20163079-ngj04_out

File ../files_to_test/20163079-ngj04.py differs from original due to handling of invisible characters

PROJECT CODE

```
##    Name:              Joshua Ng
##    Student Number:    20163079
##    version:           15/4/2017

import os.path

## 1. getCandidates(f) returns a list containing the candidates' names from the file f.
## The names will be one per line with no extraneous characters. Disregard any blank
## lines in f. If f doesn't exist, print an appropriate error message and return the empty list.
## For example, getCandidates("candidates.txt") = ["Major Clanger", "Soup Dragon",
##                                                 "Froglet", "Iron Chicken", "The Cloud"]
.
def getCandidates(f): #getCandidates(f): >> ["Major Clanger", "..."], >> print(error); []
    candidates = []
    if os.path.isfile(f):
        file_f = open(f, "r")
        for names in file_f:
            if names not in ['\n', '\r\n']:
                candidates.append(names.strip())
        file_f.close()
    else:
        print("file doesn't exist")
    return candidates

# 2. parseVote(s) returns the vote from s: parseVote("15") = parseVote(" 15 ") = 15.
# return 0 for an empty vote i.e. parseVote("") = parseVote(" ") = 0,
#-1 if there are any non-digits i.e. parseVote("-3") = parseVote("no") = parseVote("1 5") = -1,
def parseVote(s): #parseVote(s): " " >> 0, "no" >> -1, "9" >> 9
    vote = s.strip()
    if vote == "":
        return 0
    elif not vote.isdigit():
        return -1
    else:
        return int(s)


# 3. parsePaper(s, n) returns the votes from the ballot paper s in an
#    election with n candidates, plus an error message if appropriate.
# If s is formal, return the list of numbers found in s and the empty string.
#   i.e. parsePaper("14, , 2", 4) = ([14, 0, 2], "")
# If s is informal, return an empty list of numbers and the appropriate string below.
#   i.e. parsePaper(", , ", 4) = parsePaper("0, 0", 4) = ([ ], "blank"),
#   i.e. parsePaper("4, -8, 0", 4) = parsePaper("4, 7.8, 0", 4) = parsePaper("pointless,
#       5, 5", 4) = ([ ], "non-digits"),
#   i.e. parsePaper("1,2,,4,5", 4) = ([ ], "too long").
# parsePaper(s,n): >> ([14, 0, 2], ""), >> ([ ], "blank"), >> ([ ], "non-digits"), >> ([ ], "too long")
def parsePaper(s, n):
    ballot = s.split(",")
    len_ballot = len(ballot)
    votes = []
    blank_vote = 0

    if len_ballot > n:
        return ([], "too long")

    for vote in ballot:
        parse_vote = parseVote(vote) #parseVote(s): " " >> 0, "no" >> -1, "9" >> 9
        if parse_vote == -1:
            return ([], "non-digits")
        elif parse_vote == 0:
            blank_vote+=1
```

```
            votes.append(0)
        else:
            votes.append(parse_vote)

    if blank_vote == len_ballot:
        return ([], "blank")

    return (votes, "")


## 4. getPapers(f, n) returns a list containing the ballot papers from the file f, in an
election
##    with n candidates. Treat each line of the file as a separate paper. If f doesn't ex
ist,
##    print an appropriate error message and return the empty list. For example:
##    getPapers("smallfile.txt", 4) = [([1, 2, 3, 4], ""), ([], "blank"), ([0, 23, 0], ""
),
##                                    ([], "non-digits"), ([], "non-digits"), ([4,0,4,4],
 ""),
##                                    ([], "too long"), ([], "blank")].
def getPapers(f, n):
    papers = []

    if os.path.isfile(f):
        file_f = open(f, "r")
        for line in file_f:
            papers.append(parsePaper(line,n))
            #parsePaper(s,n): >> ([14, 0, 2], ""), >> ([ ], "blank"),
            #                 >> ([ ], "non-digits"), >> ([ ], "too long")
        file_f.close()
    else:
        print("file doesn't exist")
    return papers


## 5. normalisePaper(p, n) returns p with each vote scaled according to its total,
##    and padded to contain n votes.
##    For example:
##        normalisePaper([1,2,3,4], 4) = [0.1, 0.2, 0.3, 0.4],
##        normalisePaper([2], 3) = [1.0, 0.0, 0.0],
##        normalisePaper([0, 4, 496], 3) = [0.000, 0.008, 0.992]
def normalisePaper(p, n): # sum(p) > 0
    paper = []
    sum_p = sum(p)

    for vote in p:
        paper.append(float(vote)/sum_p)

    for i in range(len(paper), n):
        paper.append(0.0)
    return paper



## 6. normalisePapers(ps, n) returns ps with each paper normalised, in an election with n
 candidates.
##    e.g. normalisePapers([[2], [7, 2, 1]], 3) = [[1.0, 0.0, 0.0], [0.7, 0.2, 0.1]].
##    normalisePapers will use normalisePaper.
def normalisePapers(ps, n): # for every p on ps, sum(p) > 0
    papers = []
    for p in ps:
        papers.append(normalisePaper(p,n)) #normalisePaper(p,n): >> [0.1, 0.2, 0.3, 0.4]
    return papers


## 7. countVotes(cs, ps) returns a list of lists containing the counts for the candidates
 cs
##    from the ballot papers ps, in descending order of total number of votes.
##    For example, countVotes(["A", "B", "C"], [[0.5, 0.5, 0], [0.05, 0.3, 0.65]])
##                           = [[0.8, "B"], [0.65, "C"], [0.55, "A"]].
```

