1

Подготовка среды для Python

Это самая скучная, но тем не менее важная часть книги. В ней мы разберем процесс подготовки среды для написания и тестирования кода на Python. Мы пройдем ускоренный курс настройки виртуальной машины (ВМ) Kali Linux, создания виртуального окружения для Python 3 и установки удобной интегрированной среды разработки (integrated development environment, IDE), чтобы у вас было все необходимое для написания кода. По прочтении этой главы вы должны быть готовы к решению упражнений, представленных в ее конце, и анализу примеров кода, приведенных в оставшихся главах.

Прежде чем начать, скачайте и установите клиент виртуализации на основе гипервизора, такой как VMware Player, VirtualBox или Hyper-V, если вы этого еще не сделали. Также советуем держать наготове ВМ с Windows 10, пробную копию можно взять здесь: https://developer.microsoft.com/ru-ru/windows/downloads/virtual-machines/.

Установка Kali Linux

Kali — это реинкарнация дистрибутива BackTrack Linux, разработанная компанией Offensive Security в качестве операционной системы для тестирования на проникновение. В ее состав входит целый ряд предустановленных

инструментов, но поскольку она основана на Debian Linux, вам доступен широкий выбор дополнительных программ и библиотек.

Вы будете использовать Kali в своей гостевой виртуальной машине. То есть скачаете образ Kali и запустите его на своем компьютере с помощью любого гипервизора по своему выбору. Образ Kali имеется на странице https://www.kali.org/downloads/. Следуйте инструкциям, приведенным в документации Kali: https://www.kali.org/docs/installation/.

Выполнив все этапы установки, вы должны получить полноценное окружение рабочего стола Kali (рис. 1.1).

Рис. 1.1. Рабочий стол Kali Linux

Поскольку с момента создания образа Kali могли выйти важные обновления, давайте обновим гостевую систему до последней версии. Введите в командной оболочке Kali (Applications ▸ Accessories ▸ Terminal (Приложения ▸ Стандартные ▸ Терминал)):

```
tim@kali:~$ sudo apt update
tim@kali:~$ apt list --upgradable
tim@kali:~$ sudo apt upgrade
tim@kali:~$ sudo apt dist-upgrade
tim@kali:~$ sudo apt autoremove
```

Подготовка Python 3

Первым делом убедимся в том, что у нас установлена подходящая версия Python (проекты в этой книге рассчитаны на Python 3.6 и выше). Запустим Python в командной оболочке Kali и посмотрим:

```
tim@kali:~$ python
```

Вот как выглядит вывод на компьютере с Kali:

```
Python 2.7.17 (default, Oct 19 2019, 23:36:22)
[GCC 9.2.1 20191008] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

Не совсем то, что нам нужно. На момент написания этих строк в Kali по умолчанию использовалась версия Python 2.7.18. Но это не проблема, так как версия Python 3 тоже должна быть установлена:

```
tim@kali:~$ python3
Python 3.7.5 (default, Oct 27 2019, 15:43:29)
[GCC 9.2.1 20191022] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

Здесь используется версия 3.7.5. Если у вас она ниже 3.6, обновите дистрибутив с помощью следующей команды:

```
$ sudo apt-get upgrade python3
```

Мы будем использовать Python 3 в *виртуальной среде* — автономном дереве каталогов, которое содержит Python и набор любых дополнительных пакетов, которые вы установите. Виртуальная среда — это один из незаменимых инструментов разработчика на Python. С ее помощью можно разделять проекты с разными потребностями. Например, одну виртуальную среду можно использовать для проектов, связанных с анализом сетевых пакетов, а другую — для проектов по анализу двоичных файлов.

Благодаря наличию отдельных сред вы можете поддерживать свой код в порядке, не усложняя его без необходимости. Таким образом, у каждой среды может быть собственный набор зависимостей и модулей, которые не влияют на работу других ваших проектов.

