

СОДЕРЖАНИЕ

Об авторе	23
О техническом рецензенте	23
Введение	25
Для кого предназначена эта книга	26
Исходные предположения	27
Что такое программирование	27
Что такое Python	28
Программисту не обязательно в совершенстве знать математику	29
Учиться программированию никогда не поздно	30
Программирование — творческий вид деятельности	30
Структура книги	31
Загрузка и установка Python	33
Загрузка и установка Mu	34
Запуск Mu	34
Запуск IDLE	35
Интерактивная оболочка	35
Установка сторонних модулей	36
Как получить справку	36
Правильно формулируйте вопросы, ответы на которые ищите	37
Файлы примеров	39
Резюме	40
Часть I. Основы программирования на языке Python	41
Глава 1. Основные понятия языка Python	43
Ввод выражений в интерактивной оболочке	44
Типы данных: целые числа, вещественные числа и строки	47
Конкатенация и репликация строк	48
Сохранение значений в переменных	49
Операции присваивания	49
Имена переменных	50
Ваша первая программа	52
Анализ программы	53
Комментарии	53
Функция <code>print()</code>	54
Функция <code>input()</code>	54
Вывод имени пользователя	55
Функция <code>len()</code>	55

Функции <code>str()</code> , <code>int()</code> и <code>float()</code>	56
Резюме	59
Контрольные вопросы	60
Глава 2. Порядок выполнения программы	61
Булевы значения	63
Операторы сравнения	63
Булевы операторы	65
Бинарные булевы операторы	65
Оператор <code>not</code>	66
Сочетание операторов сравнения и булевых операторов	67
Элементы структурирования программы	68
Условия	68
Блоки кода	68
Выполнение программы	69
Управляющие инструкции	69
Инструкция <code>if</code>	69
Инструкция <code>else</code>	70
Инструкция <code>elif</code>	71
Цикл <code>while</code>	77
Инструкция <code>break</code>	81
Инструкция <code>continue</code>	82
Цикл <code>for</code> и функция <code>range()</code>	85
Импорт модулей	89
Инструкция <code>from import</code>	90
Досрочное завершение программы с помощью функции <code>sys.exit()</code>	91
Короткая программа: угадай число	92
Короткая программа: камень, ножницы, бумага	94
Резюме	98
Контрольные вопросы	98
Глава 3. Функции	101
Инструкции <code>def</code> с параметрами	103
Терминология функций	104
Инструкция <code>return</code> и возвращаемые значения	104
Значение <code>None</code>	106
Именованные аргументы и функция <code>print()</code>	107
Стек вызовов	108
Локальная и глобальная области видимости	110
Локальные переменные не могут использоваться в глобальной области видимости	112
В локальных областях видимости не могут использоваться переменные из других локальных областей видимости	112

Глобальные переменные доступны из локальной области видимости	113
Локальные и глобальные переменные с одинаковыми именами	114
Инструкция <code>global</code>	115
Обработка исключений	117
Короткая программа: зигзаг	119
Резюме	122
Контрольные вопросы	122
Учебные проекты	123
Последовательность Коллатца	123
Проверка корректности ввода	124
Глава 4. Списки	125
Что такое список	126
Доступ к элементам списка с помощью индексов	126
Отрицательные индексы	128
Получение фрагмента списка с помощью среза	129
Определение длины списка с помощью функции <code>len()</code>	129
Изменение элементов списка с помощью индексов	130
Конкатенация и репликация списков	130
Удаление значений из списка с помощью инструкции <code>del</code>	130
Работа со списками	131
Использование циклов <code>for</code> со списками	133
Операторы <code>in</code> и <code>not in</code>	134
Трюк с групповым присваиванием	135
Использование функции <code>enumerate()</code> со списками	135
Использование функций <code>random.choice()</code> и <code>random.shuffle()</code> со списками	136
Комбинированные операторы присваивания	136
Методы	137
Поиск значения в списке с помощью метода <code>index()</code>	138
Добавление значений в список с помощью методов <code>append()</code> и <code>insert()</code>	138
Удаление значений из списка с помощью метода <code>remove()</code>	139
Сортировка списка с помощью метода <code>sort()</code>	140
Инверсия списка с помощью метода <code>reverse()</code>	141
Пример программы: Magic 8 Ball со списком	142
Списковые типы данных	143
Изменяемые и неизменяемые типы данных	144
Кортежи	146
Преобразование типов с помощью функций <code>list()</code> и <code>tuple()</code>	147
Ссылки	147
Тождественность и функция <code>id()</code>	150

Передача ссылок	151
Функции <code>copy()</code> и <code>deepcopy()</code>	152
Короткая программа: игра “Жизнь”	153
Резюме	159
Контрольные вопросы	159
Учебные проекты	160
Запятая в качестве разделителя	160
Эксперименты с монетой	160
Символьная сетка	161
Глава 5. Словари	163
Что такое словарь	164
Сравнение словарей и списков	164
Методы <code>keys()</code> , <code>values()</code> и <code>items()</code>	166
Проверка наличия ключа или значения в словаре	168
Метод <code>get()</code>	169
Метод <code>setdefault()</code>	169
Красивый вывод	171
Использование структур данных для моделирования реальных объектов	172
Поле для игры в “крестики-нолики”	173
Вложенные словари и списки	179
Резюме	180
Контрольные вопросы	180
Учебные проекты	181
Валидатор словаря для игры в шахматы	181
Инвентарь приключенческой игры	181
Функция добавления списка в словарь для приключенческой игры	182
Глава 6. Строки	185
Работа со строками	186
Строковые литералы	186
Индексирование строк и извлечение срезов	189
Использование операторов <code>in</code> и <code>not in</code> со строками	190
Вставка строк в другие строки	190
Полезные методы для работы со строками	191
Методы <code>upper()</code> , <code>lower()</code> , <code>isupper()</code> и <code>islower()</code>	191
Строковые методы <code>isX()</code>	193
Методы <code>startswith()</code> и <code>endswith()</code>	195
Методы <code>join()</code> и <code>split()</code>	196
Разбиение строк с помощью метода <code>partition()</code>	197
Выравнивание текста с помощью методов <code>rjust()</code> , <code>ljust()</code> и <code>center()</code>	198
Удаление пробелов с помощью методов <code>strip()</code> , <code>rstrip()</code> и <code>rstrip()</code>	200

