МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение высшего образования

«СЕВЕРОКАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Кафедра инфокоммуникаций

Институт цифрового развития

ОТЧЁТ

по лабораторной работе №4.4

Дисциплина: «Объектно-ориентированное программирование»

Тема: «Работа с исключениями в языке Python»

Вариант 5

Выполнила: студентка 3 курса,

группы ИВТ-б-о-21-1

Диченко Дина Алексеевна

Ставрополь 2023

Цель работы: приобретение навыков по работе с исключениями при написании программ с помощью языка программирования Python версии 3.x.

Практическая часть:

1. Создала репозиторий.
2. Выполнила клонирование.
3. Дополнила файл .gitignore.
4. Организовала репозиторий в соответствии в git-flow.
5. Проработала примеры лабораторной работы.

```
C:\Users\super\OneDrive\Рабочий стол\DI\ВУЗЬ\Воронкин х3\Lab4_4\prog>python prim1.py
>>> add
Фамилия и инициалы? Диченко Д.А.
Должность? стажер
Год поступления? 2023
>>> add
Фамилия и инициалы? Иванов И.И.
Должность? бухгалтер
Год поступления? 2012
>>> list
+------+----------------------------------+----------------------+----------+
|  №   |             Ф.И.О.               |      Должность       |   Год    |
+------+----------------------------------+----------------------+----------+
|    1 | Диченко Д.А.                     | стажер               |     2023 |
|    2 | Иванов И.И.                      | бухгалтер            |     2012 |
+------+----------------------------------+----------------------+----------+
```

Рисунок 1. Результат работы примера

```
>>> help
Список команд:

add - добавить работника;
list - вывести список работников;
select <стаж> - запросить работников со стажем;
load <имя_файла> - загрузить данные из файла;
save <имя_файла> - сохранить данные в файл;
help - отобразить справку;
exit - завершить работу с программой.
>>> save work
>>> save work.txt
```

Рисунок 2. Результат работы примера

```
C:\Users\super\OneDrive\Рабочий стол\DI\ВУЗЬ\Воронкин х3\Lab4_4\prog>python prim1.py
>>> load work.txt
>>> list
+------+----------------------------------+----------------------+----------+
|  №   |             Ф.И.О.               |      Должность       |   Год    |
+------+----------------------------------+----------------------+----------+
|    1 | Диченко Д.А.                     | стажер               |     2020 |
|    2 | Иванов И.И.                      | бухгалтер            |     2012 |
+------+----------------------------------+----------------------+----------+
>>> select 6
  1: Иванов И.И.
```

Рисунок 3. Результат работы примера

6. Решите следующую задачу: напишите программу, которая запрашивает ввод двух значений. Если хотя бы одно из них не является числом, то должна выполняться конкатенация, т. е. соединение, строк. В остальных случаях введенные числа суммируются.

```
Введите первое значение: 2
Введите второе значение: a
2a
```

Рисунок 4. Общее задание 1

7. Решите следующую задачу: напишите программу, которая будет генерировать матрицу из случайных целых чисел. Пользователь может указать число строк и столбцов, а также диапазон целых чисел. Произведите обработку ошибок ввода пользователя.

```
Введите высоту матрицы: 3
Введите ширину матрицы: 3
Введите левый диапазон чисел: 10
Введите правый диапазон чисел: 30
[[25, 11, 25], [27, 14, 18], [14, 20, 12]]
```

Рисунок 5. Общее задание 2

8. Выполнила индивидуальные задания.

Задание 1. Выполнить индивидуальное задание 1 лабораторной работы 2.19, добавив возможность работы с исключениями и логгирование.

```
C:\Users\super\OneDrive\Рабочий стол\DI\ВУЗЬ\Воронкин х3\Lab4_4\prog>python individual1.py
>>> load ind1.txt
>>> add
Пункт назначения? Stavropol
Номер поезда? 43
Время отправления? 00:40
>>> list
+------+--------------------------+-----------------+----------------------+
|  №   |     Пункт назначения     |  Номер поезда   |  Время отправления   |
+------+--------------------------+-----------------+----------------------+
|    1 | Moscow                   | 1               |             21:42:00 |
|    2 | Sd                       | 12              |             12:12:00 |
|    3 | Stavropol                | 43              |                00:40 |
+------+--------------------------+-----------------+----------------------+
>>> add
Пункт назначения? Vladimir
Номер поезда? 1
Время отправления? 10:35
>>> list
+------+--------------------------+-----------------+----------------------+
|  №   |     Пункт назначения     |  Номер поезда   |  Время отправления   |
+------+--------------------------+-----------------+----------------------+
|    1 | Moscow                   | 1               |             21:42:00 |
|    2 | Sd                       | 12              |             12:12:00 |
|    3 | Stavropol                | 43              |                00:40 |
|    4 | Vladimir                 | 1               |                10:35 |
+------+--------------------------+-----------------+----------------------+
>>> select 1
  1: Moscow
  2: Vladimir
```

