**національний університет БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ україни**

**ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ**

Кафедра комп’ютерних наук

**ЛАБОРАТОРНА РОБОТА №3**

Виконав:

студент групи ІПЗ-21007б

Щур В. М.

Київ – 2024

**Хід виконання**

***1.*** *Створити віртуальне оточення (ім'я оточення - прізвище студента). В цьому оточенні створити проект Python.*

*Зображення, що містить текст, знімок екрана, Шрифт, програмне забезпечення

Автоматично згенерований опис*

***2.*** *Створити пакет, який складається з двох модулів.*

*Зображення, що містить текст, знімок екрана, програмне забезпечення, Шрифт

Автоматично згенерований опис*

***3.*** *В файлі \_\_init\_\_.py задати наступні змінні:*

*Зображення, що містить текст, знімок екрана, Шрифт, годинник

Автоматично згенерований опис*

***4.*** *В першому модулі створити наступні функції, використав інструментарій пакету googletrans:*

* def TransLate(text : str, scr : str, dest : str) -> str
* def LangDetect(text : str, set : str) -> str
* def CodeLang(lang : str) -> str
* def LanguageList(out : str, text : str) -> str

from googletrans import Translator, LANGUAGES  
  
translator = Translator()  
  
def TransLate(text: str, src: str, dest: str) -> str:  
 try:  
 translation = translator.translate(text, src=src, dest=dest)  
 return translation.text  
 except Exception as e:  
 return f"Помилка: {e}"  
  
def LangDetect(text: str, set: str = "all") -> str:  
 try:  
 detection = translator.detect(text)  
 if set == "lang":  
 return detection.lang  
 elif set == "confidence":  
 return str(detection.confidence)  
 else:  
 return f"Detected(lang={detection.lang}, confidence={detection.confidence})"  
 except Exception as e:  
 return f"Помилка: {e}"  
  
def CodeLang(lang: str) -> str:  
 lang = lang.lower()  
 if lang in LANGUAGES:  
 return LANGUAGES[lang].capitalize()  
 else:  
 for code, name in LANGUAGES.items():  
 if name.lower() == lang:  
 return code  
 return "Помилка - не знайдено відповідного коду або назви мови!"  
  
def LanguageList(out: str = "screen", text: str = None) -> str:  
 try:  
 languages = sorted(LANGUAGES.items(), key=lambda x: x[1])  
 table = "N Language ISO-639 code Text\n"  
 table += "-" \* 50 + "\n"  
 for i, (code, name) in enumerate(languages, 1):  
 translated\_text = text if text is None else TransLate(text, 'auto', code)  
 table += f"{i} {name.capitalize()} {code} {translated\_text}\n"  
  
 if out == "screen":  
 print(table)  
 elif out == "file":  
 with open("languages.txt", "w", encoding="utf-8") as file:  
 file.write(table)  
 else:  
 return "Помилка - неправильний параметр 'out'!"  
  
 return "Ok"  
 except Exception as e:  
 return f"Помилка: {e}"

***5.*** *В другому модулі створити вище наведені функції, використав інструментарій пакету deep\_translator. Для визначення мови можна використати пакет langdetect.*

from deep\_translator import GoogleTranslator  
from langdetect import detect, DetectorFactory  
from langdetect.lang\_detect\_exception import LangDetectException  
  
DetectorFactory.seed = 0  
  
def TransLate(text: str, src: str, dest: str) -> str:  
 try:  
 translator = GoogleTranslator(source=src, target=dest)  
 translation = translator.translate(text)  
 return translation  
 except Exception as e:  
 return f"Помилка: {e}"  
  
def LangDetect(text: str, set: str = "all") -> str:  
 try:  
 lang = detect(text)  
 if set == "lang":  
 return lang  
 elif set == "confidence":  
 return "N/A"  
 else:  
 return f"Detected(lang={lang}, confidence=N/A)"  
 except LangDetectException as e:  
 return f"Помилка: {e}"  
  
