

# ОГЛАВЛЕНИЕ

<b>Об авторе</b> .....	15
<b>Введение</b> .....	16
Для кого эта книга? .....	17
О книге .....	18
Как пользоваться этой книгой .....	19
Номера строк и отступов .....	19
Длинные строки кода .....	20
Загрузка и установка Python .....	21
Запуск интерактивной среды разработки .....	22
Поиск справочной документации .....	23
<b>Глава 1. Интерактивная среда разработки</b> .....	24
Немного арифметики .....	24
Целые числа и числа с плавающей запятой.....	25
Выражения .....	26
Вычисление выражений.....	26
Синтаксические ошибки.....	27
Сохранение значений в переменных .....	28
Заключение.....	33
<b>Глава 2. Написание программ</b> .....	34
Строковые значения .....	34
Конкатенация строк.....	35
Написание кода в среде разработки .....	36
Создание программы «Привет, мир!».....	37
Сохранение программы.....	38
Запуск программы.....	39
Как работает программа «Привет, мир!».....	40
Комментарии для программиста .....	40
Функции: Мини-программы внутри программ.....	41
Функция <code>print()</code> .....	41
Функция <code>input()</code> .....	41
Выражения в вызовах функций .....	42
Завершение программы.....	42
Имена переменных.....	43
Заключение.....	44
<b>Глава 3. Игра «Угадай число»</b> .....	45
Пример запуска игры «Угадай число» .....	46
Исходный код игры «Угадай число» .....	46
Импорт модуля <code>random</code> .....	47
Генерация случайных чисел при помощи функции <code>random.randint()</code> .....	49
Приветствие игрока .....	51

Инструкции управления потоком.....	51
Использование циклов для повторения кода.....	51
Группировка в блоки.....	52
Создание циклов с инструкцией <code>for</code> .....	53
Игрок угадывает число.....	55
Преобразование значений при помощи функций <code>int()</code> , <code>float()</code> и <code>str()</code> .....	55
Логический тип данных.....	57
Операторы сравнения.....	58
Проверка условий и определение истинности/ложности.....	58
Эксперименты с логическими операторами, операторами сравнения и условиями.....	59
Разница между операторами <code>=</code> и <code>==</code> .....	61
Инструкции <code>if</code> .....	61
Выход из цикла до его завершения при помощи инструкции <code>break</code> .....	62
Проверка, победил ли игрок.....	62
Проверка, проиграл ли игрок.....	63
Заключение.....	64
<b>Глава 4. Программа-шутник.....</b>	<b>65</b>
Пример запуска программы «Шутки».....	65
Исходный код программы «Шутки».....	66
Как работает код.....	66
Экранируемые символы.....	67
Одинарные и двойные кавычки.....	68
Параметр <code>end</code> функции <code>print()</code> .....	70
Заключение.....	71
<b>Глава 5. Игра «Царство драконов».....</b>	<b>72</b>
Как играть в «Царство драконов».....	72
Пример запуска игры «Царство драконов».....	73
Блок-схема игры «Царство драконов».....	73
Исходный код игры «Царство драконов».....	74
Импорт модулей <code>random</code> и <code>time</code> .....	76
Функции в игре «Царство драконов».....	76
Инструкции <code>def</code> .....	76
Вызов функции.....	77
Где указывать определение функций?.....	77
Многострочный текст.....	78
Выполнение циклов с помощью инструкций <code>while</code> .....	79
Логические операторы.....	80
Оператор <code>and</code> .....	80
Оператор <code>or</code> .....	81
Оператор <code>not</code> .....	82
Вычисление логических операций.....	82
Возвращаемые значения функций.....	84
Глобальная и локальная области видимости переменных.....	84
Параметры функции.....	86
Отображение результатов игры.....	87
Определение пещеры с дружелюбным драконом.....	88

