PYTHON 329 HW №7
Что мы прошли? 🤔

Привет! 👋
На прошедших парах (№13 и №4) мы узнали:

- set()
- try except finally
- raise
- Если мы получаем "ошибку", это не конец! Мы можем её обработать.
- Например, мы делим на рандомное число. И так может получится, что туда попадает 0. В математике на 0 делить нельзя, и Пайтон даст нам `ZeroDevisionError` исключение.
- И мы можем его обработать в блоке `Try - Except` - например пропустив эту операцию и не выполнив деление.
- Мы поговорили о том, что можем сами вызвать исключение, и указать пользователю, что именно он делает не так. Это делается ключевым словом `rise`.
- Посмотреть типы встроенных исключений в Пайтон можно тут
- На текущий момент, самые частые для нас это `TypeError` `ValueError`
- **TypeError** - операция применена к объекту несоответствующего типа.
- **ValueError** - функция получает аргумент правильного типа, но некорректного значения.
- Так же, мы посмотрели на некоторые из методов списков. Посмотреть полный список можно тут.

Домашнее задание 📄

Выполните эти задания в отдельных файлах. Упакуйте в один архив. Файлы виртуального окружения в архив не добавляйте.

Задача №1 🍰

Из внешнего источника мы получили список строк. Мы ожидаем, что там будут числовые значения.

Объявите переменную с новым, пустым списком.

Объявите цикл, пройдитесь по списку, примените `int()` к каждому элементу.

Используйте конструкцию `try - except`. Ведь неизвестно, пришли ли валидные данные.

В случае успеха `int()` - добавьте это число в новый список.

В случае неудачи, сделайте принт `f-строки` о том, что данные невалидны (подставьте переменную)

Можете использовать этот набор данных.

```
data_lst = ['1', '2', '3', '4d', 5]
```

Сделайте принт нового списка.

Критерии проверки 👌

Info

1. Используйте блоки `try-except

2. НЕ используйте проверку методом строк
3. Цикл должен завершится полностью
4. Используйте `f-строку`, и покажите, какие именно данные невалидны
5. В новом списке будет число 5 (**подтверждение** пункта 3)

Задача №2 🤓

В этой задаче мы не будем обрабатывать исключения. Мы будем вызвать их! 😎

Мы пишем программу, которая делает проверку валидности номера телефона.

И если номер не валидный, программа "падает с ошибкой"

`ValueError номер телефона {подставлен номер} не соответствует формату!`

Пользователь вводит номера телефонов через точку с запятой (без пробелов). Мы разбиваем это на список.

Номера должны быть:

- 11 чисел
- могут начинаться на 8
- могут начинаться на +7 (+ не считается за символ)

Следующие номера считаются валидными

```
#Валидные номера

nums= [
'+77053183958',
'+77773183958',
'87773183958',
'+(777)73183958',
'+7(777)-731-83-58',
'+7(777) 731 83 58']
```

После того, как мы разобьем пользовательский ввод на список по точке с запятой (без пробелов), нам надо пройтись циклом и проверить данные!

Если номера не соответствуют требованиям (11 чисел, начало не с 8 или +7) мы вызываем исключение.

Это выглядит примерно так:

```
if # очищенный номер больше 11 знаков:
        rise ValueError(f'номер {подстваили номер} больше 11 знаков')
```

Критерии проверки 👌

Info

1. Вы очистили номер от скобок, пробелов, тире и плюса методами строк
2. Вы использовали синтаксис `rise` для вызова исключений, как минимум для 3 разных случаев. Это длина номера, номер начинается не с 8 и не с +7 и в номере есть что-то кроме чисел.
3. Очень хорошо, если вы придумаете и другие проверки и реализуете их.

4. Использование f-строк для rise - чтобы показать какие именно данные вызвали исключение

Связано

Python 329
Академия TOP
Python 329. Python. Lesson 14