def CodeLang(lang: str) -> str:  
 lang = lang.lower()  
 try:  
 language\_dict = {  
 'af': 'afrikaans', 'sq': 'albanian', 'ar': 'arabic', 'hy': 'armenian',  
 'ca': 'catalan', 'hr': 'croatian', 'cs': 'czech', 'da': 'danish',  
 'nl': 'dutch', 'en': 'english', 'eo': 'esperanto', 'et': 'estonian',  
 'fi': 'finnish', 'fr': 'french', 'de': 'german', 'el': 'greek',  
 'gu': 'gujarati', 'ht': 'haitian creole', 'ha': 'hausa', 'he': 'hebrew',  
 'hi': 'hindi', 'hu': 'hungarian', 'is': 'icelandic', 'id': 'indonesian',  
 'it': 'italian', 'ja': 'japanese', 'jw': 'javanese', 'kn': 'kannada',  
 'kk': 'kazakh', 'km': 'khmer', 'ko': 'korean', 'la': 'latin',  
 'lv': 'latvian', 'lt': 'lithuanian', 'lb': 'luxembourgish', 'mk': 'macedonian',  
 'ml': 'malayalam', 'mn': 'mongolian', 'my': 'myanmar', 'ne': 'nepali',  
 'no': 'norwegian', 'or': 'odia', 'ps': 'pashto', 'pl': 'polish',  
 'pt': 'portuguese', 'pa': 'punjabi', 'ro': 'romanian', 'ru': 'russian',  
 'sm': 'samoan', 'gd': 'scots gaelic', 'sr': 'serbian', 'st': 'sesotho',  
 'sn': 'shona', 'sd': 'sindhi', 'si': 'sinhala', 'sk': 'slovak',  
 'sl': 'slovenian', 'so': 'somali', 'es': 'spanish', 'su': 'sundanese',  
 'sw': 'swahili', 'sv': 'swedish', 'tl': 'tagalog', 'ta': 'tamil',  
 'te': 'telugu', 'th': 'thai', 'tr': 'turkish', 'uk': 'ukrainian',  
 'ur': 'urdu', 'vi': 'vietnamese', 'cy': 'welsh', 'xh': 'xhosa',  
 'yi': 'yiddish', 'yo': 'yoruba', 'zu': 'zulu'  
 }  
 lang = lang.lower()  
 if lang in language\_dict.values():  
 return [code for code, name in language\_dict.items() if name == lang][0]  
 elif lang in language\_dict:  
 return language\_dict[lang].capitalize()  
 else:  
 return "Помилка - не знайдено відповідного коду або назви мови!"  
 except Exception as e:  
 return f"Помилка: {e}"  
  