Получение числовых значений символов с помощью функций <code>ord()</code> и <code>chr()</code>	201
Копирование и вставка строк с помощью модуля <code>pyperclip</code>	202
Проект: автоматическая рассылка сообщений с помощью нескольких буферов обмена	203
Шаг 1. Проектирование программы и структур данных	203
Шаг 2. Обработка аргументов командной строки	204
Шаг 3. Копирование фразы в буфер	204
Проект: добавление маркеров в разметку Wiki-документов	205
Шаг 1. Копирование и вставка посредством буфера обмена	206
Шаг 2. Разбивка текста на строки и добавление звездочек	207
Шаг 3. Объединение измененных строк	208
Короткая программа: пороссячья латынь	208
Резюме	212
Контрольные вопросы	213
Учебные проекты	214
Табличный вывод данных	214
Боты <code>Zombie Dice</code>	215
Часть II. Автоматизация задач	219
Глава 7. Регулярные выражения	221
Поиск образцов текста без использования регулярных выражений	222
Поиск образцов текста с помощью регулярных выражений	224
Создание объектов <code>Regex</code>	225
Поиск соответствий объектам <code>Regex</code>	226
Пошаговая процедура	226
Другие шаблоны регулярных выражений	227
Создание групп с помощью круглых скобок	227
Выбор альтернативных групп с помощью канала	229
Указание необязательной группы с помощью вопросительного знака	230
Указание группы, повторяющейся нуль или несколько раз, с помощью звездочки	231
Указание группы, повторяющейся один или несколько раз, с помощью знака “плюс”	231
Указание количества повторений с помощью фигурных скобок	232
Жадный и нежадный виды поиска	233
Метод <code>findall()</code>	234
Символьные классы	235
Создание собственных символьных классов	235
Символ <code>^</code> и знак доллара	236

Символ подстановки	237
Поиск любого текста с помощью комбинации “точка — звездочка”	238
Поиск символов новой строки с помощью точки	239
Сводка синтаксиса регулярных выражений	239
Поиск без учета регистра	240
Замена строк с помощью метода <code>sub()</code>	241
Работа со сложными регулярными выражениями	241
Комбинация констант <code>re.IGNORECASE</code> , <code>re.DOTALL</code> и <code>re.VERBOSE</code>	242
Проект: извлечение телефонных номеров и адресов электронной почты	243
Шаг 1. Создание регулярного выражения для поиска телефонных номеров	244
Шаг 2. Создание регулярного выражения для поиска адресов электронной почты	245
Шаг 3. Поиск всех совпадений в тексте, скопированном в буфер обмена	246
Шаг 4. Объединение совпадений в одну строку для копирования в буфер обмена	247
Запуск программы	248
Идеи для создания похожих программ	248
Резюме	249
Контрольные вопросы	249
Учебные проекты	251
Обнаружение даты	251
Выявление сильных паролей	252
Версия метода <code>strip()</code> , использующая регулярные выражения	252
Глава 8. Проверка ввода	253
Модуль <code>PyInputPlus</code>	254
Именованные аргументы <code>min</code> , <code>max</code> , <code>greaterThan</code> и <code>lessThan</code>	257
Именованный аргумент <code>blank</code>	257
Именованные аргументы <code>limit</code> , <code>timeout</code> и <code>default</code>	258
Именованные аргументы <code>allowRegexes</code> и <code>blockRegexes</code>	259
Передача пользовательской функции проверки в функцию <code>inputCustom()</code>	260
Проект: как занять дурака на несколько часов	261
Проект: тест на умножение	263
Резюме	265
Контрольные вопросы	266
Учебные проекты	266
Изготовитель бутербродов	267
Собственный тест на умножение	267

Глава 9. Чтение и запись файлов	269
Файлы и папки	270
Использование обратной косой черты в Windows и косой черты в macOS и Linux	271
Использование оператора / для объединения путей	272
Текущий каталог	274
Домашний каталог	275
Абсолютные и относительные пути	275
Создание новых папок с помощью функции <code>os.makedirs()</code>	276
Обработка абсолютных и относительных путей	277
Получение отдельных частей пути	279
Определение размеров файлов и содержимого папок	281
Изменение списка файлов с помощью шаблонов	282
Проверка существования пути	284
Процесс чтения и записи файлов	285
Открытие файла с помощью функции <code>open()</code>	286
Чтение содержимого файла	287
Запись в файл	288
Сохранение переменных с помощью модуля <code>shelve</code>	289
Сохранение переменных с помощью функции <code>pprint.pformat()</code>	291
Проект: генерирование случайных билетов	292
Шаг 1. Сохранение данных в словаре	293
Шаг 2. Создание файлов билетов и перемешивание вопросов	294
Шаг 3. Создание вариантов ответов	295
Шаг 4. Запись содержимого в файлы билетов и ключей ответов	296
Проект: множественный буфер обмена	298
Шаг 1. Комментарии и настройка хранилища	299
Шаг 2. Сохранение содержимого буфера обмена с ключевым словом	299
Шаг 3. Построение списка ключевых слов и загрузка содержимого, ассоциированного с ключевым словом	300
Резюме	301
Контрольные вопросы	302
Учебные проекты	302
Расширение возможностей множественного буфера обмена	302
Программа Mad Libs	302
Поиск с помощью регулярных выражений	303
Глава 10. Управление файлами	305
Модуль <code>shutil</code>	306
Копирование файлов и папок	306
Перемещение и переименование файлов и папок	307
Безвозвратное удаление файлов и папок	309

Безопасное удаление с помощью модуля <code>send2trash</code>	310
Обход дерева каталогов	310
Сжатие файлов с помощью модуля <code>zipfile</code>	312
Чтение ZIP-файлов	313
Извлечение файлов из ZIP-архива	314
Создание ZIP-архивов и добавление в них файлов	315
Проект: переименование файлов с заменой американского формата дат европейским	315
Шаг 1. Создание регулярного выражения для поиска дат в американском формате	316
Шаг 2. Идентификация фрагментов имен файлов, соответствующих датам	318
Шаг 3. Создание нового имени файла и переименование файлов	319
Идеи для создания похожих программ	320
Проект: создание резервной копии папки в виде ZIP-файла	320
Шаг 1. Определение имени, которое следует присвоить ZIP-файлу	320
Шаг 2. Создание нового ZIP-файла	322
Шаг 3. Обход дерева каталогов и добавление содержимого в ZIP-файл	322
Идеи для создания похожих программ	324
Резюме	324
Контрольные вопросы	325
Учебные проекты	325
Выборочное копирование	325
Удаление ненужных файлов	325
Заполнение пропусков в нумерации файлов	326
Глава 11. Отладка	327
Генерирование исключений	328
Сохранение обратной трассировки стека вызовов в виде строки	330
Утверждения	332
Использование утверждений в программе, имитирующей работу светофора	333
Протоколирование	335
Использование модуля <code>logging</code>	335
Не выполняйте отладку с помощью функции <code>print()</code>	337
Уровень протоколирования	338
Отключение протоколирования	339
Запись сообщений в файл журнала	339
Отладчик Mu	340
Кнопка <code>Continue</code>	340
Кнопка <code>Step In</code>	340
Кнопка <code>Step Over</code>	341
Кнопка <code>Step Out</code>	341

