基础

整型、浮点型和字符串数据类型

字符串连接和复制

- ```
  'abc' + 'def'
  ```
- ```
  'abc' * 5 #字符串变为 abc 重复5次，必须为整型
  ```

变量命名规则

1. 只能是一个词。
2. 只能包含字母、数字和下划线。
3. 不能以数字开头。

print()函数

关键字：

- seq 会指定print()内元素间隔用什么取代，默认是空格

  ```
  print('hello', 'world', seq='')
  ```
- end 会指定两个`print()'之间是否换行，默认是自动添加换行

  ```
  print('hello',end='')
  print('world')
  ```

input()函数

- 下例会在输入前输出 Your name is 作为提示，然后将输入赋给 name

  ```
  name = input('Your name is: ')
  ```

len()函数

- 获取长度，如字符串长度，数组长度

str() float() int()函数

- 返回相应类型的数据

控制流

布尔值

- True
- False
- 首字母大写

比较操作符

布尔操作符

- and
- not
- or

控制流语句

代码块规则：

1. 缩进增加时，代码块开始。
2. 代码块可以包含其他代码块。
3. 缩进减少为零，或减少为外面包围代码块的缩进，代码块就结束了。

if else elif

```
if name == 'Alice':
  print('Hi, Alice.')
elif age < 12:
  print('You are not Alice, kiddo.')
else:
  print('You are neither Alice nor a little kid.')
```

while

```
name = ''
while name != 'your name':
  print('Please type your name.')
  name = input()
  print('Thank you!')
```

break跳出该层循环

continue直接进行下次循环

for 循环和range()函数

```
name = ''
for i in range(7):
  print(i)
```

$[0,7)$

range()的开始、停止和步长参数

```
range(12,16)  #[12,16)
range(0,10,2) #第三个元素为步长 0 2 4 6 8
```

导入模块

import 语句包含以下部分：

- import 关键字；
- 模块的名称；
- 可选的更多模块名称，之间用逗号隔开。

```
import random
import random, sys, os, math
# 调用模块内函数时候
random.randit(1,7)  #必须以模块名.函数名的形式
from random import *  #若以该形式导入模块就直接使用函数名调用
```

用sys.exit()提前结束程序

函数

```
  def hello(name):
    print('Hello ' + name)
    return name
```

None 值

- None 是NoneType 数据类型的唯一值（其他编程语言可能称这个值为null、nil 或undefined）。就像布尔值True和False 一样，None 必须大写首字母N。

局部和全局作用域

名称相同的局部变量和全局变量

```
def spam():
  eggs = 'spam local'
  print(eggs) # prints 'spam local'

def bacon():
```

```
  eggs = 'bacon local'
  print(eggs) # prints 'bacon local'
  spam()
  print(eggs) # prints 'bacon local'

eggs = 'global'
bacon()
print(eggs) # prints 'global'
```

结果会是

> bacon local
> spam local
> bacon local
> global
> 变量名相同时，会屏蔽全局变量

global 语句

如果需要在一个函数内修改全局变量，就使用global 语句

```
def spam():
  global eggs
  eggs = 'spam'

eggs = 'global'
spam()
print(eggs)
```

异常梳理

如果在try 子句中的代码导致一个错误，程序执行就立即转到except 子句的代 码。

```
def spam(divideBy):
try:
  return 42 / divideBy
except ZeroDivisionError:
  print('Error: Invalid argument.')
```

列表

列表数据类型

- ['cat', 'bat', 'rat', 'elephant']
- []是一个空列表，不包含任何值，类似于空字符串

用下标取得列表中的单个值

```

```

```
spam = ['cat', 'bat', 'rat', 'elephant']
spam[1]

spam = [['cat', 'bat'], [10, 20, 30, 40, 50]]
spam[0]
spam[0][1]
```

负数下标

- 整数值-1 指的是列表中的最后一个下标
- -2 指的是列表中倒数第二个下标，以此类推

利用切片取得子列表

```
spam[1:3] #左闭右开
spam[:2]
spam[:]
```

用下标改变列表中的值

列表连接和列表复制

```
[1, 2, 3] + ['A', 'B', 'C']
['X', 'Y', 'Z'] * 3
```

用del 语句从列表中删除值

- del 语句将删除列表中下标处的值，表中被删除值后面的所有值，都将向前移 动一个下标

列表用于循环

in 和not in 操作符

```
'howdy' in ['hello', 'hi', 'howdy', 'heyas']
#返回布尔值
```

多重赋值技巧

```
cat = ['fat', 'black', 'loud']
size, color, disposition = cat
# size color disposition 将依次被赋予与列表cat顺序相同的值
```

增强的赋值操作

```
bacon *= 3
```

方法

用index()方法在列表中查找值

- 列表值有一个index()方法，可以传入一个值，如果该值存在于列表中，就返回它 的下标。如果该值不在列表中，Python 就报 **ValueError**
- 如果列表中存在重复的值，就返回它第一次出现的下标

```
spam = ['hello', 'hi', 'howdy', 'heyas']
spam.index('hello') #返回下标0
```

用append()和insert()方法在列表中添加值