Міністерство освіти і науки України

Державний університет “Житомирська політехніка”

Кафедра інженерії програмного забезпечення

Група: ВТ-21-1[1]

Програмування мовою Python

Лабораторна робота № 4

«Розгалуження та цикли»

Виконав: Бабушко А. С.

Прийняв: Морозов Д. С.

***Мета роботи:*** ознайомитися зі рядками в мові Python, діями над ними.

***Хід роботи:***

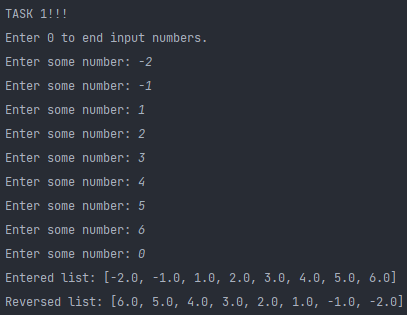
***Завдання на лабораторну роботу:***

Завдання 1. Дано список, що складається з N цілочисельних елементів. Список вводиться з клавіатури. Знайти максимальний елемент. Вивести список на екран у зворотному порядку.

***Лістинг програми:***

*# task 1  
def* task\_1\_output\_reverse\_list(someList):  
 someList.reverse()  
 *return* someList  
  
  
*def* enter\_some\_list():  
 print('Enter 0 to end input numbers.')  
 newList = []  
 *while True*:  
 number = float(input('Enter some number: '))  
 *if* number == 0:  
 *return* newList  
 *else*:  
 newList.append(number)  
  
  
print('\nTASK 1!!!')  
  
  
task\_1\_list = enter\_some\_list()  
print(f'Entered list: {task\_1\_list}')  
print(f'Reversed list: {task\_1\_output\_reverse\_list(task\_1\_list)}')

***Результат програми:***

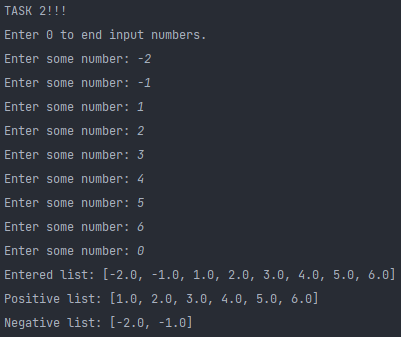
******

Завдання 2. Дано список, що складається з N цілочисельних елементів Список вводиться з клавіатури. Переписати всі додатні елементи в другій масив, а решту - в третій.

***Лістинг програми:***

*# task 2  
def* task\_2\_get\_positive\_negative\_lists(someList):  
 positive\_list = []  
 negative\_list = []  
 *for* number *in* someList:  
 *if* number >= 0:  
 positive\_list.append(number)  
 *else*:  
 negative\_list.append(number)  
 *return* [positive\_list, negative\_list]  
  
  
print('\nTASK 2!!!')  
  
  
task\_2\_list = enter\_some\_list()  
print(f'Entered list: {task\_2\_list}')  
print(f'Positive list: {task\_2\_get\_positive\_negative\_lists(task\_2\_list)[0]}')  
print(f'Negative list: {task\_2\_get\_positive\_negative\_lists(task\_2\_list)[1]}')

***Результат програми:***

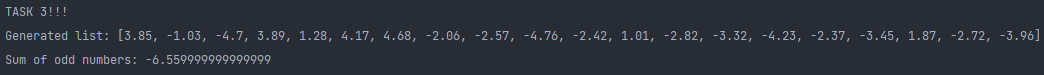


Завдання 3. В списку довжиною 20 обчислити суму елементів з непарними індексами. Вивести на екран список, отриману суму.

***Лістинг програми:***

*# task 3  
def* task\_3\_get\_index\_odd\_sum\_list(someList):  
 odd\_sum = 0  
  
 *for* i *in* range(0, len(someList)):  
 *if* i % 2 != 0:  
 odd\_sum += someList[i]  
  
 *return* odd\_sum  
  
  
*def* get\_random\_float\_number(minRandom, maxRandom):  
 *import* random *as* random  
 *return* round((random.random() \* (maxRandom - minRandom) + minRandom), 2)  
  
  
*def* create\_random\_float\_list(n, minRandom, maxRandom):  
 newList = []  
  
 *for* i *in* range(0, n):  
 newList.append(get\_random\_float\_number(minRandom, maxRandom))  
  
