Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

Механіко-технологічний факультет

ЗВІТ

ПРО ВИКОНАННЯ ЛАБОРАТОРНОЇ РОБОТИ № 2

з навчальної дисципліни

“Скриптові мови програмування”

Функції у Python

ВИКОНАВ

студент академічної групи КБ-20

Прокопенко Є. С.

ПЕРЕВІРИВ

викладач кафедри кібербезпеки   
та програмного забезпечення

Савеленко О.К.

Кропивницький – 2021

**Мета:** навчитися створювати власні функції у мові Python

**Варіант 16**

**Завдання:**

Розробити функції для здійснення наступних операцій зі списками:

1. Швидке сортування;

2. Пошук елементу за значенням;

3. Пошук послідовності елементів;

4. Пошук перших п’яти мінімальних елементів;

5. Пошук перших п’яти максимальних елементів;

6. Пошук середнього арифметичного;

7. Повернення списку, що сформований з початкового списку, але не містить повторів (залишається лише перший з однакових елементів).

Помістити функції в окремий модуль. Реалізуйте програму, яка використовує всі функції зі створеного модуля. Зробити описи Doc strings для кожної реалізованої функції.

**Завдання 1**

* 1. **Теоретичні відомості**

Швидке сортування є істотно поліпшеним варіантом алгоритму сортування за допомогою прямого обміну (його варіанти відомі як «Бульбашкова сортування» та «шейкерні сортування»), відомого в тому числі своєю низькою ефективністю. Принципова відмінність полягає в тому, що в першу чергу проводяться перестановки на найбільшому можливій відстані і після кожного проходу елементи діляться на дві незалежні групи. (Таким чином поліпшення самого неефективного прямого методу сортування дало в результаті один з найбільш ефективних поліпшених методів.) Загальна ідея алгоритму полягає в наступному: Вибрати з масиву елемент, званий опорним. Це може бути будь-який з елементів масиву. Від вибору опорного елемента не залежить коректність алгоритму, але в окремих випадках може сильно залежати його ефективність (див. Нижче). Порівняти всі інші елементи з опорним і переставити їх у масиві так, щоб розбити масив на три безперервних відрізка, що слідують один за одним: «елементи менші опорного», «рівні» і «великі». Для відрізків «менших» і «великих» значень виконати рекурсивно ту ж послідовність операцій, якщо довжина відрізка більше одиниці.

* 1. **Лістинг до функції 1**

import random

def quickSort(numbers):

"""

The function sorts the list in ascending order. Returns an already sorted list.

Parameters

----------

numbers : list.

Returns

-------

list.

"""

if ( len(numbers) < 2 ):

return numbers

else:

centerNumber = random.choice(numbers)

smallNumbers = [ number for number in numbers if ( number < centerNumber )]

largNumbers = [ number for number in numbers if ( number > centerNumber )]

centerNumbers = [centerNumber] \* numbers.count(centerNumber)

return quickSort(smallNumbers) + centerNumbers + quickSort(largNumbers)

**Завдання 2**

**2.1Теоретичні відомості**

Зазвичай для пошуку елементу у масиві використовуються цикли, але у пайтоні можливо обійтися одним порівнянням щоб дізнатися чи існує такий елемент у масиві та пошуку індексу елементу стандартною функцією list.index(elem).

* 1. **Лістинг функції 2**

def findElem(elem,list):

"""

The function looks for a given element in the list. If such an element exists, then its index in the list will be returned, and if there is no such element, then None will be returned.

Parameters

----------

elem : int or float or str .

list : list.

Returns

-------

int or None.

"""

if (elem in list):

return list.index(elem)

else:

return None

**Завдання 3**

* 1. **Теоретичні відомості**

Пошук списку у списку здійснюється за допомогою двох циклів в яких будуть порівнюватись елементи обох списків на відповідність.

* 1. **Лістинг до функції 3**

def findListInList(mainList,listForFind):

"""

The function looks for a list of elements in another list. If such a situation exists, then the initial index of the list will be returned, which should be found in the main list, and if such a situation does not exist, then None will be returned.

Parameters

----------

mainList : list.

listForFind : list.

Returns

-------

int or None.

"""

for i in range(len(mainList)):

tempArray = []

if len(listForFind) <= len(mainList) - i:

for j in range(len(listForFind)):

if listForFind[j] == mainList[i+j]:

tempArray.append(listForFind[j])

else:

break

if tempArray == listForFind:

return i

return None

**Завдання 4**

**4.1 Теоретичні відомості**

Для пошуку 5 найбільших елементів достатньо одного циклу та стандартної функції max(list).

**4.2 Лістинг функції 4**

def findFiveMax(list):

"""

The function returns the 5 elements with the largest value in the list. If the list contains less than 5 elements, a descending sorted list will be returned.

Parameters

----------

list : list.

Returns

-------

list.

"""

if ( len(list) < 6 ):

list = quickSort(list)

list.reverse()

return list

else:

maxList = []

list2 = []

list2 = list.copy()

for i in range(5):

maxList.append(max(list2))

list2.remove(max(list2))

return maxList

**Завдання 5**

**5.1 Теоретичні відомості**

Для пошуку 5 найменших елементів достатньо одного циклу та стандартної функції mіn(list).