Рисунок 6. Индивидуальное задание

Задание 2. Изучить возможности модуля logging. Добавить для предыдущего задания вывод в файлы лога даты и времени выполнения пользовательской команды с точностью до миллисекунды.

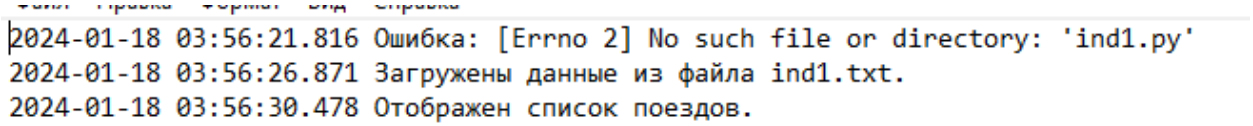

Рисунок 7. Добавление даты и времени в логи

Контрольные вопросы:

1. Какие существуют виды ошибок в языке программирования Python?

Синтаксические ошибки возникают в случае, если программа написана с нарушениями требований Python к синтаксису. Определяются они в процессе парсинга программы.

Второй вид ошибок – это исключения. Они возникают в случае, если синтаксически программакорректна, но в процессе выполнения возникает ошибка (деление на ноль и т.п.). Более подробно про понятие исключения написано выше, в разделе “исключения в языках программирования”.

2. Как осуществляется обработка исключений в языке программирования Python?

Обработка исключений нужна для того, чтобы приложение не завершалось аварийно каждый раз, когда возникает исключение. Для этого блок кода, в котором возможно появление исключительной ситуации необходимо поместить во внутрь синтаксической конструкции try… except.

3. Для чего нужны блоки finally и else при обработке исключений?

Для выполнения определенного программного кода при выходе из блока try/except, используйте оператор finally.

Не зависимо от того, возникнет или нет во время выполнения кода в блоке try исключение, код в блоке finally все равно будет выполнен.

Если необходимо выполнить какой-то программный код, в случае если в процессе выполнения блока try не возникло исключений, то можно использовать оператор else.

4. Как осуществляется генерация исключений в языке Python?

Для принудительной генерации исключения используется инструкция raise.

5. Как создаются классы пользовательский исключений в языке Python?

Для реализации собственного типа исключения необходимо создать класс, являющийся наследником от одного из классов исключений.

Пример:

```
class NegValException(Exception):
  pass
try:
  val = int(input("input positive number: "))
  if val < 0:
     raise NegValException("Neg val: " + str(val))
  print(val + 10)
except NegValException as e:
 print(e)
```

6. Каково назначение модуля logging?

С помощью logging на Python можно записывать в лог и исключения. Обычно лог пишется в файл, зададим его как log.txt. Уровень INFO указывает, что сообщения уровней ниже (в данном случае debug) не будут отражаться в логе.

7. Какие уровни логгирования поддерживаются модулем logging?

Debug (10): самый низкий уровень логирования, предназначенный для отладочных сообщений, для вывода диагностической информации о приложении.

Info (20): этот уровень предназначен для вывода данных о фрагментах кода, работающих так, как ожидается.

Warning (30): этот уровень логирования предусматривает вывод предупреждений, он применяется для записи сведений о событиях, на которые программист обычно обращает внимание. Такие события вполне могут привести к проблемам при работе приложения. Если явно не задать уровень логирования — по умолчанию используется именно warning.

Error (40): этот уровень логирования предусматривает вывод сведений об ошибках — о том, что часть приложения работает не так как ожидается, о том, что программа не смогла правильно выполниться.

Critical (50): этот уровень используется для вывода сведений об очень серьёзных ошибках, наличие которых угрожает нормальному функционированию всего приложения. Если не исправить такую ошибку — это может привести к тому, что приложение прекратит работу.

Вывод: в результате выполнения работы были приобретены навыки по работе с исключениями при написании программ с помощью языка программирования Python версии 3.x.