 *return* newList  
  
  
print('\nTASK 3!!!')  
task\_3\_list = create\_random\_float\_list(20, -5, 5)  
print(f'Generated list: {task\_3\_list}')  
print(f'Sum of odd numbers: {task\_3\_get\_index\_odd\_sum\_list(task\_3\_list)}')

***Результат програми:***

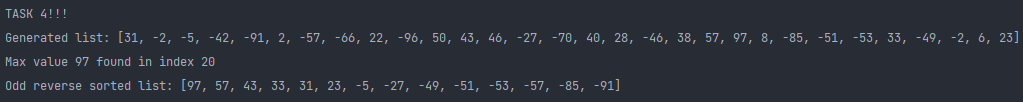


Завдання 4. Сформувати список з 30 випадкових цілих чисел від -100 до + 100. Знайти максимальний елемент списку і його порядковий номер. Отримати інший список, що складається тільки з непарних чисел вихідного списку або повідомити, що таких чисел немає. Отриманий список вивести в порядку зменшення елементів.

***Лістинг програми:***

*# task 4  
def* get\_max\_elem\_list(some\_list):  
 *return* max(some\_list)  
  
  
*def* get\_max\_index\_list(some\_list):  
 *return* some\_list.index(max(some\_list))  
  
  
*def* get\_random\_int\_number(min\_random, max\_random):  
 *import* random *as* random  
 *return* round((random.random() \* (max\_random - min\_random) + min\_random))  
  
  
*def* create\_random\_int\_list(n, min\_random, max\_random):  
 new\_list = []  
  
 *for* i *in* range(0, n):  
 new\_list.append(get\_random\_int\_number(min\_random, max\_random))  
  
 *return* new\_list  
  
  
*def* task\_5\_get\_sorted\_odd\_list(some\_list):  
 odd\_number\_list = []  
  
 *for* number *in* some\_list:  
 *if* number % 2 != 0:  
 odd\_number\_list.append(number)  
  
 odd\_number\_list.sort(reverse=*True*)  
 *return* odd\_number\_list  
  
  
print('\nTASK 4!!!')  
task\_4\_list = create\_random\_int\_list(30, -100, 100)  
print(f'Generated list: {task\_4\_list}')  
print(f'Max value {get\_max\_elem\_list(task\_4\_list)} found in index {get\_max\_index\_list(task\_4\_list)}')  
print(f'Odd reverse sorted list: {task\_5\_get\_sorted\_odd\_list(task\_4\_list)}')

***Результат програми:***

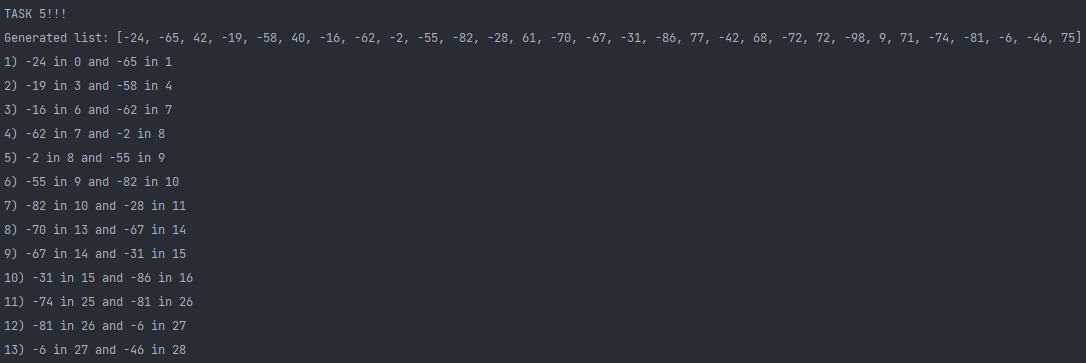
******

Завдання 5. Сформувати список з 30 випадкових цілих чисел від -100 до + 100. Вивести пари від’ємних чисел, що стоять поруч.

