Міністерство освіти і науки України

Державний університет “Житомирська політехніка”

Кафедра інженерії програмного забезпечення

Група: ВТ-21-1[1]

Програмування мовою Python

Лабораторна робота № 6

«Робота з файлами»

Виконав: Бабушко А. С.

Прийняв: Морозов Д. С.

***Мета роботи:*** ознайомитися з засобами роботи з файлами в мові Python, діями над ними.

***Хід роботи:***

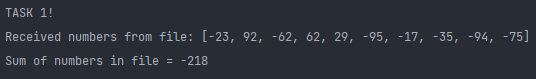
***Завдання на лабораторну роботу:***

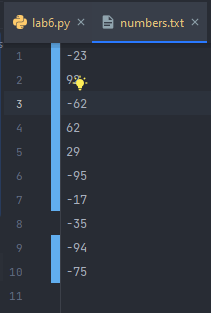
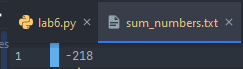
Завдання 1. Створіть новий файл numbers.txt у текстовому редакторі і запишіть у нього 10 чисел, кожне з нового рядка. Напишіть програму, яка зчитує ці числа з файлу і обчислює їх суму, виводить цю суму на екран і, водночас, записує цю суму у інший файл з назвою sum\_numbers.txt.

***Лістинг програми:***

*""" Lab 6. Python. Andrii Babushko. Repository: https://github.com/AndriiBabushko/Python """  
import* csv  
*import* os  
*import* io  
*from* datetime *import* datetime  
*from* datetime *import* date  
*from* time *import* perf\_counter  
  
*# task 1*"""   
 Завдання 1. Створіть новий файл numbers.txt у текстовому редакторі і запишіть у нього 10 чисел, кожне з нового   
 рядка. Напишіть програму, яка зчитує ці числа з файлу і обчислює їх суму, виводить цю суму на екран і, водночас,   
 записує цю суму у інший файл з назвою sum\_numbers.txt.  
"""  
  
  
*def* task\_1():  
 print('\nTASK 1!')  
 *if not* os.path.isdir(r'./task1'):  
 os.mkdir(r'./task1')  
  
 *with* io.open(r'./task1/numbers.txt', 'wt', encoding='utf-8') *as* numbers\_txt:  
 numbers\_txt.write(create\_numbers\_string())  
  
 *with* io.open(r'./task1/numbers.txt', 'rt', encoding='utf-8') *as* numbers\_txt:  
 task\_1\_sum = 0  
 task\_1\_number\_list = []  
  
 *for* number *in* numbers\_txt:  
 *try*:  
 task\_1\_sum += int(number)  
 task\_1\_number\_list.append(int(number))  
 *except* ValueError *as* err:  
 print(f'ERROR! {ValueError}')  
  
 *with* io.open(r'./task1/sum\_numbers.txt', 'wt', encoding='utf-8') *as* sum\_numbers\_txt:  
 sum\_numbers\_txt.write(str(task\_1\_sum))  
  
 print(f'Received numbers from file: {task\_1\_number\_list}')  
 print(f'Sum of numbers in file = {task\_1\_sum}')  
  
  
*def* create\_numbers\_string():  
 *from* random *import* randint  
 new\_string = ''  
  
 *for* new\_number *in* range(10):  
 new\_string += f'{randint(-100, 100)}\n'  
  
 *return* new\_string  
  
  
task\_1()

***Результат програми:***

******

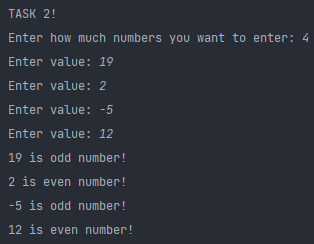
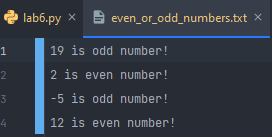
*** ***

Завдання 2. Реалізуйте програму, яка зчитує довільну кількість цілих чисел, що вводяться з командного рядка, і записує у текстовий файл інформацію, щодо парності або непарності чисел.

***Лістинг програми:***

*# task 2*"""  
 Завдання 2. Реалізуйте програму, яка зчитує довільну кількість цілих чисел, що вводяться з командного рядка, і   
 записує у текстовий файл інформацію, щодо парності або непарності чисел.  
"""  
  
  
*def* task\_2():  
 print('\nTASK 2!')  
 task\_2\_enter\_count = int(input('Enter how much numbers you want to enter: '))  
 task\_2\_number\_list = task\_2\_enter\_some\_numbers(task\_2\_enter\_count)  
  
 *if not* os.path.isdir(r'./task2'):  
 os.mkdir(r'./task2')  
  
 *with* io.open(r'./task2/even\_or\_odd\_numbers.txt', 'wt', encoding='utf-8') *as* even\_or\_odd\_numbers\_txt:  
 *for* number *in* task\_2\_number\_list:  
 *if* number % 2 == 0:  
 even\_or\_odd\_numbers\_txt.write(f'{number} is even number!\n')  
 *else*:  
 even\_or\_odd\_numbers\_txt.write(f'{number} is odd number!\n')  
  
 *with* io.open(r'./task2/even\_or\_odd\_numbers.txt', 'rt', encoding='utf-8') *as* even\_or\_odd\_numbers\_txt:  
 *for* line *in* even\_or\_odd\_numbers\_txt:  
 print(line, end="")  
  
  
*def* task\_2\_enter\_some\_numbers(count):  
 numbers\_list = []  
  
 *while* count != 0:  
 *try*:  
 int\_number = int(input('Enter value: '))  
 numbers\_list.append(int\_number)  
 count -= 1  
 *except* ValueError *as* value\_error:  
 numbers\_list.pop()  
 print('ERROR:', value\_error)  
  
 *return* numbers\_list  
  
  
task\_2()

***Результат програми:***

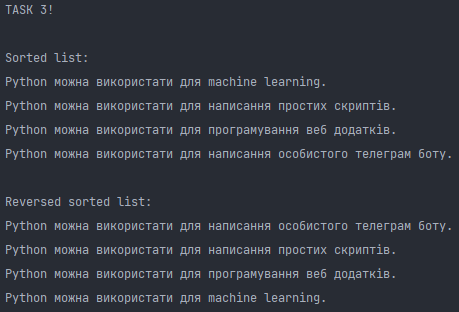
*** ***

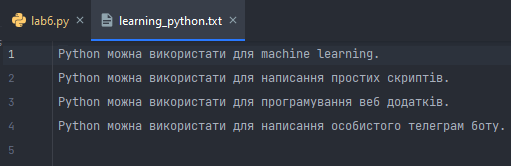
Завдання 3. Створіть новий файл у текстовому редакторі і напишіть кілька рядків тексту у ньому про можливості Python. Кожен рядок повинен починатися з фрази: «Python можна використати для ...» . Збережіть файл з ім’ям learning\_python.txt. Напишіть програму, яка зчитує файл і виводить текст з перебором рядків файла і зі збереженням рядків у списку з подальшим сортуванням списку за довжиною рядків в ньому від найбільшого до найменшого.

