ))
237
238                     fi
239             done
240
241     #if highest_count hasnt been changed yet and the curr_word (in the outer loop) is n
on-null set it as the current highest
242     #ie set the first word encountered as the highest initially even if it is only in t
hat posiiton once
243     if [ -z $highest_count ] && [ ! -z $curr_word ]
244     then
245     highest_count=$curr_count
246     highest_wname=$curr_word
247
248     #if the current word is found to belong in more files at that rank than the current
 highest they will be swapped
249     elif [[ "$highest_count" -lt "$curr_count" ]]
250     then
251     highest_count=$curr_count
252          highest_wname=$curr_word
253
254     fi
255
256
257     done
258
259
260
261     #if highest_wname is never set, a non-null word has not been encountered at that ra
nk. then the Nth word does not exist within any of the textfiles ie Nth is too large
262     if [[ -z $highest_wname ]]
263     then
264             echo ""$provided_N"th word is outside of the range for this directory"
265             exit 0 #program has still ran as intended
266     fi
267
268
269     #output for the user
270     echo "The "$provided_N"th most commmon word is "\""$highest_wname"\"" across $highe
st_count files"
271     exit 0
272
273  fi
274
275
276
277  #end of script
278
```

File: malaria_incidence

```
  1  #!/bin/bash
  2
  3  # Author: Samuel Kent 22704037
  4
  5  # malaria_incidence takes 1 arg only. reads from incedenceOfMalaria.csv and uses title
_case.py
  6
  7  # if arg is year: report country with highest incidence of malaria for that year + the
 incidence value
  8  # if arg is Country: report year with highest incidence for that country + the inciden
ce value
  9
 10  #THE PYTHON SCRIPT IS EXPECTED TO BE IN THE SAME DIRECTORY FOR THIS SCRIPT. IS NAMED t
```

kents06-22704037_13937_rerun

itle_case

```
 11
 12
 13  #set the internal field seperator to be by line for later
 14  IFS=$'\n'
 15
 16
 17  #Some variables with global scope for later:
 18
 19  #variable to check if is a valid int
 20  export intgroup='^[0-9]+$'
 21
 22  export year
 23  export country
 24
 25  export validcountry
 26  export validyear
 27
 28  export country_no_brackets
 29  export i_no_brackets
 30
 31
 32
 33
 34
 35  # CHECK USAGE
 36  #check only 1 non-zero arg is given
 37  if [[ $# -ne 1 ]] || [[ ! -n $1 ]]
 38  then
 39          echo "Illegal number of parameters"
 40     echo "Usage: $0 <argument>"
 41     exit 1 #exit failure is 1 for this program, success 0
 42  fi
 43
 44
 45
 46  #if arg is not a number save it as a country
 47  #fully capitalize all the words in the country name (using title_case.py)
 48  if ! [[ $1 =~ $intgroup ]]
 49  then
 50     country=$( echo $1 | ./title_case.py -)
 51     # country=$( echo $1 | ./title_case -)  Changed by MJW to add .py
 52     country_no_brackets=$( echo $country | sed -e 's/([^()]*)//g' | sed -e 's/ //g'  )
#remove the brackets and the brackets contents for the sake of continuity
 53  fi
 54
 55
 56
 57  #if arg is a number (ie belongs to intgroup) and fits within the year range save it as
 the year, otherwise exit failure if outside range
 58  if [[ $1 =~ $intgroup && $1 < 2019 && $1 > 1999 ]]
 59  then
 60          year=$1
 61     validyear=true #is set to later activate the year function
 62
 63  elif [[ $1 =~ $intgroup && $1 > 2018 || $1 < 2000 ]]
 64  then
 65     echo "invalid year (must be between 2000 - 2018 inclusive)"
 66     exit 1
 67  fi
 68
 69
 70  #Handle unique edge case for Vietnam
 71  if [ "$1" == "Vietnam" ] || [ "$1" == "vietnam" ]
 72  then
 73     validcountry=true
 74     country="Viet Nam"
 75     country_no_brackets=$( echo $country | sed -e 's/([^()]*)//g' | sed -e 's/ //g'  )
```

