Python 编程

Python 的特点：

一、Python 的安装、使用、保存（以 Windows7 为例）

1、用网页浏览器打开 http;//www.python.org/，然后下载最新版的 Python3 安装程序（installer）。

2、双击图标，按提示安装：

选择“Install for all Users”⟹ 点击“Next”⟹ 留意安装路径（eg C;\Python32 或 C:\Python34) ⟹点击“Next”⟹ 点击“Next” ⟹

3、创建 Python3 快捷方式到桌面

右键点击桌面 ⟹在弹出的菜单中选择“新建——快捷方式” ⟹在注有“输入项目的位置”的框中输入 C:\Python34\Lib\idlelib\idle.pyw -n ⟹ 点击“下一步” ⟹ ⟹

输入 IDLE 作为名字 ⟹ 点击“完成”来创建快捷方式。

4、安装完成后，创建第一个 Python 程序

双击图标打开“Python shell”程序 ⟹ 在提示符（>>>）后面输入一些命令

eg ：>>>print（“Hello World”）回车键

5、保存 Python 程序：

打开 IDLE 程序 ⟹ 选择“文件——新窗口”，然后会出现一个空白窗口，在菜单上有“Untitled”字样，在新窗口中输入：print（“Hello World”） ⟹选择“文件——保存” （当提示输入文件名时输入 hell.py，并把文件保存在桌面）⟹选择“运行——运行模块”

二、计算与变量

1、用 Python 来做计算

（1）Python 的运算符

@运算符：Python 用来做数学运算的那些基本符号叫运算符。

如： + 加 - 减 * 乘 / 除

（2）运算顺序：括号优先、先乘除后加减。（括号可以改变运算顺序)

@运算：任何用到运算符的东西都是一个运算。

2、使用变量

@变量：在编程时，变量指一个存储信息的地方，例如数字、文本、由数字和文本组成的列表等。（变量可以看做是贴在东西上的标签）

@变量名：可以由数字、字母和下划线字符（_）组成，但数字不能开头。变量名不能包含空格，要用下划线来代替。

三、字符串、列表、元祖和字典

1、字符串（string）

@字符串：在编程术语中，通常把文字称为“字符串”。

（1）创建字符串：通过给文本添加引号来创建字符串

！引号用英文的，可以是‘’或“”。

!要在字符串中使用多于一行的文字，即多行字符串，需要使用三个单引号(‘‘‘),并在行之间输入回车。

@Syntax：语法，指语句中文字、符号的排列和顺序。

（2）处理字符串相关的问题

！要在字符串中加入单引号和双引号时可以用三个单引号，即多行字符串，或者是在用单或双引号括起来的字符串中的每个引号前加上一个反斜杠\（这叫做“转义”escaping），反斜杠不会出现在打印出来的字符串里。

（3）在字符串里嵌入值：用%s 占位符,使用%来告诉 Python 把%s 替换成什么。

对于同一个占位符，可以用不同的变量来传给它不同的值。

在一个字符串中可以使用多个占位符，在输出时要把替换的值用括号括起来，值排放的顺序就是它们在字符中被引用的顺序

（4）字符串乘法

10*a 在 Python 中显示的是 aaaaaaaaaa

可以采用这一功能来用一定数量的空格对齐字符串。

2、列表

通过[]来创建列表

列表比字符串更有用，我们可以对列表进行操作。如，

①打印列表中的指定元素：

在方框[]中输入列表中的位置（这叫“索引位置”),就可以打印该元素。列表中的第一个元素是 0，第二个是 1，第三个是 2。

②改变列表中的某个元素：

③显示列表的一个子集：

通过在方括号中使用冒号来做到这一点。[2:5]是显示从索引位置 2 直到（但不包括）索引位置 5 的元素，即元素 2、3、4.