**5.2 Лістинг функції 5**

def findFiveMin(list):

"""

The function returns the 5 elements with the smallest value in the list. If the list contains less than 5 elements, an ascending sorted list will be returned.

Parameters

----------

list : list.

Returns

-------

list.

"""

if ( len(list) < 6 ):

return quickSort(list)

else:

minList = []

list2 = list.copy()

for i in range(5):

minList.append(min(list2))

list2.remove(min(list2))

return minList

**Завдання 6**

**6.1 Теоретичні відомості**

Для знаходження середнього арифметичного всіх елементів масиву потрібно використати дві стандартні функції : sum(list) – яка знаходить суму всіх елементів у списку та len(list) – яка повертає кількість елементів у списку.

**6.2 Лістинг функції 6**

def average(list):

"""

The function returns the arithmetic mean of all the elements in the list. If the number of elements in the list is less than one, it will be return value of None

Parameters

----------

list : list.

Returns

-------

float.

"""

if len(list) > 0:

return sum(list)/len(list)

return None

**Завдання 7**

**7.1 Теоретичні відомості**

**Для вирішення цієї задачі нам потрімно «перевернути» список та за допомогую стандартних функцій роботи із списками list.count(object) та list.remove(object) видалити всі «продубльовані» елементи.**

**7.2 Лістинг функції 7**

def deleteClone(list):

"""

The function removes all duplicate elements in the list and leaves only the first element of the same elements from the beginning of the list.

Parameters

----------

list : list.

Returns

-------

list : list.

"""

list.reverse()

for i in list:

object = i

while(list.count(object) > 1):

list.remove(object)

list.reverse()

return list

**Тестування функцій**

Для тестування функцій був створений модуль який містить всі реалізовані функції – modulesPROKOPENKO.py. Взагалі модулі у пайтоні підключаються доволі легко. Достатньо лиш кинути файл із кодом модулів у папку із проектом або дожати його у папку зі стандартними модулями пайтон.

Для тестування модулів був створений програмний застосунок testModules.py.

**Лістинг testModules.py**

from modulesPROKOPENKO import quickSort,findElem, findFiveMax, findFiveMin, findListInList, average,deleteClone

mainList = [13,9,8,4,5,6,6,7,8,9]

listForFind = [6,6,7]

print("Відсортований список " + str(mainList) + ": " + str(quickSort(mainList)))

print("Пошук елемента зі значенням \" 5 \" у списку " + str(mainList))

print("Індекс цього елемента у списку: " + str(findElem(5,mainList)))

print("Пошук елемента зі значенням \" 666 \" у списку " + str(mainList) )

print("Індекс цього елемента у списку: " + str(findElem(666,mainList)))

print("Пошук cписку "+str(listForFind)+ " у списку " + str(mainList))

print("Індекс початку шуканого списку у головному списку: " + str(findListInList(mainList,listForFind)))

print("Пошук cписку "+str([65,54,78])+ " у списку " + str(mainList))

print("Індекс початку шуканого списку у головному списку: " + str(findListInList(mainList,[65,54,78])))

print("5 максимальних елементів із списку"+str(mainList)+": "+ str(findFiveMax(mainList)))

print("5 максимальних елементів із списку"+str(listForFind)+": "+ str(findFiveMax(listForFind)))

print("5 мінімальних елементів із списку"+str(mainList)+": "+ str(findFiveMin(mainList)))

print("5 мінімальних елементів із списку"+str(listForFind)+": "+ str(findFiveMin(listForFind)))

print("Середнє арифметичне списку" +str(mainList)+": " + str(average(mainList)))

print("Середнє арифметичне списку" +str([])+": " + str(average([])))

print("Список" + str(mainList)+" без \"повторів\": "+str(deleteClone(mainList)))