```
 76  fi
 77
 78
 79
 80
 81  #if a word given check if it's a valid country
 82  #take list of unique countries from incedenceOfMalaria.csv (also fully capitalized) ,l
oop through them to see if a valid country has been given
 83  if ! [[ $1 =~ $intgroup ]]
 84  then
 85          validcountries=$( tail incedenceOfMalaria.csv -n +2 | cut -d, -f1 | sort | uni
q) #list of validcountries from the .csv file, ignore first line and take names only
 86
 87     #loop through all country names from the .csv
 88          for i in $validcountries
 89          do
 90
 91          curr_country=$( echo $i | ./title_case.py - )
 92          # curr_country=$( echo $i | ./title_case - ) Changed by MJW
 93          curr_country_no_brackets=$( echo $curr_country | sed -e 's/([^()]*)//g' | sed
-e 's/ //g' ) #also remove brackets for continuity
 94
 95
 96                       #compare them both, both capitilzed and brackets removed
 97                            if [[ $curr_country_no_brackets == $country_no_brackets ]]
 98                            then
 99
100                            validcountry=true #flag meaning the country is in the .csv
101                            country=$i #save as the countries orginal name within the .csv
 file
102
103                            break
104                            fi
105
106          done
107
108  #throw error if the given country cannot be found
109  if [[ $validcountry != "true" ]]
110  then
111
112          echo "Invalid country for this database"
113          exit 1
114  fi
115
116
117  fi
118
119
120
121  #If a valid country that is in the .csv is given then analyse the incidence
122  if [[ $validcountry == "true" ]]
123  then
124     #use the capitilzed and brackets removed version
125          country_analysing=$country_no_brackets
126
127     #initialize variables
128     highest_incidence=0
129     highest_year=0
130
131
132     #loop through the .csv file line by line for that particular country, remove the 2n
d column as it is useless
133          for j in $( tail incedenceOfMalaria.csv -n +2 | cut -d, -f1,3,4 | grep -F "$co
untry"  )
134          do
135
136
137             #take the information for the current line
```

kents06-22704037_13937_rerun

```
138                 curr_incidence=$( echo $j | cut -d, -f3 )
139                 curr_country=$( echo $j | cut -d, -f1 | sed -e 's/([^()]*)//g' | sed -
e 's/ //g' )
140             curr_year=$( echo $j | cut -d, -f2 )
141
142
143             #if a greater incidence is found then set it as the highest
144                 if [[ "$curr_incidence" -gt "$highest_incidence" ]]
145                 then
146                                 highest_incidence=$curr_incidence
147
148                             highest_year=$curr_year
149
150                 fi
151
152         done
153
154     #output to the user and exit success
155         echo "For the country $country, the year with the highest incidence was $highe
st_year, with a rate of $highest_incidence per 1,000"
156     exit 0
157  fi
158
159
160
161
162
163  #Function for if a valid year has been given
164  if [[ $validyear == "true" ]]
165  then
166     #initialize variables
167     highest_incidence=0
168     curr_incidence=0
169
170
171     #loop through .csv file line by line where the year is the same as the given year,
also remove the 2nd column again
172     for a in $( tail incedenceOfMalaria.csv -n +2 | cut -d, -f1,3,4 | grep -F "$year"
)
173     do
174             #info taken from the current line
175             curr_incidence=$( echo $a | cut -d, -f3 )
176             curr_country=$( echo $a | cut -d, -f1 )
177
178
179             #if the current incidence of this line is greater than the saved one, swap
them
180             if [[ "$curr_incidence" -gt "$highest_incidence" ]]
181                 then
182
183                                 highest_incidence=$curr_incidence
184                                 highest_country=$curr_country
185
186             fi
187
188     done
189
190     #output to the user and exit success
191     echo "For the year $year, the country with the highest incidence was $highest_count
ry, with a rate of $highest_incidence per 1,000"
192     exit 0
193  fi
194
195
196
197
198
199
```

kents06-22704037_13937_rerun

File: title_case