④列表可以用来存放各种元素，如：数字、字符串（字符串用有引号）或二者的混合

⑤列表中可以保存其他列表

（1）添加元素到列表：用到 append 函数，可以把一个元素添加到列表的最后

（函数：就是让 Python 做某些事情的一段代码）

（2）从列表中删除元素

Eg 从巫师的列表中删除第六个元素“蛇蜕皮”

```
>>>del wizard_list[5]
```

（3）列表上的算术: 只能用+和*，-和/会出现错误。

使用 + 可以把俩个列表加起来，即成为一个更大的列表。使用 * 可以使列表中的元素重复，如 list*5 就表示列

表重复 5 次。

3、元组

@元组：就像是一个使用括号（）的列表，与列表不同的是元组一旦创建就不能再做改动了

4、map

@字典（dict），也叫 map，映射。也是一堆东西的组合。字典与列表和元组不同的地方在于字典中的每个元素都有一个键（key）和一个对应的值（value），不能用+运算符来把两个字典连接起来。

字典中的所有元素都用大括号｛｝括起来，用冒号：把每个键和它的值分开，每个键和值都分别用单引号括起来，没对键和值之间用逗号，隔开。

①用键来访问字典就可以得到它的对应值

Eg: >>>print(favourite_sports[‘Rebecca Clarke’]

Netball

②用键来删除字典中的值：

Eg: >>>del favourite_sports[‘Rebecca Clarke’]

③用键来替换字典中的值

四、用海龟画图

@模块：给别的程序提供有用的代码的一种方式。

1、使用 Python 的 turtle（海龟）模块

Turtle 模块提供了编写向量图的方法，可以画简单的直线、点和曲线。

（1）让 Python 引入 turtle 模块

>>>import turtle 回车

（2）创建画布

即调用 turtle 模块中的 pen 函数，它会自动创建一个画布。在 PythonShell 中输入

>>>t=turtle.Pen（）回车

之后会出现一个空白的方块（画布），中间有一个箭头。

（3）移动海龟

>>>t.forward(50)	向前移动 50 个像素
>>>t.backward(50)	向后移动 50 个像素
>>>t.left(90)	左转 90 度
>>>t.right(90)	右转 90 度
>>>t.reset()	擦除画布
>>>t.chesr（）	擦除画布
>>>t.up()	停止作画
>>>t.down()	继续开始作画

五、用 if 和 else 来提问

1、if 语句

是由关键字 if 构成的，后面跟着一个条件和一个冒号

（;），eg if age>20:。冒号之后的代码行必须放到一个语句块中。

2、语句块：就是一组程序语句

!在 Python 中空白是有意义的，如制表符（tab）和空格（）；处在同一位置的代码，即缩进同样数量的空格，组成一个代码块。

！当新起一行，并且用来比前一行多的空格，那么你就开始了一个新的代码块，这个代码块是前一个代码块的一部分。

3、用条件语句来做比较

用于条件的符号：

==	等于
!=	不等于
>	大于
<	小于
>=	大于等于
<=	小于等于

4、If—then—else 语句

该语句相当于说，“如果某事为真，那么这样做；否则那样做。”

关键字 else 是在各种条件都不为真时执行另一段代码。Else 顶齐，后用冒号，然后另起一行写代码。

Eg else：

Print（。。。）

5、If 和 elif 语句

elif 是 else_if（否则—如果）的缩写，该语句与 if_then_else 语句不同在于，在同一个语句中可以有多个 elif。

6、组合条件

！可以使用 and 和 or 来把条件组合起来，这样会产生更加简短的代码。

7、没有值的变量—None

在 Python 中我们可以给变量赋值为什么也没有，或者说空的值，我们把空的值叫做 None，它的含意就是没有值。

！注意 None 和 0 这个值是不同的它代表没有值，而不是一个值为 0 的数字。

变量 None 可以用来把变量重置到它初始为空的状态。

8、字符串与数字之间的不同

在 Python 中数字 10 和字符串‘10’不同，Python 中有函数可以把字符串变成数字，把数字变成字符串。如：

⑴把字符串转换成数字用 int，（int 函数需要的是一个整数，不适用于小数）

>>>age=‘10’

