МИНОБРНАУКИ РОССИИ
Федеральное государственное автономное образовательное
учреждение высшего образования
«Санкт-Петербургский политехнический университет Петра Великого»
(ФГАОУ ВО «СПбПУ»)

Физико-механический институт
Кафедра «Прикладная математика»

Отчет по лабораторной работе «Кодер-декодер»
Вариант 4 (ROT1)

Студент: Ироносов Артемий Вячеславович
Преподаватель: Козлов Константин Николаевич

Группа: 5030102/10401

Санкт-Петербург 2024

Формулировка задания

В рамках лабораторной работы **нужно реализовать** предоставленный интерфейс:

- При реализации нужно создать собственный Exception (дописать класс ConfigException), который будет выбрасываться при некорректных данных в конфигурационном файле.

- Реализация функции read_file FileReaderInterface должна работать с менеджером контекста.

- При реализации ConfigReaderInterface функция read_config в) должна записывать данные в словарь, и при ошибках в конфигурационном файле должна выбрасывать исключение с описанием ошибки, ошибка должна логироваться.

- CoderInterface должен содержать реализацию кодировщика вашего варианта.

- MainClass. Изменить параметры в конструкторе на собственные реализации классов.

Параметры конфигурационного файла:

1) Размер буфера считывания (используется как аргумент в read_file).

2) Режим работы кодировщика (code или decode используется как аргумент функии run CoderInterface).

3) Путь к файлу, с которым будет работать алгоритм.

Структура программы

- ConfigException

Класс ConfigException наследуется от класса Exception и в случае возникновения исключения передает сообщение об ошибке.

```python
class ConfigException(Exception):
    def __init__(self, *args):
        if args:
            self.message = str(args[0])
        else:
            self.message = None

    def __str__(self):
        if self.message:
            return "ConfigException\n" + self.message
        else:
            return "ConfigException"
```

ConfigException вызывается в случаях:

1. Неверно передан режим работы кодировщика
2. В конфиге указаны не все параметры
3. В конфиге указаны неизвестные параметры

- ConfigReader

В классе ConfigReader есть метод read_config, который принимает имя файла и возвращает считанные из него параметры в виде словаря

```python
class ConfigReader(ConfigReaderInterface):                                    ⚠ 2
    1 usage
    def read_config(self, config_file_name: str) -> dict:
        keys = {self.buffer_size_param_name, self.file_name_for_coder_param_name, self.coder_run_option_param_name}
        config_dict = {}

        with open(config_file_name, "r") as file:
            lines = file.readlines()
            for line in lines:
                key, value = line.split(" " + self._param_delimiter + " ")
                value = re.sub( pattern: r"\n",  repl: "", value)
                if key in keys:
                    keys.remove(key)
                    if key == self.coder_run_option_param_name and not (value == "code" or value == "decode"):
                        raise ConfigException("Coder_option must be 'code' or 'decode'!")
                    elif key == self.buffer_size_param_name:
                        value = int(value)
                    config_dict[key] = value

                else:
                    raise ConfigException("Unknown parameter: {}".format(key))

            if len(keys) > 0:
                raise ConfigException("Missed parameters: " + "".join(str(i) + " " for i in keys))

        return config_dict
```

- FileReader

В классе FileReader есть метод read_file, который обернут в декоратор @contextmanager, он принимает имя файла и размер буфера, возвращает список chunks с фрагментами файла заданного размера.

```
class FileReader(FileReaderInterface):
    1 usage
    @contextmanager
    def read_file(self, file_name: str, buffer_size: int):
        try:
            file = open(file_name, "r")
            chunks = []
            chunk = file.read(buffer_size)
            while len(chunk) > 0:
                chunks.append(chunk)
                chunk = file.read(buffer_size)
            yield chunks
        except FileNotFoundError:
            print("File not found")
        finally:
            file.close()
```

- Coder

Данный класс содержит 3 метода:

1. _code – каждая буква в сообщении заменяется на букву, находящуюся в алфавите на одну позицию правее
2. _decode – каждая буква в сообщении заменяется на букву, находящуюся в алфавите на одну позицию левее
3. run – в зависимости от аргумента coder_info вызывает либо _code, либо _decode

```python
class Coder(CoderInterface):
    1 usage
    def run(self, coder_info: str, string_to_process: str) -> str:
        if coder_info == "code":
            return self._code(string_to_process)
        else:
            return self._decode(string_to_process)

    1 usage
    def _code(self, string_to_code: str) -> str:
        encoded_string = ""
        for char in string_to_code:
            if char.isalpha():
                shifted_char = chr(((ord(char) - 65 + 1) % 26) + 65) if char.isupper() else chr(
                    ((ord(char) - 97 + 1) % 26) + 97)
                encoded_string += shifted_char
            else:
                encoded_string += char
        return encoded_string

    1 usage
    def _decode(self, string_to_code: str) -> str:
        decoded_string = ""
        for char in string_to_code:
            if char.isalpha():
                shifted_char = chr(((ord(char) - 65 - 1) % 26) + 65) if char.isupper() else chr(
                    ((ord(char) - 97 - 1) % 26) + 97)
                decoded_string += shifted_char
            else:
                decoded_string += char
        return decoded_string
```

Как запускать

Чтобы запустить программу, нужно написать в командной строке (Terminal):

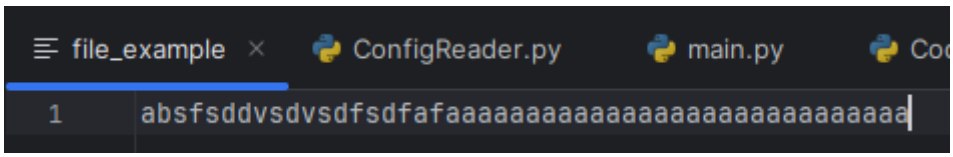
Python main.py config

Примеры работы программы

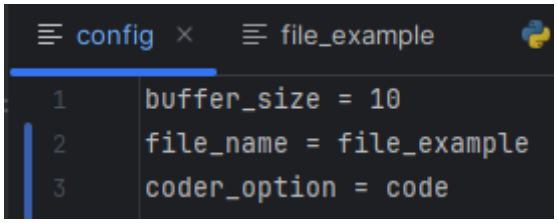
- Кодирование

Закодируем сообщение
“absfsddvsdvsdfsdfafaaaaaaaaaaaaaaaaaaaaaaaaaaaa”

Для этого запишем его в файле file_example



И сделаем файл config



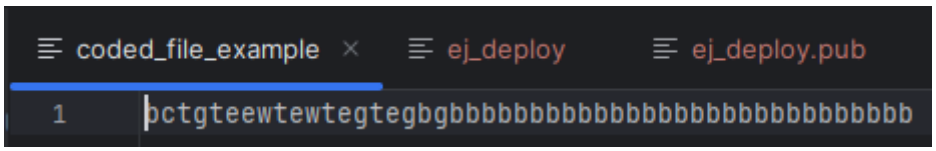
Запустим программу:

```
(.venv) PS C:\Users\Артемий\PycharmProjects\coder_decoder> python main.py config
bctgteewtewtegtegbgbbbbbbbbbbbbbbbbbbbbbbbbbbbb
```

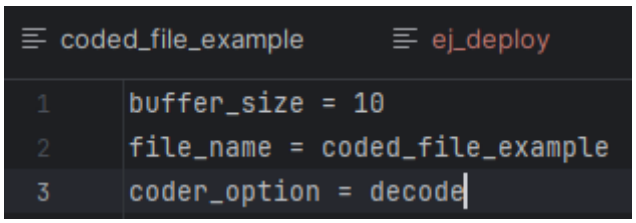
Как видим, все буквы сдвинулись на 1 вправо

- Декодирование

Теперь сделаем обратную процедуру. Запишем **“bctgteewtewtegtegbgbbbbbbbbbbbbbbbbbbbbbbbbbbbb”** в файл coded_file_example



И сделаем файл config:



Запустим программу:

```
(.venv) PS C:\Users\Артемий\PycharmProjects\coder_decoder> python main.py config
absfsddvsdvsdfsdfafaaaaaaaaaaaaaaaaaaaaaaaaaaaa
```

Видим, что все буквы сдвинулись на 1 влево

Выводы

В данной лабораторной работе я научился работать с классами в Python, также научился работать с конфигурационными файлами.