```
  1  #!/usr/bin/env python
  2  # Convert input string to title case
  3
  4  import sys
  5
  6  common_words = set(["a", "about", "after", "against", "all", "also", "an", "and", "another", "any",
  7           "are", "as", "at", "back", "be", "because", "been", "before", "being", "between", "both",
  8           "but", "by", "came", "can", "come", "could", "day", "did", "do", "down", "each", "even",
  9           "first", "for", "from", "get", "go", "good", "great", "had", "has", "have", "he", "her",
 10           "here", "him", "his", "how", "i", "if", "in", "into", "is", "it", "its", "just", "know",
 11           "last", "life", "like", "little", "long", "made", "make", "man", "many", "may", "me", "men",
 12           "might", "more", "most", "mr", "much", "must", "my", "never", "new", "no", "not", "now",
 13           "of", "off", "old", "on", "one", "only", "or", "other", "our", "out", "over", "own", "people",
 14           "right", "said", "same", "see", "she", "should", "since", "so", "some", "state", "still",
 15           "such", "take", "than", "that", "the", "their", "them", "then", "there", "these", "they",
 16           "this", "those", "three", "through", "time", "to", "too", "two", "under", "up", "us", "used",
 17           "very", "was", "way", "we", "well", "were", "what", "when", "where", "which", "while", "who",
 18           "will", "with", "work", "world", "would", "year", "years", "you", "your"])
 19
 20  def convert() :
 21    args = sys.argv
 22    if len(args) != 2 :
 23      sys.stdout.write("Usage: {0} <input string or -, if being used as a pipe>\n".format(args[0]))
 24      sys.exit(0)
 25
 26    if args[1] != '-' :
 27      sys.stdout.write("{0}\n".format(convert_string(args[1])))
 28      return
 29
 30
```

kents06-22704037_13937_rerun

```
 31    line = sys.stdin.readline()
 32    while line != "" :
 33      sys.stdout.write("{0}\n".format(convert_string(line)))
 34      line = sys.stdin.readline()
 35    return
 36
 37  # converts string to title case
 38  def convert_string(instring) :
 39    words = instring.split()
 40    new_words = [words[0].capitalize()]
 41    for w in words[1:] :
 42      if w in common_words :
 43        new_words.append(w)
 44      else :
 45        new_words.append(w.lower().capitalize())
 46    return(" ".join(new_words))
 47
 48  if __name__ == "__main__" :
 49    convert()
```

File: x

```
  1  PROJECT CODE
  2
  3  File: common_words
  4
  5    1  #!/bin/bash
  6    2
  7    3  #Author: Samuel Kent 22704037
  8    4
  9    5  #common_words program that finds all the words in a text and from that the most c
ommon word
 10    6  #Can have two optional args: -nth or -w that are used seperately
 11    7
 12    8  # -nth option is followed by an integer N. then program is to report the word tha
t is the Nth most common for the
 13    9  #largest number of files in the specified directory. Your program must also repor
t the number of files for which that
 14   10  #word is Nth most common
 15   11
 16   12  # -w option is followed by a word. and then determines the frequency rank for tha
t word in each text file and reports   #the highest rank
 17   13
 18   14
 19   15  #variables to be used throughout
 20   16
 21   17
 22   18  export intgroup='^[1-9][0-9]*$' #int group regular expression, must be an int gre
ater than 0
 23   19  export wordgroup='^[a-zA-Z]+$' #for this assignment a word is a string of letters
 and case is preserved
 24   20
 25   21
 26   22  export N_of_textfiles=0
 27   23  export N_of_empty=0
 28   24
 29   25  #variables that become true if a certain arg given
 30   26  export w_on
 31   27  export nth_on
 32   28  export most_common_on
 33   29
 34   30
 35   31  export highest_freq
 36   32  export highest_fname
 37   33  export curr_freq
 38   34
```

kents06-22704037_13937_rerun

