wqydyrrtb

July 29, 2023

NAME: PAVITHIRAN Roll.no 235229122

Programming and Data Structures with Python Lab Lab4. Python File Processing

Question 1

Write a program for Password Management System File creation: Ask user to enter N user names and their passwords. Store usernames and passwords into a file named "loginfile.txt". Store each user and password in one line. File Processing: Write a program that opens your "security.txt" file and reads usernames and passwords from it. Store user names in one list and passwords in another lists. Querying: ask user to enter user name and password for verification. If they match the values stored in the lists, print a message "Login Successful". Otherwise print a message "Login Failed, try again".

[56]:
```python
user_list = list()
password_list = list()
with open('loginfile.txt', 'w+') as a, open('security.txt', 'w+') as s:
    while True:
        check = input("Enter n to stop the entry: ")
        if check == 'n':
            break
        user = input("Enter the user name: ")
        pw = input("Enter the password: ")
        a.write('{} {}\n'.format(user, pw))


with open('loginfile.txt', 'r') as a, open('security.txt','w') as s:
    for y in a.readlines():
        a, b = map(str, y.split())
        user_list.append(a)
        password_list.append(b)
    s.write(str(user_list))
    s.write('\n')
    s.write(str(password_list))


with open('security.txt', 'r') as a:
    name = input("Enter your user name:")
    password = input("Enter your password:")
    temname = a.readline()
    tempass = a.readline()
    if name in temname and  password in tempass:
        print("Login Successful")
    else:
        print("Login Failed, try again")
```

```
Enter n to stop the entry:
Enter the user name: pavi
Enter the password: 578
Enter n to stop the entry: n
Enter your user name:lkhgy
Enter your password:879
Login Failed, try again
```

Question 2

Write a program for Student Performance Analysis   Create a text file, 'marks.txt', with N marks as floating point numbers. Open the file, read marks from it and compute and print the highest mark.   If the user runs the program more than once you should not overwrite the previous text file – simply append the marks to the end of the file.   Modify the above program so that it also prints Top-3 highest marks (Note: you may need to use list concept)   Modify the above program so that it also prints the Lowest-3 marks.

[15]:
```
def mark_entry():
    with open("marks.txt", 'a') as g:
        while True:
            mark = float(input("Enter the mark: "))
            if mark < 0:
                break
            g.write(str(mark))
            g.write("\n")

def mark_reader():
    lis1 = list()
    with open("marks.txt", 'r') as f:
        for y in f.readlines():
            lis1.append(float(y))
        return lis1

def max_check(l1):
    l2 = sorted(l1, reverse = True)
    return l2[0:3]

def min_check(l1):
    l2 = sorted(l1)
    return l2[0:3]

def printer(lst):
    for i in lst:
        if i == lst[-1]:
            print(i, end=".")
        else:
            print(i, end=" ")

def main():
    mark_entry()
    mark_list = mark_reader()
    print("The top 3 highest marks are: ", end = " ")
    printer(max_check(mark_list))
    print()
    print("The top 3 lowest marks are: ", end = " ")
    printer(min_check(mark_list))

if __name__ == "__main__":
    main()
```

```
Enter the mark:  98
Enter the mark:  97.89
Enter the mark:  94.90
Enter the mark:  12.99
Enter the mark:  24.6
Enter the mark:  76.56
Enter the mark:  -1

The top 3 highest marks are:  98.0 98.0 97.89.
The top 3 lowest marks are:  12.99 23.0 24.6.
```

Question 3

Write a program for Stock Price Analysis   File Creation: Continually prompt a user for stock name, followed by price values for 5 days. Each row indicates stock name and daily prices of one stock. Store these values in a text file called "stock-prices.txt". Open the file in Append Mode. Prompt message "Do you want to continue? " and stop reading values accordingly. Then, you can close your file.   File Processing: Now, open your file for processing. Print stock name, minimum price, maximum price and average price values.   You can also print which day stock price was lowest in the week and which day stock price was highest. So, modify your print statement to print stock name, minimum price & day of minimum price, maximum price & day of maximum price and average price values. (Hint: Use enumerate to get index values)