***Лістинг програми:***

*# task 5  
def* task\_5\_get\_negative\_pairs(some\_list):  
 negative\_pairs\_counter = 0  
 *for* i *in* range(0, len(some\_list) - 1):  
 *if* some\_list[i] < 0 *and* some\_list[i + 1] < 0:  
 negative\_pairs\_counter += 1  
 print(f'{negative\_pairs\_counter}) {some\_list[i]} in {i} and {some\_list[i + 1]} in {i + 1}')  
  
  
print('\nTASK 5!!!')  
task\_5\_list = create\_random\_int\_list(30, -100, 100)  
print(f'Generated list: {task\_5\_list}')  
task\_5\_get\_negative\_pairs(task\_5\_list)

***Результат програми:***

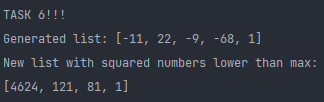
******

Завдання 6. Дано список з 10 цілих чисел. Знайти максимальний елемент і порівняти з ним інші елементи. Квадрати менших чисел записати в другий список в порядку зменшення.

***Лістинг програми:***

*# task 6  
def* task\_6\_get\_squares\_list\_lower\_than\_max(some\_list):  
 squares\_list = []  
 max\_number = max(some\_list)  
  
 *for* number *in* some\_list:  
 *if* number < max\_number:  
 squares\_list.append(number \*\* 2)  
  
 squares\_list.sort(reverse=*True*)  
 *return* squares\_list  
  
  
print('\nTASK 6!!!')  
task\_6\_list = create\_random\_int\_list(5, -100, 100)  
print(f'Generated list: {task\_6\_list}')  
print(f'New list with squared numbers lower than max:\n{task\_6\_get\_squares\_list\_lower\_than\_max(task\_6\_list)}')

***Результат програми:***

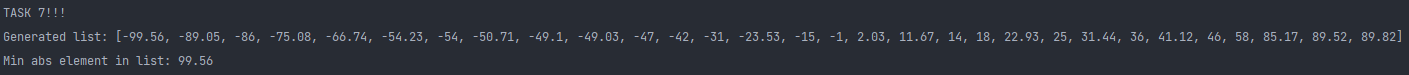
******

Завдання 7. Дано список з 30 випадкових цілих і дробових чисел від -100 до + 100. Знайти та вивести мінімальний по модулю елемент. Вивести список на екран в порядку збільшення значення.

***Лістинг програми:***

*# task 7  
def* task\_7\_get\_min\_abs\_elem(some\_list):  
 *return* abs(min(some\_list))  
  
  
*def* create\_random\_int\_float\_list(n, min\_random, max\_random):  
 *import* random *as* random  
  
 new\_list = []  
 bools = [*True*, *False*]  
  
 *for* i *in* range(0, n):  
 bool\_type = bools[random.randint(0, 1)]  
 *if* bool\_type:  
 new\_list.append(get\_random\_int\_number(min\_random, max\_random))  
 *else*:  
 new\_list.append(get\_random\_float\_number(min\_random, max\_random))  
  
 new\_list.sort()  
 *return* new\_list  
  
  
print('\nTASK 7!!!')  
task\_7\_list = create\_random\_int\_float\_list(30, -100, 100)  
print(f'Generated list: {task\_7\_list}')  
print(f'Min abs element in list: {task\_7\_get\_min\_abs\_elem(task\_7\_list)}')

***Результат програми:***

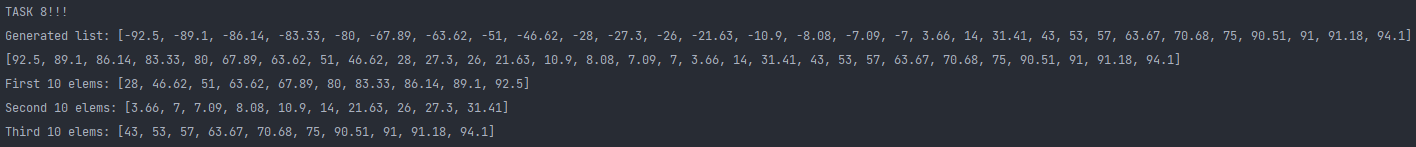
******

Завдання 8. Дано список з 30 випадкових цілих і дробових чисел від -100 до + 100. Сформувати зі списку 10 списків по 3 елементи. Вивести отримані списки в порядку зростання за сумою абсолютних значень окремих елементів.