***Лістинг програми:***

*# task 3*"""  
 Завдання 3. Створіть новий файл у текстовому редакторі і напишіть кілька рядків тексту у ньому про можливості   
 Python. Кожен рядок повинен починатися з фрази: «Python можна використати для ...» . Збережіть файл з ім’ям   
 learning\_python.txt. Напишіть програму, яка зчитує файл і виводить текст з перебором рядків файла і зі збереженням   
 рядків у списку з подальшим сортуванням списку за довжиною рядків в ньому від найбільшого до найменшого.  
"""  
  
  
*def* task\_3():  
 print('\nTASK 3!')  
  
 *if not* os.path.isdir(r'./task3'):  
 os.mkdir('./task3')  
  
 *with* io.open(r'./task3/learning\_python.txt', 'wt', encoding='utf-8') *as* learning\_python\_txt:  
 learning\_python\_txt.writelines(  
 ["Python можна використати для machine learning.\n",  
 "Python можна використати для написання простих скриптів.\n",  
 "Python можна використати для програмування веб додатків.\n",  
 "Python можна використати для написання особистого телеграм боту.\n"])  
  
 *with* io.open(r'./task3/learning\_python.txt', 'rt', encoding='utf-8') *as* learning\_python\_txt:  
 python\_string\_list = []  
  
 *for* line *in* learning\_python\_txt:  
 python\_string\_list.append(line)  
  
 python\_string\_sorted\_list = python\_string\_list[:]  
 python\_string\_sorted\_list.sort(key=len)  
 print('\nSorted list:')  
 *for* string *in* python\_string\_sorted\_list:  
 print(string, end="")  
  
 python\_string\_reversed\_sorted\_list = python\_string\_list[:]  
 python\_string\_reversed\_sorted\_list.sort(key=len, reverse=*True*)  
 print('\nReversed sorted list:')  
 *for* string *in* python\_string\_reversed\_sorted\_list:  
 print(string, end="")  
  
  
task\_3()

***Результат програми:***

******

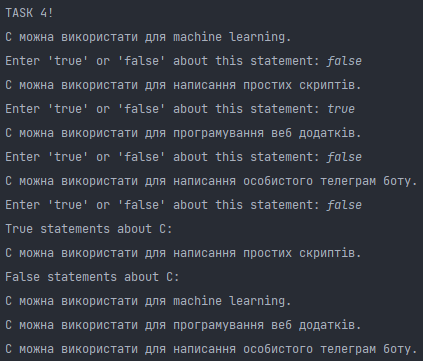
******

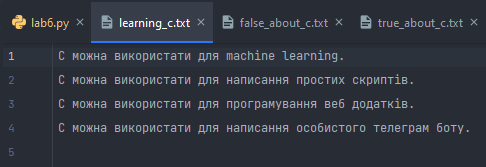
Завдання 4. Прочитайте кожен рядок зі створеного у попередньому завданні файла learning\_python.txt і замініть слово Python назвою іншої мови, наприклад C при виведенні на екран. Отриманий файл має бути створений в новому каталозі, що розміщується в поточному. Відкрийте файл пострічково і дайте можливість користувачеві визначити які змінені фрази є актуальними, наприклад для мови С, а які ні. Всі хибні твердження запишіть в інший файл, а істинні – в поточний.

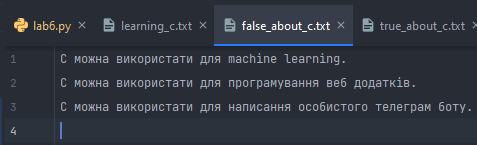
***Лістинг програми:***

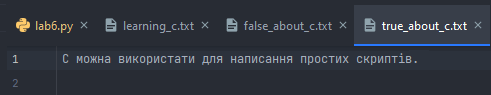
*# task 4*"""  
 Завдання 4. Прочитайте кожен рядок зі створеного у попередньому завданні файла learning\_python.txt і замініть   
 слово Python назвою іншої мови, наприклад C при виведенні на екран. Отриманий файл має бути створений в новому   
 каталозі, що розміщується в поточному. Відкрийте файл пострічково і дайте можливість користувачеві визначити які   
 змінені фрази є актуальними, наприклад для мови С, а які ні. Всі хибні твердження запишіть в інший файл, а істинні   
 – в поточний.  
"""  
  
  
*def* task\_4():  
 print('\nTASK 4!')  
  
 *if not* os.path.isdir(r'./task4'):  
 os.mkdir(r'./task4')  
  
 *with* io.open(r'./task3/learning\_python.txt', 'rt', encoding='utf-8') *as* learning\_python\_txt:  
 python\_strings = []  
  
 *for* string *in* learning\_python\_txt:  
 python\_strings.append(string.split(' '))  
  
 *for* i *in* range(0, len(python\_strings)):  
 python\_strings[i][0] = 'C'  
 python\_strings[i] = " ".join(python\_strings[i])  
  
 *with* io.open(r'./task4/learning\_c.txt', 'wt', encoding='utf-8') *as* true\_about\_c\_txt:  
 *for* c\_string *in* python\_strings:  
 true\_about\_c\_txt.write(c\_string)  
  
