Основы Python

Синтаксис, стиль, философия...

- zen, pep8
- консоль, типы данных, операторы
- terminal, tools, IDE
- исключения и разбор стектрейса
- синтаксис управляющих конструкций

Философия прежде всего!

- код мы чаще читаем чем пишем
- лаконичность синтаксиса и минимум мусора
 - пробелы имеют значение!
 - строгие гайды форматирования pep8
 - не синтаксисом и форматированием достигается zen

import this

Beautiful is better than ugly.
Explicit is better than implicit.
Simple is better than complex.
Complex is better than complicated.
Flat is better than nested.
Sparse is better than dense.
Readability counts.
Special cases aren't special enough to break the rules.
Although practicality beats purity.
Errors should never pass silently.
Unless explicitly silenced.
In the face of ambiguity, refuse the temptation to guess.
There should be one-- and preferably only one --obvious way to do it.
Although that way may not be obvious at first unless you're Dutch.
Now is better than never.
Although never is often better than *right* now.
If the implementation is hard to explain, it's a bad idea.
If the implementation is easy to explain, it may be a good idea.
Namespaces are one honking great idea -- let's do more of those!

- http://legacy.python.org/dev/peps/pep-0008/

- https://docs.python.org/2/tutorial/

- https://docs.python.org/2/

Hello world!

- своя консоль, запуск из коноли
- print it!
- типизация - динамическая, но строгая
- встроеные простые типы данных:
  - int, str, bool
- преобразование типов
- встроеные сложные типы данных:
  - list, tuple, dict
- встроенные функции
- функции типов данных

int

- операторы: + - * / // % **
- можно пользоваться как калькулятором

bool

- True, False
- булевые операторы:
  > >= == != <= < and or is not
- преобразование типов данных к булевому значению

builtin functions

- __builtin__
- dir(__builtin__)
- **типы данных** (int, bool, str, list, tuple, dict)
- **некоторые функции** (len, dir, type, ord, chr)

str

- ' ' " " ''' ''' """ """
- операторы: + * %
- строка это итерируемый тип
- функции:
  - title, capitalize, upper, lower
  - strip, lstrip, rstrip
  - count, index
  - isalnum, isalpha, isdigit

list

- [ ]
- операторы: + *
- индексы
- слайсы
- **функции**
  - index, count
  - sort, reverse
  - append, extend, insert, pop, remove
- **функции строк** split, join
- range, min, max, sum, sorted

tuple

- ( )
- операторы: + *
- индексы
- слайсы
- функции
  - index, count

dict

- {}
- особенности: не сортированный, изменяемый
- ключ-значение, взятие значения по индексу
- преобразования
- функции
  - keys, values, items
  - pop, popitem, update
  - get, has_key (operator ‘in’)

execute file

- bash terminal (навигация, по директориям, создание файлов и директорий)
- редактирование файла в IDE
- альтернативные интерактивные консоли

Exceptions!

- наши лудшие друзья
- читаем трейс ошибки
- ловим ошибки

if … elif … else

- вычисляет логическое значение
- выполняет соответствующий блок

(отступы имеют значение!)

```
if True:
    print “True!”
else:
    print “False!”
```

for … in …

- проходит по всем значениям последовательности
- выполняет для каждого тело цикла

```
for i in (1, 2, 3):
    t = i**2
    print t
```

- break, continue, else, pass

while …

- проверяет логическое условие
- выполняет тело цикла пока выполняется условие

```
while True:
    print u“Выводим это сообщение бесконечно”
```

- break, continue, else

try … except …

- просто идет выполнение первого блока
- если происходит исключение описаное в except прерывает выполнение и переходит к блоку except

```
try:
    a = 10 / 0
except ZeroDivisionError:
    print u“Деление на ноль!”
```

- else, finally

интересности

- распаковка
- генераторы списков
- unicode
- # -*- coding: utf-8 -*-