Кнопка Stop	342
Отладка программы сложения чисел	342
Точки останова	344
Резюме	346
Контрольные вопросы	346
Учебный проект	347
Отладка программы, имитирующей подбрасывание монеты	347
Глава 12. Веб-скрейпинг	349
Проект: программа <code>mapIt.py</code> с модулем <code>webbrowser</code>	350
Шаг 1. Определение URL-адреса	351
Шаг 2. Обработка аргументов командной строки	352
Шаг 3. Обработка содержимого буфера обмена и запуск браузера	353
Идеи для создания похожих программ	354
Загрузка файлов из Интернета с помощью модуля <code>requests</code>	354
Загрузка веб-страницы с помощью функции <code>requests.get()</code>	354
Проверка ошибок	355
Сохранение загруженных файлов на жестком диске	356
HTML	358
Ресурсы для изучения HTML	358
Краткие сведения об HTML	358
Просмотр HTML-кода веб-страницы	360
Открытие окна инструментов веб-разработки в браузере	360
Использование инструментов веб-разработки для поиска HTML-элементов	362
Парсинг HTML-разметки с помощью модуля <code>bs4</code>	364
Создание объекта <code>BeautifulSoup</code> на основе HTML-разметки	365
Поиск элемента с помощью метода <code>select()</code>	365
Получение данных из атрибутов элемента	368
Проект: открытие всех результатов поиска	368
Шаг 1. Получение аргументов командной строки и запрос поисковой страницы	369
Шаг 2. Поиск всех результатов	370
Шаг 3. Открытие браузера для каждого из результатов поиска	371
Идеи для создания похожих программ	372
Проект: загрузка всех комиксов на сайте XKCD	372
Шаг 1. Проектирование программы	373
Шаг 2. Загрузка веб-страницы	374
Шаг 3. Поиск и загрузка изображения комикса	375
Шаг 4. Сохранение изображения и поиск предыдущего комикса	376
Идеи для создания похожих программ	378
Управление браузером с помощью модуля <code>selenium</code>	378

Запуск браузера под управлением Selenium	379
Поиск элементов на веб-странице	381
Щелчок на веб-странице	382
Заполнение и отправка веб-форм	383
Отправка кодов специальных клавиш	384
Щелчки на кнопках браузера	385
Получение дополнительной информации о модуле selenium	385
Резюме	385
Контрольные вопросы	385
Учебные проекты	387
Программа для отправки электронной почты из командной строки	387
Загрузчик изображений из Интернета	387
2048	387
Верификация гиперссылок	387
Глава 13. Работа с таблицами Excel	389
Документы Excel	390
Установка модуля openpyxl	390
Чтение документов Excel	391
Открытие документов Excel с помощью модуля openpyxl	392
Получение списка листов рабочей книги	392
Получение ячеек рабочих листов	393
Преобразование буквенных и числовых обозначений столбцов	394
Получение строк и столбцов рабочих листов	395
Рабочие книги, листы и ячейки	397
Проект: чтение данных электронной таблицы	397
Шаг 1. Чтение электронной таблицы	398
Шаг 2. Заполнение структуры данных	399
Шаг 3. Запись результатов в файл	401
Идеи для создания похожих программ	402
Запись документов Excel	403
Создание и сохранение документов Excel	403
Создание и удаление рабочих листов	404
Запись значений в ячейки	405
Проект: обновление электронной таблицы	405
Шаг 1. Создание структуры, содержащей данные для обновления	406
Шаг 2. Проверка всех строк и обновление некорректных цен	407
Идеи для создания похожих программ	408
Настройка шрифтов ячеек	409
Объекты Font	409
Формулы	411
Настройка строк и столбцов	412

Настройка высоты строк и ширины столбцов	412
Объединение и отмена объединения ячеек	413
Закрепление областей	414
Диаграммы	415
Резюме	418
Контрольные вопросы	418
Учебные проекты	419
Генератор таблиц умножения	419
Программа для вставки пустых строк	419
Транспонирование электронной таблицы	420
Преобразование текстовых файлов в электронную таблицу	420
Преобразование электронной таблицы в текстовые файлы	421
Глава 14. Работа с приложением Google Таблицы	423
Установка и настройка модуля EZSheets	424
Получение файлов учетных данных и токенов	424
Отзыв файла учетных данных	426
Объекты Spreadsheet	427
Создание, выгрузка и отображение электронных таблиц	427
Атрибуты объекта Spreadsheet	429
Загрузка и выгрузка электронных таблиц	430
Удаление электронной таблицы	431
Объекты Sheet	431
Чтение и запись данных	432
Создание и удаление листов	437
Копирование листов	439
Квоты приложения Google Таблицы	440
Резюме	440
Контрольные вопросы	441
Учебные проекты	441
Загрузка данных из приложения Google Формы	441
Преобразование электронных таблиц в другие форматы	442
Поиск ошибок в электронной таблице	442
Глава 15. Работа с документами PDF и Word	445
PDF-документы	446
Извлечение текста из PDF-файлов	446
Дешифровка PDF-документов	448
Создание PDF-документов	449
Проект: объединение выбранных страниц из многих PDF-документов	455
Шаг 1. Поиск всех PDF-файлов	456
Шаг 2. Открытие PDF-файлов	456
Шаг 3. Добавление страниц	457

Шаг 4. Сохранение результатов	458
Идеи для создания похожих программ	458
Документы Word	459
Чтение документов Word	460
Получение всего текста из файла .docx	461
Стилевое оформление абзаца и объекты Run	462
Создание документов Word с нестандартными стилями	463
Атрибуты объекта Run	464
Запись документов Word	466
Добавление заголовков	468
Добавление разрывов строк и страниц	469
Добавление изображений	470
Создание документов PDF на основе документов Word	470
Резюме	471
Контрольные вопросы	472
Учебные проекты	472
PDF-паранойя	472
Персонализированные приглашения в виде документов Word	473
Взлом паролей PDF-файлов методом грубой силы	474
Глава 16. Работа с CSV-файлами и данными в формате JSON	475
Модуль csv	476
Объекты reader	477
Чтение данных из объекта reader в цикле for	478
Объекты writer	479
Именованные аргументы delimiter и lineterminator	480
Объекты DictReader и DictWriter	481
Проект: удаление заголовков из CSV-файла	483
Шаг 1. Цикл по всем CSV-файлам	484
Шаг 2. Чтение CSV-файла	485
Шаг 3. Запись CSV-файла без первой строки	486
Идеи для создания похожих программ	487
JSON и программные интерфейсы	487
Модуль json	488
Чтение данных JSON с помощью функции loads ()	489
Запись данных JSON с помощью функции dumps ()	489
Проект: получение текущего прогноза погоды	489
Шаг 1. Определение местоположения с помощью аргумента командной строки	490
Шаг 2. Загрузка данных JSON	491
Шаг 3. Запись данных JSON и вывод прогноза погоды	492
Идеи для создания похожих программ	494

