## 关于Python的基础知识（Basic knowlegde on Python）

## 常用的数字数据类型（Common used numeric data types）

| **Name** | **Notation** | **Declaration e.g.** |
| --- | --- | --- |
| Integers | int | a = 10 |
| Floating | float | b = 3.14 |
| Complex | complex | c = 1 + 2j |
| String | str | d = 'Python' |

* 备注：
* 在Python 2.7中，int与另一个int运算将导致int结果。 但是，一个浮点运算与int会导致浮点数。
* 在Python 3.x中，int与另一个int运算将导致浮点数。

## 算数运算符（Arithmetic operators）

| **Name** | **Notation** | **Examples** |
| --- | --- | --- |
| Addition | + | a + b |
| Subtraction | - | c - b |
| Multiplication | \* | x\*y |
| Division | / | x/z |
| Modulus | % | x%a |
| Exponent | \*\* | a\*\*x |

## 浮点数的精度（Precision of float）

* 尝试在您的控制台中键入0.1 + 0.2。 你会发现这个值是
* 0.30000000000000004
* 这是二进制浮点的本质。 您可以在支持硬件浮点运算的所有语言中看到同样的东西。
* 可以使用“round（）”功能控制显示精度，但也有上述情况，这意味着round(9.995,2)返回9.99而不是10，因为9.995的存储稍小于9.995。
* decimal Library将给出精确的存储值，请参见以下示例。

## 例子（Example）

import decimal # import the libray "decimal" # display 2 decimal precisionprint (round (3\*1415 , 2)) # result 3. 14print (round (9 .995 , 2)) # result 9. 99

#call function "Decimal " from lib "decimal"print (decimal.Decimal (9.995))

The last "print" returns9.9949999999999992184029906638897955417633056640625,

which is exactly how 9.995 is stored in the hardware.

## 练习3：字符串str的一些功能（Exercise3:Some functions for str）

t = 'He is a string. Who are you?'print(t.capitalize()) # Cap first letterprint(t.split()) # split by wordsprint(t.find('i')) # return index of 'i'print(t.find('in')) # index of 'i' in 'in'print(t.find('Python')) # find sth not inprint(t[0:4]) # returu from index 0 to 3print(t.replace(' ','|')) # replace by '|'

w = 'http://www.google.com'print(w.strip('http://')) #delete sth

## Python功能：索引（Python features: Indexing）

* Python具有与C ++类似的索引规则，其起始索引为0。
* 当通过索引返回值时，间隔实际上是[，]样式，这意味着不包括终端索引。

## 基本的数据结构（Basic data structures）

| **Name** | **Nation** | **Declaration e.g.** |
| --- | --- | --- |
| Tuple | tuple | b = (1,2.5, 'data') |
| List | list | c = [1,2.5,'data'] |
| Dictionary | dict | d = {'Name': 'Kobe', 'Country':'US'} |
| Set | set | e = set(['u','d','ud','d','du']) |

* 元组（tuple）只有几种方法可以更改。
* 列表（list）比元组更灵活。
* 字典（dict）是一个键值对存储对象。
* 集合（set）是对象中唯一的无序集合对象。

## 列表的一些有用功能（Some useful functions for list）

l = [1, 2, 3.14, 'data'] #listprint (type(l))l.append ([4, 3])print(l)l.extend (['delta' ,5 ,6] ) #add a listprint(l)l.insert(3, 'beta') #insert before index 3print(l)l.remove ('data') #delete an elementprint(l)

## Python功能：参考对象（Python features: Refer for object）

* 在Python中，如果要将值从一个对象传递给另一个对象，则=(等号)将按地址传递值。
* 例如，

x = [1, 2. 3, 4]

y = x

y[0] = 5print(x)

x = [1, 2, 3, 4]

z = x.copy()

z[0] = 5print (x)

输出将会是[5,2,3,4],[1,2,3,4]

## 多维列表（Multidimensional list）

* 我们可以创建一个包含多行的列表

a = [[1,2 , 3 ,4],[1,2 ,3,4],[1,2 ,3]] print(a)print(a[0][3])

* 但请注意： 多维列表不是矩阵。

## 条件语句（Conditions）

* 条件控制元素包括if，else和elif。
* 对于条件语句，我们有如下几种：

| **Name** | **Notation** |
| --- | --- |
| larger | > |
| smaller | < |
| equal | == |
| larger or equal | >= |
| sma |  |