Создадим виртуальную среду. Для начала нужно установить пакет `python3-venv`:

```
tim@kali:~$ sudo apt-get install python3-venv
[sudo] password for tim:
...
```

Теперь можно создать виртуальную среду. Для этого понадобится новый каталог, в котором мы будем работать:

```
tim@kali:~$ mkdir bhp
tim@kali:~$ cd bhp
tim@kali:~/bhp$ python3 -m venv venv3
tim@kali:~/bhp$ source venv3/bin/activate
(venv3) tim@kali:~/bhp$ python
```

Таким образом, мы получим в текущем каталоге папку `bhp`. Затем создается новая виртуальная среда путем вызова пакета `venv` с флагом `-m` и названием, которое мы хотим ей назначить. Мы назвали свою среду `venv3`, но вы можете выбрать любое другое имя. В этом каталоге будут находиться скрипты, пакеты и исполняемые файлы Python. Дальше мы активируем нашу среду с помощью скрипта `activate`. Обратите внимание на то, что в результате этого меняется приглашение командной строки — теперь перед ним выводится название среды (в нашем случае это `venv3`). Позже, когда закончите работать со средой, используйте команду `deactivate`.

Итак, вы подготовили Python и активировали виртуальную среду. Так как в этой среде используется Python 3, при запуске интерпретатора можно указывать просто `python` вместо `python3`. Иными словами, после активации каждая команда Python будет интерпретироваться относительно вашей виртуальной среды. Пожалуйста, учтите, что использование другой версии Python может нарушить работу некоторых примеров кода, приводимых в книге.

Для установки пакетов Python в виртуальной среде можно задействовать исполняемый файл `pip`. Он во многом похож на диспетчер пакетов `apt`, так как позволяет автоматизировать процесс установки, чтобы вам не нужно было скачивать, распаковывать и устанавливать пакеты вручную.

С помощью `pip` вы можете искать пакеты и устанавливать их в своей виртуальной среде:

```
(venv3) tim@kali:~/bhp: pip search hashcrack
```

Давайте проведем небольшой эксперимент и установим модуль `lxml`, который будет применяться в главе 5 для создания программы, извлекающей содержимое веб-страниц. Введите в своем терминале следующее:

```
(venv3) tim@kali:~/bhp: pip install lxml
```

Вывод в терминале должен сигнализировать о том, что библиотека скачивается и устанавливается. По завершении процесса запустите командную оболочку Python и подтвердите, что установка прошла корректно:

```
(venv3) tim@kali:~/bhp$ python
Python 3.7.5 (default, Oct 27 2019, 15:43:29)
[GCC 9.2.1 20191022] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> from lxml import etree
>>> exit()
(venv3) tim@kali:~/bhp$
```

Если вместо этого вы получите ошибку или версию Python 2, убедитесь в том, что выполнили все шаги, перечисленные ранее, и что у вас установлена актуальная версия Kali.

Имейте в виду, что код большинства примеров, представленных в этой книге, можно разрабатывать в разнообразных операционных системах, включая macOS, Linux и Windows. Вам также, наверное, стоит использовать отдельные виртуальные среды для разных проектов или глав. Некоторые главы относятся исключительно к Windows, и в их начале мы об этом упомянем.

Итак, мы подготовили виртуальную машину и виртуальную среду с Python 3. Теперь установим IDE для разработки.

Установка IDE

Интегрированная среда разработки (integrated development environment, IDE) предоставляет набор инструментов для написания кода. Она, как правило, состоит из редактора с подсветкой синтаксиса и автоматическим анализом кода и отладчика. IDE предназначена для облегчения написания и отладки программ. При программировании на Python ее использовать необязательно — для небольших программ-прототипов можно обойтись любым текстовым редактором, таким как vim, nano, Блокнот или emacs. Но в более крупных и сложных проектах IDE будет чрезвычайно полезна, например она может выделить

определенные, но неиспользуемые переменные, обнаружить опечатки в именах переменных или найти недостающие инструкции импорта пакетов.