Игровой цикл .....	89
Вызов функций в программе.....	90
Запрос «сыграть снова» .....	91
Заключение.....	91
<b>Глава 6. Использование отладчика .....</b>	<b>93</b>
Типы багов.....	93
Отладка .....	95
Запуск отладчика.....	95
Пошаговое выполнение программы с помощью отладчика.....	97
Область глобальных переменных.....	97
Область локальных переменных.....	98
Обычное выполнение и завершение работы.....	98
Навигация по коду .....	98
Поиск багов .....	100
Установка точек останова.....	104
Использование точек останова .....	105
Заключение.....	107
<b>Глава 7. Проектирование игры «Виселица» с помощью блок-схем .....</b>	<b>108</b>
Правила игры «Виселица» .....	108
Пример запуска игры «Виселица» .....	109
ASCII-графика .....	110
Проектирование игры с помощью блок-схем .....	111
Создание блок-схем .....	112
Ветвление в блок-схемах .....	113
Заканчиваем или начинаем игру сначала .....	115
Следующая попытка.....	115
Обратная связь с игроком .....	117
Заключение.....	118
<b>Глава 8. Написание кода игры «Виселица» .....</b>	<b>119</b>
Исходный код игры «Виселица» .....	119
Импорт модуля <code>random</code> .....	123
Константы .....	123
Списки.....	124
Доступ к элементам по их индексам .....	125
Индекс за пределами диапазона и ошибка <code>IndexError</code> .....	126
Присваивание индексов элементам .....	126
Конкатенация списков .....	126
Оператор <code>in</code> .....	127
Вызов методов .....	127
Методы списков <code>reverse()</code> и <code>append()</code> .....	128
Строковый метод <code>split()</code> .....	128
Получение секретного слова из списка.....	129
Отображение игрового поля для игрока .....	130
Функции <code>list()</code> и <code>range()</code> .....	131
Срезы списков и строк.....	132
Вывод секретного слова с пробелами .....	134

Получение предположений игрока .....	136
Строковые методы <code>lower()</code> и <code>upper()</code> .....	136
Завершение цикла <code>while</code> .....	138
Инструкции <code>elif</code> .....	138
Проверка допустимости предположения игрока .....	139
Предложение игроку сыграть заново .....	140
Обзор функций игры .....	141
Игровой цикл .....	141
Вызов функции <code>displayBoard()</code> .....	142
Ввод игроком угадываемой буквы .....	142
Проверка наличия буквы в секретном слове .....	143
Проверка — не победил ли игрок .....	143
Обработка ошибочных предположений .....	144
Проверка — не проиграл ли игрок .....	144
Завершение или перезагрузка игры .....	145
Заключение .....	146
<b>Глава 9. Доработка игры «Виселица» .....</b>	<b>147</b>
Увеличение числа угадываний .....	147
Словари .....	148
Определение размера словаря с помощью функции <code>len()</code> .....	149
Различия между списком и словарем .....	150
Методы словаря <code>keys()</code> и <code>values()</code> .....	151
Использование словаря слов в игре «Виселица» .....	152
Случайный выбор из списка .....	152
Удаление элементов списка .....	154
Множественное присваивание .....	156
Выбор игроком категории слов .....	157
Заключение .....	158
<b>Глава 10. Игра «Крестики-нолики» .....</b>	<b>159</b>
Пример запуска игры «Крестики-нолики» .....	160
Исходный код игры «Крестики-нолики» .....	161
Проектирование программы .....	166
Данные для прорисовки игрового поля .....	166
Стратегия игры ИИ .....	167
Импорт модуля <code>random</code> .....	168
Вывод игрового поля на экран .....	169
Предоставление игроку выбора между «X» или «O» .....	170
Выбор — кто будет ходить первым .....	171
Размещение меток на игровом поле .....	171
Ссылки на список .....	172
Использование ссылок на списки в функции <code>makeMove()</code> .....	175
Проверка — не победил ли игрок .....	175
Дублирование данных игрового поля .....	178
Проверка — свободна ли клетка игрового поля .....	178
Разрешение игроку сделать ход .....	179
Вычисление по короткой схеме .....	180

