初学 Python

2.3.1 使用方法修改字符串的大小写

name.title()

name.upper()

name.lower()

2.3.4 删除空白

>>> favorite_language = ' python '

>>> favorite_language.rstrip()

' python'

>>> favorite_language.lstrip()

'python '

>>> favorite_language.strip()

'python'

2.4 数字

2.4.1 整数

Python 使用两个乘号表示乘方运算：

3.1.1 访问列表元素

索引-2 返回倒数第二个列表元素，

索引-3 返回倒数第三个列表元素，以此类推。

3.2.2 在列表中添加元素

1. 在列表末尾添加元素

motorcycles.append('ducati')

你可以先创建一个空列表，再使用一系列的

append()语句添加元素。

motorcycles = []

motorcycles.append('honda')

motorcycles.append('yamaha')

motorcycles.append('suzuki')

print(motorcycles)

2. 在列表中插入元素

motorcycles.insert(0, 'ducati')

3.2.3 从列表中删除元素

1.del motorcycles[0]

2. 使用方法 pop()删除元素

可以使用 pop()来删除列表中任何位置的元素，只需在括号中指定要删除的元素的索引即可。

pop()不指定，弹出栈顶

4. 根据值删除元素

motorcycles.remove('ducati')

使用 remove()从列表中删除元素时，也可接着使用它的值。

方法 remove()只删除第一个指定的值。如果要删除的值可能在列表中出现多次，就需要使用循环来判断是否删除了所有这样的值。

3.3 组织列表

3.3.1 使用方法 sort()对列表进行永久性排序

cars.sort()

print(cars)

汽车是按字母顺序排列的， 再也无法恢复到原来的排列顺序。

还可以按与字母顺序相反的顺序排列列表元素，为此，只需向 sort()方法传递参数 reverse=True。

cars.sort(reverse=True)

print(cars)

3.3.2 使用函数 sorted()对列表进行临时排序

函数 sorted()让你能够按特定顺序显示列表元素，同时不影响它们在列表中的原始排列顺序。

调用函数 sorted()后，列表元素的排列顺序并没有变。

如果你要按与字母顺序相反的顺序显示，也可向函数 sorted()传递参数 reverse=True。

3.3.3 倒着打印列表

cars = ['bmw', 'audi', 'toyota', 'subaru']

```
print(cars)
cars.reverse()
print(cars)
```

方法 reverse()永久性地修改列表元素的排列顺序，但可随时恢复到原来的排列顺序，为此只需对列表再次调用 reverse()即可。

3.3.4 确定列表的长度

Python 计算列表元素数时从 1 开始，因此确定列表长度时，你应该不会遇到差一错误。

```
>>> cars = ['bmw', 'audi', 'toyota', 'subaru']
>>> len(cars)
4
```

4.1 遍历整个列表

```
magicians = ['alice', 'david', 'carolina']
for magician in magicians:
    print(magician)
    print(magician.title() + ", that was a great trick!") //在 for 循环中执行更多的操作
    + ".\n"
```

注意： for 循环结束后执行一些操作、避免缩进错误、忘记缩进、忘记缩进额外的代码行、不必要的缩进、循环后不必要的缩进、遗漏了冒号

4.3 创建数值列表

4.3.1 使用函数 range()

```
for value in range(1,5):
    print(value)
```

range()只是打印数字 1~4，这是你在编程语言中经常看到的差一行为的结果。

要打印数字 1~5，需要使用 range(1,6)

4.3.2 使用 range()创建数字列表

要创建数字列表，可使用函数 list()将 range()的结果直接转换为列表。

```
numbers = list(range(1,6))
print(numbers)
```

结果如下：

[1, 2, 3, 4, 5]

使用函数 range()时，还可指定步长。例如，下面的代码打印 1~10 内的偶数：

```
even_numbers = list(range(2,11,2)) //步长
print(even_numbers)
```

两个星号（**）表示乘方运算

```
    squares = []
    for value in range(1,11):
      square = value**2
      squares.append(square) ==squares.append(value**2)
      print(squares)
```