* 对于多条件同时使用的情况，我么使用and, or 或者 not作为关键字来相互衔接

## 示例和练习：条件语句 (Example & exercise: conditions)

a = [24, 16, 54]

b = []if a[0]< a[1] and a[0]< a[2] :

b.append(a[0])

if a[1] < a[2]:

b.append(a[1])

b.append(a[2])

else:

b.append(a[2])

b.append(a[1])

## 循环语句 （Loops）

* 循环语句有很多不同的样式代码，我们只提供两个常用的语句。

for...in ... ：statement A

是循环中最常用的语句，通常与range（start,end,step）一起使用,start为起始值，end为结束值，step为步长。 例如，

range(0,8,1) 给出[0，1，2，3，4，5，6，7]

2/

while ...：statement A

将会执行A语句，直到满足while的条件。

## 举例：循环语言(Example: loops)

# for和range的例子 example of for and range # 初始值默认值为default start of range is O # 步长默认值为default step of range is 1 for i in range(2, 10, 3):

print(i)

l= i\*\*2

print(l)

# white to sum up 1 to 100

a = 0

sumup = Owhile a < 100 :

a + 1

sumup += a

print ( sumup)

## break和continue.

* 你可以在循环语句中使用关键字break，以跳出循环。(you can use the keyword break inside a loop to leave the loop..)
* 你也可以在循环语句中使用关键字continue，以暂停当前循环执行后面的语句。(you can use the keyword continue inside a loop to stop pracessing the current iteration of the loop and immediately go on to the next round.)
* 举例 E.g.

# search the first# be divided by 17for i in range(300, 351):

if i % 17 == O:

print (i)

break

else :

cantinue

## 循环语句嵌套（Loop inside the loop）

* 循环语句可以内嵌在另一个循环语句中(Loop can be written inside another loop)

for i in range (10):

print (i)

for j in range (5):

print (j)

## 练习(Exercises)

* 计算从1到1000的累计值。
* 计算从1到1000的偶数之和。

## 功能 (Functions)

## 功能(方法or函数)的声明 (Function declaration)

* 方法定义如下(Functions are defined as)

def TheNameOfFunction(paral, para2):

...

return Outcome

* 函数（方法）返回的输出结果会在函数被调用的地方出现。

## 举例：求两个变量最大值的函数

def MaxOfTwo (x1, x2):

if x1 >= x2:

return x1

else:

return x2

a = l

b = 2

c = MaxOfTwo(a, b)print(c)

## 默认参数

* Python中的函数没有函数重载(function overloading)。 这意味着你不能有两个共享同名的功能。 但是操作符重载（operator overloading）是正常的。
* 您可以为函数的参数提供默认值，例如。

def MaxOfTwo(xl, x2 = 1): ...

* 请将您的默认参数放在一堆函数参数的末尾处。

## 具有两个输出的函数（方法）

* 函数还可以返回两个或多个输出。 在这种情况下，你应该这样写出代码：

def f (x1, x2, x3, ...):

......

return(y1, y2, y3, ...)

a1,b1,c1 = f(...)

## 读取/写入文件 (Reading/ writing files)

## 内建open()方法 [Built-in open() function]

* 要打开一个文件，使用Python内建的open()函数。 open()方法返回一个文件对象，最常用的一般使用两个参数。

file\_object = open(filename, mode)

* 这个mode参数可以是（The mode can be）
  + 'r' 只读模式（when the file will only be read）
  + 'w' 只写模式（与一个现存文件文件同名，则被清除）
  + 'a' 添加模式，即任意写入文件的数据都被自动添加到末尾
  + 'r+' 打开文件，可以读、写

## 创建一个文件 (Creating new file)

file = open('newfile.txt', 'w')

file.write('I am created for the course. \n')

file.write('How about you? ')

file.write('How is your exam?')

file.close()

## 读取一个文件 (Reading a file)

file = open('newfile.txt', 'r')#show whole efileprint(file.read())#show first ten characterrsprint(file.read(10))#view by lineprint(file.readline())#view all by lineprint(file.readlines())file.close()

## 循环读取一个文件 (Looping over a file object)

file = open('newfile.txt', 'r')for line in file:

print (line)file.close()

输出将是

我是为这个课程而诞生的

你怎么样？你的考试如何

\* ------以下是英文原文--------------------

Output would be:

I am created for the course

How about you? How is your exam?

