Міністерство освіти і науки України

Національний університет «Запорізька Політехніка»

Кафедра програмних засобів

ЗВІТ

з лабораторної роботи №2

з дисципліни «Системний аналіз» на тему:

«Використання функціонального програмування для побудови програмної системи»

Виконав

Студент групи КНТ-122 О. А. Онищенко

Прийняли

Викладач Л. Ю. Дейнега

2024

ВИКОРИСТАННЯ ФУНКЦІОНАЛЬНОГО ПРОГРАМУВАННЯ ДЛЯ ПОБУДОВИ ПРОГРАМНОЇ СИСТЕМИ

Мета роботи

Ознайомитися з основними особливостями функціональної парадигми програмування та особливостями її реалізації в мові програмування Python.

Навчитися розробляти програми мовою програмування Python на основі використання парадигми функціонального програмування.

Завдання

- Вивести всі коректні комбінації пар круглих дужок, які можна сформувати з n дужок, що закриваються і відкриваються. Наприклад, коректна комбінація (()()), некоректна ())((. Кількість дужок задається користувачем

- Визначити кількість слів у тексті, що зберігається у файлі, та довжину найкоротшого слова. Слова відділяються одне від одного не тільки пробілами, але й будь-якими знаками пунктуації

Код програми

```python
# main.py

import inquirer
from rich.console import Console
from rich.traceback import install

import one as taskOne
import two as taskTwo
```

```
install()
console = Console()


def main() -> None:
    availableTasks = [
        "First - Bracket Pairs",
        "Second - Words Counter",
    ]
    selectedTask = inquirer.prompt(
        [
            inquirer.List(
                "task",
                message="Which task would you like to look at?",
                choices=availableTasks,
            )
        ]
    )["task"]

    if selectedTask == availableTasks[0]:
        taskOne.main()
    elif selectedTask == availableTasks[1]:
        taskTwo.main()


if __name__ == "__main__":
    main()
```

```
# one.py

import inquirer
from rich.console import Console
from rich.traceback import install

install()
console = Console()


def functional(n: int) -> None:
    def helper(openCount: int, closeCount: int, currentCombination:
list[str]) -> None:
        if openCount == closeCount == n:
            console.print("".join(currentCombination))
            return

        if openCount < n:
            currentCombination.append("(")
            helper(openCount + 1, closeCount, currentCombination)
```

```
            currentCombination.pop()

        if closeCount < openCount:
            currentCombination.append(")")
            helper(openCount, closeCount + 1, currentCombination)
            currentCombination.pop()

    helper(0, 0, [])


def imperative(n: int) -> None:
    stack: list[tuple[str, int, int]] = []
    stack.append(("", 0, 0))

    while stack:
        currentCombination, openCount, closeCount = stack.pop()

        if openCount == closeCount == n:
            console.print(currentCombination)
            continue

        if openCount < n:
            stack.append((currentCombination + "(", openCount + 1,
closeCount))

        if closeCount < openCount:
            stack.append((currentCombination + ")", openCount, closeCount +
1))


def main() -> None:
    options = [
        "Functional Programming",
        "Imperative Programming",
    ]
    choice = inquirer.prompt(
        [
            inquirer.List(
                "solution",
                message="Which solution would you like to use?",
                choices=options,
            )
        ]
    )["solution"]

    bracketsNumber = inquirer.prompt(
        [
            inquirer.Text(
                "number of brackets",
```

```
                message="How many brackets would you like to use?",
                validate=lambda _, x: x != "" and x.isdigit() and int(x) > 0,
            )
        ]
    )["number of brackets"]
    bracketsNumber = int(bracketsNumber)

    console.print(f"\nAll possible bracket pairs for {bracketsNumber} bracket
pairs:")
    if choice == options[0]:
        functional(n=bracketsNumber)
    elif choice == options[1]:
        imperative(n=bracketsNumber)


if __name__ == "__main__":
    main()
```

```
# two.py

import inquirer
from os import path
from functools import reduce
from rich.console import Console
from rich.traceback import install

install()
console = Console()


def functional(filePath: str) -> tuple[int, int]:
    with open(filePath, "r", encoding="utf-8") as file:
        text = file.read()

    punctuation = '!@#$%^&*()_+{}|:"<>?`~'
    table = str.maketrans("", "", punctuation)

    words = text.translate(table).split()
    wordsCount = len(words)

    shortestWordLength = reduce(
        lambda shortestSoFar, word: min(shortestSoFar, len(word)), words,
float("inf")
    )

    return wordsCount, shortestWordLength
```

```python
def main() -> None:
    currentDir: str = path.dirname(path.abspath(__file__))
    filePath: str = path.join(currentDir, "..", "data", "input.txt")

    choices = [
        "Use provided file",
        "Enter path for my own",
    ]
    decision = inquirer.prompt(
        [
            inquirer.List(
                "choose",
                message="Which path would you like to use?",
                choices=choices,
            )
        ]
    )["choose"]

    if decision == choices[1]:
        filePath = inquirer.prompt(
            [
                inquirer.Text(
                    "path",
                    message="Enter full path to your file",
                    validate=lambda _, x: path.exists(x) and path.isfile(x),
                )
            ]
        )["path"]
        console.print()

    wordsCount, shortestWordLength = functional(filePath)
    console.print(f"Number of words: {wordsCount}")
    console.print(f"Length of shortest word: {shortestWordLength}")


if __name__ == "__main__":
    main()
```