4. 3. 3 对数字列表执行简单的统计计算

```
>>> digits = [1, 2, 3, 4, 5, 6, 7, 8, 9, 0]
>>> min(digits)
0
>>> max(digits)
9
>>> sum(digits)
45
```

4. 3. 4 列表解析

```
squares = [value**2 for value in range(1,11)] //表达式，for 循环的值给表达式
      print(squares)
[1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
```

4. 4 使用列表的一部分

4. 4. 1 切片

要输出列表中的前三个元素，需要指定索引 0~3，

这将输出分别为 0、1 和 2 的元素。

```
players = ['charles', 'martina', 'michael', 'florence', 'eli']
    print(players[0:3])
```

['charles', 'martina', 'michael']

如果你没有指定第一个索引，Python 将自动从列表开头开始：

print(players[:4])

要让切片终止于列表末尾，

print(players[2:])

负数索引返回离列表末尾相应距离的元素

players = ['charles', 'martina', 'michael', 'florence', 'eli']

print(players[-3:])

上述代码打印最后三名队员的名字，即便队员名单的长度发生变化，也依然如此。

4. 4. 2 遍历切片

如果要遍历列表的部分元素，可在 for 循环中使用切片。

players = ['charles', 'martina', 'michael', 'florence', 'eli']

print("Here are the first three players on my team:")

for player in players[:3]:

print(player.title())

4. 4. 3 复制列表

要复制列表，可创建一个包含整个列表的切片，方法是同时省略起始索引和终止索引（[:]）。

my_foods = ['pizza', 'falafel', 'carrot cake']

friend_foods = my_foods[:]

print("My favorite foods are:")

print(my_foods)

print("\nMy friend's favorite foods are:")

print(friend_foods)

my_foods.append('cannoli') //互不影响

friend_foods.append('ice cream')

倘若我们只是简单地将 my_foods 赋给 friend_foods，就不能得到两个列表。

friend_foods = my_foods

这种语法实际上是让 Python 将新变量 friend_foods 关联到包含在 my_foods 中的列表，因此这两个变量都指向同一个列表。//互相影响

4.5 元组

列表是可以修改的

Python 将不能修改的值称为不可变的，而不可变的列表被称为元组。

4.5.1 定义元组

元组看起来犹如列表，但使用圆括号而不是方括号来标识。定义元组后，就可以使用索引来访问其元素，就像访问列表元素一样。

```
dimensions = (200, 50)
    print(dimensions[0])
print(dimensions[1])
```

4.5.2 遍历元组中的所有值

```
dimensions = (200, 50)
for dimension in dimensions:
print(dimension)
```

4.5.3 修改元组变量

虽然不能修改元组的元素，但可以给存储元组的变量赋值。因此，如果要修改前述矩形的尺寸，可重新定义整个元组：

```
    dimensions = (200, 50)
print("Original dimensions:")
for dimension in dimensions:
print(dimension)


    dimensions = (400, 100)
    print("\nModified dimensions:")
for dimension in dimensions:
print(dimension)
```

5、if 语句

```
cars = ['audi', 'bmw', 'subaru', 'toyota']
for car in cars:
```

```
if car == 'bmw':
    print(car.upper())
else:
    print(car.title())
```

5.2.2 检查是否相等时不考虑大小写

```
>>> car = 'Audi'
>>> car == 'audi'
False

>>> car = 'Audi'
>>> car.lower() == 'audi'
True
```

函数 lower() 不会修改存储在变量 car 中的值，因此进行这样的比较时不会影响原来的变量：

5.2.5 检查多个条件

1. 使用 and 检查多个条件

```
age_0 >= 21 and age_1 >= 21
```

为改善可读性，可将每个测试都分别放在一对括号内，

```
(age_0 >= 21) and (age_1 >= 21)
```

2. 使用 or 检查多个条件

5.2.6 检查特定值是否包含在列表中

要判断特定的值是否已包含在列表中，可使用关键字 in。

```
>>> requested_toppings = ['mushrooms', 'onions', 'pineapple']
    >>> 'mushrooms' in requested_toppings
True
    >>> 'pepperoni' in requested_toppings
False
```

