Івано-Франківський національний технічний університет нафти і газу

Кафедра комп’ютерних систем і мереж

**ЛАБОРАТОРНА РОБОТА №6**

з дисципліни Практикум Python

Тема: Модулі в Python. Механізми обробки винятків

Виконав:

Ст. гр. КІ-23-2 (Боднар Р.В.)

Перевірив: (Кропивницький Д. Р.)

м. Івано-Франківськ

2025

**Мета роботи:** Познайомитися з принципами створення та імпорту модулів на мові Python, з механізмом обробки виняткових ситуацій.

**Варіант 1**

**Завдання:**

Завдання 1.

Файл models.py містить програмний код, поданий нижче, що імітує друкування 3D-моделей різних об’єктів. Перенесіть функції print\_models() і show\_completed\_models() у окремий файл з ім’ям printing\_functions.py. Виконайте імпорт цих функцій у файл models.py, змінивши файл так, щоб у ньому імпортовані функції можна було використовувати.

Завдання 2.

Розробіть функції для здійснення наступних операцій зі списками:

1. 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).

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

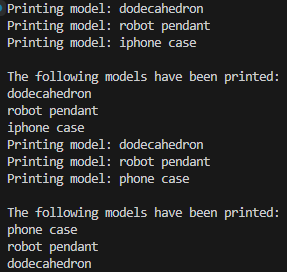
**Хід роботи**

**Код програми до завдання 1:**

import printing\_functions as pf

pf.print\_models(['phone case', 'robot pendant', 'dodecahedron'], [])

pf.show\_completed\_models(['phone case', 'robot pendant', 'dodecahedron'])

****

**Рисунок 1 - Результат виконання завдання 1**

**Код програми до завдання 2:**

import functions as f

def main():

    lst = [5, 3, 7, 2, 5, 0, 3, 8, 1]

    print("Original list:", lst)

    print("Sorted list:", f.sort\_list(lst))

    print("Find 7:", f.find(7, lst))

    print("Find sequence [2, 5, 0]:", f.find\_sequence\_in\_list(lst, [2, 5, 0]))

    print("First five minimal:", f.first\_five\_minimal(lst))

    print("First five maximal:", f.first\_five\_maximal(lst))

    print("Average value:", f.average\_value(lst))

    print("List without duplicates:", f.list\_without\_duplicates(lst))

if \_\_name\_\_ == "\_\_main\_\_":

    main()

"""

Module list\_utils

Contains utility functions for working with lists.

"""

def sort\_list(lst):

    """

    Sort a list in ascending order.

    Args:

        lst (list): List of elements.

    Returns:

        list: Sorted list.

    """

    return sorted(lst)

def find(item, lst):

    """

    Check if an item exists in a list.

    Args:

        item: Element to find.

        lst (list): List to search in.

    Returns:

        bool: True if item is found, False otherwise.

    """

    return item in lst

def find\_sequence\_in\_list(lst, subseq):

    """

    Find a subsequence in a list.

    Args:

        lst (list): Source list.

        subseq (list): Subsequence to find.

    Returns:

        int or str: Starting index of subsequence if found,

                    otherwise "Sequence not found".

    """

    for i in range(len(lst) - len(subseq) + 1):

        if lst[i:i+len(subseq)] == subseq:

            return i

    return "Sequence not found"

def first\_five\_minimal(lst):

    """

    Get the first five minimal elements from the list.

    Args:

        lst (list): List of elements.

    Returns:

        list: List of five minimal elements (or fewer if list is shorter).

    """

    return sorted(lst)[:5]

def first\_five\_maximal(lst):

    """

    Get the first five maximal elements from the list.

    Args:

        lst (list): List of elements.

    Returns:

        list: List of five maximal elements (or fewer if list is shorter).

    """

    return sorted(lst, reverse=True)[:5]

def average\_value(lst):

    """

    Calculate the average value of a list.

    Args:

        lst (list): List of numbers.

    Returns:

        float: Average value, or 0 if list is empty.

    """

    return sum(lst) / len(lst) if lst else 0

def list\_without\_duplicates(lst):

    """

    Remove duplicates from a list while preserving the first occurrence.

    Args:

        lst (list): List with possible duplicates.

    Returns:

        list: List without duplicates (order preserved).

    """

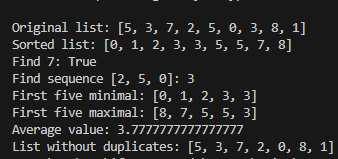
    result = []

    for x in lst:

        if x not in result:

            result.append(x)

    return result

****

**Рисунок 2 - Результат виконання завдання 2**

**Висновок:** У роботі реалізовано програму з використанням власних функцій в окремому модулі, а також написано опис функцій.