Выбор хода из списка.....	182
Значение None .....	183
Создание искусственного интеллекта .....	184
Проверка — сможет ли компьютер победить, сделав ход.....	185
Проверка — сможет ли игрок победить, сделав ход.....	185
Проверка угловых, центральной и боковых клеток (в порядке очереди) .....	186
Проверка — заполнено ли поле.....	187
Игровой цикл .....	187
Выбор буквы игрока и того, кто будет ходить первым .....	188
Переменная <code>turn</code> в значении 'Человек' .....	188
Переменная <code>turn</code> в значении 'Компьютер' .....	190
Предложение игроку сыграть заново.....	190
Заключение.....	191
<b>Глава 11. Дедуктивная игра «Холодно-горячо» .....</b>	<b>192</b>
Пример запуска игры «Холодно-горячо» .....	193
Исходный код игры «Холодно-горячо».....	193
Блок-схема игры «Холодно-горячо» .....	195
Импорт модуля <code>random</code> и определение функции <code>getSecretNum()</code> .....	196
Перетасовка уникального набора цифр .....	197
Изменение порядка элементов списка с помощью функции <code>random.shuffle()</code> .....	197
Получение секретного числа из перетасованных цифр.....	198
Расширенные операторы присваивания.....	198
Подсчет выдаваемых подсказок.....	200
Метод списка <code>sort()</code> .....	201
Строковый метод <code>join()</code> .....	202
Проверка на содержание в строке только чисел.....	203
Начало игры .....	203
Интерполяция строк.....	204
Игровой цикл .....	205
Получение предположения игрока.....	206
Получение подсказок в зависимости от предположения игрока.....	206
Проверка победы или поражения игрока.....	207
Предложение сыграть снова.....	207
Заключение.....	208
<b>Глава 12. Декартова система координат .....</b>	<b>209</b>
Сетки и декартовы координаты.....	209
Отрицательные числа .....	211
Система координат компьютерного экрана.....	213
Математические хитрости.....	213
Хитрость 1: минус «съедает» плюс слева от себя.....	214
Хитрость 2: два минуса в сумме дают плюс .....	214
Хитрость 3: два слагаемых числа можно переставлять местами .....	214
Абсолютные величины и функция <code>abs()</code> .....	215
Заключение.....	216

<b>Глава 13. Игра «Охотник за сокровищами»</b> .....	<b>217</b>
Пример запуска игры «Охотник за сокровищами» .....	218
Исходный код игры «Охотник за сокровищами» .....	222
Проектирование программы .....	227
Импорт модулей <code>random</code> , <code>sys</code> и <code>math</code> .....	227
Создание поля для новой игры .....	228
Генерация игрового поля .....	229
Изображение координат $x$ вдоль верхней части поля .....	230
Рисование океана .....	231
Вывод ряда в океане .....	232
Изображение координат $x$ вдоль нижней части игрового поля .....	233
Создание случайных сундуков с сокровищами .....	233
Определение допустимости хода .....	234
Отражение хода на игровом поле .....	234
Поиск ближайшего сундука с сокровищами .....	235
Удаление значений с помощью метода списка <code>remove()</code> .....	238
Получение хода игрока .....	239
Вывод игроку инструкций по игре .....	241
Игровой цикл .....	242
Демонстрация игроку статуса игры .....	243
Обработка хода игрока .....	243
Нахождение затонувшего сундука с сокровищами .....	244
Проверка победы игрока .....	245
Проверка проигрыша игрока .....	245
Завершение работы программы с помощью функции <code>sys.exit()</code> .....	246
Заключение .....	246
<b>Глава 14. Шифр Цезаря</b> .....	<b>248</b>
Криптография и шифрование .....	248
Как работает шифр Цезаря .....	249
Пример запуска программы «Шифр Цезаря» .....	251
Исходный код программы «Шифр Цезаря» .....	252
Установление максимальной длины ключа .....	253
Выбор между шифрованием и расшифровыванием сообщения .....	254
Получение сообщения от игрока .....	254
Получение ключа от игрока .....	255
Шифрование/расшифровывание сообщения .....	255
Нахождение переданных строк с помощью строчного метода <code>find()</code> .....	256
Шифрование/расшифровка каждой буквы .....	257
Запуск программы .....	258
Полный перебор .....	259
Добавление режима полного перебора .....	260
Заключение .....	261
<b>Глава 15. Игра «Реверси»</b> .....	<b>263</b>
Как играть в «Реверси» .....	263
Пример запуска игры «Реверси» .....	266
Исходный код игры «Реверси» .....	268