```
 39   35
 40   36  export provided_word
 41   37  export provided_N
 42   38  export dir
 43   39
 44   40  #usage
 45   41  #appears when no args are given or if more than 3 args are given
 46   42  if [ $# -eq 0 ] || [ $# -gt 3 ]
 47   43  then
 48   44        echo "Usage: common_words [-w word | -nth N] <directory of text files>"
 49   45        exit 1
 50   46  fi
 51   47
 52   48
 53   49  #process the given arguments and ensure correct usage using case, shift and while
 54   50  while [[ $# -gt 0 ]]
 55   51  do
 56   52        case $1 in
 57   53
 58   54                -w)     if [[ -d $2 ]] #check that an operand for -w has been given
, if dir is $2 than it cannot have been given
 59   55                        then
 60   56                        echo "Operand for -w not provided"
 61   57                        exit 1
 62   58
 63   59                                elif ! [[ $2 =~ $wordgroup ]] #check that -w operan
d is a valid word (cannot be word-word )
 64   60                                then
 65   61                                echo "Operand for -w is not a valid word"
 66   62                                exit 1
 67   63
 68   64                                        elif ! [[ -d $3 ]] #check that a directory
given after the -w operand (ie at $3)
 69   65                                        then
 70   66                                        echo "A directory must be provided"
 71   67                                        exit 1
 72   68                                        fi
 73   69
 74   70                        #set the neccasary variables for the -w arg after it's been
 checked to be valid
 75   71                        provided_word=$2
 76   72                        dir=$3
 77   73                        w_on="true"
 78   74                        shift;; #shift such that $2 becomes $1 etc..
 79   75
 80   76                -nth)   if [[ -d $2 ]]
 81   77                          then
 82   78                          echo "Operand for -nth not provided"
 83   79                          exit 1
 84   80
 85   81                                elif ! [[ $2 =~ $intgroup ]] #operand belong to int
group, ie is an int greater than 0
 86   82                                  then
 87   83                                  echo "Operand for -nth argument must be an Intege
r and be greater than 0"
 88   84                                  exit 1
 89   85
 90   86                                        elif ! [[ -d $3 ]]
 91   87                                          then
 92   88                                          echo "A directory must be provided"
 93   89                                          exit 1
 94   90                                          fi
 95   91
 96   92                        provided_N=$2
 97   93                        dir=$3
 98   94                        nth_on="true"
 99   95                        shift;;
100   96
```

kents06-22704037_13937_rerun

```
101   97                -*)     echo "Unknown argument $1" #if anything else following "-" is given is not a valid option
102   98                          exit 1;;
103   99
104  100                 *)  if ! [[ -d $1 ]] #if only one arg given check it is a directory
105  101                          then
106  102                          echo "A directory must be provided"
107  103                          exit 1
108  104                          fi
109  105
110  106                         #if neither -nth or -w has been used and only a dir is given, than find the 1st most common
111  107                         if [ -z $nth_on ] && [ -z $w_on ]
112  108                         then
113  109                           dir=$1
114  110                         most_common_on="true"
115  111                         fi
116  112
117  113                           shift;;
118  114
119  115                esac
120  116
121  117                shift
122  118
123  119        done
124  120
125  121
126  122
127  123
128  124  #checking that the Directory is Usable:
129  125  #check there is atleast 1 non empty textfile in the directory
130  126  if ! [ "$(ls -A $dir)" ] #returns number of files in a dir
131  127  then
132  128        echo "Given directory $dir is empty"
133  129          exit 1
134  130  fi
135  131
136  132  #for loop to find the number of textfiles in the directory, to be used later
137  133  for f in $dir/*
138  134  do
139  135        N_of_textfiles=$(( $N_of_textfiles + 1 )) #iterate when a new txt file found
140  136
141  137
142  138        if [[ "$( wc -w $f | cut -d" " -f1 )" == "0" ]] #checks to see if the curretn text file is empty
143  139        then
144  140                N_of_empty=$(( $N_of_empty + 1)) #iterate if empty file found
145  141        fi
146  142
147  143  done
148  144
149  145  #if the number of textfiles is equal to the N of empty textfiles than the dir has no textfiles that are not empty
150  146  if [[ "$N_of_textfiles" == "$N_of_empty" ]]
151  147  then
152  148        echo "must be atleast 1 non-empty textfile in the directory"
153  149        exit 1
154  150  fi
155  151
156  152
157  153
158  154
159  155
160  156
161  157
162  158  #process for if -w used
```

kents06-22704037_13937_rerun

