Практическое задание.

Тема: Исключения Python.

Цель: Изучить исключения в Python и их практическое применение, для улучшения работы программ.

Задачи: Изучить конструкцию построения исключений. Применить теоретические навыки.

Теоретическая часть.

Исключения в Python - это события, которые возникают во время выполнения программы и изменяют обычный порядок ее работы. Если исключение не обрабатывается, программа завершается с ошибкой. Например, попытка деления на ноль вызывает исключение типа ZeroDivisionError, а попытка обращения к несуществующему ключу в словаре вызывает исключение KeyError.

Синтаксис конструкции исключений:

```
try:
    # код, который может вызвать исключение
except SomeException:
    # код, который выполнится, если возникнет исключение SomeException
else:
    # код, который выполнится, если исключений не возникло (если выполнился try)
```

В блоке try находится код, который может вызвать исключение. Если в блоке try возникает исключение, управление передается блоку except, и выполняется код внутри него. Если исключений не возникло, выполняется код в блоке else.

Где <error type> - возможное исключение.

Самые распространённые исключения: TypeError и ValueError.

- TypeError - операция применена к объекту несоответствующего типа.
- ValueError - функция получает аргумент правильного типа, но некорректного значения.
- ZeroDivisionError - деление на ноль.

* Больше исключений в документации Python.

Пример программы с исключением:

```
try:
    x = 1 / 0
except ZeroDivisionError:
    print("Вы попытались разделить на ноль.")
```

В этом примере, мы пытаемся поделить число на ноль, что вызывает исключение ZeroDivisionError. Блок try пытается выполнить код, а except обрабатывает исключение, выводя сообщение об ошибке.

Если бы программа была написана так:

```
x = 1 / 0

print(x)
```

То до вывода x, была бы ошибка деления на ноль (ZeroDivisionError).

Практическая часть.

Задание 1.

Пользователь вводит целое, положительное число x. Если x действительно является числом, то выведете диапазон чисел от 0 до x. Иначе, вывести сообщение «x - не число. Повторите ввод.» В таком случае, пользователь сможет ещё раз повторить ввод.

Пример входных/выходных данных:

```
>>> число: abc

>>> abc - не число. Повторите ввод.

>>> число: x

>>> x - не число. Повторите ввод.

>>> число: 6

>>> 0 1 2 3 4 5 6
```

Задание 2.

Создайте список с произвольным количеством числовых элементов в нём. Поделите каждое число списка на его индекс (используйте enumerate). Чтобы предотвратить ошибку деление первого элемента на 0, создайте блок try и вместо деления выведете «Деление на 0! Элемент».

Например, дан список:

```
any_list = [4, 3.2, 16, 9, 13.5, 67]
```

Выходные данные:

```
Деление на 0! Элемент: 4
3.2 / 1 = 3.2
16 / 2 = 8.0
9 / 3 = 3.0
13.5 / 4 = 3.375
67 / 5 = 13.4
```

Задание 3.

Создать пустой список и наполнить его пятью числами. Числа вводятся с клавиатуры, если введено не число, то не добавлять его в список. В конце вывести весь список.

Пример входных данных:

```
>>> 8
>>> abc
>>> 11
>>> -2
>>> 9
>>> txt
>>> 4
```

Пример выходных данных:

```
>>> Числа в списке: [8, 11, -2, 9, 4]
```

Критерии выставления оценки.

Оценка 5 – сделано 3 задания.

Оценка 4 – сделано 2 задания.

Оценка 3 – сделано 1 задание.

Оценка 2 – ни одного задания не сделано или работа не сдана.