Импорт модулей и создание констант .....	275
Структура данных игрового поля .....	275
Отображение на экране структуры данных игрового поля.....	276
Создание структуры данных нового игрового поля.....	277
Проверка допустимости хода .....	278
Проверка каждого из восьми направлений .....	279
Определение наличия фишек, которые можно перевернуть .....	280
Проверка допустимости координат .....	282
Получение списка со всеми допустимыми ходами .....	282
Вызов функции <code>bool()</code> .....	283
Получение игрового счета.....	284
Получение сделанного игроком выбора фишки.....	285
Определение первого игрока .....	285
Помещение фишки на поле .....	286
Создание копии структуры данных игрового поля.....	286
Определение того, находится ли клетка в углу .....	287
Получение хода игрока.....	287
Получение хода компьютера.....	290
Разработка стратегии с угловыми ходами .....	290
Получения списка самых результативных ходов .....	291
Вывод игрового счета на экран .....	292
Начало игры .....	292
Проверка на попадание в тупик .....	293
Выполнение хода игроком .....	294
Выполнение хода компьютером .....	295
Игровой цикл .....	296
Предложение сыграть снова .....	297
Заключение.....	298
<b>Глава 16. Искусственный интеллект игры «Реверси» .....</b>	<b>299</b>
Компьютер против компьютера .....	300
Пример запуска модели 1 .....	300
Исходный код модели 1 .....	301
Удаление приглашений для игрока и добавление игрока-компьютера.....	303
Компьютер против компьютера — несколько партий.....	304
Пример запуска модели 2 .....	304
Исходный код модели 2 .....	305
Отслеживание результатов партий .....	306
Отключение вызовов функции <code>print()</code> .....	307
Оценка искусственных интеллектов в процентном соотношении.....	307
Деление приводит к получению числа с плавающей запятой .....	308
Функция <code>round()</code> .....	309
Сравнение различных алгоритмов ИИ .....	309
Исходный код модели 3 .....	310
Принципы работы искусственных интеллектов в модели 3 .....	312
ИИ лучшего углового хода .....	312
ИИ худшего хода.....	312

ИИ случайного хода .....	313
Проверка на граничные ходы .....	314
ИИ лучшего углового-граничного хода .....	314
Сравнение искусственных интеллектов .....	315
ИИ худшего хода против ИИ лучшего углового хода .....	315
ИИ случайного хода против ИИ лучшего углового хода .....	316
ИИ лучшего углового-граничного хода против ИИ лучшего углового хода .....	317
Заключение .....	318
<b>Глава 17. Создание графики .....</b>	<b>319</b>
Установка pygame .....	320
Привет, pygame! .....	320
Пример запуска pygame-программы «Привет, мир!» .....	321
Исходный код pygame-программы «Привет, мир!» .....	322
Импорт модуля pygame .....	323
Инициализация pygame .....	324
Настройка окна pygame .....	324
Кортежи .....	325
Объекты поверхности .....	326
Работа с цветом .....	326
Вывод текста в окне pygame .....	327
Использование шрифтов для оформления текста .....	327
Рендеринг объекта Font .....	328
Настройка местоположения текста с помощью атрибутов Rect .....	329
Заливка цветом объекта Surface .....	331
Функции рисования pygame .....	332
Рисование многоугольника .....	332
Рисование линии .....	333
Рисование круга .....	333
Рисование эллипса .....	334
Рисование прямоугольника .....	334
Окрашивание пикселей .....	335
Метод blit () для объектов Surface .....	336
Вывод объекта Surface на экран .....	336
События и игровой цикл .....	336
Получение объектов Event .....	337
Выход из программы .....	338
Заключение .....	338
<b>Глава 18. Анимированная графика .....</b>	<b>339</b>
Пример запуска игры программы .....	339
Исходный код программы .....	339
Перемещение и контроль отскока блоков .....	342
Создание констант .....	343
Константы для направлений .....	344
Константы для цвета .....	345
Создание структуры данных блока .....	345

