Exceptions

exceptions

Exceptions(исключения) это ошибки возникающие при работе программы.

Если в программе есть ошибки и они не обрабатываются, программа перестает работать
Например:

```
print(2+5)
print(
print(2**3)
print(25%5)

Ответ: File "C:/class work.py", line 4
    print(25%5)
         ^
SyntaxError: invalid syntax
```

Программа может работать до ошибки либо нет. Это зависит от типа ошибки.

```
print(25//4)
print(36%7)
print(25/0)
print(“Exception.")
```

Ответ:

```
Traceback (most recent call last):
6
1
  File "C:/Users/PycharmProjects/class work.py", line 3, in <module>
    print(25/0)
ZeroDivisionError: division by zero
```

Иерархия исключении

В питоне есть иерархия классов для обработки исключении. С помощью этих классов мы можем узнать тип исключения

```
BaseException
   +-- SystemExit
   +-- KeyboardInterrupt
   +-- GeneratorExit
   +-- Exception
      +-- StopIteration
      +-- StopAsyncIteration
      +-- ArithmeticError
         +-- FloatingPointError
         +-- OverflowError
         +-- ZeroDivisionError
```

Exceptions конструкции

- try/except exception_name
- try/finally
- try/except/else
- raise
- assert
- with/ as

try/except exception_name

С помощью конструкций try/except exception_name можно обработать исключение программы. Оно состоит из двух частей: try, except. try пишется код который может вызвать ошибку. А в except пишется обработка ошибки. Пример:

```
def fetcher(obj,index):
    return obj[index] #возвращает элемент по индексу
name= 'студент' #
n = fetcher(name,2)
print("The character with index two is",n)

n = fetcher(name,7)
print("The character with index seven is",n)
```

Продолжение

```
Ответ:
The character with index two is a
Traceback (most recent call last): File "File_example1.py",
line 6, in <module>
n = fetcher(name,7)
File "File_example1.py", line 2, in fetcher
return obj[index]
IndexError: string index out of range
```

Нет инедекса 7 в элементе name, IndexError это наше исключение

Продолжение

Используем try/except exception_name

```
def fetcher(obj,index):
    return obj[index]
name=‘студент'
try:
    n = fetcher(name,2)
print("The character with index two is",n)
except IndexError:
    print("The index requested (2) is not valid
try:
    n = fetcher(name,7
    print("The character with index seven is",n)
except IndexError:
    print("The index requested (7) is not valid.")
```

Ответ:

```
The character with index two is a
The index requested (7) is not valid.
```

try/finally

Конструкция try/finally состоит из 2 частей: try, finally. В try пишется вероятно ошибочный код. finally пишется код который выполнится в любом случае. Пример:

```
try:
    result="d"*"d"
    print(result)
except(TypeError):
    print("Exception was raised!")
finally:
        print("Finally clause text.")
Ответ:
Exception was raised!
Finally clause text.
```

try/finally

```
try:
    result="d"*"d"
    print(result)
finally:
    print("Finally clause text.")
```

Ответ:

```
Traceback (most recent call last):
  File "class work.py", line 2, in <module>
    result="d"*"d"
TypeError: can't multiply sequence by non-int of type 'str'
Finally clause text.
```

try/except/else

```
try/except/else:
    try…
    except...
    else вызывается если все ок.
```

try/except/else

```
try:
    file=open("файл.txt","r")
    print(file.encoding)
    print(file.name)
except(SyntaxError):
    print("You have a syntax error.")
except(IOError):
    print("IOError exception was raised.")
else:
    print("You have no syntax and input/output errors.")
```

Ответ:

```
cp1251
файл.txt
You have not syntax or and input/output errors.
```

try/except/else

```
try:
    file=open("фай.txt","r")
    print(file.encoding)
    print(file.name)
except(SyntaxError):
    print("You have a syntax error.")
except(IOError):
    print("IOError exception was raised.")
else:
    print("You have no syntax and input/output errors.")
```

Файла фай.txt не был найден и вышло исключение IOError.

Ответ:
IOError exception was raised.

Конструкция assert

Конструкция assert: assert condition, data . Condition – условие. Data – (string) текст.

Если в конструкции assert условие False, выводит data. Если True, не выводит data.

Пример

```
result=42
assert result==41, "Фаренгейт не правильно. Проверьте
формулу"

Ответ:
Traceback (most recent call last):
  File "C:/Users/problem2.py", line 8, in <module>
    assert result==41, "Фаренгейт не правильно. Проверьте
формулу."
AssertionError: Фаренгейт не правильно. Проверьте формулу.
```

Продолжение

```
result=41
assert result==41, " Фаренгейт не правильно. Проверьте
формулу "

Ответ:
Fahrenheit: 41.0
```

raise оператор

raise создает exception. Синтаксис оператора raise:

1)raise exception_class_name

2)raise exception_object_name

То есть exception можно получить по ошибке или оператором raise

Например:

```
print(4/0)
ZeroDivisionError  без raise
```

```
raise ZeroDivisionError
ZeroDivisionError  с raise
```

синтаксис raise exception_object_name

Задание: Пользователь вводит число. Если число не в интервале [3;5] ValueError. Иначе делим на 25

```
n=int(input())
value_object=ValueError()
if n<3 or n>5:
    raise value_object
print(n/25)
```