 *with* io.open(r'./task4/learning\_c.txt', 'rt', encoding='utf-8') *as* learning\_c\_txt:  
 *for* c\_string *in* learning\_c\_txt:  
 print(c\_string, end='')  
 answer = enter\_true\_or\_false()  
 *if* answer:  
 *with* io.open(r'./task4/true\_about\_c.txt', 'at', encoding='utf-8') *as* true\_about\_c\_txt:  
 true\_about\_c\_txt.write(c\_string)  
 *else*:  
 *with* io.open(r'./task4/false\_about\_c.txt', 'at', encoding='utf-8') *as* false\_about\_c\_txt:  
 false\_about\_c\_txt.write(c\_string)  
  
 *with* io.open(r'./task4/true\_about\_c.txt', 'rt', encoding='utf-8') *as* true\_about\_c\_txt\_read:  
 *for* c\_string *in* true\_about\_c\_txt\_read:  
 print(c\_string, end='')  
  
 *with* io.open(r'./task4/false\_about\_c.txt', 'rt', encoding='utf-8') *as* false\_about\_c\_txt\_read:  
 *for* c\_string *in* false\_about\_c\_txt\_read:  
 print(c\_string, end='')  
  
  
*def* enter\_true\_or\_false():  
 *while True*:  
 *try*:  
 answer = input("Enter 'true' or 'false' about this statement: ")  
 *if* answer != 'true' *and* answer != 'false':  
 *raise* ValueError('Value was entered incorrectly!')  
 *else*:  
 *pass  
 if* answer == 'true':  
 *return True  
 elif* answer == 'false':  
 *return False  
 except* ValueError *as* value\_err:  
 print(f'ERROR! {value\_err}')  
  
  
task\_4()

***Результат програми:***

******

******

******

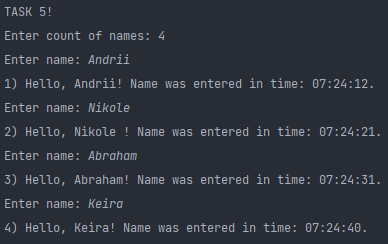
******

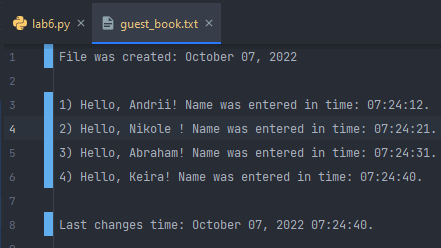
Завдання 5. Створіть порожній файл guest\_book.txt у текстовому редакторі. Напишіть програму, яка запитує у користувачів імена. При введенні кожного імені виведіть на екран рядок з вітанням для користувача і запишіть рядок вітання у файл з ім’ям guest\_book.txt. Простежте за тим, щоб кожне повідомлення розміщувалося в окремому рядку файла з зазначенням часу внесення цього повідомлення. Передбачте зазначення в файлі часу його створення і вказання в ньому часу останніх внесених змін

***Лістинг програми:***

*# task 5*"""  
 Завдання 5. Створіть порожній файл guest\_book.txt у текстовому редакторі. Напишіть програму, яка запитує у   
 користувачів імена. При введенні кожного імені виведіть на екран рядок з вітанням для користувача і запишіть   
 рядок вітання у файл з ім’ям guest\_book.txt. Простежте за тим, щоб кожне повідомлення розміщувалося в окремому   
 рядку файла з зазначенням часу внесення цього повідомлення. Передбачте зазначення в файлі часу його створення і   
 вказання в ньому часу останніх внесених змін  
"""  
  
  
*def* task\_5():  
 print('\nTASK 5!')  
  
 *if not* os.path.isdir(r'./task5'):  
 os.mkdir(r'./task5')  
  
 task\_5\_count = enter\_count()  
 file\_create\_time = date.today().strftime('%B %d, %Y')  
  
 *with* io.open(r'./task5/guest\_book.txt', 'wt', encoding='utf-8') *as* guest\_book\_txt:  
 guest\_book\_txt.write(f'File was created: {file\_create\_time}\n\n')  
  
 *for* counter *in* range(0, task\_5\_count):  
 name\_greeting = f'{counter + 1}) Hello, ' + enter\_name() + '! '  
 message\_time = datetime.now().strftime('%H:%M:%S')  
 name\_greeting += f'Name was entered in time: {message\_time}.\n'  
 print(name\_greeting, end="")  
 guest\_book\_txt.write(name\_greeting)  
  
 last\_changes\_date = date.today().strftime('%B %d, %Y')  
 last\_changes\_time = datetime.now().strftime('%H:%M:%S')  
 guest\_book\_txt.write(f'\nLast changes time: {last\_changes\_date} {last\_changes\_time}.\n')  
  
  
*def* enter\_name():  
 *while True*:  
 *try*:  
 name = input('Enter name: ')  
 *if* len(name) < 2:  
 *raise* ValueError('Name\'s length is less than 2!')  
 *else*:  
 *pass  
 return* name  
 *except* ValueError *as* value\_err:  
 print(f'ERROR! {value\_err}')  
  
  
*def* enter\_count():  
 *while True*:  
 *try*:  
 count = int(input('Enter count of names: '))  
 *if* count <= 0:  
 *raise* ValueError('Count names was entered incorrectly!')  
 *else*:  
 *pass  
 return* count  
 *except* ValueError *as* value\_err:  
 print(f'ERROR! {value\_err}')  
  
  
task\_5()

***Результат програми:***

******

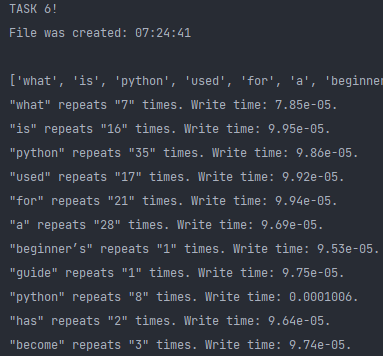
******

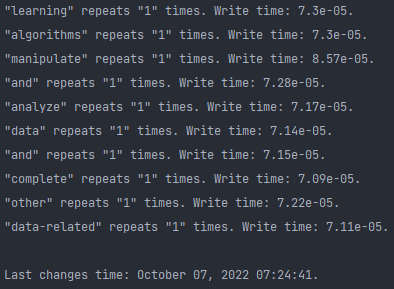
Завдання 6. Збережіть в тектовому файлі публікацію про Python на 3000 слів англійською мовою. Напишіть програму, що аналізуватиме частоту з якою в тексті зустрічастимуться окремі літери чи слова незалежно від їх регістру. Результат робот програми має виводитись в консоль і зберігатись в окремому файлі з зазначенням часу його стоврення, часу виконнання окремих змін, результатів пошкуку і часу, що знадовся на виконнання цього пошуку.