По результатам недавнего опроса, двумя наиболее популярными IDE у разработчиков на Python являются PyCharm (имеет коммерческую и бесплатную версии) и Visual Studio Code (бесплатная). Джастин обожает WingIDE (имеет коммерческую и бесплатную версии), а Тим использует Visual Studio Code (VS Code). Все эти три IDE доступны в Windows, macOS и Linux.

PyCharm и WingIDE можно скачать на страницах https://www.jetbrains.com/pycharm/download/ и https://wingware.com/downloads/ соответственно. VS Code можно установить в командной строке Kali:

```
tim@kali#: apt-get install code
```

Последняя версия VS Code доступна также на странице https://code.visualstudio.com/download/, ее можно установить с помощью `apt-get`:

```
tim@kali#: apt-get install -f ./code_1.39.2-1571154070_amd64.deb
```

Номер версии, указанный в имени файла, наверняка будет отличаться от того, который показан здесь, поэтому убедитесь в том, что имя в вашей команде совпадает с тем, что вы скачали.

Правила оформления кода

Что бы вы ни использовали для написания своих программ, всегда следует соблюдать рекомендации по форматированию кода. Они сделают ваш код на Python более удобочитаемым и упорядоченным. Его будет легче понять как вам, когда вы попытаетесь его прочитать через какое-то время, так и тем, с кем вы решили им поделиться. У сообщества Python есть соответствующее руководство под названием PEP 8. Его полная версия находится по адресу https://www.python.org/dev/peps/pep-0008/.

Примеры в этой книге, за несколькими исключениями, в целом соответствуют PEP 8. Как вы сами увидите, представленный здесь код оформлен по следующему принципу:

```
from lxml import etree ❶
from subprocess import Popen

import argparse ❷
import os
```

```
def get_ip(machine_name): ❸
    pass

class Scanner: ❹
    def __init__(self):
        pass

if __name__ == '__main__': ❺
    scan = Scanner()
    print('hello')
```

В верхней части программы импортируем нужные нам пакеты. Первый блок инструкций импорта ❶ имеет вид `from` *`XXX`* `import` *`YYY`*. Все строчки в нем размещены в алфавитном порядке.

То же самое относится к импорту модулей — они тоже отсортированы по алфавиту ❷. Это позволяет быстро определить, был ли импортирован пакет, не перечитывая каждую строчку с инструкциями `import`. Благодаря этому вы также не импортируете один и тот же пакет дважды. Такой подход направлен на то, чтобы держать код в порядке и избавить вас от лишних раздумий при его повторном чтении.

Затем идут функции ❸ и определения ❹, если таковые имеются. Некоторые программисты полностью отказываются от классов и оформляют все в виде функций. На этот счет нет какого-то железного правила, но если ваше состояние хранится в глобальных переменных или одни и те же структуры данных передаются разным функциям, это может быть признаком того, что программа была бы понятней, если бы ее переписали с применением классов.

Наконец, главный блок внизу ❺ позволяет вам использовать свой код двумя способами. Во-первых, можете выполнить его из командной строки. В этом случае внутренним именем модуля будет `__main__` и выполняться будет главный блок. Так, если файл с кодом называется `scan.py`, его можно запустить из командной строки следующим образом:

```
python scan.py
```

Интерпретатор загрузит функции и классы, находящиеся в `scan.py`, и выполнит главный блок. Вы увидите в консоли ответ `hello`.

Во-вторых, можете импортировать свой код в другой программе без побочных эффектов, например вот так:

```
import scan
```

Поскольку в качестве внутреннего имени используется название модуля Python `scan`, а не `__main__`, вы получите доступ ко всем функциям и классам, определенным в данном модуле, но главный блок при этом не будет выполняться.

Вы также заметите, что мы стараемся не давать переменным бессмысленные имена. Чем лучше будете называть свои переменные, тем понятней будет программа.

Итак, у вас должны быть виртуальная машина, Python 3, виртуальная среда и IDE. Теперь начинается настоящее веселье!