def LanguageList(out: str = "screen", text: str = None) -> str:  
 try:  
 language\_dict = {  
 'af': 'afrikaans', 'sq': 'albanian', 'ar': 'arabic', 'hy': 'armenian',  
 'ca': 'catalan', 'hr': 'croatian', 'cs': 'czech', 'da': 'danish',  
 'nl': 'dutch', 'en': 'english', 'eo': 'esperanto', 'et': 'estonian',  
 'fi': 'finnish', 'fr': 'french', 'de': 'german', 'el': 'greek',  
 'gu': 'gujarati', 'ht': 'haitian creole', 'ha': 'hausa', 'he': 'hebrew',  
 'hi': 'hindi', 'hu': 'hungarian', 'is': 'icelandic', 'id': 'indonesian',  
 'it': 'italian', 'ja': 'japanese', 'jw': 'javanese', 'kn': 'kannada',  
 'kk': 'kazakh', 'km': 'khmer', 'ko': 'korean', 'la': 'latin',  
 'lv': 'latvian', 'lt': 'lithuanian', 'lb': 'luxembourgish', 'mk': 'macedonian',  
 'ml': 'malayalam', 'mn': 'mongolian', 'my': 'myanmar', 'ne': 'nepali',  
 'no': 'norwegian', 'or': 'odia', 'ps': 'pashto', 'pl': 'polish',  
 'pt': 'portuguese', 'pa': 'punjabi', 'ro': 'romanian', 'ru': 'russian',  
 'sm': 'samoan', 'gd': 'scots gaelic', 'sr': 'serbian', 'st': 'sesotho',  
 'sn': 'shona', 'sd': 'sindhi', 'si': 'sinhala', 'sk': 'slovak',  
 'sl': 'slovenian', 'so': 'somali', 'es': 'spanish', 'su': 'sundanese',  
 'sw': 'swahili', 'sv': 'swedish', 'tl': 'tagalog', 'ta': 'tamil',  
 'te': 'telugu', 'th': 'thai', 'tr': 'turkish', 'uk': 'ukrainian',  
 'ur': 'urdu', 'vi': 'vietnamese', 'cy': 'welsh', 'xh': 'xhosa',  
 'yi': 'yiddish', 'yo': 'yoruba', 'zu': 'zulu'  
 }  
 table = "N Language ISO-639 code Text\n"  
 table += "-" \* 50 + "\n"  
 for i, (code, name) in enumerate(language\_dict.items(), 1):  
 translated\_text = text if text is None else TransLate(text, 'auto', code)  
 table += f"{i} {name.capitalize()} {code} {translated\_text}\n"  
  
 if out == "screen":  
 print(table)  
 elif out == "file":  
 with open("languages.txt", "w", encoding="utf-8") as file:  
 file.write(table)  
 else:  
 return "Помилка- неправильний параметр 'out'!"  
  
 return "Ok"  
 except Exception as e:  
 return f"Помилка: {e}"

***6.*** *В кореневому каталозі проекту створити три файли: gtrans.py, deeptr.py, filetr.py. В файлі gtrans.py написати програму, яка демонструє роботу функцій пакету з першого модуля.*

*Зображення, що містить текст, знімок екрана, програмне забезпечення, Шрифт

Автоматично згенерований опис*

**Код gtrans:**

from package.module1 import TransLate, LangDetect, CodeLang, LanguageList  
  
def translation\_program():  
 input\_text = "Hello, world!"  
 source\_language = "en"  
 target\_language = "es"  
 translated\_result = TransLate(input\_text, source\_language, target\_language)  
 print(f"Translated text from '{source\_language}' to '{target\_language}': {translated\_result}")  
  
 text\_for\_detection = "Bonjour tout le monde!"  
 identified\_language = LangDetect(text\_for\_detection, set="lang")  
 print(f"Detected language: {identified\_language}")  
  
 language\_identifier = CodeLang("French")  
 print(f"Language code for 'French': {language\_identifier}")  
  
 language\_output = LanguageList(out="screen", text="Good morning")  
 print(f"Language list output: {language\_output}")  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 translation\_program()

**Результат:**

**Зображення, що містить текст, знімок екрана, Шрифт

Автоматично згенерований опис**

***7.*** *В файлі deeptr.py написати програму, яка демонструє роботу функцій пакету з другого модуля.*

**Код deeptr:**

from package.module2 import TransLate, LangDetect, CodeLang, LanguageList  
  
def language\_tests():  
 input\_text = "Hello, world!"  
 source\_lang = "en"  
 target\_lang = "fr"  
 translated\_output = TransLate(input\_text, source\_lang, target\_lang)  
 print(f"Translated text from '{source\_lang}' to '{target\_lang}': {translated\_output}")  
  
 detection\_text = "Hola, ¿cómo estás?"  
 language\_and\_confidence = LangDetect(detection\_text, set="all")  
 print(f"Detected language and confidence: {language\_and\_confidence}")  
  
 lang\_name = CodeLang("es")  
 print(f"Language name for code 'es': {lang\_name}")  
  
 language\_list\_result = LanguageList(out="screen", text="Good evening")  
 print(f"Language list output: {language\_list\_result}")  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 language\_tests()