20163079-ngj04_out

```
def countVotes(cs, ps): # ps have been normalised for an election with len(cs) candidates
    results = []

    for candidate in range(len(cs)):
        sum = 0
        for vote in ps:
            sum += vote[candidate]
        results.append([sum, cs[candidate]])

    results.sort(reverse=True)
    return results


## 8. printCount(c) displays the election count c, i.e. the result from countVotes.
def printCount(c):
    for i in range(len(c)):
        print(" {0:0.2f} {1}".format(c[i][0], c[i][1]))


## 9. main() prompts the user for the names of the necessary files, then conducts the ele
ction.
##    main() will use getCandidates, getPapers, normalisePapers, countVotes, and printCou
nt.
def main():
    candidates_file = str(input("Please enter the candidates' filename: "))
    candidates = getCandidates(candidates_file)
    if candidates == []:
        return
    number_candidates = len(candidates)

    papers_file = str(input("Please enter the ballot papers' filename: "))
    papers = getPapers(papers_file, number_candidates)
    if papers == []:
        return

    formal_papers = []
    for p in papers:
        if not p[0] == []:
            formal_papers.append(p[0])

    normal_papers = normalisePapers(formal_papers, number_candidates)
    results = countVotes(candidates, normal_papers)

    number_formal = len(formal_papers)
    number_informal = len(papers) - number_formal

    print("Nerdvanian election 2017\n")
    print("There were {0} informal votes".format(number_informal))
    print("There were {0} formal votes\n".format(number_formal))
    printCount(results)
pass
```

UNIT TESTING

```
file doesn't exist
      getCandidates: all 2 test(s) correct
          parseVote: all 3 test(s) correct
         parsePaper: all 5 test(s) correct
file doesn't exist
           getPapers: all 2 test(s) correct
     normalisePaper: all 3 test(s) correct
    normalisePapers: all 1 test(s) correct
         countVotes: all 2 test(s) correct
```

20163079-ngj04_out

```
END TO END TESTING

Handling of Formal Votes

All good (no empty slots). candidate file: candidates.txt  papers file: papers0.txt
Please enter the candidates' filename: Please enter the ballot papers' filename: Nerdvani
an election 2017

There were 0 informal votes
There were 5 formal votes

 1.43 Iron Chicken
 1.22 Soup Dragon
 1.17 Major Clanger
 0.83 Froglet
 0.35 The Cloud

---------------------------
All good (with empty slots). candidate file: candidates.txt  papers file: papers1.txt
Please enter the candidates' filename: Please enter the ballot papers' filename: Nerdvani
an election 2017

There were 0 informal votes
There were 3 formal votes

 0.84 The Cloud
 0.75 Froglet
 0.56 Major Clanger
 0.55 Iron Chicken
 0.31 Soup Dragon

---------------------------
One candidate. Single vote. candidate file: candidates1.txt  papers file: papers2.txt
Please enter the candidates' filename: Please enter the ballot papers' filename: Nerdvani
an election 2017

There were 0 informal votes
There were 1 formal votes

 1.00 Major Clanger

---------------------------
One good, 1 short vote (padded with zeros) (5 candidates). candidate file: candidates.txt
  papers file: papers4.txt
Please enter the candidates' filename: Please enter the ballot papers' filename: Nerdvani
an election 2017

There were 0 informal votes
There were 2 formal votes

 1.14 Froglet
 0.28 Soup Dragon
 0.26 Major Clanger
 0.22 Iron Chicken
 0.10 The Cloud

---------------------------
One good vote, with white space (5 candidates). candidate file: candidates.txt  papers fi
le: papers7.txt
Please enter the candidates' filename: Please enter the ballot papers' filename: Nerdvani
an election 2017

There were 0 informal votes
There were 1 formal votes

 0.36 Soup Dragon
 0.33 Major Clanger
 0.18 Froglet
 0.12 The Cloud
```

20163079-ngj04_out

```
 0.00 Iron Chicken

---------------------------

Handling of Informal Votes

All zero votes (5 candidates). candidate file: candidates.txt  papers file: papers3.txt
Please enter the candidates' filename: Please enter the ballot papers' filename: Nerdvani
an election 2017

There were 1 informal votes
There were 0 formal votes

 0.00 The Cloud
 0.00 Soup Dragon
 0.00 Major Clanger
 0.00 Iron Chicken
 0.00 Froglet

---------------------------
One floating point vote (5 candidates). candidate file: candidates.txt  papers file: pape
rs5.txt
Please enter the candidates' filename: Please enter the ballot papers' filename: Nerdvani
an election 2017

There were 1 informal votes
There were 1 formal votes

 0.33 The Cloud
 0.27 Iron Chicken
 0.20 Froglet
 0.13 Soup Dragon
 0.07 Major Clanger

---------------------------
One negative vote (5 candidates). candidate file: candidates.txt  papers file: papers6.tx
t
Please enter the candidates' filename: Please enter the ballot papers' filename: Nerdvani
an election 2017

There were 1 informal votes
There were 1 formal votes

 0.33 The Cloud
 0.27 Iron Chicken
 0.20 Froglet
 0.13 Soup Dragon
 0.07 Major Clanger

---------------------------
Full Monty (5 candidates). candidate file: candidates.txt  papers file: papers.txt
Please enter the candidates' filename: Please enter the ballot papers' filename: Nerdvani
an election 2017

There were 44 informal votes
There were 250 formal votes

 61.99 Soup Dragon
 57.09 Froglet
 50.52 Major Clanger
 42.40 Iron Chicken
 37.99 The Cloud

---------------------------
```