Результати виконання

Перше завдання

Після виконання коду до першого завдання отримуємо наступні результати:

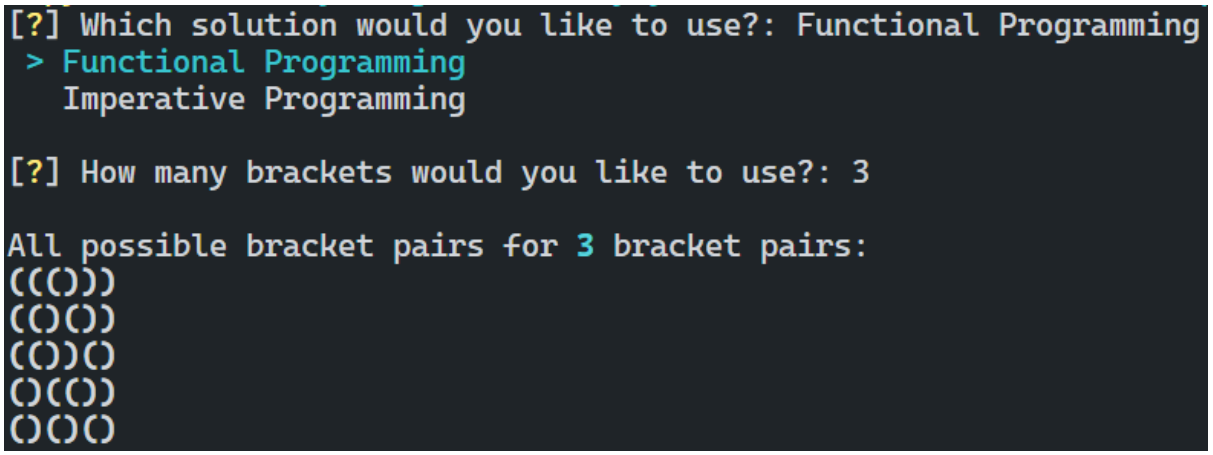

Рисунок 1.1 – Результати виконання першого завдання

Друге завдання

Під час виконання другого завдання як вхідний використався файл наступного змісту:

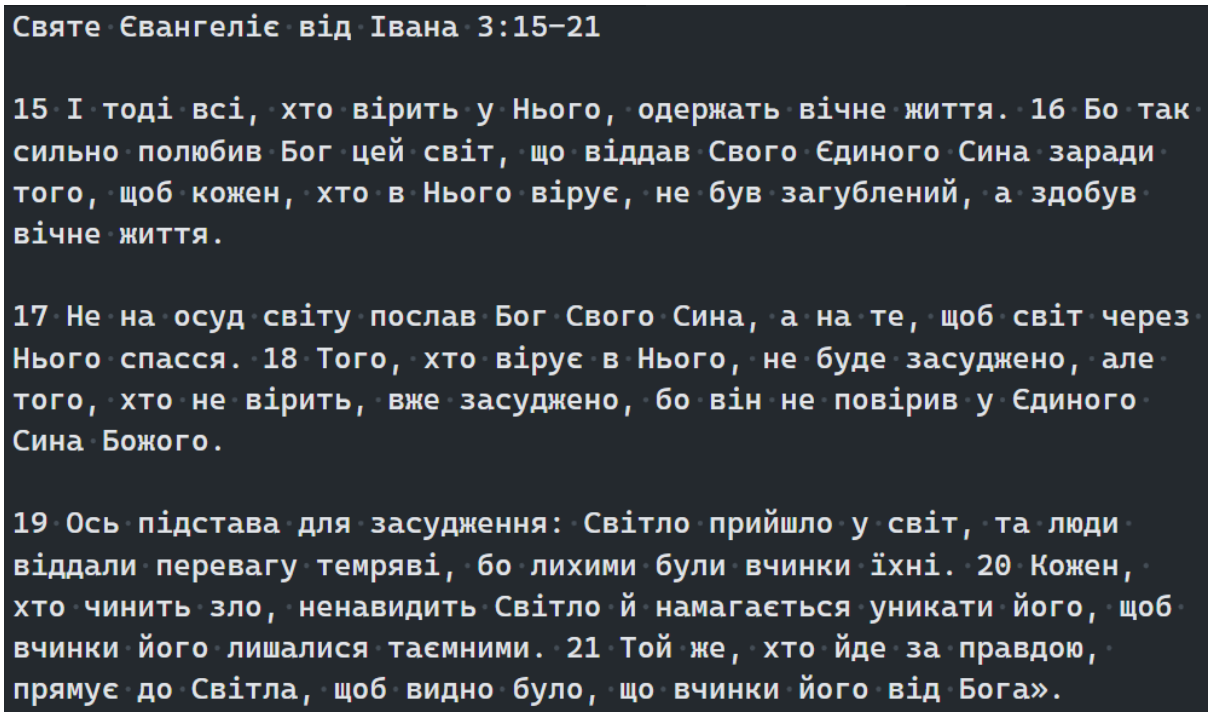

Рисунок 1.2 – Зміст файлу вхідних даних для другого завдання

Після виконання коду до другого завдання маємо наступний результат:

```
[?] Which path would you like to use?: Use provided file
 > Use provided file
   Enter path for my own

Number of words: 138
Length of shortest word: 1
```

Рисунок 1.3 – Результати виконання другого завдання

Висновки

Таким чином, ми ознайомилися з основними особливостями функціональної парадигми програмування та особливостями її реалізації в мові програмування Python; а також навчилися розробляти програми мовою програмування Python на основі використання парадигми функціонального програмування.

Контрольні питання

Яким чином виконати форматування рядка?

Форматувати рядок у Python можна за допомогою методу format() або f-рядків (форматованих рядкових літералів). Наприклад, з допомогою f-рядка:

```
ймення = "Микола"
вік = 30
форматований_рядок = f"Мене звати {ймення} і мені {вік} років"
```

Які дії можна виконувати над переліками?

Списки в Python є змінюваними, що означає, що над ними можна виконувати різні дії, такі як:

- Додавання елементів (append(), extend(), insert())
- Видалення елементів (remove(), pop(), clear())
- Пошук елементів (index(), ключове слово in)
- Підрахунок елементів (count())
- Сортування (sort())
- Реверсування (reverse())

Що таке множина та яким чином її визначити?

Множина - це невпорядкована колекція унікальних елементів. Визначити множину в Python можна за допомогою фігурних дужок {} або функції set(). Наприклад:

```
довільна_множина = {1, 2, 3}
# або з допомогою функції set()
довільна_множина = set([1, 2, 3])
```