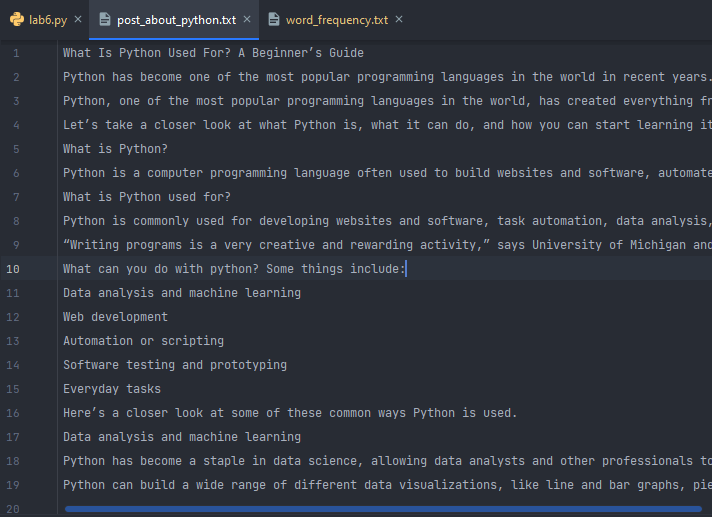
***Лістинг програми:***

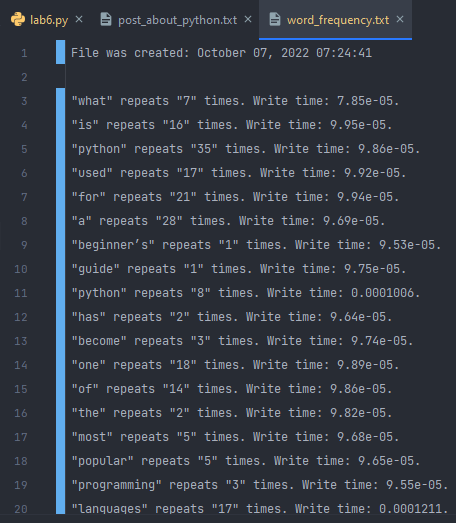
*# task 6*"""  
 Завдання 6. Збережіть в текстовому файлі публікацію про Python на 3000 слів англійською мовою. Напишіть програму,   
 що аналізуватиме частоту з якою в тексті зустрічатимуться окремі літери чи слова незалежно від їх регістру.   
 Результат робот програми має виводитись в консоль і зберігатись в окремому файлі з зазначенням часу його створення,   
 часу виконання окремих змін, результатів пошуку і часу, що знадобився на виконання цього пошуку.  
"""  
  
  
*def* task\_6():  
 *import* re  
 print('\nTASK 6!')  
  
 *with* io.open(r'./task6/post\_about\_python.txt', 'rt', encoding='utf-8') *as* post\_about\_python\_txt:  
 words = re.split(r' |\n|\? |\.|\, |\: |\;', post\_about\_python\_txt.read().lower())  
  
 file\_create\_date = date.today().strftime('%B %d, %Y')  
 file\_create\_time = datetime.now().strftime('%H:%M:%S')  
 *with* io.open(r'./task6/word\_frequency.txt', 'wt', encoding='utf-8') *as* word\_frequency\_txt:  
 word\_frequency\_txt.write(f'File was created: {file\_create\_date} {file\_create\_time}\n\n')  
 print(f'File was created: {file\_create\_time}\n')  
  
 words\_without\_duplicates = []  
 *for* string *in* words:  
 *if* string *not in* words\_without\_duplicates:  
 words\_without\_duplicates.append(string)  
  
 print(words\_without\_duplicates)  
 *for* i *in* range(0, len(words\_without\_duplicates)):  
 time\_started = perf\_counter()  
 counter = 0  
 *for* j *in* range(0, len(words)):  
 *if* words\_without\_duplicates[i] == words[j]:  
 counter += 1  
 time\_finished = perf\_counter()  
 write\_time = round(time\_finished - time\_started, 8)  
 word\_frequency\_txt.write(f'"{words[i]}" repeats "{counter}" times. Write time: {write\_time}.\n')  
 print(f'"{words[i]}" repeats "{counter}" times. Write time: {write\_time}.')  
  
 last\_changes\_date = date.today().strftime('%B %d, %Y')  
 last\_changes\_time = datetime.now().strftime('%H:%M:%S')  
 word\_frequency\_txt.write(f'\nLast changes time: {last\_changes\_date} {last\_changes\_time}.\n')  
 print(f'\nLast changes time: {last\_changes\_date} {last\_changes\_time}.')  
  
  
task\_6()

***Результат програми:***

******

******

******

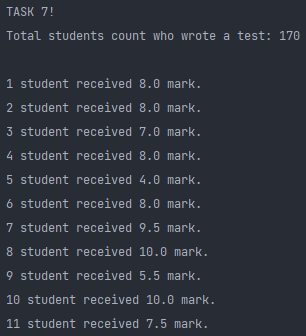
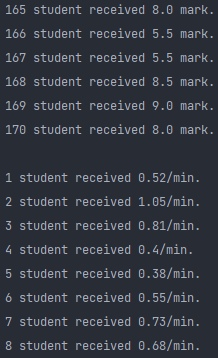
******

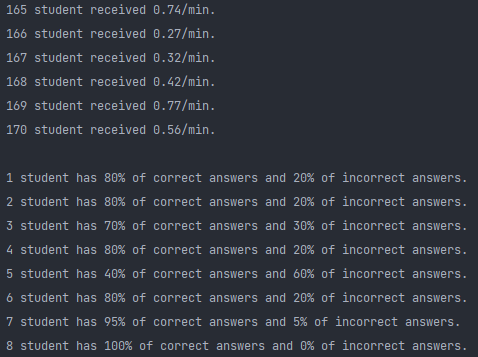
Завдання 7. Завантажте файл marks.csv і визначте кількість студентів, що проходили тестування. Виведіть інформацію про те яку оцінку набрали відповідна кільксть студентів. Виведіть інформацію яку середню оцінку отримував студент за певний час виконання КМР (крок – 1 хв). Створіть текстовий файл і запишіть в нього статистику по правильним відповідям для кожного окремого питання (який відсоток правильних і неправильних відповідей на питання дали студенти). В цей же файл внесіть інформацію про 5 найкращих оцінок в співвідношенні оцінка/час витрачений складання КМР.