***Лістинг програми:***

*# task 8  
def* task\_8\_create\_3\_lists\_of\_10\_elems(some\_list):  
 *for* i *in* range(0, len(some\_list)):  
 some\_list[i] = abs(some\_list[i])  
  
 print(some\_list)  
 first\_10\_elems = some\_list[:10]  
 first\_10\_elems.sort()  
 second\_10\_elems = some\_list[10:20]  
 second\_10\_elems.sort()  
 third\_10\_elems = some\_list[20:30]  
 third\_10\_elems.sort()  
  
 *return* [first\_10\_elems, second\_10\_elems, third\_10\_elems]  
  
  
print('\nTASK 8!!!')  
task\_8\_list = create\_random\_int\_float\_list(30, -100, 100)  
print(f'Generated list: {task\_8\_list}')  
three\_lists\_of\_10\_elems = task\_8\_create\_3\_lists\_of\_10\_elems(task\_8\_list)  
print(f'First 10 elems: {three\_lists\_of\_10\_elems[0]}')  
print(f'Second 10 elems: {three\_lists\_of\_10\_elems[1]}')  
print(f'Third 10 elems: {three\_lists\_of\_10\_elems[2]}')

***Результат програми:***

******

***Увесь лістинг програми:***

*""" Lab 4. Python. Andrii Babushko. Repository: https://github.com/AndriiBabushko/Python """  
  
  
# task 1  
def* task\_1\_output\_reverse\_list(someList):  
 someList.reverse()  
 *return* someList  
  
  
*def* enter\_some\_list():  
 print('Enter 0 to end input numbers.')  
 newList = []  
 *while True*:  
 number = float(input('Enter some number: '))  
 *if* number == 0:  
 *return* newList  
 *else*:  
 newList.append(number)  
  
  
print('\nTASK 1!!!')  
  
  
task\_1\_list = enter\_some\_list()  
print(f'Entered list: {task\_1\_list}')  
print(f'Reversed list: {task\_1\_output\_reverse\_list(task\_1\_list)}')  
  
  
*# task 2  
def* task\_2\_get\_positive\_negative\_lists(someList):  
 positive\_list = []  
 negative\_list = []  
 *for* number *in* someList:  
 *if* number >= 0:  
 positive\_list.append(number)  
 *else*:  
 negative\_list.append(number)  
 *return* [positive\_list, negative\_list]  
  
  
print('\nTASK 2!!!')  
  
  
task\_2\_list = enter\_some\_list()  
print(f'Entered list: {task\_2\_list}')  
print(f'Positive list: {task\_2\_get\_positive\_negative\_lists(task\_2\_list)[0]}')  
print(f'Negative list: {task\_2\_get\_positive\_negative\_lists(task\_2\_list)[1]}')  
  
  
*# task 3  
def* task\_3\_get\_index\_odd\_sum\_list(someList):  
 odd\_sum = 0  
  
 *for* i *in* range(0, len(someList)):  
 *if* i % 2 != 0:  
 odd\_sum += someList[i]  
  
 *return* odd\_sum  
  
  
*def* get\_random\_float\_number(minRandom, maxRandom):  
 *import* random *as* random  
 *return* round((random.random() \* (maxRandom - minRandom) + minRandom), 2)  
  
  
*def* create\_random\_float\_list(n, minRandom, maxRandom):  
 newList = []  
  
 *for* i *in* range(0, n):  
 newList.append(get\_random\_float\_number(minRandom, maxRandom))  
  
 *return* newList  
  
  
print('\nTASK 3!!!')  
task\_3\_list = create\_random\_float\_list(20, -5, 5)  
print(f'Generated list: {task\_3\_list}')  
print(f'Sum of odd numbers: {task\_3\_get\_index\_odd\_sum\_list(task\_3\_list)}')  
  
  
*# task 4  
def* get\_max\_elem\_list(some\_list):  
 *return* max(some\_list)  
  
  
*def* get\_max\_index\_list(some\_list):  
 *return* some\_list.index(max(some\_list))  
  
  
*def* get\_random\_int\_number(min\_random, max\_random):  
 *import* random *as* random  
 *return* round((random.random() \* (max\_random - min\_random) + min\_random))  
  
  
*def* create\_random\_int\_list(n, min\_random, max\_random):  
 new\_list = []  
  
 *for* i *in* range(0, n):  
 new\_list.append(get\_random\_int\_number(min\_random, max\_random))  
  
 *return* new\_list  
  
  
*def* task\_5\_get\_sorted\_odd\_list(some\_list):  
 odd\_number\_list = []  
  