## 增加一个文件 (Adding in a file)

file = open（'newfile.txt', 'a')

file.write('\nI am back again. \n'）

file.write('Do you miss me?\n')

file.close()

## with语句 (The with statement)

* 很容易忘记关闭{close()}文件。 由于这个和其他原因，最好使用with语句:

with open(“humpty.txt”) as f：

* 这样可以确保文件正确关闭，即使读取时发生错误。

## 文件路径 (File paths)

* 也可以使用绝对路径指定文件名。
* 示例（Mac / Linux）：

open ('/etc/gimp/2.O/gtkrc')

* 示例（Windows）：

open('C:\Users\user\Documents\file.txt')

* 请注意，反斜杠需要转义（写入两次'\'）

## 导入模块 (Import modules)

* 可以通过输入import关键字来导入模块

import numpy

* 或者使用简称，即将模块通过as关键字来命名一个简称

import numpy as np

* 有时您不必导入整个模块，就像

from scipy.stats import norm

## 我们会遇到的模块

NumPy：多维数组的有效操作。 高效的数学函数。

Matplotlib：可视化：2D和（最近）3D图

SciPy：大型库实现各种数值算法，例如：

* + 线性和非线性方程的解
  + 优化
  + 数值整合

Sympy：符号计算（解析的 Analytical）

Pandas：统计与数据分析

## Numpy

## ndarray类型

* NumPy提供了一种新的数据类型：ndarray（n维数组）。
  + 与元组和列表不同，数组只能存储相同类型的对象（例如只有floats或只有ints）
  + 这使得数组上的操作比列表快得多; 此外，阵列占用的内存少于列表。
  + 数组为列表索引机制提供强大的扩展。

## 创建ndarray

* 我们导入Numpy（在脚本的开头或终端）：

import numpy as np

* 然后我们创建numpy数组：

In [1] : np.array([2, 3, 6, 7])

Out[l] : array([2, 3, 6, 7])

In [2] : np.array([2, 3, 6, 7.])

Out [2] :  array([ 2.,  3.,  6., 7.])  <- Hamogenaous

In  [3] :  np.array( [2,  3,  6,  7+ij])

Out [3] :  array([ 2.+O.j,  3.+O.j,  6.+O.j,  7.+1.j])

------------------------------以下为英文原文-------------------------------------

Create the ndarray

* We import Numpy (at the beginning of the script or in the terminal):

import numpy as np

* Then we create numpy arrays:

In [1] : np.array([2, 3, 6, 7])

Out[l] : array([2, 3, 6, 7])

In [2] : np.array([2, 3, 6, 7.])

Out [2] :  array([ 2.,  3.,  6., 7.])  <- Hamogenaous

In  [3] :  np.array( [2,  3,  6,  7+ij])

Out [3] :  array([ 2.+O.j,  3.+O.j,  6.+O.j,  7.+1.j])

## 创建均匀间隔的数组

* arange：

in[1]：np.arange（5）

Out [l]：array（[0，1，2，3，4]）

range(start, stop, step)的所有三个参数即起始值，结束值，步长都是可以用的 另外还有一个数据的dtype参数

in[2]：np.arange（10，100，20，dtype = float）

Out [2]：array（[10.，30.，50.，70.，90.]）

* linspace（start，stop，num）返回数字间隔均匀的样本，按区间[start，stop]计算：

in[3]：np.linspace（0.，2.5，5）

Out [3]：array([0.，0.625，1.25，1.875，2.5])

   这在生成plots图表中非常有用。

* 注释：即从0开始，到2.5结束，然后分成5等份

## 多维数组矩阵 (Matrix by multidimensional array)

In [1] : a = np.array([[l, 2, 3]  [4, 5, 6]])

                         ^ 第一行 (Row 1)

In [2] : a

Out [2] : array([[l, 2,  3] ,   [4,  5,  6]])

In  [3] : a.shape  #<- 行、列数等 (Number of rows, columns etc.)

Out [3] : (2,3)

In  [4] : a.ndim   #<- 维度数 (Number of dimensions)

Out [4] : 2

In  [5] : a,size   #<- 元素数量 (Total number of elements)

Out [5] : 6

## 形状变化 (Shape changing)

import numpy as np

a = np .arange(0, 20, 1) ＃1维

b = a.reshape((4, 5)) ＃4行5列

c = a.reshape((20, 1)) ＃2维

d = a.reshape((-1, 4)) ＃-1：自动确定

＃a.shape =(4, 5) ＃改变a的形状

## Size（N，），（N，1）和（1，N）是不同的!!!

* Size（N, ）表示数组是一维的。
* Size（N，1）表示数组是维数为2, N列和1行。
* Size（1，N）表示数组是维数为2, 1行和N列。

Reference

<https://zhuanlan.zhihu.com/p/29184240>

https://liam.page/2017/07/23/modules-and-packages-of-python/