Лекция 4
Разработка программ на Python

Давыдов Виталий Валерьевич (ГАИШ МГУ, Постгрес-ПРО)

Повторение пройденного

- Форматирование кода (отступы определяют блоки)
- Условные операторы (if, else)
- Операторы цикла (while, for)
- Функции:
  - Обычные функции
  - Генераторы
  - Лямбда-функции
- Области видимости
  - Правило LEGB (local, enclosed, global, built-in)

Аргументы функции

- **Позиционные и именованные аргументы**
  - `def f(a, b, c, d, e)`
  - Все аргументы обязательные
- **Можно задавать дефалтовые значения (необязательные арг.)**
  - `def f(a, b = 20)`
  - Обязательные аргументы должны быть до необязательных
- **Можно задавать переменное число позиционных аргументов**
  - `def f(a, *b): for arg in b: print(arg)`
- **Можно задавать переменное число именованных аргументов**
  - `def f(**kwargs): for arg in kwargs.keys(): print(arg, kwargs[arg])`
- **В вызове функции можно указывать аргументы по имени:**
  - `def f(a, b = 20): ...` `f(b = 1, a = 2)`
- **Разрешить только фиксированные именованные аргументы**
  - `def f(a, *, b, c):...` `f(10, b = 20, c = 30)`

Встроенные функции (функциональн. прогр.)

- **map**
  - Применяет унарную функцию к каждому элементу последовательности
  - map(func, seq)
  - def f(x): print(x) map(f, [1, 2, 3, 4, 5])
  - map(lambda x: x + 1, [1, 2, 3, 4, 5])
- **reduce (functools)**
  - Применяет бинарную функцию к предыдущему результату и следующему элементу
  - reduce(func, seq, startval)
  - reduce(lambda x, y: x + y, [1,2,3,4,5])
- **filter**
  - Фильтрует входящую последовательность, возвращает новую с отфильтрованными элементами
  - filter(func, seq)
  - filter(lambda x: x % 3 == 0, range(100))
- **apply**
  - Python2: apply(f, ...)
  - Python3: f(*[arglist], **[kwarglist])

Функции - Вопросы

```
def f(): pass
a = f()
print(a)
```

```
def a(a): print(a)
a(10)
```

```
def f(a, *, b): pass
f(10, b = 20, c = 30)
```

```
def f(*a):
    for x in a: print(x)
f(1, 2, 3, 4)
```

```
if x == 2:
    def f():
        print("x == 2")
f()
```

```
a = 10
def f():
    a = a + 1
f()
```

```
def f():
    a = 10
    def g():
        def d():
            print(a)
        d()
    g()
f()
```

```
a = lambda: 1
print(a())
```

```
a = lambda x, y = 2: x + y
print(a())
```

```
(lambda x: print(x))(10)
```

```
f = lambda x, y: [x + y, x * y]
a, b = f(1, 2)
print(a, b)
```

```
a = map(lambda x:x+1, [1,2,3])
print(a)
print(*a)
```

Списковые включения и генераторные выражения

- **Списковые включения (list comprehensions, python 2.0)**
  - [expr **for** iter_var **in** iterable **if** cond_expr]
  - Вложенные итерации: [expr for iv1 in seq1 for iv2 in seq2]
- **Генераторные выражения (generator expressions, python 2.2)**
  - Lazy calculations
  - (expr **for** iter_var **in** iterable **if** cond_expr)
  - Вложенные итерации: (expr for iv1 in seq1 for iv2 in seq2)

```
L = [ x **2 for x in range(100) ]
L = [ (x, y) for x in range(3) for y in range(3) ]

R = ( x**2 for x in range(100) )
```

Списковые включения - Вопросы

Что делает код:

```
f = open('hhga.txt', 'r')
print(len([ word for line in f for word in line.split() ]))

f.seek(0)
sum([ len(word) for line in f for word in line.split() ])
```