Резюме	494
Контрольные вопросы	495
Учебный проект	495
Программа для преобразования данных из формата Excel в формат CSV	495
Глава 17. Работа с датой и временем, планирование заданий и запуск программ	497
Модуль <code>time</code>	498
Функция <code>time.time()</code>	498
Функция <code>time.sleep()</code>	500
Округление чисел	500
Проект: суперсекундомер	501
Шаг 1. Создание программы для отслеживания времени	502
Шаг 2. Отслеживание и вывод длительности замеров	502
Идеи для создания похожих программ	503
Модуль <code>datetime</code>	504
Тип данных <code>timedelta</code>	506
Пауза до наступления заданной даты	507
Преобразование объектов <code>datetime</code> в строки	508
Преобразование строк в объекты <code>datetime</code>	509
Обзор функций Python для работы с датой и временем	510
Многопоточность	511
Передача аргументов целевой функции потока	513
Проблемы параллелизма	514
Проект: многопоточный загрузчик файлов с сайта XKCD	514
Шаг 1. Модификация программы путем вынесения ее кода в функцию	515
Шаг 2. Создание и запуск потоков выполнения	516
Шаг 3. Ожидание завершения всех потоков	517
Запуск других программ из Python	518
Передача аргументов командной строки в функцию <code>Popen()</code>	520
Планировщик заданий Windows, демон <code>launchd</code> и планировщик <code>cron</code>	521
Открытие веб-сайтов с помощью Python	521
Запуск других сценариев Python	521
Открытие файлов приложениями, заданными по умолчанию	522
Проект: простая программа обратного отсчета времени	523
Шаг 1. Обратный отсчет	523
Шаг 2. Воспроизведение звукового файла	524
Идеи для создания похожих программ	525
Резюме	525
Контрольные вопросы	526

Учебные проекты	526
Наглядный секундомер	526
Загрузка веб-комиксов по расписанию	527
Глава 18. Отправка электронной почты и текстовых сообщений	529
Отправка и получение электронной почты с помощью Gmail API	530
Подключение Gmail API	531
Отправка электронной почты через учетную запись Gmail	532
Чтение электронной почты с помощью учетной записи Gmail	533
Поиск почты в учетной записи Gmail	534
Загрузка вложений из писем Gmail	535
SMTP	535
Отправка электронной почты по протоколу SMTP	536
Подключение к серверу SMTP	537
Отправка строки приветствия серверу SMTP	538
Начало TLS-шифрования	538
Регистрация на сервере SMTP	539
Отправка письма	539
Разрыв соединения с сервером SMTP	540
IMAP	540
Получение и удаление сообщений электронной почты по протоколу IMAP	541
Подключение к серверу IMAP	542
Регистрация на сервере IMAP	542
Поиск сообщений	543
Получение сообщений электронной почты и пометка их как прочитанных	547
Получение адресов электронной почты из необработанных сообщений	548
Получение тела письма из необработанного сообщения	549
Удаление писем	550
Разрыв соединения с сервером IMAP	551
Проект: рассылка напоминаний об уплате членских взносов	551
Шаг 1. Открытие файла Excel	552
Шаг 2. Поиск всех членов клуба, не уплативших взнос	553
Шаг 3. Отправка персональных напоминаний по электронной почте	554
Отправка текстовых сообщений с помощью почтового шлюза SMS	556
Отправка текстовых сообщений с помощью Twilio	557
Создание учетной записи Twilio	558
Отправка текстовых сообщений	559
Проект: модуль “Черкни мне”	561
Резюме	562

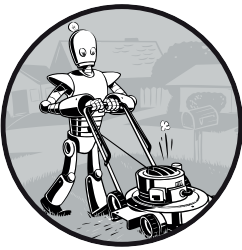
Контрольные вопросы	563
Учебные проекты	564
Произвольное распределение заданий путем рассылки по электронной почте	564
Напоминание о зонтике	564
Автоматический отказ от подписки	564
Дистанционное управление компьютером по электронной почте	565
Глава 19. Работа с изображениями	567
Основы компьютерной обработки изображений	568
Цвета и значения RGBA	568
Кортежи координат и прямоугольников	570
Обработка изображений с помощью модуля Pillow	571
Работа с объектами Image	572
Обрезка изображений	574
Копирование и вставка изображений в другие изображения	574
Изменение размеров изображения	577
Поворот и зеркальное отражение изображений	579
Изменение отдельных пикселей	581
Проект: добавление логотипа	583
Шаг 1. Открытие изображения логотипа	584
Шаг 2. Цикл по всем файлам и открытие изображений	585
Шаг 3. Масштабирование изображений	586
Шаг 4. Добавление логотипа и сохранение изменений	587
Идеи для создания похожих программ	589
Рисование на изображениях	589
Рисование фигур	590
Рисование текста	592
Резюме	594
Контрольные вопросы	595
Учебные проекты	595
Доработка основного проекта главы	596
Поиск папок с фотографиями на жестком диске	596
Персональные приглашения	598
Глава 20. Управление клавиатурой и мышью	599
Установка модуля PyAutoGUI	600
Настройка доступности в macOS	601
Контроль над клавиатурой и мышью	601
Паузы и безопасное завершение работы	601
Прекращение выполнения всех задач путем выхода из учетной записи	602
Управление перемещениями мыши	602

Перемещение указателя мыши	603
Получение позиции указателя	604
Управление взаимодействием с мышью	605
Щелчки мышью	605
Перетаскивание указателя мыши	606
Прокрутка	608
Планирование перемещений указателя	608
Работа с экраном	610
Получение снимка экрана	610
Анализ снимка экрана	610
Распознавание изображений	612
Получение информации об окне	613
Определение активного окна	614
Другие способы получения информации об окнах	615
Манипулирование окнами	616
Управление клавиатурой	618
Отправка строки, набранной на виртуальной клавиатуре	618
Названия клавиш	619
Нажатие и отпускание клавиш	621
Горячие клавиши	621
Настройка собственных сценариев GUI-автоматизации	622
Обзор функций PyAutoGUI	623
Проект: автоматическое заполнение формы	624
Шаг 1. Составление плана действий	626
Шаг 2. Настройка координат	627
Шаг 3. Начало ввода данных	629
Шаг 4. Обработка списков выбора и переключателей	630
Шаг 5. Отправка формы и ожидание	631
Отображение окон сообщений	632
Резюме	633
Контрольные вопросы	634
Учебные проекты	635
Как притвориться занятым	635
Использование буфера обмена для чтения текстового поля	635
Бот для отправки мгновенных сообщений	636
Руководство по созданию игрового бота	636
Приложение А. Установка сторонних модулей	639
Утилита pip	640
Инсталляция модулей	640
Установка модулей для редактора Mu	643

