Министерство Просвещения, Культуры и Исследований Республики Молдова

Технический университет Молдовы

Факультет Вычислительной Техники, Информатики и Микроэлектроники

Департамент Микроэлектроники и Биоинженерии

**Отчет**

**По предмету Programarea Declarativă**

**По лабораторной работе №5**

**Тема:** «Разработка собственного модуля Python»

Выполнил Cuțar Alexandr (TI-218)

Проверил A.Leah

Кишинев 2023

**Цель**

Целью этой лабораторной работы является разработка собственного модуля Python, который предоставит полезные функциональности для вашего проекта.

**Задачи**

1. Создайте новый проект Python и настройте его среду разработки (например, используя Visual Studio Code или PyCharm).
2. Создайте новый модуль Python в вашем проекте. Модуль должен предоставлять как минимум две функции, которые могли бы быть полезны для вашего проекта. Например, вы можете создать модуль для работы с датами, файлами, или любой другой функциональностью, которая связана с проектом.
3. Напишите документацию для вашего модуля, используя формат docstring. Документация должна включать в себя описания функций, их параметров и возвращаемых значений.

**Код date\_utils.py**

from datetime import datetime, timedelta  
  
def is\_leap\_year(year):  
 *"""  
 Проверяет, является ли год високосным.  
  
 Параметры:  
 year (int): Год, который нужно проверить.  
  
 Возвращает:  
 bool: True, если год високосный, иначе False.  
 """* return year % 4 == 0 and (year % 100 != 0 or year % 400 == 0)  
  
def days\_between\_dates(date1, date2):  
 *"""  
 Рассчитывает разницу в днях между двумя датами.  
  
 Параметры:  
 date1 (str): Первая дата в формате 'YYYY-MM-DD'.  
 date2 (str): Вторая дата в формате 'YYYY-MM-DD'.  
  
 Возвращает:  
 int: Количество дней между датами. Если первая дата идет после второй,  
 возвращается отрицательное число.  
 """* d1 = datetime.strptime(date1, "%Y-%m-%d")  
 d2 = datetime.strptime(date2, "%Y-%m-%d")  
 delta = d2 - d1  
 return delta.days  
  
def get\_current\_date(format="%Y-%m-%d"):  
 *"""  
 Возвращает текущую дату в заданном формате.  
  
 Параметры:  
 format (str): Строка формата, в котором должна быть возвращена текущая дата.  
  
 Возвращает:  
 str: Текущая дата в заданном формате.  
 """* return datetime.now().strftime(format)  
  
def add\_days\_to\_date(date\_str, days, date\_format="%Y-%m-%d"):  
 *"""  
 Добавляет заданное количество дней к дате.  
  
 Параметры:  
 date\_str (str): Дата, к которой нужно добавить дни, в виде строки.  
 days (int): Количество дней для добавления.  
 date\_format (str): Формат даты входной строки.  
  
 Возвращает:  
 str: Новая дата после добавления указанного количества дней.  
 """* date = datetime.strptime(date\_str, date\_format)  
 new\_date = date + timedelta(days=days)  
 return new\_date.strftime(date\_format)  
  
def has\_date\_passed(target\_date, date\_format="%Y-%m-%d"):  
 *"""  
 Проверяет, наступила ли уже заданная дата по сравнению с текущей датой.  
  
 Параметры:  
 target\_date (str): Дата для проверки.  
 date\_format (str): Формат даты входной строки.  
  
 Возвращает:  
 bool: True если заданная дата уже наступила, иначе False.  
 """* current\_date = datetime.now()  
 check\_date = datetime.strptime(target\_date, date\_format)  
 return current\_date >= check\_date