```
163  159  if [[ "$w_on" == "true" ]] #turned on earlier when -w <word> provided
164  160  then
165  161
166  162          #loop through each file in the dir
167  163          for f in $dir/*
168  164          do
169  165
170  166                  #get everyword in the file, remove duplicates, sort by order of how
 often they appear (highest on top), find the provided_word (ie the -w operand) and take th
e line number only (frequency)
171  167                  grab_freq=$(tr -cs '[A-Za-z]' '\012' < $f | sort | uniq -c | sort -
k 1nr | grep -nw "$provided_word" | cut -d':' -f1)
172  168
173  169                  if ! [[ -z $grab_freq ]] #if $grab_freq is not null ie exists in th
e current file, set it as the curr_frequency
174  170                  then
175  171                          curr_freq=$grab_freq
176  172                  fi
177  173
178  174                  if [[ -z "$highest_freq" ]] #first time highest_freq is set, is whe
n a word is found of any frequency ie is null
179  175                  then
180  176                  highest_freq=$curr_freq
181  177                  highest_fname=$f
182  178
183  179                          elif [[ "$curr_freq" -lt "$highest_freq" ]] #if curr_freq i
s lower, ie higher rank and more frequent. set it as the new highest_freq
184  180                          then
185  181                          highest_freq=$curr_freq
186  182                          highest_fname=$f
187  183                          fi
188  184
189  185          done
190  186
191  187                  #if highest_freq is never set then the word was not found in any fi
les
192  188                  if [[ -z "$highest_freq" ]]
193  189                  then
194  190                          echo "Word Not found in any files of this directory"
195  191                          exit 0 #exit success the program has still worked as intend
ed
196  192                  fi
197  193
198  194  #Output the result to the user, use sed to cut the directory name ie leave only t
he name of the file.txt
199  195  echo "The most significant rank for the word $provided_word is $highest_freq in f
ile $( sed 's#.*/##' <<< "$highest_fname") "
200  196
201  197  fi
202  198
203  199
204  200
205  201
206  202
207  203
208  204
209  205  #process for when -nth is used
210  206  if [[ "$nth_on" == "true" || "$most_common_on" == "true" ]] #used if either -nth
or no arg
211  207  then
212  208          if [ "$most_common_on" == "true" ] && [ -z $nth_on ] #if no arg given set p
rovided_N to 1 and act as if -nth 1 given
213  209          then
214  210                  provided_N="1"
215  211          fi
216  212
217  213
218  214          #iterate through each file, outer loop
```

kents06-22704037_13937_rerun

