НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

«ХАРКІВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

Кафедра «ОБЧИСЛЮВАЛЬНА ТЕХНІКА ТА ПРОГРАМУВАННЯ»

Звiт з лабораторної роботи №6

Тема: «Функції»

Виконав:   
ст. гр. КІТ-118в  
Міщенко Д.С.

Перевірив:  
Поворознюк О.А.

Харків – 2021

***Мета:*** Ознайомлення та отримання практичних навичок у використанні функцій

***Загальне завдання***

*Завдання 1*

Розробіть функції для здійснення наступних операцій зі списками:

1. Швидке сортування;

2. Пошук елементу за значенням;

3. Пошук перших 𝑛 мінімальних елементів;

4. Пошук перших 𝑛 максимальних елементів;

5. Пошук середнього арифметичного;

6. Повернення списку, що сформований з початкового списку, але не містить

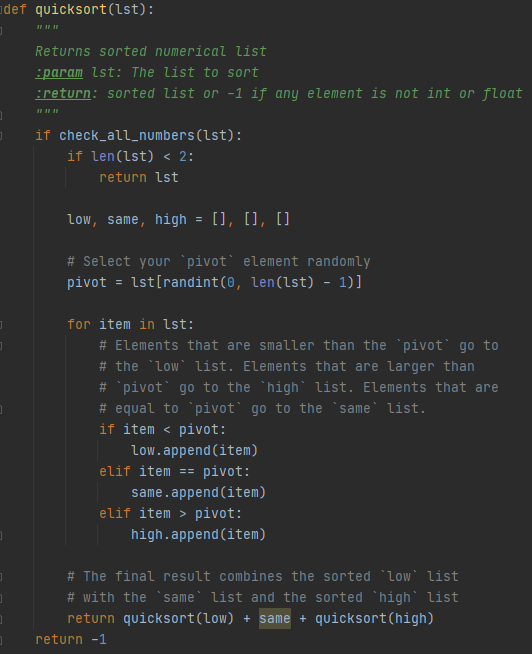
повторів (залишається лише перший з однакових елементів).

*Завдання 2*

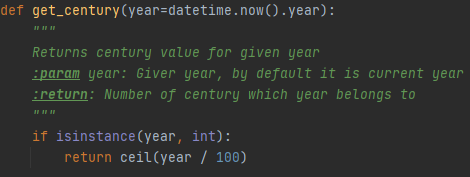
3. Визначити, до якого століття належить заданий рік

***Хід роботи***

***Фрагменти програми***

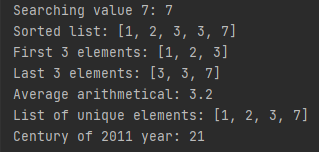
******

**Рисунок 1** ─ Сортування

******

**Рисунок 2** ─ Отримання століття з заданого року

***Результати роботи***



**Рисунок 3** ─ Отримання століття з заданого року

***Код програми***

**Numeric\_list\_operations.py**

from random import randint  
  
  
def quicksort(lst):  
 *"""  
 Returns sorted numerical list* ***:param*** *lst: The list to sort* ***:return****: sorted list or -1 if any element is not int or float  
 """* if check\_all\_numbers(lst):  
 if len(lst) < 2:  
 return lst  
  
 low, same, high = [], [], []  
  
 # Select your `pivot` element randomly  
 pivot = lst[randint(0, len(lst) - 1)]  
  
 for item in lst:  
 # Elements that are smaller than the `pivot` go to  
 # the `low` list. Elements that are larger than  
 # `pivot` go to the `high` list. Elements that are  
 # equal to `pivot` go to the `same` list.  
 if item < pivot:  
 low.append(item)  
 elif item == pivot:  
 same.append(item)  
 elif item > pivot:  
 high.append(item)  
  
 # The final result combines the sorted `low` list  
 # with the `same` list and the sorted `high` list  
 return quicksort(low) + same + quicksort(high)  
 return -1  
  
  
def search\_by\_value(lst, value):  
 *"""  
 Searches element in list by value* ***:param*** *lst: List in which searching is used* ***:param*** *value: value to search* ***:return****: value if is found or -1 if in isn`t found or list contains non-numeric values  
 """* if check\_all\_numbers(lst):  
 for i in lst:  
 if i == value:  
 return i  
 return -1  
  
  
def get\_n\_minimum\_elements(lst, n):  
 *"""  
 Gets first n elements with minimum value* ***:param*** *lst: List of values* ***:param*** *n: Count of elements to get* ***:return****: List of selected values of -1 if there is 1+ non-numeric value in list  
 """* if check\_all\_numbers(lst):  
 tmp = quicksort(lst)  
 return tmp[0:n]  
 return -1  
  
  
def get\_n\_maximum\_elements(lst, n):  
 *"""  
 Gets first n elements with maximum value* ***:param*** *lst: List of values* ***:param*** *n: Count of elements to get* ***:return****: List of selected values or -1 if there is 1+ non-numeric value in list  
 """* if check\_all\_numbers(lst):  
 tmp = quicksort(lst)  
 return tmp[-n:]  
 return -1  
  
  
def get\_avg\_arithmetical(lst):  
 *"""  
 Gets average arithmetical value from list of numeric values* ***:param*** *lst: List of values* ***:return****: Average arithmetical value or None value if there is 1+ non-numeric value in list  
 """* if check\_all\_numbers(lst):  
 return sum(lst) / len(lst)  
 return None  
  
  
def get\_unique\_list(lst):  
 *"""  
 Gets list of unique elements from given list* ***:param*** *lst: Given list* ***:return****: List of unique elements or -1 if there is 1+ non-numeric value in list  
 """* if check\_all\_numbers(lst):  
 return list(set(lst))  
 return -1  
  
  
def get\_unique\_numbers\_list(lst):  
 *"""  
 Gets list of unique elements from given list* ***:param*** *lst: Given list* ***:return****: List of unique elements or -1 if there is 1+ non-numeric value in list  
 """* unique = []  
 for number in lst:  
 if number not in unique:  
 unique.append(number)  
 return unique  
  
  
def check\_all\_numbers(lst):  
 *"""  
 Checks if every value in list is numeric* ***:param*** *lst: Given list* ***:return****: True if every value in list is numeric or False if there is 1+ non-numeric value in list  
 """* return all(isinstance(e, (int, float)) for e in lst)

**main.py**

from datetime import datetime  
from math import ceil  
  
from numeric\_list\_operations import search\_by\_value, quicksort, get\_n\_minimum\_elements, get\_n\_maximum\_elements, \  
 get\_avg\_arithmetical, get\_unique\_list  
  
  
def get\_century(year=datetime.now().year):  
 *"""  
 Returns century value for given year* ***:param*** *year: Giver year, by default it is current year* ***:return****: Number of century which year belongs to  
 """* if isinstance(year, int):  
 return ceil(year / 100)  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 l = [1, 7, 2, 3, 3]  
 print(f"Searching value 7: {search\_by\_value(l, 7)}")  
 print(f"Sorted list: {quicksort(l)}")  
 print(f"First 3 elements: {get\_n\_minimum\_elements(l, 3)}")  
 print(f"Last 3 elements: {get\_n\_maximum\_elements(l, 3)}")  
 print(f"Average arithmetical: {get\_avg\_arithmetical(l)}")  
 print(f"List of unique elements: {get\_unique\_list(l)}")  
 print(f"Century of 2011 year: {get\_century(2011)}")

**Висновки:** в ході виконання лабораторної роботи були набуті практичні навички з роботи з функціями в мові Python.