Приложение Б. Запуск программ	645
Запуск программ в окне терминала	646
Запуск сценариев Python в Windows	647
Запуск сценариев Python в macOS	648
Запуск сценариев Python в Ubuntu Linux	649
Запуск сценариев Python с отключенными проверками	650
Приложение В. Ответы на контрольные вопросы	651
Глава 1	652
Глава 2	652
Глава 3	654
Глава 4	655
Глава 5	656
Глава 6	656
Глава 7	657
Глава 8	658
Глава 9	658
Глава 10	659
Глава 11	659
Глава 12	660
Глава 13	661
Глава 14	662
Глава 15	663
Глава 16	663
Глава 17	664
Глава 18	664
Глава 19	665
Глава 20	665
Предметный указатель	667

8

ПРОВЕРКА ВВОДА



В программе необходимо проверять корректность значений, вводимых пользователем с помощью той же функции `input()`. В частности, если пользователь должен указать свой возраст, то программа не должна принимать бессмысленные ответы, например отрицательные числа (недопустимый диапазон целых чисел) или слова (неправильный тип данных). Проверка ввода также позволяет предотвратить ошибки. Если вы реализуете функцию `withdrawFromAccount()`, которая содержит аргумент для снятия суммы со счета, то убедитесь, что эта сумма положительна. При попытке снять со счета отрицательную сумму функция в конечном итоге добавит деньги на счет!

Как правило, мы выполняем проверку ввода, неоднократно запрашивая данные у пользователя, пока он не введет корректный текст, как в следующем примере.

```
while True:
    print('Укажите ваш возраст:')
    age = input()
    try:
        age = int(age)
    except:
        print('Пожалуйста, введите цифры.')
        continue
    if age < 1:
        print('Пожалуйста, введите положительное число.')
        continue
    break

print(f'Вам {age} лет.')
```

После запуска программы результат будет выглядеть примерно так.

```
Укажите ваш возраст:
тридцать
Пожалуйста, введите цифры.
Укажите ваш возраст:
-2
Пожалуйста, введите положительное число.
Укажите ваш возраст:
30
Вам 30 лет.
```

Программа предлагает ввести возраст до тех пор, пока не будет введено правильное значение. Тем самым гарантируется, что при выходе из цикла `while` переменная `age` будет содержать допустимое значение, которое впоследствии не приведет к сбою программы.

Впрочем, писать код проверки для каждого вызова функции `input()` быстро надоедает. Кроме того, возникает риск пропустить те или иные ошибочные варианты ввода, и в результате в программе окажутся некорректные данные. В этой главе мы рассмотрим использование модуля `PyInputPlus` для проверки ввода.

Модуль `PyInputPlus`

Модуль `PyInputPlus` содержит функции, аналогичные `input()`, которые предназначены для ввода различных типов данных: чисел, дат, адресов электронной почты и т.п. Если пользователь введет недопустимое значение, например неверно отформатированную дату или число за пределами

допустимого диапазона, модуль `PyInputPlus` предложит ввести данные повторно, как в приведенном выше примере. В модуле `PyInputPlus` реализованы и другие полезные возможности, такие как ограничение количества повторных запросов или тайм-аут, если пользователь должен уложиться в определенные временные рамки.

Модуль `PyInputPlus` не является частью стандартной библиотеки Python, поэтому его нужно установить отдельно с помощью утилиты `pip` (сведения по установке сторонних модулей приведены в приложении А). Для этого в командной строке выполните следующую инструкцию:

```
pip install --user pyinputplus
```

Чтобы проверить, корректно ли установлен модуль `PyInputPlus`, импортируйте его в интерактивной оболочке:

```
>>> import pyinputplus
```

Если при импорте модуля не появилось никаких сообщений об ошибках, значит, он успешно установлен.

Модуль `PyInputPlus` содержит несколько функций, предназначенных для обработки различных типов вводимых данных.

- `inputStr()`. Аналогична встроенной функции `input()`, но поддерживает расширенные возможности модуля `PyInputPlus`. Ей можно передать пользовательскую функцию для проверки введенных данных.
- `inputNum()`. Гарантирует ввод числа и возвращает значение типа `int` или `float`, в зависимости от того, содержит ли введенное число десятичную точку.
- `inputChoice()`. Гарантирует выбор одного из предложенных вариантов.
- `inputMenu()`. Аналогична функции `inputChoice()`, но отображает меню с числовыми или буквенными вариантами.
- `inputDatetime()`. Гарантирует ввод значений даты и времени.
- `inputYesNo()`. Гарантирует, что пользователь введет ответ “да/нет”.
- `inputBool()`. Аналогична функции `inputYesNo()`, но распознает ответ `'True'` или `'False'` и возвращает соответствующее булево значение.
- `inputEmail()`. Гарантирует ввод корректного адреса электронной почты.

- `inputFilepath()`. Гарантирует ввод правильного имени файла (включая путь) и может дополнительно проверять, существует ли файл с таким именем.
- `inputPassword()`. Аналогична встроенной функции `input()`, но отображает символы * вместо вводимых символов, что позволяет безопасно вводить пароли и другую конфиденциальную информацию.

Эти функции автоматически выводят новый запрос до тех пор, пока не будут введены корректные данные.

```
>>> import pyinputplus as pyip
>>> response = pyip.inputNum()
пять
'пять' is not a number.
42
>>> response
42
```

Выражение `as pyip` в инструкции `import` избавляет от необходимости указывать `pyinputplus` каждый раз, когда нужно вызвать функцию из этого модуля. Вместо длинного имени используется более короткий псевдоним `pyip`. В отличие от функции `input()`, функция `inputNum()` возвращает значение типа `int` или `float`: в данном случае это 42, а не '42'.

Как и в случае функции `input()`, функциям модуля `PyInputPlus` можно передать строку приглашения с помощью именованного аргумента `prompt`.

```
>>> response = input('Введите число: ')
Введите число: 42
>>> response
'42'
>>> import pyinputplus as pyip
>>> response = pyip.inputInt(prompt='Введите число: ')
Введите число: кот
'кот' is not an integer.
Введите число: 42
>>> response
42
```

Чтобы больше узнать о каждой из этих функций, воспользуйтесь функцией `help()`. Например, если ввести в интерактивной оболочке `help(pyip.inputChoice)`, отобразится справочная информация по функции `inputChoice()`. Полная документация к модулю доступна по адресу <https://pyinputplus.readthedocs.io/>.

В отличие от встроенной функции `input()`, функции модуля `PyInputPlus` имеют ряд дополнительных возможностей проверки, о которых будет сказано далее.