Игровой цикл .....	346
Обработка решения игрока завершить игру .....	346
Перемещение каждого блока.....	347
Управление отскакиванием блока.....	349
Отображение в окне блоков в новых положениях .....	350
Отображение окна на экране .....	350
Заключение.....	351
<b>Глава 19. Обнаружение столкновений.....</b>	<b>352</b>
Пример работы программы .....	353
Исходный код программы.....	353
Импорт модулей .....	356
Использование объекта <code>clock</code> для управления скоростью работы программы.....	357
Настройка окна и структур данных.....	357
Создание переменных для отслеживания перемещения.....	359
Обработка событий.....	359
Обработка события <code>KEYDOWN</code> .....	361
Обработка события <code>KEYUP</code> .....	363
Телепортация игрока .....	364
Добавление новых блоков «еды» .....	364
Перемещение игрока по окну .....	365
Отображение блока игрока в окне.....	366
Проверка на столкновения .....	366
Отображение блоков «еды».....	367
Заключение.....	368
<b>Глава 20. Использование звуков и изображений .....</b>	<b>369</b>
Добавление изображений с помощью спрайтов .....	369
Графические и звуковые файлы .....	370
Пример запуска игры.....	371
Исходный код программы.....	371
Настройка окна и создание структуры данных.....	375
Добавление спрайта.....	375
Изменение размера спрайта.....	376
Установка музыки и звуков.....	376
Добавление аудиофайлов .....	376
Включение/отключение звука.....	377
Отображение спрайта игрока в окне.....	378
Проверка на столкновения.....	378
Отображение спрайтов вишен в окне .....	379
Заключение.....	380
<b>Глава 21. Игра «Ловкач» с графикой и звуком.....</b>	<b>381</b>
Обзор основных типов данных <code>pygame</code> .....	381
Пример запуска игры «Ловкач» .....	383
Исходный код игры «Ловкач».....	383
Импорт модулей .....	388
Создание констант .....	389

Определение функций.....	390
Завершение игры и добавление паузы.....	390
Отслеживание столкновений со злодеями.....	391
Отображение текста в окне.....	392
Инициализация ругаме и настройка окна .....	393
Установка шрифтов, изображений и звуков.....	394
Отображение начального экрана.....	395
Начало игры.....	396
Игровой цикл .....	398
Обработка событий клавиатуры.....	398
Обработка событий мыши.....	400
Добавление новых злодеев .....	401
Перемещение спрайтов игрока и злодеев.....	402
Реализация чит-кодов .....	404
Удаление спрайтов злодеев.....	404
Отображение окна.....	405
Отображение очков игрока .....	405
Отображение спрайтов игрока и злодеев .....	406
Проверка на столкновения.....	406
Экран окончания игры.....	407
Изменение игры «Ловкач» .....	408
Заключение.....	409
<b>Предметный указатель .....</b>	<b>410</b>

# 1

## ИНТЕРАКТИВНАЯ СРЕДА РАЗРАБОТКИ



Прежде чем приступить к разработке игр, вам нужно изучить несколько базовых концепций программирования. Эту главу вы начнете с того, что научитесь использовать интерактивную среду разработки Python и выполнять базовые арифметические задачи.