```
219  215        for f in $dir/*
220  216        do
221  217
222  218        #order each word in current file ($f), arrange by frequency, search for lin
e number N and take the name of the word at that frequency (line number) using awk
223  219        curr_word=$(tr -cs '[A-Za-z]' '\012' < $f | sort | uniq -c | sort -k 1nr |
sed -n "$provided_N"p | awk '{s=$2} END {print s}')
224  220
225  221        #since a word has been found in the first file set the count to 1, if no wo
rd is there (N too large for that file) and curr_word is null will be checked later
226  222        #ie reset the count as a new word is now being compared against the other f
iles
227  223        curr_count=1
228  224
229  225                #inner loop that compares the word found in $f at Nth rank to the w
ords in the same rank in every other file
230  226                for i in $dir/*
231  227                do
232  228                        #find the word at the same N (frequency rank) but for $i
233  229                        curr_word_i=$(tr -cs '[A-Za-z]' '\012' < $i | sort | uniq -
c | sort -k 1nr | sed -n "$provided_N"p | awk '{s=$2} END {print s}')
234  230
235  231                        #compare the word found in the outer loop to word in the sa
me pos for all other files, must be a different file to outer loop, and the word must not b
e null
236  232                        #if they are the same word than iterate the count, the word
 has been found at the same rank for another file
237  233                        if [ "$i" != "$f" ] && [ "$curr_word_i" == "$curr_word" ] &
& [ ! -z $curr_word_i ]
238  234                        then
239  235
240  236                        curr_count=$(( $curr_count + 1 ))
241  237
242  238                        fi
243  239                done
244  240
245  241        #if highest_count hasnt been changed yet and the curr_word (in the outer lo
op) is non-null set it as the current highest
246  242        #ie set the first word encountered as the highest initially even if it is o
nly in that posiiton once
247  243        if [ -z $highest_count ] && [ ! -z $curr_word ]
248  244        then
249  245        highest_count=$curr_count
250  246        highest_wname=$curr_word
251  247
252  248        #if the current word is found to belong in more files at that rank than the
 current highest they will be swapped
253  249        elif [[ "$highest_count" -lt "$curr_count" ]]
254  250        then
255  251        highest_count=$curr_count
256  252          highest_wname=$curr_word
257  253
258  254        fi
259  255
260  256
261  257        done
262  258
263  259
264  260
265  261        #if highest_wname is never set, a non-null word has not been encountered at
 that rank. then the Nth word does not exist within any of the textfiles ie Nth is too larg
e
266  262        if [[ -z $highest_wname ]]
267  263        then
268  264                echo ""$provided_N"th word is outside of the range for this directo
ry"
269  265                exit 0 #program has still ran as intended
270  266        fi
```

kents06-22704037_13937_rerun

```
271  267
272  268
273  269         #output for the user
274  270         echo "The "$provided_N"th most commmon word is "\""$highest_wname"\"" acros
s $highest_count files"
275  271         exit 0
276  272
277  273  fi
278  274
279  275
280  276
281  277  #end of script
282  278
283
284
285
286  File: malaria_incidence
287
288    1  #!/bin/bash
289    2
290    3  # Author: Samuel Kent 22704037
291    4
292    5  # malaria_incidence takes 1 arg only. reads from incedenceOfMalaria.csv and uses
title_case.py
293    6
294    7  # if arg is year: report country with highest incidence of malaria for that year
+ the incidence value
295    8  # if arg is Country: report year with highest incidence for that country + the in
cidence value
296    9
297   10  #THE PYTHON SCRIPT IS EXPECTED TO BE IN THE SAME DIRECTORY FOR THIS SCRIPT. IS NA
MED title_case
298   11
299   12
300   13  #set the internal field seperator to be by line for later
301   14  IFS=$'\n'
302   15
303   16
304   17  #Some variables with global scope for later:
305   18
306   19  #variable to check if is a valid int
307   20  export intgroup='^[0-9]+$'
308   21
309   22  export year
310   23  export country
311   24
312   25  export validcountry
313   26  export validyear
314   27
315   28  export country_no_brackets
316   29  export i_no_brackets
317   30
318   31
319   32
320   33
321   34
322   35  # CHECK USAGE
323   36  #check only 1 non-zero arg is given
324   37  if [[ $# -ne 1 ]] || [[ ! -n $1 ]]
325   38  then
326   39          echo "Illegal number of parameters"
327   40        echo "Usage: $0 <argument>"
328   41        exit 1 #exit failure is 1 for this program, success 0
329   42  fi
330   43
331   44
332   45
333   46  #if arg is not a number save it as a country
```

kents06-22704037_13937_rerun