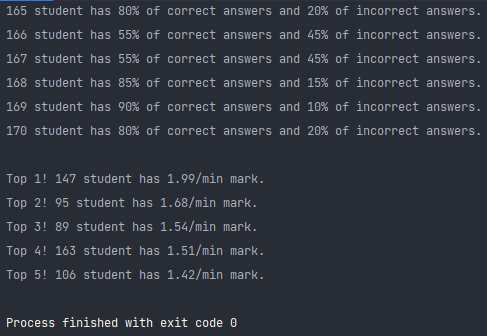
***Лістинг програми:***

*# task 7*"""  
 Завдання 7. Завантажте файл marks.csv і визначте кількість студентів, що проходили тестування. Виведіть інформацію   
 про те яку оцінку набрали відповідна кільксть студентів. Виведіть інформацію яку середню оцінку отримував студент   
 за певний час виконання КМР (крок – 1 хв). Створіть текстовий файл і запишіть в нього статистику по правильним   
 відповідям для кожного окремого питання (який відсоток правильних і неправильних відповідей на питання дали   
 студенти). В цей же файл внесіть інформацію про 5 найкращих оцінок в співвідношенні оцінка/час витрачений   
 складання КМР.  
"""  
  
  
*def* task\_7():  
 print('\nTASK 7!')  
  
 *if not* os.path.isdir(r'./task7'):  
 os.mkdir(r'./task7')  
  
 *with* io.open(r'./task7/marks.lab6.csv', 'rt', encoding='utf-8') *as* marks\_lab6\_csv:  
 marks\_csv = csv.reader(marks\_lab6\_csv)  
 marks = [line *for* line *in* marks\_csv]  
  
 students\_count = len(marks)  
 print(f'Total students count who wrote a test: {students\_count}')  
  
 students\_marks = []  
 *for* i *in* range(students\_count):  
 mark\_string = marks[i][4].split(',')  
 mark\_number = float(mark\_string[0]) + float(int(mark\_string[1]) / 100)  
 students\_marks.append(mark\_number)  
  
 print()  
 iterator = 1  
 *for* mark *in* students\_marks:  
 print(f'{iterator} student received {mark} mark.')  
 iterator += 1  
  
 students\_time = []  
 *for* i *in* range(students\_count):  
 time = marks[i][3].split(' ')  
 *if* len(time) > 2:  
 minutes = int(time[0])  
 seconds = int(time[2]) + minutes \* 60  
 *else*:  
 minutes = int(time[0])  
 seconds = minutes \* 60  
 students\_time.append(seconds)  
  
 students\_mark\_per\_min = []  
 *for* i *in* range(students\_count):  
 students\_mark\_per\_min.append(round((students\_marks[i] / students\_time[i]) \* 60, 2))  
  
 print()  
 iterator = 1  
 *for* average\_mark *in* students\_mark\_per\_min:  
 print(f'{iterator} student received {average\_mark}/min.')  
 iterator += 1  
  
 file\_create\_date = date.today().strftime('%B %d, %Y')  
 file\_create\_time = datetime.now().strftime('%H:%M:%S')  
 *with* io.open(r'./task7/statistics.txt', 'wt', encoding='utf-8') *as* statistics\_txt:  
 statistics\_txt.write(f'File was created: {file\_create\_date} {file\_create\_time}\n\n')  
  
 student\_answers = []  
 *for* answers *in* marks:  
 student\_answers.append(answers[5:])  
  
 print()  
 iterator = 1  
 *for* answer *in* student\_answers:  
 correct\_answers = 0  
 incorrect\_answers = 0  
  
 *for* i *in* range(0, len(answer)):  
 *if* answer[i] == '0,50':  
 correct\_answers += 1  
 *else*:  
 incorrect\_answers += 1  
  
 correct\_answers\_percentage = round((correct\_answers / len(answer)) \* 100)  
 incorrect\_answers\_percentage = round((incorrect\_answers / len(answer)) \* 100)  
 statistics\_txt.write(f'{iterator} student has {correct\_answers\_percentage}% of correct answers '  
 f'and {incorrect\_answers\_percentage}% of incorrect answers.\n')  
 print(f'{iterator} student has {correct\_answers\_percentage}% of correct answers '  
 f'and {incorrect\_answers\_percentage}% of incorrect answers.')  
 iterator += 1  
  
 print()  
 statistics\_txt.write('\n')  
 top\_5\_results = []  
 students\_mark\_per\_min\_copy = students\_mark\_per\_min[:]  
 *for* i *in* range(0, 5):  
 max\_average\_mark = max(students\_mark\_per\_min\_copy)  
 top\_5\_results.append(max\_average\_mark)  
 students\_mark\_per\_min\_copy.pop(students\_mark\_per\_min\_copy.index(max\_average\_mark))  
 index\_of\_student = students\_mark\_per\_min.index(top\_5\_results[i])  
 statistics\_txt.write(f'Top {i + 1}! {index\_of\_student + 1} student has {top\_5\_results[i]}/min mark.\n')  
 print(f'Top {i + 1}! {index\_of\_student + 1} student has {top\_5\_results[i]}/min mark.')  
  
 last\_changes\_date = date.today().strftime('%B %d, %Y')  
 last\_changes\_time = datetime.now().strftime('%H:%M:%S')  
 statistics\_txt.write(f'\nLast changes time: {last\_changes\_date} {last\_changes\_time}.\n')  
  
  
task\_7()

***Результат програми:***

*** ***

******

******

***Увесь лістинг програми:***

*""" Lab 6. Python. Andrii Babushko. Repository: https://github.com/AndriiBabushko/Python """  
import* csv  
*import* os  
*import* io  
*from* datetime *import* datetime  
*from* datetime *import* date  
*from* time *import* perf\_counter  
  