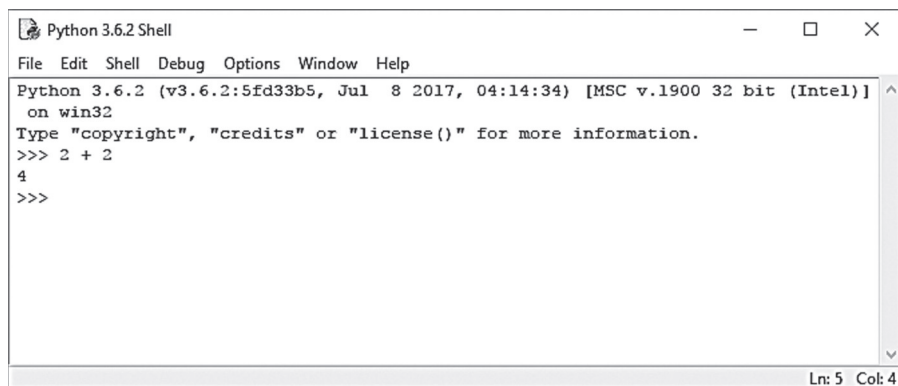
### В ЭТОЙ ГЛАВЕ РАССМАТРИВАЮТСЯ СЛЕДУЮЩИЕ ТЕМЫ:

- Операторы
- Целые числа и числа с плавающей запятой
- Значения
- Выражения
- Синтаксические ошибки в коде
- Сохранение значений в переменных

### Немного арифметики

Запустите IDLE, выполнив указания из раздела «Запуск интерактивной среды разработки» во введении. На первых порах вы будете использовать Python для решения простых арифметических задач. IDLE может быть использована как калькулятор. Введите  $2 + 2$  в интерактивной оболочке после символов приглашения, `>>>`, и нажмите клавишу **Enter**. (На некоторых клавиатурах эта клавиша обозначена как **Return**.) На рис. 1.1 показано, как

эта арифметическая задача выглядит в интерактивной оболочке — обратите внимание, что получен ответ в виде числа 4.



```
Python 3.6.2 Shell
File Edit Shell Debug Options Window Help
Python 3.6.2 (v3.6.2:5fd33b5, Jul 8 2017, 04:14:34) [MSC v.1900 32 bit (Intel)]
on win32
Type "copyright", "credits" or "license()" for more information.
>>> 2 + 2
4
>>>
```

Рис. 1.1. Ввод  $2 + 2$  в интерактивной оболочке

Эта арифметическая задача — простая инструкция программирования. Символ «плюс» (+) указывает компьютеру сложить 2 и 2. Компьютер выполняет операцию и отвечает числом 4 в следующей строке. Таблица 1.1 содержит доступные в Python арифметические операторы.

Таблица 1.1. Арифметические операторы

Оператор	Операция
+	Прибавить
-	Вычесть
*	Умножить
/	Разделить

Символ минус (-) вычитает числа, звездочка (\*) умножает, а косая черта (/) делит. При использовании таким образом, знаки +, -, \* и / называются *операторами*. Операторы сообщают Python, какую операцию необходимо произвести над числами.

### Целые числа и числа с плавающей запятой

*Целые числа* (на английском int) представляют собой числа, такие как 4, 99 или 0. *Числа с плавающей запятой* (на английском float) представляют собой дроби или числа с десятичными точками, такие как 3.5, 42.1 и 5.0. Для Python число 5 — целое, а 5.0 — число с плавающей запятой.

Эти числа называют *значениями*. Позже вы узнаете о других видах значений помимо чисел. В арифметической задаче, введенной в интерактивной оболочке, 2 и 2 — это целые (или *целочисленные*) значения.

## Выражения

Арифметическая задача  $2 + 2$  — пример *выражения*. Как демонстрирует рис. 1.2, выражения состоят из значений (чисел), связанных между собой с помощью операторов (арифметических символов), что создает новое значение, которое может быть использовано в коде. Компьютер способен вычислить значения миллиардов выражений в секунду.

Введите следующие выражения в интерактивной оболочке, нажимая клавишу **Enter** после каждой строки.



**Рис. 1.2.** Выражение, состоящее из значений и операторов

---

```
>>> 2+2+2+2+2
10
>>> 8*6
48
>>> 10-5+6
11
>>> 2 + 2
4
```

---

Все эти выражения выглядят как обычные арифметические уравнения, но обратите внимание на пробелы в примере  $2 + 2$ . В Python вы можете поставить любое количество пробелов между значениями и операторами. Тем не менее вы всегда должны начинать инструкцию в начале строки без пробелов при вводе их в интерактивной оболочке.

