本次笔记包含四个章节：

1. 入门
2. 变量和简单数据类型
3. 列表简介
4. 列表的操作（以及元组）

1 入门

python主页：https://www.python.org/

MAC从终端输入python即可进入python编程界面。

从终端运行python程序：

```
python helloworld.py
```

第一个python程序：

```
print("Hello World")
```

查询python函数的用法：

```
# 使用help()，例查询sum函数的用法
help(sum)
```

2 变量和简单数据类型

2.1 变量的命名和使用规则

- 变量名只能包含字母、数字和下划线。变量名可以字母或下划线开头，但不能以数字开头。如：num_1。
- 变量名不能包含空格，但可以使用下划线来分割其中的单词。如：hello_word。
- 变量名应既简短又有描述性。如：student_name。
- 慎用小写字母l和大写字母O，因为他们可能被人错看成数字1和0。
- 最好使用小写的Python变量名
- 不要将Python关键字和函数名作为变量名
- **Python关键字**

False	**class**	**finally**	**is**	**return**
None	**continue**	**for**	**lamda**	**try**
True	**def**	**from**	**nonlocal**	**while**
and	**del**	**global**	**not**	**with**
as	**elif**	**if**	**or**	**yield**
assert	**else**	**import**	**pass**	
break	**except**	**in**	**raise**	

- **Python内置函数名**

abs()	**divmod()**	**input()**	**open()**	**staticmethod()**
all()	**enumerate()**	**int()**	**ord()**	**str()**
any()	**eval()**	**isinstance()**	**pow()**	**sum()**
basestring()	**execfile()**	**issubclass()**	**print()**	**super()**
bin()	**file()**	**iter()**	**property()**	**tuple()**
bool()	**filter()**	**len()**	**range()**	**type()**
bytearray()	**float()**	**list()**	**raw_input()**	**unichr()**
callable()	**format()**	**locals()**	**reduce()**	**unicode()**
chr()	**frozenset()**	**long()**	**reload()**	**vars()**
classmethod()	**getattr()**	**map()**	**repr()**	**xrange()**
cmp()	**globals()**	**max()**	**reversed()**	**zip()** **Zip()**

abs()	**divmod()**	**input()**	**open()**	**staticmethod()**
compile()	**hasattr()**	**memoryview()**	**round()**	***_import_()***
complex()	**hash()**	**min()**	**set()**	**apply()**
delattr()	**help()**	**next()**	**setattr()**	**buffer()**
dict()	**hex()**	**object()**	**slice()**	**corece()**
dir()	**id()**	**otc()**	**sorted()**	**interm()**

2.2 字符串

字符串就是一系列字符。在Python中，用引号括起的都是字符串，其中的引号可以是单引号也可以是双引号。

```
print('I told my friend, "Python is my favorite language!"') #字符串中有双引号要用单引号表示字符串
print("It's beautiful!") #字符串中有单引号要用双引号表示字符串
print('''"It's beautiful!"''') #字符串中既有双引号又有单引号
print("""one #三引号用于多行输出
two
three
""")
```

2.2.1 使用方法修改字符串的大小写

```
name = "wade frank"
#title()
print(name.title()) #以首字母大写的方式显示每个单词
#upper()
print(name.upper()) #将字符串改为全部大写
#lower()
print(name.lower()) #将字符串改为全部小写
```

2.2.2 合并字符串

```
first_name = "Wade"
last_name = "Frank"
#使用"+"合并字符串
full_name = first_name + " " + last_name
print(full_name)
print("Hello, " + full_name + "!")
```

2.2.3 使用制表符或换行符来添加空白

在编程中，空白泛指任何非打印字符，如空格、制表符和换行符。你可以使用空白来组织输出，使其更易读。

```
print("\tPython")
print("Python\nC\nJavaScript")
```

2.2.4 删除空白

```
string = " hello "
#rstrip()
string.rstrip() #暂时删除字符串结尾的空白
string = string.rstrip() #永久删除字符串结尾的空白
#lstrip()
string.lstrip() #暂时删除字符串开头的空白
string = string.lstrip() #永久删除字符串开头的空白
#strip()
string.strip() #暂时删除字符串两端的空白
string = string.strip() #永久删除字符串两端的空白
```

2.3 数字

2.3.1 整数

```
2 + 3 #加，结果为5
3 - 2 #减，结果为1
2 * 3 #乘，结果为6
6 / 2 #除，结果为3
3 ** 2 #乘方，结果为9
```

2.3.2 浮点数

带小数点的数，需要注意的是，结果包含的小数点可能是不确定的。

```
>>> 3/2
1.5
>>> 0.2 + 0.1
0.30000000000000004
```

2.3.3 使用str()函数避免类型错误

```
age = 23
message = "Happy " + str(age) + "rd Birthday!"
print(message)
```

2.4 Python之禅

```
>>> import this
The Zen of Python, by Tim Peters

Beautiful is better than ugly.
Explicit is better than implicit.
Simple is better than complex.
Complex is better than complicated.
Flat is better than nested.
Sparse is better than dense.
Readability counts.
Special cases aren't special enough to break the rules.
Although practicality beats purity.
Errors should never pass silently.
Unless explicitly silenced.
In the face of ambiguity, refuse the temptation to guess.
There should be one-- and preferably only one --obvious way to do it.
Although that way may not be obvious at first unless you're Dutch.
Now is better than never.
Although never is often better than *right* now.
If the implementation is hard to explain, it's a bad idea.
If the implementation is easy to explain, it may be a good idea.
Namespaces are one honking great idea -- let's do more of those!
```

3 列表简介

3.1 列表的定义

列表是由一系列按特定顺序排列的元素组成。**列表中的元素可以是不同的类型**。

```
subjects = ["Math", "Chinese", "English"]
print(subjects) #输出整个列表，带方括号
```