*# task 1*"""   
 Завдання 1. Створіть новий файл numbers.txt у текстовому редакторі і запишіть у нього 10 чисел, кожне з нового   
 рядка. Напишіть програму, яка зчитує ці числа з файлу і обчислює їх суму, виводить цю суму на екран і, водночас,   
 записує цю суму у інший файл з назвою sum\_numbers.txt.  
"""  
  
  
*def* task\_1():  
 print('\nTASK 1!')  
 *if not* os.path.isdir(r'./task1'):  
 os.mkdir(r'./task1')  
  
 *with* io.open(r'./task1/numbers.txt', 'wt', encoding='utf-8') *as* numbers\_txt:  
 numbers\_txt.write(create\_numbers\_string())  
  
 *with* io.open(r'./task1/numbers.txt', 'rt', encoding='utf-8') *as* numbers\_txt:  
 task\_1\_sum = 0  
 task\_1\_number\_list = []  
  
 *for* number *in* numbers\_txt:  
 *try*:  
 task\_1\_sum += int(number)  
 task\_1\_number\_list.append(int(number))  
 *except* ValueError *as* err:  
 print(f'ERROR! {ValueError}')  
  
 *with* io.open(r'./task1/sum\_numbers.txt', 'wt', encoding='utf-8') *as* sum\_numbers\_txt:  
 sum\_numbers\_txt.write(str(task\_1\_sum))  
  
 print(f'Received numbers from file: {task\_1\_number\_list}')  
 print(f'Sum of numbers in file = {task\_1\_sum}')  
  
  
*def* create\_numbers\_string():  
 *from* random *import* randint  
 new\_string = ''  
  
 *for* new\_number *in* range(10):  
 new\_string += f'{randint(-100, 100)}\n'  
  
 *return* new\_string  
  
  
task\_1()  
  
*# task 2*"""   
 Завдання 2. Реалізуйте програму, яка зчитує довільну кількість цілих чисел, що вводяться з командного рядка, і   
 записує у текстовий файл інформацію, щодо парності або непарності чисел.  
"""  
  
  
*def* task\_2():  
 print('\nTASK 2!')  
 task\_2\_enter\_count = int(input('Enter how much numbers you want to enter: '))  
 task\_2\_number\_list = task\_2\_enter\_some\_numbers(task\_2\_enter\_count)  
  
 *if not* os.path.isdir(r'./task2'):  
 os.mkdir(r'./task2')  
  
 *with* io.open(r'./task2/even\_or\_odd\_numbers.txt', 'wt', encoding='utf-8') *as* even\_or\_odd\_numbers\_txt:  
 *for* number *in* task\_2\_number\_list:  
 *if* number % 2 == 0:  
 even\_or\_odd\_numbers\_txt.write(f'{number} is even number!\n')  
 *else*:  
 even\_or\_odd\_numbers\_txt.write(f'{number} is odd number!\n')  
  
 *with* io.open(r'./task2/even\_or\_odd\_numbers.txt', 'rt', encoding='utf-8') *as* even\_or\_odd\_numbers\_txt:  
 *for* line *in* even\_or\_odd\_numbers\_txt:  
 print(line, end="")  
  
  
*def* task\_2\_enter\_some\_numbers(count):  
 numbers\_list = []  
  
 *while* count != 0:  
 *try*:  
 int\_number = int(input('Enter value: '))  
 numbers\_list.append(int\_number)  
 count -= 1  
 *except* ValueError *as* value\_error:  
 numbers\_list.pop()  
 print('ERROR:', value\_error)  
  
 *return* numbers\_list  
  
  
task\_2()  
  
*# task 3*"""  
 Завдання 3. Створіть новий файл у текстовому редакторі і напишіть кілька рядків тексту у ньому про можливості   
 Python. Кожен рядок повинен починатися з фрази: «Python можна використати для ...» . Збережіть файл з ім’ям   
 learning\_python.txt. Напишіть програму, яка зчитує файл і виводить текст з перебором рядків файла і зі збереженням   
 рядків у списку з подальшим сортуванням списку за довжиною рядків в ньому від найбільшого до найменшого.  
"""  
  
  
*def* task\_3():  
 print('\nTASK 3!')  
  
 *if not* os.path.isdir(r'./task3'):  
 os.mkdir('./task3')  
  
 *with* io.open(r'./task3/learning\_python.txt', 'wt', encoding='utf-8') *as* learning\_python\_txt:  
 learning\_python\_txt.writelines(  
 ["Python можна використати для machine learning.\n",  
 "Python можна використати для написання простих скриптів.\n",  
 "Python можна використати для програмування веб додатків.\n",  
 "Python можна використати для написання особистого телеграм боту.\n"])  
  
 *with* io.open(r'./task3/learning\_python.txt', 'rt', encoding='utf-8') *as* learning\_python\_txt:  
 python\_string\_list = []  
  
 *for* line *in* learning\_python\_txt:  
 python\_string\_list.append(line)  
  
 python\_string\_sorted\_list = python\_string\_list[:]  
 python\_string\_sorted\_list.sort(key=len)  
 print('\nSorted list:')  
 *for* string *in* python\_string\_sorted\_list:  
 print(string, end="")  
  
 python\_string\_reversed\_sorted\_list = python\_string\_list[:]  
 python\_string\_reversed\_sorted\_list.sort(key=len, reverse=*True*)  
 print('\nReversed sorted list:')  
 *for* string *in* python\_string\_reversed\_sorted\_list:  
 print(string, end="")  
  
  
task\_3()  
  
*# task 4*"""  
 Завдання 4. Прочитайте кожен рядок зі створеного у попередньому завданні файла learning\_python.txt і замініть   
 слово Python назвою іншої мови, наприклад C при виведенні на екран. Отриманий файл має бути створений в новому   
 каталозі, що розміщується в поточному. Відкрийте файл пострічково і дайте можливість користувачеві визначити які   
 змінені фрази є актуальними, наприклад для мови С, а які ні. Всі хибні твердження запишіть в інший файл, а істинні   
 – в поточний.  
"""  
  
  
*def* task\_4():  
 print('\nTASK 4!')  
  