Именованные аргументы `min`, `max`, `greaterThan` и `lessThan`

Функции `inputNum()`, `inputInt()` и `inputFloat()`, которые работают с целыми числами и числами с плавающей точкой, поддерживают именованные аргументы `min`, `max`, `greaterThan` и `lessThan`, с помощью которых можно задать диапазон допустимых значений. Введите в интерактивной оболочке следующие инструкции.

```
>>> import pyinputplus as pyip
>>> response = pyip.inputNum('Введите число: ', min=4)
Введите число: 3
Input must be at minimum 4.
Введите число: 4
>>> response
4
>>> response = pyip.inputNum('Введите число: ', greaterThan=4)
Введите число: 4
Input must be greater than 4.
Введите число: 5
>>> response
5
>>> response = pyip.inputNum('>', min=4, lessThan=6)
> 6
Input must be less than 6.
> 3
Input must be at minimum 4.
> 4
>>> response
4
```

Это необязательные аргументы, но если они указаны, то вводимые значения не могут быть меньше аргумента `min` или больше аргумента `max` (но могут быть равны им). Кроме того, вводимые значения должны быть больше, чем аргумент `moreThan`, и меньше, чем аргумент `lessThan` (т.е. они не могут быть равны им).

Именованный аргумент `blank`

По умолчанию ввод пустой строки не допускается, если только для аргумента `blank` не задано значение `True`.

```
>>> import pyinputplus as pyip
>>> response = pyip.inputNum('Введите число: ')
Введите число: (введено пустое значение)
```

```
Blank values are not allowed.
Введите число: 42
>>> response
42
>>> response = pyip.inputNum(blank=True)
(введено пустое значение)
>>> response
''
```

Используйте аргумент `blank = True` в том случае, когда пользователю разрешается ничего не вводить.

Именованные аргументы `limit`, `timeout` и `default`

По умолчанию функции модуля `PyInputPlus` будут продолжать запрашивать у пользователя ввод корректных данных бесконечно долго (ну или до тех пор, пока выполняется программа). Если хотите, чтобы функция перестала запрашивать данные после определенного количества попыток или по истечении определенного времени, используйте именованные аргументы `limit` и `timeout`. Целочисленный аргумент `limit` определяет, сколько попыток будет предпринято функцией для получения корректных данных, прежде чем она завершит работу, а целочисленный аргумент `timeout` определяет, сколько секунд отведено пользователю для ввода корректных данных, прежде чем функция завершится.

Если пользователь так и не введет корректных данных за указанное количество попыток или отведенное время, функции сгенерируют исключение `RetryLimitException` или `TimeoutException` соответственно. Например, введите в интерактивной оболочке следующий код.

```
>>> import pyinputplus as pyip
>>> response = pyip.inputNum(limit=2)
бла-бла-бла
'бла-бла-бла' is not a number.
Enter num: число
'число' is not a number.
Traceback (most recent call last):
  -- Опущено --
pyinputplus.RetryLimitException
>>> response = pyip.inputNum(timeout=10)
42 (введено после 10 секунд ожидания)
Traceback (most recent call last):
  -- Опущено --
pyinputplus.TimeoutException
```

Если помимо этих аргументов указан также аргумент `default`, функция не сгенерирует исключение, а вернет значение, переданное ей с помощью этого аргумента. Введите в интерактивной оболочке следующий код.

```
>>> response = pyip.inputNum(limit=2, default='N/A')
Здравствуй
'Здравствуй' is not a number.
мир
'мир' is not a number.
>>> response
'N/A'
```

Вместо того чтобы генерировать исключение `RetryLimitException`, функция `inputNum()` возвращает строку `'N/A'`.

Именованные аргументы `allowRegexes` и `blockRegexes`

Указывать допустимые значения можно также с помощью регулярных выражений. Именованные аргументы `allowRegexes` и `blockRegexes` позволяют задать списки регулярных выражений, определяющих, какие значения функция принимает в качестве допустимых, а какие — отклоняет. Например, введите в интерактивной оболочке следующий код, чтобы функция `inputNum()` помимо обычных чисел поддерживала римские цифры.

```
>>> import pyinputplus as pyip
>>> response=pyip.inputNum(allowRegexes=[r'(I|V|X|L|C|D|M)+',
                                     r'zero'])
XLII
>>> response
'XLII'
>>> response=pyip.inputNum(allowRegexes=[r'(i|v|x|l|c|d|m)+',
                                     r'zero'])
xlii
>>> response
'xlii'
```

Конечно, эти регулярные выражения задают только буквы, которые разрешается вводить пользователю. Порядок цифр в числе может оказаться неправильным, например `'XVX'` или `'MILLI'`, потому что регулярное выражение `r'(I|V|X|L|C|D|M)+'` допускает такое число.

С помощью именованного аргумента `blockRegexes` можно также указать список регулярных выражений, которые функция не будет принимать. Введите в интерактивной оболочке следующий код, чтобы функция `inputNum()` не принимала четные числа.

```
>>> import pyinputplus as pyip
>>> response = pyip.inputNum(blockRegexes=[r'[02468]$\'])
42
This response is invalid.
44
This response is invalid.
```



```
43
>>> response
43
```

Если указать оба аргумента — и `allowRegexes`, и `blockRegexes`, — то список разрешений перекрывает список блокировок. Например, введите в интерактивной оболочке следующий код, который разрешает ввод слов 'caterpillar' и 'category', но блокирует любые другие строки, содержащие слово 'cat'.

```
>>> import pyinputplus as pyip
>>> response = pyip.inputStr(allowRegexes=[r'caterpillar',
...                                     'category'],
...                           blockRegexes=[r'cat'])
cat
This response is invalid.
catastrophe
This response is invalid.
category
>>> response
'category'
```

Функции модуля `PyInputPlus` помогут избавить вас от утомительного написания кода проверки вводимых данных. Полная документация к этому модулю доступна по адресу <https://pyinputplus.readthedocs.io/>.

Передача пользовательской функции проверки в функцию `inputCustom()`

Можно написать функцию, реализующую требуемую логику проверки, и передать ее функции `inputCustom()`. Допустим, необходимо, чтобы пользователь ввел последовательность цифр, в сумме равных 10. Функции `pyinputplus.inputAddsUpToTen()` не существует, но можно создать свою собственную функцию, которая:

- имеет один строковый аргумент (значение, введенное пользователем);
- генерирует исключение, если строка не проходит проверку;
- возвращает `None` (инструкция `return` может быть опущена), если функция `inputCustom()` должна вернуть строку без изменений;
- возвращает значение, отличное от `None`, если функция `inputCustom()` должна вернуть строку, отличную от той, которую ввел пользователь;
- передается в качестве первого аргумента функции `inputCustom()`.