## Вычисление выражений

Когда среда вычисляет выражение  $10 + 5$  и возвращает значение 15, это значит, что он его *вычислил*. Вычисление выражения сводит его к единому значению, так же как и решение арифметической задачи сводит ее к единственному числу — ответу. Например, два выражения  $10 + 5$  и  $10 + 3 + 2$  вычисляются с результатом 15.

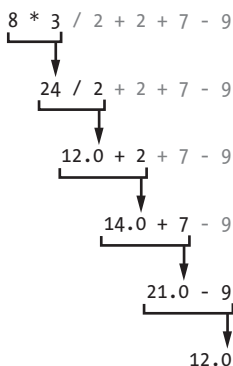
Когда Python вычисляет выражение, он придерживается того же порядка арифметических операций, какому вас научили в школе.

Правила следующие:

- Сначала вычисляются части выражения внутри круглых скобок.
- Умножение и деление выполняются перед сложением и вычитанием.
- Порядок действий слева направо.

Выражение  $1 + 2 * 3 + 4$  равно 11, а не 13, потому что сначала выполняется умножение  $2 * 3$ . Если бы выражение выглядело как  $(1 + 2) * (3 + 4)$ , оно бы равнялось 21, потому что части  $(1 + 2)$  и  $(3 + 4)$  находятся внутри скобок и вычисляются перед умножением.

Выражения могут быть любого размера, и Python вычислит значение каждого из них. Даже одиночные цифры — это выражения. Например, выражение 15 является выражением со значением 15. Выражение  $8 * 3 / 2 + 2 + 7 - 9$  равняется значению 12.0 после следующих вычислений:



Несмотря на то что компьютер выполняет все эти шаги, вы не видите их в интерактивной оболочке. Интерактивная среда разработки показывает вам только результат.

---

```
>>> 8 * 3 / 2 + 2 + 7 - 9
12.0
```

---

Обратите внимание, что выражения с оператором (/) деления Python в итоге приводит к числу с плавающей запятой; Например,  $24 / 2$  вычисляется как 12.0. Математические операции хотя бы с одним значением с плавающей запятой так же записываются и в дальнейшем, поэтому  $12.0 + 2$  вычисляется как 14.0.

## Синтаксические ошибки

Если в интерактивной оболочке вы напишете  $5 +$  и нажмете клавишу **Enter**, то получите следующее сообщение об ошибке:

---

```
>>> 5 +
```

```
SyntaxError: invalid syntax
```

---

Эта ошибка произошла потому, что `5 +` не является выражением. Выражения состоят из значений, соединенных операторами. В данном случае оператор сложения `+` подразумевает наличие значений с *двух* сторон. Сообщение об ошибке появляется, когда ожидаемое значение отсутствует.

«Синтаксическая ошибка» означает, что Python не понимает инструкцию, потому что вы задали ее неверно. При написании программ важно не только давать инструкции компьютеру, но и знать, как делать это правильно.

Впрочем, не переживайте из-за ошибок в коде. Ошибки не нанесут вреда вашему компьютеру. Просто правильно перепишите инструкцию в интерактивной оболочке после следующего приглашения `>>>`.

## Сохранение значений в переменных

Вы можете использовать полученные значения выражений, сохраняя их в *переменных*. Для простоты можете представить, что переменная — это такая «коробка», в которой вы можете хранить значения. Для сохранения значения выражения в переменной используйте *инструкцию присваивания*. Введите имя переменной, затем знак равенства (`=`), который называется *оператором присваивания*, и затем значение. Например, введите в интерактивной оболочке следующую команду:

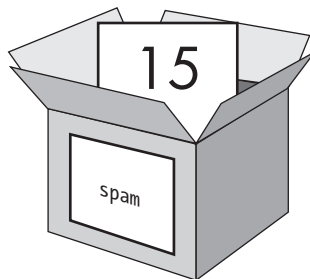
---

```
>>> spam = 15
```

```
>>>
```

---

В «коробке» переменной `spam` теперь хранится значение `15`, как показано на рис. 1.3.



**Рис. 1.3.** Переменные подобны «коробкам», которые могут содержать значения