[24]:
```
def stk_priz_write():
    with open("stock-prices.txt", 'a') as stk:
        while True:
```

```python
            stk_name = input("Enter the stock name(or 'quit' to stop) :")
            if stk_name.lower() == 'quit':
                break
            stk_price = list()
            for day in range(1, 6):
                price = float(input("Enter the {} day price: ".format(day)))
                stk_price.append(price)
            stk.write("{},{}\n".format(stk_name,','.join(map(str, stk_price))))

def stk_priz_read():
    stk_data = {}
    with open("stock-prices.txt", 'r') as stk:
        for s in stk:
            stk_info = s.strip().split(',')
            stk_name = stk_info[0]
            prices = list(map(float, stk_info[1:]))
            stk_data[stk_name] = prices
    return stk_data

def analizing_stk_prz(stocks_data):
    for stock_name, prices in stocks_data.items():
        min_price = min(prices)
        max_price = max(prices)
        avg_price = sum(prices) / len(prices)

        min_day = prices.index(min_price) + 1
        max_day = prices.index(max_price) + 1

        print(f"Stock Name: {stock_name}")
        print(f"Minimum Price: {min_price} (Day {min_day})")
        print(f"Maximum Price: {max_price} (Day {max_day})")
        print(f"Average Price: {avg_price:.2f}")
        print()

stk_priz_write()
stk_data = stk_priz_read()
analizing_stk_prz(stk_data)
```

```
Enter the stock name(or 'quit' to stop) : kumar
Enter the 1 day price:  2
Enter the 2 day price:  3
Enter the 3 day price:  3
Enter the 4 day price:  4
Enter the 5 day price:  53
Enter the stock name(or 'quit' to stop) : raj
Enter the 1 day price:  3
Enter the 2 day price:  3
Enter the 3 day price:  4
Enter the 4 day price:  53
Enter the 5 day price:  5
Enter the stock name(or 'quit' to stop) : yogi
Enter the 1 day price:  3
Enter the 2 day price:  4
Enter the 3 day price:  43
Enter the 4 day price:  4
Enter the 5 day price:  34
Enter the stock name(or 'quit' to stop) : joker
Enter the 1 day price:  3
Enter the 2 day price:  4
Enter the 3 day price:  5
Enter the 4 day price:  645
Enter the 5 day price:  4
Enter the stock name(or 'quit' to stop) : quit

Stock Name: kumar
Minimum Price: 2.0 (Day 1)
Maximum Price: 53.0 (Day 5)
Average Price: 13.00

Stock Name: raj
Minimum Price: 3.0 (Day 1)
Maximum Price: 53.0 (Day 4)
Average Price: 13.60

Stock Name: yogi
Minimum Price: 3.0 (Day 1)
Maximum Price: 43.0 (Day 3)
Average Price: 17.60

Stock Name: joker
Minimum Price: 3.0 (Day 1)
Maximum Price: 645.0 (Day 4)
Average Price: 132.20
```

Question 4

Write a program for File Explorer Display the contents of file Count the number of lines in a text file. (Use splitlines()) Count the number of unique words in a file. Find frequency of words in a given file. (Hint: Use Counter object) Show a random line in a file. (Use Random object)

