Обработка ошибок

1. Текст ошибки указывается в последней строчке
2. Все что перед ней - место, где ошибка произошла
3. Есть встроенные типы ошибок, но можно создавать и свои

Некоторые типы ошибок из документации (точнее перевода (https://pythonworld.ru/tipy-dannyx-v-python/isklyucheniya-v-python-konstrukciya-try-except-dlya-obrabotki-isklyuchenij.html)):

- ZeroDivisionError - деление на ноль
- ImportError - не удалось импортирование модуля или его атрибута (надо установить эту библиотеку)
- IndexError - индекс не входит в диапазон элементов.
- KeyError - несуществующий ключ (в словаре, множестве или другом объекте)
- MemoryError - недостаточно памяти
- SyntaxError - синтаксическая ошибка (вы опечатались или не закрыли скобку)
- TypeError - операция применена к объекту несоответствующего типа
- ValueError - функция получает аргумент правильного типа, но некорректного значения
- Warning - предупреждение (текст на красном фоне в юпитере это предупреждение, а не ошибка)

In [1]:

```
# эту строку можно перевести в число
some_num = '123'
```

In [2]:

```
float(some_num)
```

Out[2]:

```
123.0
```

In [3]:

```
# а эту уже нет (по крайней мере в десятичном счислении)
ups = '123a'
```

In [4]:

```
# ValueError - тип ошибки, далее пояснение что произошло
# ----> 1 float(ups) - в каком месте кода произошла ошибка
float(ups)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-4-b5e45a6d52d0> in <module>
      1 # ValueError - тип ошибки, далее пояснение что произошло
      2 # ----> 1 float(ups) - в каком месте кода произошла ошибка
----> 3 float(ups)

ValueError: could not convert string to float: '123a'
```

Пример ошибки внутри функции

In [8]:

```
def square_sum(*args):
    total_sum = 0
    for arg in args:
        print(arg)
        total_sum += arg**2

    return total_sum
```

In [6]:

```
square_sum(1, 2, 3)
```

Out[6]:

```
14
```

In [9]:

```
# пытаемся применить к операцию возведения в квадрат к строке
# ----> 1 square_sum(1, 2, '3') - в какой функции произошла ошибка
# ----> 4         total_sum += arg**2 - в какой именно строке произошла ошибка

square_sum(1, 2, '3')
```

```
1
2
3

---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-9-7189647907b6> in <module>
      3 # ----> 4         total_sum += arg**2 - в какой именно строке произо
шла ошибка
      4
----> 5 square_sum(1, 2, '3')

<ipython-input-8-0f60c0b38293> in square_sum(*args)
      3     for arg in args:
      4         print(arg)
----> 5         total_sum += arg**2
      6
      7     return total_sum

TypeError: unsupported operand type(s) for ** or pow(): 'str' and 'int'
```

Как сделать, чтобы цикл с расчетом не падал каждый раз

In [10]:

```
try:
    # ваш код, где может произойти ошибка
    float('123a')

except:
    # код, который выполняется в случае ошибки
    print('Кривая строка')
    pass
```

```
Кривая строка
```

In [11]:

```
data = ['90', '60', '90', '240tot']
total_sum = 0

for num in data:
    try:
        total_sum += float(num)

    except:
        print('Ошибка в данных: {}'.format(num))

print('Итого', total_sum)
```

```
Ошибка в данных: 240tot
Итого 240.0
```

Как сохранить всю информацию об ошибке?

In [12]:

```
# полная версия traceback
import traceback

try:
    float('123fff')

except Exception:
    print(traceback.print_exc())

print('Проехали')
```

```
None
Проехали

Traceback (most recent call last):
  File "<ipython-input-12-aec2760d579a>", line 5, in <module>
    float('123fff')
ValueError: could not convert string to float: '123fff'
```

Блок finally

In [13]:

```
try:
    print(stats["wednesday"])

except IndexError:
    print("Ошибка индекса")

except KeyError:
    print("Ошибка ключа")
    print(1/0)

finally:
    print('Эта строчка будет выполнена всегда')
```

```
Эта строчка будет выполнена всегда

---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-13-f6fe70bb823a> in <module>
      1 try:
----> 2     print(stats["wednesday"])
      3
      4 except IndexError:
      5     print("Ошибка индекса")

NameError: name 'stats' is not defined
```

Даты

In [ ]:

```
# иногда импортируют так
import datetime
```

In [ ]:

```
datetime.datetime.strptime()
```

In [ ]:

```
# можно и так
import datetime as dt
```

In [15]:

```
datetime.strptime()
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-15-e1a91825e15a> in <module>
----> 1 datetime.datetime.strptime()

AttributeError: type object 'datetime.datetime' has no attribute 'datetime'
```

In [ ]:

```
import datetime.py
    class datetime
```

In [14]:

```
# у нас будет вариант покороче (но это не одно и то же)
from datetime import datetime
```

In [ ]:

```
datetime.strtip
```

In [16]:

```
date_string = '09.05.2018  09:00'
```

In [ ]:

```
2018-09-05Z15:32:00.1726352
```

In [17]:

```
# сейчас date_string это просто строка
type(date_string)
```

Out[17]:

```
str
```

In [19]:

```
datetime.strptime('09.05.2018 09:00', '%d.%m.%Y %H:%M')
```

Out[19]:

```
datetime.datetime(2018, 5, 9, 9, 0)
```

In [ ]:

```
datetime.strptime('09.05.2018 09:00', '%d.%m.%Y %H:%M')
```

In [20]:

```
# https://docs.python.org/3/library/datetime.html

date_datetime = datetime.strptime( date_string, '%d.%m.%Y %H:%M' )
date_datetime
```