**Результати роботи** **testModules.py**

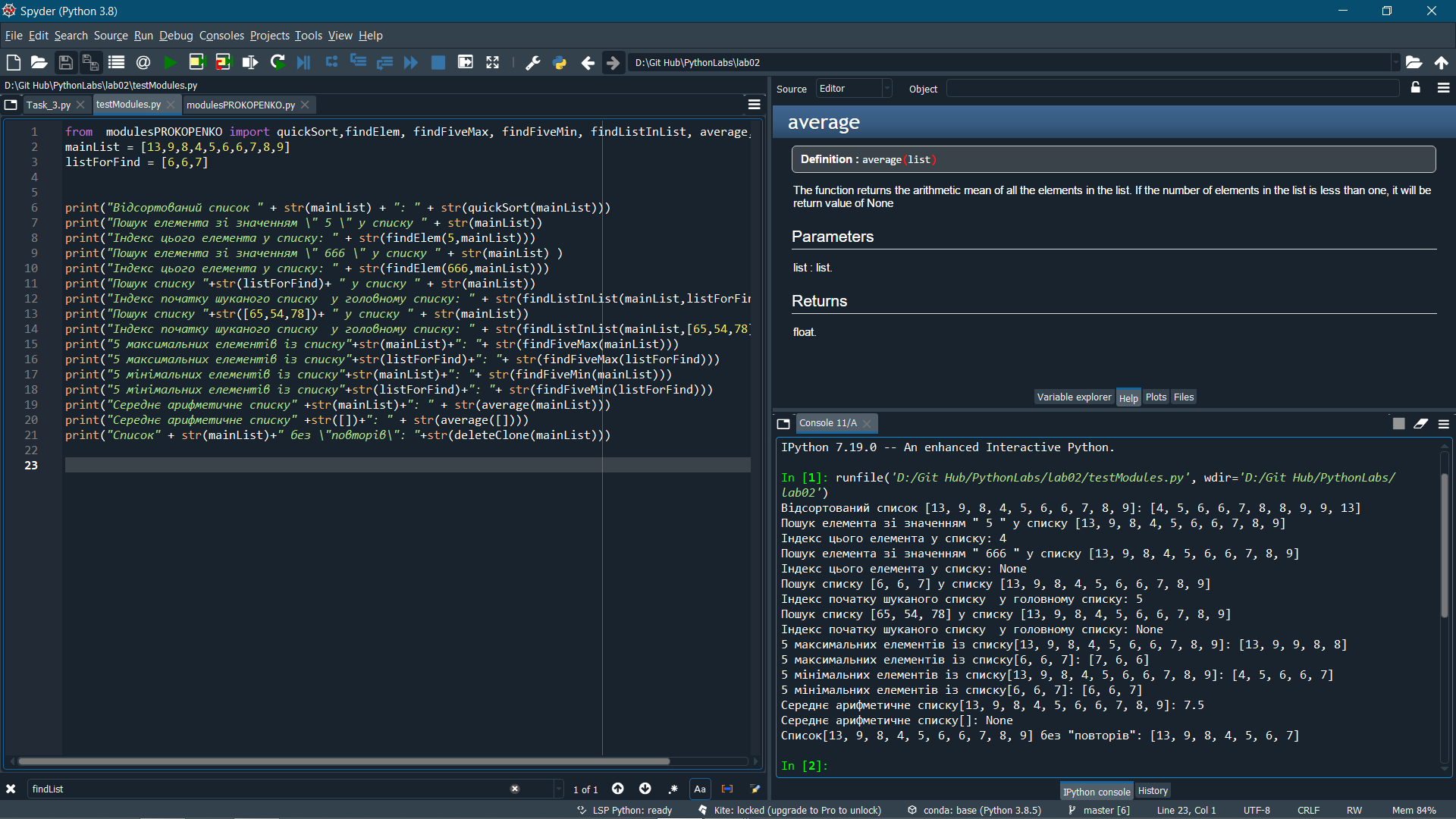


Рисунок 1 -Тестування створених функцій

**Висновки**

Дані завдання дозволяють ознайомитися із особливостями роботи зі списками та створювання модулів у мові програмування Python.

Взагалі можливо під час виконання лабораторної роботи зістрітись із такии труднощами, як копіюваня списків та створення модулів із подальшим їхнім підключенням до різних проєктів.

Якщо ви при «копіюванні» або передаванні списку у функцію робите так : new\_list = my\_list, то у вас фактично немає двох списків. Присвоювання просто копіює посилання на список, а не фактичний список, тому як new\_list, так і my\_list відносяться до того ж списку після призначення.

Щоб насправді скопіювати список, у розроника є різні способи:

* Ви можете використовувати вбудований list.copy () (доступний з python 3.3): new\_list = old\_list.copy ()
* Ви можете «відрізати» його: new\_list = old\_list [:]

На думку Алекса Мартеллі (у крайному випадку , в 2007 році), це дивний синтаксис, і використовувати його ніколи не має сенсу. (На його думку, наступний - більш читається).

* Ви можете використовувати вбудовану функцію list ():

new\_list = list (old\_list)

* Ви можете використовувати generic copy.copy ():

import copy

new\_list = copy.copy (old\_list)

Це трохи повільніше, ніж list () тому що спочатку потрібно з'ясувати тип даних old\_list.

* Якщо список містить об'єкти і ви хочете їх скопіювати, використовуйте загальний copy.deepcopy ():

import copy

new\_list = copy.deepcopy (old\_list)

Очевидно, найповільніший і найнеобхідніший для пам'яті спосіб, але іноді неминучий.

А щоб розібратися із другою проблемою треба знати , що таке модулі в пайтоні.

Модулем в Python називається будь-який файл з програмою. Отже виника єпитання куди йогог треба перемістити , щоб можливо було підключити ?

Відповідь надзвичайно проста : Туди, де його потім можна буде знайти. Шляхи пошуку модулів вказані в змінної sys.path. У нього включені поточна директорія (тобто модуль можна залишити в папці з основною програмою), а також директорії, в яких встановлено python. Крім того, змінну sys.path можна змінювати вручну, що дозволяє покласти модуль в будь-який зручний для вас місце (головне, не забути модифікувати sys.path).