Южный федеральный университет
Институт математики, механики и компьютерных наук им. И. И. Воровича
направление подготовки «Прикладная математика и информатика»

Лабораторная работа по теме «Обработка исключений»[1]

Обработка исключений используется для избежания ошибок, возникающих в успешно скомпилированной программе в процессе её выполнения.
Для обработки исключений в `Python` используется конструкция `try...except`.

Пример 1А. Обработка деления на ноль.

```
numbers = [1, 2, 3, 0, 100, 0]
for el in numbers:
    try:
        print(1/el)
    except ZeroDivisionError:
        print("Деление на ноль!")
```

Первой выполняется ветка `try`. Если при этом не возникает исключения, то ветка `except` игнорируется. Если в процессе выполнения операторов в `try` возникает исключение, то выполнение операторов в этой ветке прерывается и переходит в ветку `except`. Далее происходит обработка исключения: если тип возникшей ошибки встречается в именах, перечисленных после ключевого слова `except`, то выполняются операторы этой ветки.
Если же возникает исключение, которое отсутствует среди перечисленных в ветке `except`, то это *необработанное исключение*, и выполнение программы прекращается с сообщением об ошибке.
Один оператор `try` может иметь несколько веток `except`, обрабатывающих разные исключения, однако выполнена будет только одна из них. Одна ветка `except` может именовать несколько исключений. В этом случае все типы исключений должны быть указаны в виде кортежа:

```
except (NameError, KeyError, ZeroDivisionError):
```

Помимо веток try и except в обработчике исключений может быть еще две ветки: else — выполняется, если не возникло ни одно из исключений, и finally — выполняется в любом случае после завершения работы обработчика исключений.

Пример 1Б. Обработка деления на ноль.

```
numbers = [1, 2, 3, 0, 100, 0]
for el in numbers:
    try:
        print(1/el)
    except ZeroDivisionError:
        print("Деление на ноль!")
    else:
        print("Значение получено!")
    finally:
        print("До новых встреч!")
```

[1]Разработано А.М. Филимоновой (кафедра ВМиМФ мехмата ЮФУ)

Вызов исключений

Для генерации исключения в Python используется оператор raise, в аргументе которого указывается тип исключения:

```
raise ZeroDivisionError(1/0)    #ZeroDivisionError: division by zero
raise NameError(y)              #NameError: name "y" is not defined
```

Основные типы ошибок в Python:

NameError	переменной с таким именением не существует
KeyError	попытка получить доступ к элементу по несуществующему ключу
ZeroDivisionError	деление на ноль
TypeError	применение операции к объекту несоответствующего типа
ValueError	передача в функцию аргумента неверного значения
ModuleNotFoundError	попытка импортировать несуществующий модуль
ImportError	попытка импортировать несуществующий метод из модуля
IndexError	попытка получить доступ к индексу, которого не существует
OverflowError	переполнение
KeyboardInterrupt	выполнение программы было прервано пользователем

Исключения, определяемые пользователем

Для создания собственного исключения необходимо создать новый класс исключения, который должен быть получен из класса Exception прямо или косвенно. Для именования собственных исключений принято выбирать имена, оканчивающиеся на Error (аналогично именованию стандартных исключений в Python).

Пример 2. Проверка числа на отрицательность.

```
class NegativeNumError(Exception):
    def __init__(self, text, num):    # метод вызывается в момент генерации исключения
        self.txt = text               # текст сообщения об ошибке
        self.n = num                  # значение числа, вызвавшее исключение

a = input("Input positive integer: ")

try:
    a = int(a)
    if a < 0:
        raise NegativeNumError("Number is negative!", a) # генерация исключения

except ValueError:      # обработка случая, когда пользователь ввёл не число
    print("Error type of value!")

except NegativeNumError as NE:
    print(NE)           # кортеж из текста сообщения и значения числа a
    print(NE.args)      # кортеж из текста сообщения и значения числа a
    print(NE.args[0])   # текст сообщения как элемент кортежа
    print(NE.args[1])   # значение числа a, вызвавшее исключение, как элемент кортежа
    print(NE.txt)       # текст сообщения как атрибут экземпляра класса
    print(NE.n)       # значение числа a, вызвавшее исключение, как атрибут экземпляра класса
else:
    print(a)            # если не было вызвано ни одно из исключений
```

Здесь в ветке except используется конструкция для получения аргументов исключения в переменную NE. Порядок аргументов исключения и их тип определяется в методе __init__ класса NegativeNumError.

Для проверки значений различных переменных (или выполнения какого-то условия) помимо конструкции `try...except`, может быть использована конструкция `assert`. Эта конструкция принимает в качестве аргумента некоторое условие и вызывает исключение `AssertionError`, если значение этого выражения `False`. После чего программа завершает работу с ошибкой.
Assert'ы позволяют отлавливать ошибки на любом этапе выполнения программы, поскольку могут быть использованы в любом её месте: в начале, при вводе данных, в теле функции, в основной части, во вложенных конструкциях и т. п.

Пример 1. Проверка числа на положительность.

```
n = int(input("n = "))
assert n > 0      # в случае, если число n>0, оно выведется на экран
print("n = ", n)  # в случае, если число n<=0, возникнет исключение AssertionError
```

Как и при создании собственного исключения, в конструкцию assert можно передать сообщение пользователю о характере конкретной ошибки в данных:

```
assert n > 0, "n should be positive!"
```

Результат работы конструкции для $n = -100$: AssertionError: n should be positive!

Пример 2. Обработка исключения, вызванного assert.

```
n, m = int(input("n = ")), int(input("m = "))
try:
    assert m > 0 and n > 0, "m and n should be positive!"
except AssertionError as AE:
    print(AE)
else:
    print("n = ", n, "; m = ", m)
```

Задачи для самостоятельного решения. Обработка исключений.

---

1. С клавиатуры вводится N. Получить и вывести на экран int(N). Обработать как исключение случай неверного ввода пользователем данных. Задачу решить, используя обработку исключения через try ... except и assert.
2. С клавиатуры вводятся x и a. Посчитать и вывести на экран значения следующих функций: $\sqrt{x-a}$, $\ln x$, $\log_a x$, $\sqrt{\frac{x}{a}}$. Обработать как исключение значения x и a, не удовлетворяющие ООФ.
3. С клавиатуры вводятся два числа в формате строки. Числа могут быть только целыми. Используя конструкцию assert наибольшее из них.
4. С клавиатуры вводится натуральное число n. Заполнить матрицу $A_{n\times n}$ случайными числами в диапазоне $[-2n, n^2]$. По введённым с клавиатуры числам i, j вывести A_{ij} на экран. Обработать как исключение случай выхода за границы двумерного списка.

Задачи для самостоятельного решения. Создание собственного исключения.

6. С клавиатуры вводится целое число N. Если сумма его цифр чётная — вывести на экран сообщение, если нет —создать и вызвать собственное исключение.
7. С клавиатуры вводится строка. Проверить, состоит ли она только из букв и пробелов. Если состоит — вывести строку на экран, если нет — создать и вызвать собственное исключение.
8. С клавиатуры заполняется список. Если очередной введённый элемент чётный, то добавить его. Если нет — создать и вызвать собственное исключение. В случае отсутствия исключений, вывести список на экран.