Например, можно создать пользовательскую функцию `addUpToTen()` и передать ее функции `inputCustom()`. При этом вызов должен выглядеть как `inputCustom(addUpToTen)`, а не `inputCustom(addUpToTen())`, потому что мы передаем ссылку на функцию `addUpToTen()`, а не возвращаемое ею значение.

```
>>> import pyinputplus as pyip
>>> def addUpToTen(numbers):
...     numbersList = list(numbers)
...     for i, digit in enumerate(numbersList):
...         numbersList[i] = int(digit)
...         if sum(numbersList) != 10:
...             raise Exception('Сумма должна быть 10, а не %s.' %
(sum(numbersList)))
...     return int(numbers)      # вернуть целое число
...
>>> response = pyip.inputCustom(addUpToTen) # без скобок после имени
123
Сумма должна быть 10, а не 6.
1235
Сумма должна быть 10, а не 11.
1234
>>> response # функция inputCustom() возвращает целое число, а не строку
1234
>>> response = pyip.inputCustom(addUpToTen)
Здравствуй
invalid literal for int() with base 10: '3'
55
>>> response
55
```

Функция `inputCustom()` тоже поддерживает именованные аргументы `blank`, `limit`, `timeout`, `default`, `allowRegexes` и `blockRegexes`. Написание собственной функции проверки целесообразно в том случае, когда трудно или невозможно написать регулярное выражение для проверки допустимых данных, как в приведенном выше примере.

Проект: как занять дурака на несколько часов

Воспользуемся модулем `PyInputPlus` для создания простой программы, которая выполняет следующие действия:

- 1) спрашивает пользователя, не хотел бы он узнать, как занять дурака на протяжении нескольких часов;
- 2) завершает работу, если пользователь отвечает “нет”;
- 3) возвращается к п. 1, если пользователь отвечает “да”.

Поскольку мы не знаем, будет ли пользователь вводить что-то кроме 'да' или 'нет', необходимо проверить правильность введенных данных. Было бы также удобно, чтобы пользователь мог вводить 'д' или 'н' вместо полных слов. Эти проверки выполняет функция `inputYesNo()` из модуля `PyInputPlus`, которая независимо от регистра введенных символов возвращает строку 'да' или 'нет' в нижнем регистре.

Результаты работы программы будут выглядеть примерно так.

Хотите узнать, как занять дурака на несколько часов?

конечно

'конечно' is not a valid yes/no response.

Хотите узнать, как занять дурака на несколько часов?

да

Хотите узнать, как занять дурака на несколько часов?

д

Хотите узнать, как занять дурака на несколько часов?

Да

Хотите узнать, как занять дурака на несколько часов?

ДА

Хотите узнать, как занять дурака на несколько часов?

ДА!!!!!!

'ДА!!!!!!' is not a valid yes/no response.

Хотите узнать, как занять дурака на несколько часов?

СКАЖИ МНЕ, КАК.

'СКАЖИ МНЕ, КАК.' is not a valid yes/no response.

Хотите узнать, как занять дурака на несколько часов?

нет

Спасибо! Желаю хорошего дня.

Откройте новую вкладку в редакторе файлов и сохраните файл под именем *idiot.py*. Затем введите следующий код:

```
import pyinputplus as pyip
```

В результате будет импортирован модуль `PyInputPlus`. Поскольку имя `pyinputplus` слишком длинное, мы назначаем ему псевдоним `pyip`.

```
while True:
    prompt = 'Хотите узнать, как занять дурака на несколько часов?'
    response = pyip.inputYesNo(prompt)
```

Выражение `while True:` создает бесконечный цикл, который выполняется до тех пор, пока не встретится инструкция `break`. В цикле вызывается функция `pyip.inputYesNo()`, которая не завершится до тех пор, пока пользователь не введет корректный ответ.

```
if response == 'no':  
    break
```

Функция `pyip.inputYesNo()` гарантированно возвращает 'yes' или 'no'. Если возвращается 'no', программа выходит из бесконечного цикла и продолжает выполняться до последней строки, после чего прощается с пользователем:

```
print('Спасибо! Желаю хорошего дня.')
```

В противном случае цикл повторяется снова.

Функция `inputYesNo()` поддерживает языки, отличные от английского, благодаря именованным аргументам `yesVal` и `noVal`. Например, русскоязычная версия программы должна содержать следующий код.

```
response = pyip.inputYesNo(prompt, yesVal= 'да', noVal='нет')  
if response == 'нет':
```

Теперь пользователь может ввести 'да' или 'д' (в нижнем или верхнем регистре) вместо 'yes' или 'y' в качестве утвердительного ответа.

Проект: тест на умножение

Модуль `PyInputPlus` может оказаться полезным для создания теста на умножение с лимитом времени. Благодаря именованным аргументам `allowRegexes`, `blockRegexes`, `timeout` и `limit` функции `pyip.inputStr()` большая часть операций реализуется в самом модуле `PyInputPlus`. Чем меньше кода предстоит написать, тем быстрее можно получить готовую программу. Мы напишем программу, которая выводит пользователю 10 заданий на умножение. Откройте в редакторе файлов новую вкладку и сохраните файл под именем *multiplicationQuiz.py*.

Сначала необходимо импортировать модули `pyinputplus`, `random` и `time`. Мы будем отслеживать, сколько вопросов задала программа и сколько правильных ответов дал пользователь, с помощью переменных `numberOfQuestions` и `correctAnswers`. В цикле `for` будет случайным образом 10 раз выдаваться задание на умножение.

```
import pyinputplus as pyip  
import random, time  
  
numberOfQuestions = 10  
correctAnswers = 0  
for questionNumber in range(numberOfQuestions):
```

В цикле `for` программа выбирает две перемножаемые цифры. Эти цифры отображаются в приглашении `#Q: N × N =`, где `Q` — номер вопроса (от 1 до 10), а `N` — числа, которые нужно умножить.

```
# Выбираем два случайных числа
num1 = random.randint(0, 9)
num2 = random.randint(0, 9)
prompt = '#%s: %s x %s = ' % (questionNumber, num1, num2)
```

Основная логика программы реализована в функции `pyip.inputStr()`. Именованный аргумент `allowRegexes` представляет собой список, содержащий регулярное выражение `'^%s$'`, где `%s` заменяется правильным ответом. Символы `^` и `$` гарантируют, что ответ начинается и заканчивается правильным числом, хотя функции `PyInputPlus` сначала удаляют любые ведущие и замыкающие пробелы на тот случай, если пользователь случайно нажмет пробела до или после ответа. Именованный аргумент `blocklistRegexes` содержит список с одним кортежем `('.*', 'Неправильно!')`. Первая строка в кортеже — это регулярное выражение, которое соответствует любой возможной строке. Следовательно, если пользователь вводит неправильный ответ, программа отклоняет его. В таком случае отображается строка `'Неправильно!'`, после чего пользователю предлагается ответить снова. Кроме того, аргументы `timeout=8` и `limit=3` гарантируют, что у пользователя есть только 8 секунд и 3 попытки дать правильный ответ.

```
try:
    # Правильные ответы задаются аргументом allowRegexes;
    # неправильные ответы задаются аргументом blockRegexes
    # (в случае неправильного ответа отображается
    # пользовательское сообщение)
    pyip.inputStr(prompt, allowRegexes=['^%s$' % (num1 * num2)], \
                  blockRegexes=[('.*', 'Неправильно!')], \
                  timeout=8, limit=3)
```