 *if not* os.path.isdir(r'./task4'):  
 os.mkdir(r'./task4')  
  
 *with* io.open(r'./task3/learning\_python.txt', 'rt', encoding='utf-8') *as* learning\_python\_txt:  
 python\_strings = []  
  
 *for* string *in* learning\_python\_txt:  
 python\_strings.append(string.split(' '))  
  
 *for* i *in* range(0, len(python\_strings)):  
 python\_strings[i][0] = 'C'  
 python\_strings[i] = " ".join(python\_strings[i])  
  
 *with* io.open(r'./task4/learning\_c.txt', 'wt', encoding='utf-8') *as* true\_about\_c\_txt:  
 *for* c\_string *in* python\_strings:  
 true\_about\_c\_txt.write(c\_string)  
  
 *if* os.path.exists(r'./task4/true\_about\_c.txt'):  
 os.remove(r'./task4/true\_about\_c.txt')  
 *if* os.path.exists(r'./task4/false\_about\_c.txt'):  
 os.remove(r'./task4/false\_about\_c.txt')  
  
 *with* io.open(r'./task4/learning\_c.txt', 'rt', encoding='utf-8') *as* learning\_c\_txt:  
 *for* c\_string *in* learning\_c\_txt:  
 print(c\_string, end='')  
 answer = enter\_true\_or\_false()  
 *if* answer:  
 *with* io.open(r'./task4/true\_about\_c.txt', 'at', encoding='utf-8') *as* true\_about\_c\_txt:  
 true\_about\_c\_txt.write(c\_string)  
 *else*:  
 *with* io.open(r'./task4/false\_about\_c.txt', 'at', encoding='utf-8') *as* false\_about\_c\_txt:  
 false\_about\_c\_txt.write(c\_string)  
  
 print(f'True statements about C:')  
 *with* io.open(r'./task4/true\_about\_c.txt', 'rt', encoding='utf-8') *as* true\_about\_c\_txt\_read:  
 *for* c\_string *in* true\_about\_c\_txt\_read:  
 print(c\_string, end='')  
  
 print(f'False statements about C:')  
 *with* io.open(r'./task4/false\_about\_c.txt', 'rt', encoding='utf-8') *as* false\_about\_c\_txt\_read:  
 *for* c\_string *in* false\_about\_c\_txt\_read:  
 print(c\_string, end='')  
  
  
*def* enter\_true\_or\_false():  
 *while True*:  
 *try*:  
 answer = input("Enter 'true' or 'false' about this statement: ")  
 *if* answer != 'true' *and* answer != 'false':  
 *raise* ValueError('Value was entered incorrectly!')  
 *else*:  
 *pass  
 if* answer == 'true':  
 *return True  
 elif* answer == 'false':  
 *return False  
 except* ValueError *as* value\_err:  
 print(f'ERROR! {value\_err}')  
  
  
task\_4()  
  
*# task 5*"""  
 Завдання 5. Створіть порожній файл guest\_book.txt у текстовому редакторі. Напишіть програму, яка запитує у   
 користувачів імена. При введенні кожного імені виведіть на екран рядок з вітанням для користувача і запишіть   
 рядок вітання у файл з ім’ям guest\_book.txt. Простежте за тим, щоб кожне повідомлення розміщувалося в окремому   
 рядку файла з зазначенням часу внесення цього повідомлення. Передбачте зазначення в файлі часу його створення і   
 вказання в ньому часу останніх внесених змін  
"""  
  
  
*def* task\_5():  
 print('\nTASK 5!')  
  
 *if not* os.path.isdir(r'./task5'):  
 os.mkdir(r'./task5')  
  
 task\_5\_count = enter\_count()  
 file\_create\_time = date.today().strftime('%B %d, %Y')  
  
 *with* io.open(r'./task5/guest\_book.txt', 'wt', encoding='utf-8') *as* guest\_book\_txt:  
 guest\_book\_txt.write(f'File was created: {file\_create\_time}\n\n')  
  
 *for* counter *in* range(0, task\_5\_count):  
 name\_greeting = f'{counter + 1}) Hello, ' + enter\_name() + '! '  
 message\_time = datetime.now().strftime('%H:%M:%S')  
 name\_greeting += f'Name was entered in time: {message\_time}.\n'  
 print(name\_greeting, end="")  
 guest\_book\_txt.write(name\_greeting)  
  
 last\_changes\_date = date.today().strftime('%B %d, %Y')  
 last\_changes\_time = datetime.now().strftime('%H:%M:%S')  
 guest\_book\_txt.write(f'\nLast changes time: {last\_changes\_date} {last\_changes\_time}.\n')  
  
  
*def* enter\_name():  
 *while True*:  
 *try*:  
 name = input('Enter name: ')  
 *if* len(name) < 2:  
 *raise* ValueError('Name\'s length is less than 2!')  
 *else*:  
 *pass  
 return* name  
 *except* ValueError *as* value\_err:  
 print(f'ERROR! {value\_err}')  
  
  
*def* enter\_count():  
 *while True*:  
 *try*:  
 count = int(input('Enter count of names: '))  
 *if* count <= 0:  
 *raise* ValueError('Count names was entered incorrectly!')  
 *else*:  
 *pass  
 return* count  
 *except* ValueError *as* value\_err:  
 print(f'ERROR! {value\_err}')  
  
  
task\_5()  
  
*# task 6*"""  
 Завдання 6. Збережіть в текстовому файлі публікацію про Python на 3000 слів англійською мовою. Напишіть програму,   
 що аналізуватиме частоту з якою в тексті зустрічатимуться окремі літери чи слова незалежно від їх регістру.   
 Результат робот програми має виводитись в консоль і зберігатись в окремому файлі з зазначенням часу його створення,   
 часу виконання окремих змін, результатів пошуку і часу, що знадобився на виконання цього пошуку.  
"""  
  
  
*def* task\_6():  
 *import* re  
 print('\nTASK 6!')  
  
