Chapter1

1. pip
2. 可以
3. 是
4. str
5. 官网下载，选择适合自己操作系统的Python版本
6. 如果为不同不同变量赋值相同值，
   这个值在内存中只保存一份，
   多个变量指向同一个值的内存空间首地址，
   这样可以减少内从空间的占用，提高内存利用率

   1. / : 真除法，结果为实数
      2. // : 整除法，向下取整

   1. from 模块名 [as 别名]
      2. from 模块名 import 变量名 [as 别名]
7. 如果脚本作为模块被导入，__name__的值为模块名，
   如果脚本作为主程序运行，name__**的值为**__main.
8. 见Chapter1.py

Chapter2

1. 是一个不可变的序列类型
2. False
3. None
4. remove()
5. [6, 7, 9, 11]
6. {} key value key
7. item() key() get()
8. dict(zip(a,b))
9. b = a[::3]
10. b = [5 for i in range(10)]

11. 不可以
12. 避免改变原列表的迭代器指针

Chapter3

1. for , while
2. 如果or的前面的表达式为真，后面的表达式不会执行

Chapter5

1. global
2. None
3. yield
4. 错
5. 对
6. 在Python中，当你在一个函数内部声明一个变量时，
   这个变量将被视为局部变量，即使它与一个全局变量同名。
   这意味着在函数内部，局部变量的声明和使用会“隐藏”同名的全局变量。
   局部变量的作用域仅限于它被声明的那个函数内部。

Chapter6

1. 封装、继承、多态
2. //:整除。**:幂运算
3. ?
4. 单个下划线 _ :

通常被用作临时变量名，表示一个不需要使用的值，有时也用作循环中的计数器，如果不需要使用循环计数器的值。
在交互式解释器中，表示最后一个表达式的结果。
单个下划线结尾 _ :

在一些情况下，这种命名方式用于避免与 Python 关键字或内置函数同名。例如，class_，list_等。
双下划线开头 __ :

双下划线开头的变量名是 Python
的私有成员的约定，它们只能在类的内部访问，外部无法直接访问。
Python 使用一种名称修饰技术（name
mangling），将双下划线开头的变量名进行了变形，使得它们在类的外部变得不可见。
双下划线开头和结尾 variable：

通常用于特殊用途，例如 Python 内置的特殊方法（例如 init，repr
等），或者一些特殊的变量（例如 name，file 等）。
也可以用作某些特殊情况下的约定，例如 author 表示作者信息，version 表示版本号等。
单个下划线和双下划线开头 _variable 和 __variable：

这种命名方式通常用于表示“内部使用”或“私有”变量，但不具有 Python
中双下划线开头变量名的严格私有性，可以被访问，但不建议在类的外部直接访问。