Если пользователь отвечает по истечении 8-секундного тайм-аута, то даже в случае правильного ответа функция `pyip.inputStr()` генерирует исключение `TimeoutException`. Если пользователь отвечает неправильно более трех раз, генерируется исключение `RetryLimitException`. Оба этих типа исключений определены в модуле `PyInputPlus`, поэтому их нужно предварять префиксом `pyip`.

```
except pyip.TimeoutException:
    print('Время истекло!')
except pyip.RetryLimitException:
    print('Закончилось количество попыток!')
```

Помните, что блок `else` может следовать не только за блоком `if` или `elif`, но и за блоком `except`. Следующий код будет выполняться, если в блоке `try` не было сгенерировано исключение. В нашем случае это означает, что пользователь ввел правильный ответ.

```
else:
    # Этот блок выполняется, если в блоке try
    # не возникло исключений
    print('Правильно!')
    correctAnswers += 1
```

Независимо от того, какое из трех сообщений ('Время истекло!', 'Закончилось количество попыток!' или 'Правильно') отображается, в конце цикла делается секундная пауза, которая дает пользователю время на прочтение сообщения. После того как программа задала 10 вопросов, пользователь увидит количество правильных ответов.

```
time.sleep(1)    # короткая пауза, позволяющая пользователю
                 # увидеть результат
print('Счет: %s/%s'%(correctAnswers, numberOfQuestions))
```

Модуль `PyInputPlus` достаточно гибкий, благодаря чему его можно использовать в самых разных программах, которые принимают ввод пользователя с клавиатуры.

Резюме

Многие программисты забывают писать код для проверки вводимых данных, но без него в программах практически наверняка будут возникать ошибки. Значения, которые вы ожидаете от пользователей, и значения, которые они фактически вводят, могут оказаться совершенно разными, и программы должны быть достаточно надежными, чтобы справляться с такими ситуациями. Для создания кода проверки вводимых данных можно использовать регулярные выражения, но обычно проще использовать готовый модуль, такой как `PyInputPlus`. Импортируйте этот модуль с помощью инструкции `import pyinputplus as pyip`, чтобы при вызове функций модуля указывать более короткий псевдоним `pyip`.

Модуль `PyInputPlus` содержит функции для ввода различных типов данных, включая строки, числа, даты, булевы значения, варианты "да/нет", адреса электронной почты и файлы. В то время как функция `input()` всегда возвращает строку, функции модуля `PyInputPlus` возвращают значения соответствующего типа данных. Функция `inputChoice()` позволяет выбрать один из нескольких предварительно заданных вариантов, а функция `inputMenu()` добавляет цифры или буквы к вариантам выбора.

Все эти функции поддерживают следующие стандартные возможности: удаление ведущих и замыкающих пробелов, установка тайм-аута и количества допустимых повторов с помощью именованных аргументов `timeout` и `limit`, а также передача списков регулярных выражений с помощью аргументов `allowRegexes` и `blockRegexes`, позволяющих принимать или отвергать определенные ответы. Вам больше не нужно писать утомительные циклы `while`, в которых проверяется правильность введенных данных и выводятся повторные запросы.

Если ни одна из функций модуля `PyInputPlus` не соответствует вашим задачам, можно написать собственную функцию проверки и передать ее функции `inputCustom()`. Полный список функций модуля доступен по адресу <https://pyinputplus.readthedocs.io/en/latest/>. В онлайн-документации содержится гораздо больше информации, чем было приведено в этой главе. Не стоит изобретать велосипед — лучше научиться использовать этот модуль, чем писать (и отлаживать!) собственный код.

Теперь, когда вы умеете работать с текстом и проверять его правильность, пора научиться считывать и записывать файлы, хранящиеся на жестком диске. Этой теме посвящена следующая глава.

Контрольные вопросы

1. Входит ли модуль `PyInputPlus` в стандартную библиотеку Python?
2. Почему модуль `PyInputPlus` обычно импортируют с помощью инструкции `import pyinputplus as pyip`?
3. Чем отличается функция `inputInt()` от функции `inputFloat()`?
4. Как с помощью модуля `PyInputPlus` гарантировать, что пользователь введет целое число в диапазоне от 0 до 99?
5. Что передается с помощью именованных аргументов `allowRegexes` и `blockRegexes`?
6. Что сделает функция `inputStr(limit=3)`, если три раза ввести пустую строку?
7. Что сделает функция `inputStr(limit=3, default='привет')`, если три раза ввести пустую строку?

Учебные проекты

Чтобы закрепить полученные знания на практике, напишите программы для предложенных ниже задач.

Изготовитель бутербродов

Напишите программу, которая спрашивает пользователя о его бутербродных предпочтениях. Программа должна использовать модуль `PyInputPlus`, чтобы гарантировать ввод корректных данных.

- Используйте функцию `inputMenu()` для определения типа хлеба: цельнозерновой, белый или ржаной.
- Используйте функцию `inputMenu()` для определения типа белкового продукта: курица, индейка, ветчина или тофу.
- Используйте функцию `inputYesNo()`, чтобы спросить, хочет ли пользователь добавить сыр.
- Если пользователь ответил утвердительно, используйте функцию `inputMenu()`, чтобы узнать тип сыра: чеддер, швейцарский или моцарелла.
- Используйте функцию `inputYesNo()`, чтобы узнать у пользователя, хочет ли он добавить майонез, горчицу, салат или помидор.
- Используйте функцию `inputInt()`, чтобы узнать, сколько бутербродов хочет пользователь. Убедитесь в том, что это число не меньше 1.

Придумайте цены для каждого из параметров бутерброда, и пусть программа отобразит общую стоимость после того, как пользователь сделает свой выбор.

Собственный тест на умножение

Чтобы оценить реальные возможности модуля `PyInputPlus`, попробуйте воссоздать тест на умножение без использования этого модуля. Программа должна предложить пользователю 10 заданий на умножение в диапазоне от 0.0 до 9.9. Необходимо реализовать следующий функционал.

- Если пользователь вводит правильный ответ, программа в течение одной секунды отображает сообщение “Правильно!” и переходит к следующему вопросу.
- Пользователь получает три попытки для ввода правильного ответа, прежде чем программа перейдет к следующему вопросу.
- Через восемь секунд после первого отображения вопроса он помечается как неправильный, даже если пользователь вводит правильный ответ после 8-секундной паузы.

Сравните свой код с кодом на основе модуля `PyInputPlus`, который был приведен в главе.