Out[20]:

```
datetime.datetime(2018, 5, 9, 9, 0)
```

In [21]:

```
# теперь можем работать с датами
type(date_datetime)
```

Out[21]:

```
datetime.datetime
```

In [22]:

```
# получить номер года и часа
date_datetime.year, date_datetime.hour
```

Out[22]:

```
(2018, 9)
```

In [23]:

```
# день недели
date_datetime.weekday()
```

Out[23]:

```
2
```

In [24]:

```
# сегодня
datetime.now()
```

Out[24]:

```
datetime.datetime(2020, 6, 20, 15, 9, 40, 92581)
```

Прибавление интервала к датам

In [25]:

```
from datetime import timedelta
```

In [26]:

```
start_date = '2018-01-01'
end_date = '2018-01-07'
```

In [27]:

```
type(start_date)
```

Out[27]:

```
str
```

In [31]:

```
'logs_2017-12-31.csv' < 'logs_2018-01-01.csv'
```

Out[31]:

```
True
```

In [32]:

```
start_date_datetime = datetime.strptime(start_date, '%Y-%m-%d')
start_date_datetime
```

Out[32]:

```
datetime.datetime(2018, 1, 1, 0, 0)
```

In [36]:

```
start_date_datetime + timedelta(hours=1)
```

Out[36]:

```
datetime.datetime(2018, 1, 1, 1, 0)
```

In [37]:

```
start_date_datetime + timedelta(days=-7, minutes=-1)
```

Out[37]:

```
datetime.datetime(2017, 12, 24, 23, 59)
```

Перевод обратно в строку

In [38]:

```
date = datetime(2018, 9, 1)
date
```

Out[38]:

```
datetime.datetime(2018, 9, 1, 0, 0)
```

In [40]:

```
date.strftime('%Y-%m-%dZZZ%H')
```

Out[40]:

```
'2018-09-01ZZZ00'
```

In [41]:

```
date.strftime('%B %d %Y %I:%M%p')
```

Out[41]:

```
'September 01 2018 12:00AM'
```

In [42]:

```
datetime.now().strftime('%Y-%m-01')
```

Out[42]:

```
'2020-06-01'
```

In [ ]:

```
# как получить первый день месяца

date.strftime('%Y-%m-01')
```

In [43]:

```
start_date = '2018-01-01'
end_date = '2018-01-07'
```

In [44]:

```
start_date, end_date
```

Out[44]:

```
('2018-01-01', '2018-01-07')
```

In [45]:

```
start_date_dt = datetime.strptime(start_date, '%Y-%m-%d')
end_date_dt = datetime.strptime(end_date, '%Y-%m-%d')

print(start_date_dt, end_date_dt)
```

```
2018-01-01 00:00:00 2018-01-07 00:00:00
```

In [46]:

```
i = 0

while i < 10:
    # ...
    i += 1
    print(i)
```

```
1
2
3
4
5
6
7
8
9
10
```

In [47]:

```
current_dt = start_date_dt

while current_dt <= end_date_dt:
    print(current_dt.strftime('%Y-%m-%d'))

    current_dt += timedelta(hours=1)
```

```
2018-01-01
2018-01-02
2018-01-03
2018-01-04
2018-01-05
2018-01-06
2018-01-07
```

In [48]:

```
current_dt = start_date_dt

while current_dt.strftime('%Y-%m-%d') <= end_date:
    print(current_dt.strftime('%Y-%m-%d'))

    current_dt += timedelta(days=1)
```

```
2018-01-01
2018-01-02
2018-01-03
2018-01-04
2018-01-05
2018-01-06
2018-01-07
```

In [49]:

```
# можно и с помощью list comprehension
[(start_date_dt + timedelta(days=x)).strftime('%Y-%m-%d') for x in range(10)]
```

Out[49]:

```
['2018-01-01',
 '2018-01-02',
 '2018-01-03',
 '2018-01-04',
 '2018-01-05',
 '2018-01-06',
 '2018-01-07',
 '2018-01-08',
 '2018-01-09',
 '2018-01-10']
```

Нагрузка на систему по часам

In [59]:

```
stats = {}

with open('logs.csv', 'r') as f:
    for line in f:
        line = line.strip()

#         print(line[11:13])
#         break

#         dt = datetime.strptime(line, '%Y-%m-%dT%H:%M:%SZ')
        hour = line[11:13]
        # print(hour)

        stats.setdefault(hour, 0)
        stats[hour] += 1

        # break

        # Вычисления нагрузки на систему...

# результат
stats
```

Out[59]:

```
{'21': 59,
 '20': 67,
 '23': 36,
 '22': 37,
 '18': 72,
 '13': 56,
 '11': 52,
 '00': 22,
 '16': 56,
 '17': 56,
 '15': 68,
 '19': 63,
 '12': 64,
 '10': 77,
 '01': 10,
 '07': 22,
 '05': 10,
 '09': 33,
 '06': 19,
 '14': 57,
 '08': 42,
 '03': 7,
 '02': 8,
 '04': 7}
```

In [ ]:

```
# а в процентном соотношении?
```

In [ ]:

```
```

In [ ]:

```
```

In [ ]:

```
```

Unixtime

Количество секунд, прошедших с 1 января 1970 года по UTC

In [60]:

```
import time
from datetime import date
from datetime import datetime
```

In [61]:

```
d = date(2019, 3, 11)

unixtime = time.mktime(d.timetuple())
unixtime
```

Out[61]:

```
1552251600.0
```

In [62]:

```
from datetime import datetime
```

In [63]:

```
datetime.fromtimestamp(1552251600)
```

Out[63]:

```
datetime.datetime(2019, 3, 11, 0, 0)
```

На практике все сложнее https://habr.com/ru/post/452584/ (https://habr.com/ru/post/452584/)