Python.

Модули и исключения.

Типы ошибок

- Синтаксические ошибки
- Исключения

Синтаксические ошибки.

```
while 1 print ’Hello world’
    File "<stdin>", line 1
     while 1 print ’Hello world’
SyntaxError: invalid syntax
```

Исключения

Ошибки, обнаруженные во время выполнения.

Пример:

```
>>> 10*(1/0)
ZeroDivisionError: integer division or modulo by zero
```

Обработка исключений

```
while True:
    try:
        x = int(raw_input("Пожалуйста, введите целое число"))
    except ValueError:
        print "Вы ошиблись..."
```

Несколько типов исключений

```
except (RuntimeError, TypeError, NameError):
    pass
except RuntimeError:
    pass
except TypeError :
    pass
```

Else

```
for arg in sys.argv[1:]:
    try:
        f = open(arg, 'r')
    except IOError:
        print 'Exception' , arg
    else:
        print arg, 'Все OK! '
        f.close()
```

Генерация исключений

```
>>> raise NameError('HiThere')

Traceback (innermost last):
  File "<stdin>", line 1
NameError: HiThere
```

Пользовательские исключения

```
class MyError(Exception): pass

>>> raise MyError(1)
Traceback (innermost last):
File "<stdin>", line 1
__main__.MyError: 1
```

finally

```
f = open('file.txt', 'r')

try:
# do stuff with f
finally:
    f.close()
```

KeyboardInterrupt

- Ctrl + C порождает исключение

With statement

```
with open("myfile.txt") as f:
    for line in f:
        print(line)
```

With statement

```
class controlled_execution:
    def __enter__(self):
        set things up
        return thing
    def __exit__(self, type, value, traceback):
        tear things down

with controlled_execution() as thing:
    some code
```

Модули.

Модули

Модуль – файл, содержащий определения и другие инструкции языка Python. Имя файла образуется путем добавления к имени модуля суффикса(расширения) '.py'.

В пределах модуля его имя доступно глобальной переменной __name__.

Создание модулей

‘fibo.py’

```
def fib(n):
    '''Последовательность чисел Фибоначчи < n'''
    a, b = 0, 1
    while b < n:
        print b,
        a, b = b, a+b
```

Использование модулей

```
>>> import fibo

>>> fibo.fib(1000)
1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987

>>> fib = fibo.fib
```

Модуль представлен объектом-модулем, атрибутами которого являются имена, определенные в модуле

From statement

```
>>> from fibo import fib
>>> fib(500)

>>> from fibo import *
>>> fib(500)
```

import AS

```
import string as _string
from anydbm import open as dbopen
```

reload()

Каждый модуль импортируется лишь единожды на каждый сеанс работы с интерпретатором

reload(<имя модуля>)

Запуск модуля

$ python fibo.py <аргументы>

__name__ == "__main__"

Поиск модулей

- Текущий каталог
- PYTHONPATH

- sys.path

['', '/usr/lib/python27.zip', '/usr/lib/python2.7']

“Компилированные” модули

- Байт-код
- В файле *.pyc записано время изменения версии *.py

- $ python -m compileall .

Стандартные модули

- Библиотека стандартных модулей
- Модули sys, os
- sys.path(список строк с именами каталогов, в которых происходит поиск модулей)

Функция dir()

- Для любого объекта можно получить всю информацию о его внутренней структуре
- Отсортированный список имен атрибутов для любого переданного в нее объекта.
- Если ни один объект не указан, dir() возвращает имена в текущей области видимости.

Пакеты

- способ структурирования *пространств имён*
- A.B означает — подмодуль с именем B в пакете с именем A
- Файл __init__.py
- from *package* import * # __all__