 *with* io.open(r'./task6/post\_about\_python.txt', 'rt', encoding='utf-8') *as* post\_about\_python\_txt:  
 words = re.split(r' |\n|\? |\.|\, |\: |\;', post\_about\_python\_txt.read().lower())  
  
 file\_create\_date = date.today().strftime('%B %d, %Y')  
 file\_create\_time = datetime.now().strftime('%H:%M:%S')  
 *with* io.open(r'./task6/word\_frequency.txt', 'wt', encoding='utf-8') *as* word\_frequency\_txt:  
 word\_frequency\_txt.write(f'File was created: {file\_create\_date} {file\_create\_time}\n\n')  
 print(f'File was created: {file\_create\_time}\n')  
  
 words\_without\_duplicates = []  
 *for* string *in* words:  
 *if* string *not in* words\_without\_duplicates:  
 words\_without\_duplicates.append(string)  
  
 *for* i *in* range(0, len(words\_without\_duplicates)):  
 time\_started = perf\_counter()  
 counter = 0  
 *for* j *in* range(0, len(words)):  
 *if* words\_without\_duplicates[i] == words[j]:  
 counter += 1  
 time\_finished = perf\_counter()  
 write\_time = round(time\_finished - time\_started, 8)  
 word\_frequency\_txt.write(f'"{words[i]}" repeats "{counter}" times. Write time: {write\_time}.\n')  
 print(f'"{words[i]}" repeats "{counter}" times. Write time: {write\_time}.')  
  
 last\_changes\_date = date.today().strftime('%B %d, %Y')  
 last\_changes\_time = datetime.now().strftime('%H:%M:%S')  
 word\_frequency\_txt.write(f'\nLast changes time: {last\_changes\_date} {last\_changes\_time}.\n')  
 print(f'\nLast changes time: {last\_changes\_date} {last\_changes\_time}.')  
  
  
task\_6()  
  
*# task 7*"""  
 Завдання 7. Завантажте файл marks.csv і визначте кількість студентів, що проходили тестування. Виведіть інформацію   
 про те яку оцінку набрали відповідна кільксть студентів. Виведіть інформацію яку середню оцінку отримував студент   
 за певний час виконання КМР (крок – 1 хв). Створіть текстовий файл і запишіть в нього статистику по правильним   
 відповідям для кожного окремого питання (який відсоток правильних і неправильних відповідей на питання дали   
 студенти). В цей же файл внесіть інформацію про 5 найкращих оцінок в співвідношенні оцінка/час витрачений   
 складання КМР.  
"""  
  
  
*def* task\_7():  
 print('\nTASK 7!')  
  
 *if not* os.path.isdir(r'./task7'):  
 os.mkdir(r'./task7')  
  
 *with* io.open(r'./task7/marks.lab6.csv', 'rt', encoding='utf-8') *as* marks\_lab6\_csv:  
 marks\_csv = csv.reader(marks\_lab6\_csv)  
 marks = [line *for* line *in* marks\_csv]  
  
 students\_count = len(marks)  
 print(f'Total students count who wrote a test: {students\_count}')  
  
 students\_marks = []  
 *for* i *in* range(students\_count):  
 mark\_string = marks[i][4].split(',')  
 mark\_number = float(mark\_string[0]) + float(int(mark\_string[1]) / 100)  
 students\_marks.append(mark\_number)  
  
 print()  
 iterator = 1  
 *for* mark *in* students\_marks:  
 print(f'{iterator} student received {mark} mark.')  
 iterator += 1  
  
 students\_time = []  
 *for* i *in* range(students\_count):  
 time = marks[i][3].split(' ')  
 *if* len(time) > 2:  
 minutes = int(time[0])  
 seconds = int(time[2]) + minutes \* 60  
 *else*:  
 minutes = int(time[0])  
 seconds = minutes \* 60  
 students\_time.append(seconds)  
  
 students\_mark\_per\_min = []  
 *for* i *in* range(students\_count):  
 students\_mark\_per\_min.append(round((students\_marks[i] / students\_time[i]) \* 60, 2))  
  
 print()  
 iterator = 1  
 *for* average\_mark *in* students\_mark\_per\_min:  
 print(f'{iterator} student received {average\_mark}/min.')  
 iterator += 1  
  
 file\_create\_date = date.today().strftime('%B %d, %Y')  
 file\_create\_time = datetime.now().strftime('%H:%M:%S')  
 *with* io.open(r'./task7/statistics.txt', 'wt', encoding='utf-8') *as* statistics\_txt:  
 statistics\_txt.write(f'File was created: {file\_create\_date} {file\_create\_time}\n\n')  
  
 student\_answers = []  
 *for* answers *in* marks:  
 student\_answers.append(answers[5:])  
  
 print()  
 iterator = 1  
 *for* answer *in* student\_answers:  
 correct\_answers = 0  
 incorrect\_answers = 0  
  
 *for* i *in* range(0, len(answer)):  
 *if* answer[i] == '0,50':  
 correct\_answers += 1  
 *else*:  
 incorrect\_answers += 1  
  
 correct\_answers\_percentage = round((correct\_answers / len(answer)) \* 100)  
 incorrect\_answers\_percentage = round((incorrect\_answers / len(answer)) \* 100)  
 statistics\_txt.write(f'{iterator} student has {correct\_answers\_percentage}% of correct answers '  
 f'and {incorrect\_answers\_percentage}% of incorrect answers.\n')  
 print(f'{iterator} student has {correct\_answers\_percentage}% of correct answers '  
 f'and {incorrect\_answers\_percentage}% of incorrect answers.')  
 iterator += 1  
  
 print()  
 statistics\_txt.write('\n')  
 top\_5\_results = []  
 students\_mark\_per\_min\_copy = students\_mark\_per\_min[:]  
 *for* i *in* range(0, 5):  
 max\_average\_mark = max(students\_mark\_per\_min\_copy)  
 top\_5\_results.append(max\_average\_mark)  
 students\_mark\_per\_min\_copy.pop(students\_mark\_per\_min\_copy.index(max\_average\_mark))  
 index\_of\_student = students\_mark\_per\_min.index(top\_5\_results[i])  
 statistics\_txt.write(f'Top {i + 1}! {index\_of\_student + 1} student has {top\_5\_results[i]}/min mark.\n')  
 print(f'Top {i + 1}! {index\_of\_student + 1} student has {top\_5\_results[i]}/min mark.')  
  
 last\_changes\_date = date.today().strftime('%B %d, %Y')  
 last\_changes\_time = datetime.now().strftime('%H:%M:%S')  
 statistics\_txt.write(f'\nLast changes time: {last\_changes\_date} {last\_changes\_time}.\n')  
  
  
task\_7()

***Висновок:*** під час виконання лабораторної роботи було отримано навички роботи з текстовими та csv файлами на мові Python.