**Результат:**

**Зображення, що містить текст, знімок екрана, Шрифт, дизайн

Автоматично згенерований опис**

***8.*** *В файлі filetr.py написати програму для перекладу тексту з файлу. Попередньо створив файл з текстом і конфігураційний файл. Всі файли повинні знаходитись в кореневому каталозі проекту.*

*Зображення, що містить текст, програмне забезпечення, Мультимедійне програмне забезпечення, ряд

Автоматично згенерований опис*

*Зображення, що містить текст, знімок екрана, монітор, програмне забезпечення

Автоматично згенерований опис*

**Код filetr:**

import os  
import json  
import re  
from package.module1 import TransLate, LangDetect  
from langdetect import detect, DetectorFactory  
  
DetectorFactory.seed = 0  
  
def load\_config(config\_path):  
 try:  
 with open(config\_path, 'r') as config\_file:  
 settings = json.load(config\_file)  
 return settings  
 except Exception as error:  
 return f"Error reading config file: {error}"  
  
def evaluate\_text(text\_content):  
 character\_count = len(text\_content)  
 word\_count = len(re.findall(r'\w+', text\_content))  
 sentence\_count = len(re.findall(r'[.!?]', text\_content)) + text\_content.count('\n')  
  
 return character\_count, word\_count, sentence\_count  
  
def text\_analysis():  
 config\_settings = load\_config('config.json')  
 if isinstance(config\_settings, str):  
 print(config\_settings)  
 return  
  
 file\_path = config\_settings.get('text\_file')  
 destination\_language = config\_settings.get('target\_language')  
 display\_mode = config\_settings.get('output\_mode')  
 max\_chars = config\_settings.get('char\_limit')  
 max\_words = config\_settings.get('word\_limit')  
 max\_sentences = config\_settings.get('sentence\_limit')  
  
 if not os.path.isfile(file\_path):  
 print(f"File {file\_path} not found.")  
 return  
  
 try:  
 with open(file\_path, 'r') as text\_file:  
 text\_content = text\_file.read()  
  
 character\_count, word\_count, sentence\_count = evaluate\_text(text\_content)  
  
 print(f"File name: {file\_path}")  
 print(f"File size: {os.path.getsize(file\_path)} bytes")  
 print(f"Number of characters: {character\_count}")  
 print(f"Number of words: {word\_count}")  
 print(f"Number of sentences: {sentence\_count}")  
  
 detected\_language = detect(text\_content)  
 print(f"Detected language: {detected\_language}")  
  
 # Зчитування тексту до виконання умов  
 if (character\_count > max\_chars or word\_count > max\_words or sentence\_count > max\_sentences):  
 text\_content = text\_content[:max\_chars]  
  
 translated\_output = TransLate(text\_content, detected\_language, destination\_language)  
  
 if display\_mode == 'screen':  
 print(f"Translated text into '{destination\_language}':")  
 print(translated\_output)  
 elif display\_mode == 'file':  
 output\_filename = f"{os.path.splitext(file\_path)[0]}\_{destination\_language}.txt"  
 try:  
 with open(output\_filename, 'w') as output\_file:  
 output\_file.write(translated\_output)  
 print("Translation saved successfully!")  
 except Exception as error:  
 print(f"Error writing to file: {error}")  
 else:  
 print("Invalid output mode specified!")  
  
 except Exception as error:  
 print(f"Error: {error}")  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 text\_analysis()

**Результат:**

**Зображення, що містить текст, знімок екрана, Шрифт

Автоматично згенерований опис**

***9.*** *Створити файл requirments.txt, в якому зберігаються всі встановлені модулі та пакети.*

*Зображення, що містить текст, електроніка, знімок екрана, програмне забезпечення

Автоматично згенерований опис*

***10.*** *Завантажити проект на сервіс GitHub.*