 *for* number *in* some\_list:  
 *if* number % 2 != 0:  
 odd\_number\_list.append(number)  
  
 odd\_number\_list.sort(reverse=*True*)  
 *return* odd\_number\_list  
  
  
print('\nTASK 4!!!')  
task\_4\_list = create\_random\_int\_list(30, -100, 100)  
print(f'Generated list: {task\_4\_list}')  
print(f'Max value {get\_max\_elem\_list(task\_4\_list)} found in index {get\_max\_index\_list(task\_4\_list)}')  
print(f'Odd reverse sorted list: {task\_5\_get\_sorted\_odd\_list(task\_4\_list)}')  
  
  
*# task 5  
def* task\_5\_get\_negative\_pairs(some\_list):  
 negative\_pairs\_counter = 0  
 *for* i *in* range(0, len(some\_list) - 1):  
 *if* some\_list[i] < 0 *and* some\_list[i + 1] < 0:  
 negative\_pairs\_counter += 1  
 print(f'{negative\_pairs\_counter}) {some\_list[i]} in {i} and {some\_list[i + 1]} in {i + 1}')  
  
  
print('\nTASK 5!!!')  
task\_5\_list = create\_random\_int\_list(30, -100, 100)  
print(f'Generated list: {task\_5\_list}')  
task\_5\_get\_negative\_pairs(task\_5\_list)  
  
  
*# task 6  
def* task\_6\_get\_squares\_list\_lower\_than\_max(some\_list):  
 squares\_list = []  
 max\_number = max(some\_list)  
  
 *for* number *in* some\_list:  
 *if* number < max\_number:  
 squares\_list.append(number \*\* 2)  
  
 squares\_list.sort(reverse=*True*)  
 *return* squares\_list  
  
  
print('\nTASK 6!!!')  
task\_6\_list = create\_random\_int\_list(5, -100, 100)  
print(f'Generated list: {task\_6\_list}')  
print(f'New list with squared numbers lower than max:\n{task\_6\_get\_squares\_list\_lower\_than\_max(task\_6\_list)}')  
  
  
*# task 7  
def* task\_7\_get\_min\_abs\_elem(some\_list):  
 *return* abs(min(some\_list))  
  
  
*def* create\_random\_int\_float\_list(n, min\_random, max\_random):  
 *import* random *as* random  
  
 new\_list = []  
 bools = [*True*, *False*]  
  
 *for* i *in* range(0, n):  
 bool\_type = bools[random.randint(0, 1)]  
 *if* bool\_type:  
 new\_list.append(get\_random\_int\_number(min\_random, max\_random))  
 *else*:  
 new\_list.append(get\_random\_float\_number(min\_random, max\_random))  
  
 new\_list.sort()  
 *return* new\_list  
  
  
print('\nTASK 7!!!')  
task\_7\_list = create\_random\_int\_float\_list(30, -100, 100)  
print(f'Generated list: {task\_7\_list}')  
print(f'Min abs element in list: {task\_7\_get\_min\_abs\_elem(task\_7\_list)}')  
  
  
*# task 8  
def* task\_8\_create\_3\_lists\_of\_10\_elems(some\_list):  
 *for* i *in* range(0, len(some\_list)):  
 some\_list[i] = abs(some\_list[i])  
  
 print(some\_list)  
 first\_10\_elems = some\_list[:10]  
 first\_10\_elems.sort()  
 second\_10\_elems = some\_list[10:20]  
 second\_10\_elems.sort()  
 third\_10\_elems = some\_list[20:30]  
 third\_10\_elems.sort()  
  
 *return* [first\_10\_elems, second\_10\_elems, third\_10\_elems]  
  
  
print('\nTASK 8!!!')  
task\_8\_list = create\_random\_int\_float\_list(30, -100, 100)  
print(f'Generated list: {task\_8\_list}')  
three\_lists\_of\_10\_elems = task\_8\_create\_3\_lists\_of\_10\_elems(task\_8\_list)  
print(f'First 10 elems: {three\_lists\_of\_10\_elems[0]}')  
print(f'Second 10 elems: {three\_lists\_of\_10\_elems[1]}')  
print(f'Third 10 elems: {three\_lists\_of\_10\_elems[2]}')

***Висновок:*** під час виконання лабораторної роботи було отримано навички використання списків як структуру даних та деяких методів для роботи з ними.