5.2.7 检查特定值是否不包含在列表中

在这种情况下，可使用关键字 not in。

```
banned_users = ['andrew', 'carolina', 'david']
user = 'marie'
```

if user not in banned_users:

5.2.8 布尔表达式

game_active = True

can_edit = False

5.3 if 语句

If 条件表达式后需跟“：”，else 也是

5.3.2 if-else 语句

if age >= 18:

else:

5.3.3 if-elif-else 结构

```
alien_color='yellow'
if alien_color=='green':
    print("Get 5 dots.")
else if alien_color=='yellow':
    print("Get 10 dots.")
else if alien_color=='red':
    print("Get 15 dots.")
```

```
alien_color='yellow'
if alien_color=='green':
    print("Get 5 dots.")
elif alien_color=='yellow':
    print("Get 10 dots.")
elif alien_color=='red':
    print("Get 15 dots.")
```

5.3.5 省略 else 代码块

5.4 使用 if 语句处理列表

5.4.2 确定列表不是空的

requested_toppings = []

if requested_toppings:

for requested_topping in requested_toppings:

print("Adding " + requested_topping + ".")

print("\nFinished making your pizza!")

else:

print("Are you sure you want a plain pizza?")

5.4.3 使用多个列表

available_toppings = ['mushrooms', 'olives', 'green peppers',

'pepperoni', 'pineapple', 'extra cheese']

requested_toppings = ['mushrooms', 'french fries', 'extra cheese']

```
    for requested_topping in requested_toppings:
    if requested_topping in available_toppings:
print("Adding " + requested_topping + ".")
    else:
print("Sorry, we don't have " + requested_topping + ".")
```

6、字 典

6.1 一个简单的字典

```
alien_0 = {'color': 'green', 'points': 5}
print(alien_0['color'])
print(alien_0['points'])
```

6.2 使用字典

字典是一系列键—值对。每个键都与一个值相关联，你可以使用键来访问与之相关联的值。与键相关联的值可以是数字、字符串、列表乃至字典。事实上，可将任何 Python 对象用作字典中的值。

在 Python 中，字典用放在花括号{}中的一系列键—值对表示。

键和值之间用冒号分隔，而键—值对之间用逗号分隔。

6.2.1 访问字典中的值

```
alien_0 = {'color': 'green', 'points': 5}
    new_points = alien_0['points']
    print("You just earned " + str(new_points) + " points!")
```

6.2.2 添加键—值对

```
alien_0 = {'color': 'green', 'points': 5}
print(alien_0)
    alien_0['x_position'] = 0
    alien_0['y_position'] = 25
print(alien_0)
```

注意，键—值对的排列顺序与添加顺序不同。Python 不关心键—值对的添加顺序，而只关心键和值之间的关联关系。

6.2.3 先创建一个空字典

可先使用一对空的花括号定义一个字典，再分行添加各个键—值对。

```
alien_0 = {}
alien_0['color'] = 'green'
alien_0['points'] = 5
```

6.2.4 修改字典中的值

```
alien_0 = {'color': 'green'}
print("The alien is " + alien_0['color'] + ".")
alien_0['color'] = 'yellow'
print("The alien is now " + alien_0['color'] + ".")
```

6.2.5 删除键—值对

对于字典中不再需要的信息，可使用 del 语句将相应的键—值对彻底删除。使用 del 语句时，必须指定字典名和要删除的键。

```
del alien_0['points']
```

6.2.6 由类似对象组成的字典

```
favorite_languages = {
    'jen': 'python',
    'sarah': 'c',
    'edward': 'ruby',
    'phil': 'python',
    }
```

6.3 遍历字典

6.3.1 遍历所有的键—值对

```
user_0 = {
'username': 'efermi',
'first': 'enrico',
'last': 'fermi',
}
    for key, value in user_0.items(): //键-值
    print("\nKey: " + key)
    print("Value: " + value)
```