```
334   47  #fully capitalize all the words in the country name (using title_case.py)
335   48  if ! [[ $1 =~ $intgroup ]]
336   49  then
337   50        country=$( echo $1 | ./title_case.py -)
338   51        # country=$( echo $1 | ./title_case -)  Changed by MJW to add .py
339   52        country_no_brackets=$( echo $country | sed -e 's/([^()]*)//g' | sed -e 's/
//g'  ) #remove the brackets and the brackets contents for the sake of continuity
340   53  fi
341   54
342   55
343   56
344   57  #if arg is a number (ie belongs to intgroup) and fits within the year range save
it as the year, otherwise exit failure if outside range
345   58  if [[ $1 =~ $intgroup && $1 < 2019 && $1 > 1999 ]]
346   59  then
347   60          year=$1
348   61        validyear=true #is set to later activate the year function
349   62
350   63  elif [[ $1 =~ $intgroup && $1 > 2018 || $1 < 2000 ]]
351   64  then
352   65        echo "invalid year (must be between 2000 - 2018 inclusive)"
353   66        exit 1
354   67  fi
355   68
356   69
357   70  #Handle unique edge case for Vietnam
358   71  if [ "$1" == "Vietnam" ] || [ "$1" == "vietnam" ]
359   72  then
360   73        validcountry=true
361   74        country="Viet Nam"
362   75        country_no_brackets=$( echo $country | sed -e 's/([^()]*)//g' | sed -e 's/
//g'  )
363   76  fi
364   77
365   78
366   79
367   80
368   81  #if a word given check if it's a valid country
369   82  #take list of unique countries from incedenceOfMalaria.csv (also fully capitalize
d) ,loop through them to see if a valid country has been given
370   83  if ! [[ $1 =~ $intgroup ]]
371   84  then
372   85          validcountries=$( tail incedenceOfMalaria.csv -n +2 | cut -d, -f1 | sort
| uniq) #list of validcountries from the .csv file, ignore first line and take names only
373   86
374   87        #loop through all country names from the .csv
375   88          for i in $validcountries
376   89          do
377   90
378   91          curr_country=$( echo $i | ./title_case.py - )
379   92          # curr_country=$( echo $i | ./title_case - ) Changed by MJW
380   93          curr_country_no_brackets=$( echo $curr_country | sed -e 's/([^()]*)//g' |
 sed -e 's/ //g' ) #also remove brackets for continuity
381   94
382   95
383   96                          #compare them both, both capitilzed and brackets removed
384   97                            if [[ $curr_country_no_brackets == $country_no_brackets ]
]
385   98                            then
386   99
387  100                            validcountry=true #flag meaning the country is in the .cs
v
388  101                            country=$i #save as the countries orginal name within the
 .csv file
389  102
390  103                            break
391  104                            fi
392  105
```

kents06-22704037_13937_rerun