3.2 访问列表元素

```
subjects = ["Math", "Chinese", "English", "Computer Science"]
#使用下标访问元素
print(subjects[0]) #访问第一个元素，下标从0开始
print(subjects[-1]) #访问最后一个元素
print(subjects[-2]) #访问倒数第二个元素
```

3.3 修改、添加和删除元素

3.3.1 修改列表元素

```
names = ["Tom", "Jack", "John"]
names[0] = "KiKi"
```

3.3.2 在列表中添加元素

```
names = ["Tom", "Jack", "John"]
#append()
names.append("Frank") #在末尾添加元素
#insert()
names.insert(2, "Jordan") #在第三个位置插入元素（下标从0开始）
```

3.3.3 在列表中删除元素

```
names = ["Tom", "Jack", "John"]
#del
```

```
del names[0] #删除第一个元素，可以删除任何一个元素
#pop()
names.pop() #删除末尾的元素
names.pop(1) #删除第二个元素，可以删除任何一个元素
name = names.pop(1) #删除第二个元素,并返回给name
#remove()
#根据值删除，只删除第一个指定的值，后面的有重复的元素需要用循环来判断
names.remove("Jack") #删除"Jack"
```

3.4 组织列表

3.4.1 永久性排序方法sort()

```
cars = ['bmw', 'audi', 'toyata', 'subaru']
cars.sort() #顺序排序
cars.sort(reverse = True) #逆序排序
print(cars)
```

3.4.2 临时性排序函数sorted()

```
cars = ['bmw', 'audi', 'toyata', 'subaru']
print(sorted(cars)) #输出顺序排序
print(sorted(cars, reverse = True)) #输出逆序排序
print(cars)
```

3.4.3 永久性反转列表方法reverse()

```
cars = ['bmw', 'audi', 'toyata', 'subaru']
cars.reverse() #反转列表
print(cars)
```

3.4.4 确定列表的长度函数len()

```
cars = ['bmw', 'audi', 'toyata', 'subaru']
print(len(cars))
```

4 列表的操作（以及元组）

4.1 使用for循环遍历列表

Python根据缩进来判断代码行与前一个代码行的关系。

```
nba_teams = ['Bulls', 'Heat', 'Lakers', '76ers', 'Celtics']
for team in nba_teams:
  print(team) #自动换行，注意缩进
```

4.2 创建数字列表

4.2.1 使用函数range()

```
for i in range(5): # 从0开始，打印0，1，2，3，4，自动换行
 print(i)
for value in range(1, 5): #左闭右开，所以只能打印1，2，3，4，自动换行
  print(value)
```

4.2.2 使用函数list()将range()的结果直接转换为列表

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

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

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

4.2.3 使用for()+range()来创建任何想要的数字集

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

4.2.4 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()
print(min(digits))
#max()
print(max(digits))
#sum()
print(sum(digits))
```

4.2.5 列表解析

列表解析将for循环和创建新元素的代码合并成一行，并自动附加新元素。(可以减少代码量）

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

4.3 使用列表的一部分

4.3.1 切片

使用切片可以生成列表的任何子集。

```
nba_teams = ['Bulls', 'Heat', 'Lakers', '76ers', 'Celtics']
print(nba_teams[0:3]) # 输出前三只球队'Bulls', 'Heat', 'Lakers'
print(nba_teams[1:4]) #输出中间三只球队'Heat', 'Lakers', '76ers'
print(nba_teams[:4]) #从头开始输出到第4只球队
print(nba_teams[1:]) #从第二只球队开始输出到结尾
print(nba_teams[-3:]) #输出倒数三只球队
```

4.3.2 遍历切片

```
nba_teams = ['Bulls', 'Heat', 'Lakers', '76ers', 'Celtics']
for team in nba_teams[:3]: # 输出前三只球队
  print(team)
```

4.4 复制列表

4.4.1 共享内存的复制

以下代码只是简单的将my_foods赋给friend_foods，所以他们共享内存，内存中只存在一个列表。对my_foods的改变也会改变friend_foods，反过来一样。

```
my_foods = ['pizza', 'cake', 'apple']
friend_foods = my_foods
my_foods.append('banana')
friend_foods.append('ice cream')
print(my_foods)
print(friend_foods)
```

4.4.2 不共享内存的复制

要想新建一个列表，不共享内存空间则需要以下代码：

```
my_foods = ['pizza', 'cake', 'apple']
friend_foods = my_foods[:]
my_foods.append('banana')
friend_foods.append('ice cream')
print(my_foods)
print(friend_foods)
```

4.5 元组

元组是不可变的列表。我们可以使用圆括号来定义元组。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。

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

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，如果要修改前述矩阵的尺寸，可重新定义整个元组：

```
rectangle = (200, 50)
```

```
print(rectangle)

rectangle = (400, 100)
print(rectangle)
```

4.6 代码规范

随着你编写的程序越来越长，有必要了解一些代码格式规定，养成良好的习惯。遵循格式设置约定可以让你的代码易于阅读，同时也可以帮助别人更好的理解你的代码。

4.6.1 缩进

PEP 8（Python Enhancement Proposal）建议每级缩进使用4个空格。

最好不要空格和制表符混用，更推荐全部使用4个空格。

4.6.2 行长

很多Python程序员建议每行不超过80个字符。

PEP 8还建议注释的行长不超过72字符，因为有些工具为大型项目自动生成文档时，会在每行注释开头添加格式化字符。

4.6.3 空行

要将程序的不同部分分开，可以使用空行，但不能滥用。用一个空行来分隔开两个不同的部分是比较合适的。

关于PEP 8的更多内容可以访问：https://python.org/dev/peps/pep-0008/