要编写用于遍历字典的 for 循环，可声明两个变量，用于存储键—值对中的键和值。

for 语句的第二部分包含字典名和方法 items()（见 ），它返回一个键—值对列表。

注意，即便遍历字典时，键—值对的返回顺序也与存储顺序不同。

6.3.2 遍历字典中的所有键

```
for name in favorite_languages.keys():
    print(name.title())
```

遍历字典时，会默认遍历所有的键，因此，如果将上述代码中的 for name in favorite_languages.keys():替换为 for name in favorite_languages:，输出将不变。

方法 keys()并非只能用于遍历；实际上，它返回一个列表，其中包含字典中的所有键

可使用当前键来访问与之相关联的值。

```
❶ friends = ['phil', 'sarah']
  for name in favorite_languages.keys():
      print(name.title())

❷     if name in friends:
          print("  Hi " + name.title() +
              ", I see your favorite language is " +
❸             favorite_languages[name].title() + "!")
```

6.3.3 按顺序遍历字典中的所有键

```
for name in sorted(favorite_languages.keys()):
    print(name.title() + ", thank you for taking the poll.")
```

6.3.4 遍历字典中的所有值

如果你感兴趣的主要是字典包含的值，可使用方法 values()，它返回一个值列表，而不包含任何键。

```
for language in favorite_languages.values():
    print(language.title())
```

但如果被调查者很多，最终的列表可能包含大量的重复项。为剔除重复项，可使用集合（set）。集合类似于列表，但每个元素都必须是独一无二的：

```
for language in set(favorite_languages.values()):
    print(language.title())
```

7 用户输入和 while 循环

运算符+=在存储在 prompt 中的字符串末尾附加一个字符串。

```
prompt = "If you tell us who you are, we can personalize the messages you see."
prompt += "\nWhat is your first name? "
name = input(prompt)
print("\nHello, " + name + "!")
```

使用函数 input()时，Python 将用户输入解读为字符串。

使用函数 int()，它让 Python 将输入视为数值。函数 int()将数字的字符串表示转换为数值表示

```
>>> age = input("How old are you? ")
How old are you? 21
    >>> age = int(age)
>>> age >= 18
True
```

```
height = input("How tall are you, in inches? ")
height = int(height)
```

8 函数

形参名*toppings 中的星号让 Python 创建一个名为 toppings 的空元组，并将收到的所有值都封

装到这个元组中。

8.5.1 结合使用位置实参和任意数量实参

如果要让函数接受不同类型的实参，必须在函数定义中将接纳任意数量实参的形参放在最后。Python 先匹配位置实参和关键字实参，再将余下的实参都收集到最后一个形参中。

9.3.4 重写父类的方法

对于父类的方法，只要它不符合子类模拟的实物的行为，都可对其进行重写。为此，可在子类中定义一个这样的方法，即它与要重写的父类方法同名。这样，Python将不会考虑这个父类方法，而只关注你在子类中定义的相应方法。

假设Car类有一个名为fill_gas_tank()的方法，它对全电动汽车来说毫无意义，因此你可能想重写它。下面演示了一种重写方式：

```
def ElectricCar(Car):
    --snip--

    def fill_gas_tank():
        """电动汽车没有油箱"""
        print("This car doesn't need a gas tank!")
```

现在，如果有人对电动汽车调用方法fill_gas_tank()，Python将忽略Car类中的方法fill_gas_tank()，转而运行上述代码。使用继承时，可让子类保留从父类那里继承而来的精华，并剔除不需要的糟粕。

from collections import OrderedDict

favorite_languages = OrderedDict() 字典无序，使他按赋值的顺序

favorite_languages['jen'] = 'python'

favorite_languages['sarah'] = 'c'

favorite_languages['edward'] = 'ruby'

favorite_languages['phil'] = 'python'

for name, language in favorite_languages.items():

print(name.title() + "'s favorite language is " +

language.title() + ".")

Python 标准库，一个很不错的资源是网

站 Python Module of the Week。请访问 http://pymotw.com/