```
393  106          done
394  107
395  108  #throw error if the given country cannot be found
396  109  if [[ $validcountry != "true" ]]
397  110  then
398  111
399  112          echo "Invalid country for this database"
400  113          exit 1
401  114  fi
402  115
403  116
404  117  fi
405  118
406  119
407  120
408  121  #If a valid country that is in the .csv is given then analyse the incidence
409  122  if [[ $validcountry == "true" ]]
410  123  then
411  124        #use the capitilzed and brackets removed version
412  125          country_analysing=$country_no_brackets
413  126
414  127        #initialize variables
415  128        highest_incidence=0
416  129        highest_year=0
417  130
418  131
419  132        #loop through the .csv file line by line for that particular country, remov
e the 2nd column as it is useless
420  133          for j in $( tail incedenceOfMalaria.csv -n +2 | cut -d, -f1,3,4 | grep -F
 "$country"  )
421  134          do
422  135
423  136
424  137                #take the information for the current line
425  138                  curr_incidence=$( echo $j | cut -d, -f3 )
426  139                  curr_country=$( echo $j | cut -d, -f1 | sed -e 's/([^()]*)//g' |
sed -e 's/ //g' )
427  140                curr_year=$( echo $j | cut -d, -f2 )
428  141
429  142
430  143                #if a greater incidence is found then set it as the highest
431  144                  if [[ "$curr_incidence" -gt "$highest_incidence" ]]
432  145                  then
433  146                                  highest_incidence=$curr_incidence
434  147
435  148                                highest_year=$curr_year
436  149
437  150                  fi
438  151
439  152          done
440  153
441  154        #output to the user and exit success
442  155          echo "For the country $country, the year with the highest incidence was $
highest_year, with a rate of $highest_incidence per 1,000"
443  156        exit 0
444  157  fi
445  158
446  159
447  160
448  161
449  162
450  163  #Function for if a valid year has been given
451  164  if [[ $validyear == "true" ]]
452  165  then
453  166        #initialize variables
454  167        highest_incidence=0
455  168        curr_incidence=0
456  169
```

kents06-22704037_13937_rerun

```
457  170
458  171         #loop through .csv file line by line where the year is the same as the give
n year, also remove the 2nd column again
459  172         for a in $( tail incedenceOfMalaria.csv -n +2 | cut -d, -f1,3,4 | grep -F "
$year"  )
460  173         do
461  174                 #info taken from the current line
462  175                 curr_incidence=$( echo $a | cut -d, -f3 )
463  176                 curr_country=$( echo $a | cut -d, -f1 )
464  177
465  178
466  179                 #if the current incidence of this line is greater than the saved on
e, swap them
467  180                 if [[ "$curr_incidence" -gt "$highest_incidence" ]]
468  181                   then
469  182
470  183                                         highest_incidence=$curr_incidence
```

RUNTIME TESTING
Malaria Incidence
Test 1: malaria_incidence 2008

For the year 2008, the country with the highest incidence was Burkina Faso, with a rate of 533 per 1,000

Test 2: malaria_incidence Belize

For the country Belize, the year with the highest incidence was 2000, with a rate of 9 per 1,000

Test 3: malaria_incidence 'Timor-Leste'

For the country Timor-Leste, the year with the highest incidence was 2004, with a rate of 181 per 1,000

Test 4: malaria_incidence 'united republic of tanzania'

For the country United Republic of Tanzania, the year with the highest incidence was 2000, with a rate of 343 per 1,000

Test 5: malaria_incidence 'Venezuela'

For the country Venezuela (Bolivarian Republic of), the year with the highest incidence was 2018, with a rate of 33 per 1,000

Test 6: malaria_incidence 1999

invalid year (must be between 2000 - 2018 inclusive)

Common Words

kents06-22704037_13937_rerun

```
Test 7: common_words  -w little text_files

The most significant rank for the word little is 33 in file AliceInWonderland.txt


----
Test 8: common_words  -w cat text_files

The most significant rank for the word cat is 381 in file AliceInWonderland.txt


----
Test 9: common_words  -nth 4 text_files

The 4th most commmon word is "to" across 5 files


----
Test 10: common_words  -w Ozymandias text_files

Word Not found in any files of this directory


----
Test 11: common_words -nth 100000000 text_files

100000000th word is outside of the range for this directory


----
Test 12: common_words -w Alice test_files

A directory must be provided


----
END TO END TESTING


Execution time 48.8 seconds
```