

# Оглавление

<b>Предисловие</b> .....	<b>16</b>
Комментарии переводчика .....	18
Базовый набор библиотек для разработчика .....	18
От издательства .....	19
<b>Глава 1. Введение</b> .....	<b>20</b>
1.1. Что такое идиома Python .....	20
1.2. Чем эта книга будет полезна .....	22
1.3. Как читать эту книгу .....	23
1.4. Тонкости Python: цифровой комплект инструментов в качестве бонуса ..	24
<b>Глава 2. Шаблоны для чистого Python</b> .....	<b>25</b>
2.1. Прикрой свой з** инструкциями assert .....	25
Инструкция assert в Python — пример. ....	26
Почему просто не применить обычное исключение? .....	27
Синтаксис инструкции Python assert .....	28
Распространенные ловушки, связанные с использованием инструкции assert в Python .....	30
Предостережение № 1: не используйте инструкции assert для проверки данных .....	30
Предостережение № 2: инструкции assert, которые никогда не дают сбой	32
Инструкции assert — резюме .....	34
Ключевые выводы .....	34

2.2. Беспечное размещение запятой . . . . .	34
Ключевые выводы . . . . .	38
2.3. Менеджеры контекста и инструкция with . . . . .	38
Поддержка инструкции with в собственных объектах . . . . .	40
Написание красивых API с менеджерами контекста . . . . .	42
Ключевые выводы . . . . .	44
2.4. Подчеркивания, дандеры и другое . . . . .	44
1. Одинарный начальный символ подчеркивания: <code>_var</code> . . . . .	45
2. Одинарный замыкающий символ подчеркивания: <code>var_</code> . . . . .	47
3. Двойной начальный символ подчеркивания: <code>__var</code> . . . . .	48
Экскурс: что такое дандеры? . . . . .	52
4. Двойной начальный и замыкающий символ подчеркивания: <code>__var__</code> . . . . .	53
5. Одинарный символ подчеркивания: <code>_</code> . . . . .	54
Ключевые выводы . . . . .	55
2.5. Шокирующая правда о форматировании строковых значений . . . . .	56
№ 1. «Классическое» форматирование строковых значений. . . . .	57
№ 2. «Современное» форматирование строковых значений. . . . .	58
№ 3. Интерполяция литеральных строк (Python 3.6+) . . . . .	60
№ 4. Шаблонные строки . . . . .	62
Какой метод форматирования строк мне использовать? . . . . .	63
Ключевые выводы . . . . .	64
2.6. Пасхалка «Дзен Python» . . . . .	64
Дзен Python от Тима Питерса . . . . .	65

## **Глава 3. Эффективные функции . . . . . 66**

3.1. Функции Python — это объекты первого класса . . . . .	66
Функции — это объекты. . . . .	67
Функции могут храниться в структурах данных. . . . .	68
Функции могут передаваться другим функциям . . . . .	69
Функции могут быть вложенными. . . . .	70

Функции могут захватывать локальное состояние. . . . .	72
Объекты могут вести себя как функции . . . . .	74
Ключевые выводы . . . . .	75
3.2. Лямбды — это функции одного выражения . . . . .	75
Лямбды в вашем распоряжении . . . . .	77
А может, не надо.... . . . . .	78
Ключевые выводы . . . . .	80
3.3. Сила декораторов . . . . .	80
Основы декораторов Python . . . . .	82
Декораторы могут менять поведение . . . . .	84
Короткая пауза . . . . .	86
Применение многочисленных декораторов к функции . . . . .	86
Декорирование функций, принимающих аргументы . . . . .	88
Ключевые выводы . . . . .	91
3.4. Веселье с *args и **kwargs . . . . .	92
Переадресация необязательных или именованных аргументов . . . . .	93
Ключевые выводы . . . . .	95
3.5. Распаковка аргументов функции . . . . .	96
Ключевые выводы . . . . .	98
3.6. Здесь нечего возвращать . . . . .	98
Ключевые выводы . . . . .	101

## **Глава 4. Классы и ООП . . . . . 102**

4.1. Сравнения объектов: is против == . . . . .	102
4.2. Преобразование строк (каждому классу по __repr__) . . . . .	104
Метод __str__ против __repr__ . . . . .	107
Почему каждый класс нуждается в __repr__ . . . . .	110
Отличия Python 2.x: __unicode__ . . . . .	112
Ключевые выводы . . . . .	113

4.3. Определение своих собственных классов-исключений . . . . .	114
Ключевые выводы . . . . .	117
4.4. Клонирование объектов для дела и веселья . . . . .	118
Создание мелких копий . . . . .	119
Создание глубоких копий . . . . .	121
Копирование произвольных объектов . . . . .	122
Ключевые выводы . . . . .	125
4.5. Абстрактные базовые классы держат наследование под контролем . . . . .	125
Ключевые выводы . . . . .	128
4.6. Чем полезны именованные кортежи . . . . .	129
Именованные кортежи спешат на помощь . . . . .	130
Создание производных от Namedtuple подклассов . . . . .	133
Встроенные вспомогательные методы . . . . .	134
Когда использовать именованные кортежи. . . . .	135
Ключевые выводы . . . . .	135
4.7. Переменные класса против переменных экземпляра: подводные камни . . . . .	136
Пример без собак . . . . .	139
Ключевые выводы . . . . .	141
4.8. Срыв покровов с методов экземпляра, методов класса и статических методов . . . . .	142
Методы экземпляра . . . . .	143
Методы класса . . . . .	143
Статические методы . . . . .	144
Посмотрим на них в действии! . . . . .	144
Фабрики аппетитной пиццы с @classmethod . . . . .	147
Когда использовать статические методы . . . . .	149
Ключевые выводы . . . . .	151