[31]:
```
import random
from collections import Counter
```

```python
def display_file_contents(file_path):
    with open(file_path, 'r') as file:
        contents = file.read()
        print("File Contents:")
        print(contents)


def count_lines(file_path):
    with open(file_path, 'r') as file:
        lines = file.read().splitlines()
        num_lines = len(lines)
        print(f"Number of lines in the file: {num_lines}")


def count_unique_words(file_path):
    with open(file_path, 'r') as file:
        words = file.read().split()
        num_unique_words = len(set(words))
        print(f"Number of unique words in the file: {num_unique_words}")


def word_frequency(file_path):
    with open(file_path, 'r') as file:
        words = file.read().split()
        word_counter = Counter(words)
        print("Word Frequency:")
        for word, frequency in word_counter.items():
            print(f"{word}: {frequency}")


def show_random_line(file_path):
    with open(file_path, 'r') as file:
        lines = file.readlines()
        random_line = random.choice(lines)
        print("Random Line:")
        print(random_line.strip())


def main():
    file_path = "sample_file.txt"
    display_file_contents(file_path)
    print()
    count_lines(file_path)
    print()
    count_unique_words(file_path)
    print()
    word_frequency(file_path)
    print()
    show_random_line(file_path)
    print()


if __name__ == "__main__":
    main()
```

```
File Contents:
The European rock pipit (Anthus petrosus) is a small species of songbird that
breeds in western Europe on rocky coasts.
It has streaked greyish-brown upperparts and buff underparts, and is similar in
appearance to other European pipits.
There are three subspecies, of which only the Fennoscandian form is migratory,
wintering in shoreline habitats further south in Europe and northwest Africa.
The rock pipit is territorial at least in the breeding season, and remains so
year-round where it is resident.

 Males will sometimes enter an adjacent territory to assist the resident in
repelling an intruder, behaviour only otherwise known from an African fiddler
crab.
 Rock pipits construct a cup nest under coastal vegetation or in cliff crevices
and lay four to six speckled pale grey eggs which hatch in about two weeks.
The pipits feed mainly on small invertebrates picked off the rocks or from
shallow water, and occasionally catch insects in flight.
The bird's population is large and stable.

Number of lines in the file: 8

Number of unique words in the file: 117

Word Frequency:
The: 4
European: 2
rock: 2
pipit: 2
(Anthus: 1
petrosus): 1
is: 6
a: 2
small: 2
species: 1
of: 2
songbird: 1
that: 1
breeds: 1
in: 9
```

```
western: 1
Europe: 2
on: 2
rocky: 1
coasts.: 1
It: 1
has: 1
streaked: 1
greyish-brown: 1
upperparts: 1
and: 7
buff: 1
underparts,: 1
similar: 1
appearance: 1
to: 3
other: 1
pipits.: 1
There: 1
are: 1
three: 1
subspecies,: 1
which: 2
only: 2
the: 4
Fennoscandian: 1
form: 1
migratory,: 1
wintering: 1
shoreline: 1
habitats: 1
further: 1
south: 1
northwest: 1
Africa.: 1
territorial: 1
at: 1
least: 1
breeding: 1
season,: 1
remains: 1
so: 1
year-round: 1
where: 1
it: 1
resident.: 1
Males: 1
will: 1
```

```
sometimes: 1
enter: 1
an: 3
adjacent: 1
territory: 1
assist: 1
resident: 1
repelling: 1
intruder,: 1
behaviour: 1
otherwise: 1
known: 1
from: 2
African: 1
fiddler: 1
crab.: 1
Rock: 1
pipits: 2
construct: 1
cup: 1
nest: 1
under: 1
coastal: 1
vegetation: 1
or: 2
cliff: 1
crevices: 1
lay: 1
four: 1
six: 1
speckled: 1
pale: 1
grey: 1
eggs: 1
hatch: 1
about: 1
two: 1
weeks.: 1
feed: 1
mainly: 1
invertebrates: 1
picked: 1
off: 1
rocks: 1
shallow: 1
water,: 1
occasionally: 1
catch: 1
```

```
insects: 1
flight.: 1
bird's: 1
population: 1
large: 1
stable.: 1

Random Line:
The European rock pipit (Anthus petrosus) is a small species of songbird that
breeds in western Europe on rocky coasts.
```

Question 5

[File Searcher]. Develop an application in Python to read through the email data ("mbox-short.txt") and when you find line that starts with "From", you will split the line into words using the split function. We are interested in who sent the message, which is the second word on the From line: From stephen.marquard@uct.ac.za Sat Jan 5 09:14:16 2008. You will parse the From line and print out the second word for each From line, then you will also count the number of From (not From:) lines and print out a count at the end