Чтение и запись файлов

- Открытый файл идентифицируется объектом-файлом
- Режимы открытия: текстовый и бинарный
- Файл необходимо "открыть" и "закрыть" для освобож. ресурсов
- Функции работы с файлами
  - open(<filepath>, <mode>, <encoding>, ...) - открытие файла файл (mode = r, w, a, t, b)
  - f.close() - закрытие файла
  - f.read() - чтение всего содержимого
  - f.readline() - чтение файла построчно
  - f.readlines() - чтение всех строк файла в виде списка
  - f.write() - запись строки в файл
  - f.tell() - возвращает текущее положение в файле (в байтах)
  - f.seek(<pos>) - перемещение текущего положения в позицию pos (в байтах)

Чтение и запись файлов в бинарном режиме

- Упаковка (в массив байтов)
  - struct.pack
- Распаковка
  - struct.unpack
- Порядок байтов (MSB, LSB)
  - Big/Little Endian
  - Endianess
  - Сетевой порядок байтов: MSB (Big Endian)
  - C-функции: htonl, ntohl, ...
- fd.write() возвращает bytes
- fd.read() принимает bytes

```python
import struct

with open("data.bin", "wb") as fd:
    b = struct.pack(">II", 1, 2)
    print(b)
    fd.write(b)

with open("data.bin", "rb") as fd:
    while True:
        b = fd.read(8)
        if len(b) == 0:
            break
        print(b)
        x, y = struct.unpack(">II", b)
        print(x, y)
```

Обработка ошибок

- Используются исключения
- try, except, finally

```
# В чем проблема этого кода?

try:
    f = open("data.dat", "rt")
    content = f.read()
    print(content)
except Exception as e:
     print("ERROR: Ошибка чтения файла ({})".format(e))
finally:
    f.close()
```

Менеджер контекста

- Это объект, реализующий интерфейс МК
- Используется с оператором if
- Реализует функции __enter__, __leave__ (dunder functions)
- Используется для автоматического освобождения ресурсов

```
with open("myfile.dat", "rt") as f:
    for line in f.readline():
        process_line(line)
```

Регулярные выражения

- Используются для поиска подстрок по заданному шаблону
- Метасимволы
  - ^ - начало строки
  - $ - конец строки
  - . - любой символ
  - * - 0 или более предстоящих символов
  - ? - 0 или 1 предстоящий символ
  - + - 1 и более предстоящий символ
  - [] - любой символ из указанных в скобках
  - () - подгруппа
  - Класы символов: \d, \D, \w,]W, \s, \S
  - Экранирование: \+, \., \[, \(
- import re
- https://docs.python.org/3/library/re.html

Регулярные выражения

```
re.match(pat, str)
re.search(pat, str)
re.findall(pat, str)
re.split(pat, str, maxsplit = 0)
re.sub(pat, repl, str)
re.compile(pat)
```

```
import re

s = "this is my email ivan@mail.ru"
m = re.search("([\w\.-])@([\w\.-]+)", s)
if m:
    print(m.group())
    print(m.group(1))
    print(m.group(2))
```

Практическая Задача

Найти в директории и ее поддиректориях все файлы в формате SBIG ST-6 (по расширению .SBIG), извлечь и вывести в консоль их метаданные.

Решение Практической Задачи

```python
import os
import re

def scan_dir(rootpath):
    # Сканирование директории в поисках файлов в формате SBIG ST-6
    for dirpath, _, filenames in os.walk(rootpath):
        for fn in filter(lambda x: "SBIG" in x, filenames):
            yield os.path.join(dirpath, fn)

def sbig_read_headers(fd):
    # Чтение метаданных из SBIG файла
    for line in map(str.strip, fd):
        if "End" in line:
            return
        m = re.search("^([^=]*) = (.*)$", line)
        if m:
            yield m.group(1), m.group(2)

```

```python
def scan_sbig_files(rootpath):
    # Рекурсивный поиск файлов и чтение метаданных
    for fn in scan_dir(rootpath):
        headers = {}
        with open(fn, mode="rt", errors="ignore", newline="\n") as fd:
            for k, v in sbig_read_headers(fd):
                headers[k] = v
        yield fn, headers

for fn, headers in scan_sbig_files("/s/science/ss433"):
    print(fn)
    for k, v in headers.items():
        print("\t{} : {}".format(k, v))
```