<b>Глава 5. Общие структуры данных Python</b> . . . . .	<b>153</b>
5.1. Словари, ассоциативные массивы и хеш-таблицы . . . . .	155
dict — ваш дежурный словарь . . . . .	156
collections.OrderedDict — помнят порядок вставки ключей . . . . .	157
collections.defaultdict — возвращает значения, заданные по умолчанию для отсутствующих ключей. . . . .	158
collections.ChainMap — производит поиск в многочисленных словарях как в одной таблице соответствия . . . . .	159
types.MappingProxyType — обертка для создания словарей только для чтения . . . . .	159
Словари в Python: заключение . . . . .	160
Ключевые выводы . . . . .	161
5.2. Массивоподобные структуры данных . . . . .	161
list — изменяемые динамические массивы . . . . .	162
tuple — неизменяемые контейнеры . . . . .	163
array.array — элементарные типизированные массивы . . . . .	164
str — неизменяемые массивы символов Юникода . . . . .	165
bytes — неизменяемые массивы одиночных байтов . . . . .	167
bytearray — изменяемые массивы одиночных байтов . . . . .	168
Ключевые выводы . . . . .	169
5.3. Записи, структуры и объекты переноса данных . . . . .	170
dict — простые объекты данных . . . . .	171
tuple — неизменяемые группы объектов. . . . .	172
Написание собственного класса — больше работы, больше контроля . . . .	174
collections.namedtuple — удобные объекты данных . . . . .	175
typing.NamedTuple — усовершенствованные именованные кортежи . . . .	177
struct.Struct — сериализованные C-структуры. . . . .	178
types.SimpleNamespace — причудливый атрибутивный доступ . . . . .	179
Ключевые выводы . . . . .	180

5.4. Множества и мультимножества . . . . .	181
set — ваше дежурное множество . . . . .	182
frozenset — неизменяемые множества . . . . .	183
collections.Counter — мультимножества . . . . .	183
Ключевые выводы . . . . .	184
5.5. Стеки (с дисциплиной доступа LIFO) . . . . .	185
list — простые встроенные стеки . . . . .	186
collections.deque — быстрые и надежные стеки. . . . .	187
deque.LifoQueue — семантика блокирования для параллельных вычислений . . . . .	188
Сравнение реализаций стека в Python . . . . .	189
Ключевые выводы . . . . .	190
5.6. Очереди (с дисциплиной доступа FIFO) . . . . .	190
list — ужасно меееедленная очередь . . . . .	192
collections.deque — быстрые и надежные очереди . . . . .	193
queue.Queue — семантика блокирования для параллельных вычислений . . . . .	194
multiprocessing.Queue — очереди совместных заданий . . . . .	195
Ключевые выводы . . . . .	196
5.7. Очереди с приоритетом . . . . .	196
list — поддержание сортируемой очереди вручную . . . . .	197
heapq — двоичные кучи на основе списка . . . . .	198
queue.PriorityQueue — красивые очереди с приоритетом. . . . .	199
Ключевые выводы . . . . .	200

## **Глава 6. Циклы и итерации . . . . . 201**

6.1. Написание питоновских циклов . . . . .	201
Ключевые выводы . . . . .	204
6.2. Осмысление включений . . . . .	205
Ключевые выводы . . . . .	208

6.3. Нарезки списков и суши-оператор . . . . .	209
Ключевые выводы . . . . .	211
6.4. Красивые итераторы . . . . .	212
Бесконечное повторение . . . . .	213
Как циклы for-in работают в Python? . . . . .	215
Более простой класс-итератор . . . . .	218
Кто же захочет без конца выполнять итерации . . . . .	219
Совместимость с Python 2.x . . . . .	223
Ключевые выводы . . . . .	224
6.5. Генераторы — это упрощенные итераторы . . . . .	224
Бесконечные генераторы . . . . .	225
Генераторы, которые прекращают генерацию . . . . .	227
Ключевые выводы . . . . .	231
6.6. Выражения-генераторы . . . . .	231
Выражения-генераторы против включений в список . . . . .	233
Фильтрация значений . . . . .	235
Встраиваемые выражения-генераторы . . . . .	236
Слишком много хорошего... . . . .	236
Ключевые выводы . . . . .	238
6.7. Цепочки итераторов . . . . .	238
Ключевые выводы . . . . .	241

## **Глава 7. Трюки со словарем . . . . . 242**

7.1. Значения словаря, принимаемые по умолчанию . . . . .	242
Ключевые выводы . . . . .	245
7.2. Сортировка словарей для дела и веселья . . . . .	245
Ключевые выводы . . . . .	248
7.3. Имитация инструкций выбора на основе словарей . . . . .	248
Ключевые выводы . . . . .	253

7.4. Самое сумасшедшее выражение-словарь на западе . . . . .	253
Ключевые выводы . . . . .	260
7.5. Так много способов объединить словари . . . . .	260
Ключевые выводы . . . . .	263
7.6. Структурная печать словаря . . . . .	263
Ключевые выводы . . . . .	265

**Глава 8. Питоновские методы  
повышения производительности . . . . . 266**

8.1. Исследование модулей и объектов Python . . . . .	266
Ключевые выводы . . . . .	269
8.2. Изоляция зависимостей проекта при помощи Virtualenv . . . . .	270
Виртуальные среды спешат на помощь. . . . .	271
Ключевые выводы . . . . .	274
8.3. По ту сторону байткода . . . . .	275
Ключевые выводы . . . . .	279

**Глава 9. Итоги . . . . . 280**

9.1. Бесплатные еженедельные советы для разработчиков на Python . . . . .	281
9.2. PythonistaCafe: сообщество разработчиков на Python . . . . .	282