[1]:
```python
with open("mbox-short.txt", 'r') as file:
    count_from_lines = 0

    for line in file:
        line = line.strip()
        if line.startswith("From "):
            words = line.split()
            sender_email = words[1]
            print(sender_email)
            count_from_lines += 1

    print(f"Total number of 'From' lines: {count_from_lines}")
```

```
stephen.marquard@uct.ac.za
louis@media.berkeley.edu
zqian@umich.edu
rjlowe@iupui.edu
zqian@umich.edu
rjlowe@iupui.edu
cwen@iupui.edu
cwen@iupui.edu
gsilver@umich.edu
gsilver@umich.edu
zqian@umich.edu
gsilver@umich.edu
wagnermr@iupui.edu
```

```
zqian@umich.edu
antranig@caret.cam.ac.uk
gopal.ramasammycook@gmail.com
david.horwitz@uct.ac.za
david.horwitz@uct.ac.za
david.horwitz@uct.ac.za
david.horwitz@uct.ac.za
stephen.marquard@uct.ac.za
louis@media.berkeley.edu
louis@media.berkeley.edu
ray@media.berkeley.edu
cwen@iupui.edu
cwen@iupui.edu
cwen@iupui.edu
Total number of 'From' lines: 27
```

Question 6

Write a program to read and write CSV files File Creation: Create MS Excel file ("student_marks.csv") with 5 rows of student name, mark1, mark2, mark3, mark4. Use comma to separate each value in a row. File Display: Now, open your CSV file and display the file contents row by row (More information at: https://docs.python.org/3/library/csv.html). File Writing: Now, open ("student_marks.csv") for writing. Ask user to enter name followed by 4 marks for one new student and write them onto the file.

[3]:
```python
import csv

def create_initial_csv_file(file_path):
    # Sample data for the initial CSV file
    data = [
        ['Student Name', 'Mark1', 'Mark2', 'Mark3', 'Mark4'],
        ['John Doe', 90, 85, 78, 92],
        ['Jane Smith', 75, 88, 91, 79],
        ['Michael Johnson', 82, 79, 87, 95],
        ['Emily Brown', 95, 90, 88, 93],
        ['William Lee', 88, 86, 92, 78]
    ]

    with open(file_path, 'w', newline='') as csvfile:
        writer = csv.writer(csvfile)
        writer.writerows(data)

def display_csv_file(file_path):
    with open(file_path, 'r', newline='') as csvfile:
        reader = csv.reader(csvfile)
        for row in reader:
            print(', '.join(row))
```

```python
def write_new_student_data(file_path):
    new_student_data = []
    new_student_data.append(input("Enter the student's name: "))
    for i in range(4):
        mark = input(f"Enter Mark {i+1}: ")
        new_student_data.append(mark)

    with open(file_path, 'a', newline='') as csvfile:
        writer = csv.writer(csvfile)
        writer.writerow(new_student_data)

if __name__ == "__main__":
    file_path = "student_marks.csv"
    create_initial_csv_file(file_path)
    print("CSV File Contents:")
    display_csv_file(file_path)
    write_new_student_data(file_path)
    print("\nUpdated CSV File Contents:")
    display_csv_file(file_path)
```

```
CSV File Contents:
Student Name, Mark1, Mark2, Mark3, Mark4
John Doe, 90, 85, 78, 92
Jane Smith, 75, 88, 91, 79
Michael Johnson, 82, 79, 87, 95
Emily Brown, 95, 90, 88, 93
William Lee, 88, 86, 92, 78

Enter the student's name:  kumar
Enter Mark 1:  90
Enter Mark 2:  80
Enter Mark 3:  99
Enter Mark 4:  87


Updated CSV File Contents:
Student Name, Mark1, Mark2, Mark3, Mark4
John Doe, 90, 85, 78, 92
Jane Smith, 75, 88, 91, 79
Michael Johnson, 82, 79, 87, 95
Emily Brown, 95, 90, 88, 93
William Lee, 88, 86, 92, 78
kumar, 90, 80, 99, 87
```