>>>convertde_age=int（age）

现在变量 convertde_age 的值就是数字 10 了

⑵float 函数可以用来处理不是整数类型的数字，float 议为“浮点数”，是计算机表示小数的一种方式。

>>>age=‘10.5’

>>>convertde_age=float(age)

>>>print(convertde_age)

10.5

⑶把数字转换成字符串用 str

>>>age=10

>>>convertde_age=str（age）

现在 convertde_age 的值是字符串‘10’

六、循环

1、for 循环

使用 for 循环可以减少需要输入的数字和重复工作。

For 循环是针对指定长度的循环，而 while 循环是用于事先不知道何时停止的情况。

2、While 循环

（1）! 通常可以用关键字 break（打断）来退出循环。break 用来立刻从循环中跳出来，让循环停止，对于 for 和 while 循环都适用。

（2)While 循环的步骤：

检查条件——执行语句块中的代码——重复。

(3)While 循环的另外一个作用是创建“半永久”的循环。这种循环可能会永远执行下去，但实际上他会继续直到代码中有什么事发生，然后从里面跳出来。

七 使用函数和模块来重用你的代码

！创建一个连续数字的列表的方法：

```
>>>list（range（0,5））
[0,1,2,3,4]
```

1、使用函数——可以重复使用代码

（1）函数的组成部分

一个函数由三部分组成：名字、参数和函数体。

函数常常需要返回一个值，这就用到了 return（返回）语句。

（2）变量和作用域

在函数体内的变量在函数执行结束后就不能再用了，因为它只在函数中存在。在编写程序的世界里，这被称为“作用域”。

如果一个变量定义在函数之外，那么它的作用域则不一样。如，在创建函数之前先定义一个变量，这个变量就能够在函数内使用。

```
Eg >>>def spaceship_building（cans）:
```

```
total_cans=0
for week in range（1,53）：对于一年中每一周的循环
    total_cans=total_cans+cans
    Print（‘week %s = %s cans’ %（week，total_cans））
```

2、使用模块

@模块是用来把函数、变量、以及其他东西组织成更大的、更强的程序。

八 如何使用类和对象

1、把事物拆分成类

用关键字 class 来定义类，后面跟着一个名字。用 pass 语句来告诉 Python 我们不会给出更多的信息，当我们想提供一个类或者一个函数，却暂时不想填入具体信息的时候就可以使用 pass。

（1）父母与孩子

如果一个类是另外一个类家族的一部分，那么它是另外一个类的“孩子”，另一个类是它的“父亲”。

（2）增加属于类的对象

（3）定义类中的函数

（4）用函数来表示类的特征

（5）为什么要使用类和对象

（6）画图中的对象和类

2、对象和类的另一些实用功能

（1）函数继承

一个对象可以调用它所属类中定义的函数以及任何父类中定义的函数，因为这些函数已近被继承过来了。

（2）从函数里调用其他函数

用到 self 参数，self 参数可以用来从类中的一个函数调用另外一个函数。

```
>>> class Giraffes(Mammals):
        def find_food(self):
            self.move()
            print("I've found food!")
            self.eat_food()
```

3、初始化对象

九、Python 的内建函数

模块要先被引用才能使用。

内建函数不需要引用

1、使用内建函数

（1）abs 函数

@返回一个数字的“绝对值。

用法：abs 函数的用法很简单，把数字或变量当成参数就可以看。

Eg　>>> print(abs(-10))

```
>>> a=abs(10)+abs(-10)
>>> print(a)
20
>>> b=abs(-10)+-10
>>> print(b)
0
>>>
```

10

（2）bool 函数

@bool 是 Boolean（布尔类型）的简写，程序员们用它来表示两种可能值中的一种，通常是真（true）或者假（false）。

bool 函数只有一个参数，并根据这个参数值返回真或者假。当对数字使用 bool 都返回真。当对其他类型的 串，对于没有值的字符串，也 返回 false，否则返回真。

```
>>> print(bool(0))
False
>>> print(bool(-1))
True
>>> print(bool(None))
False
>>> print(bool(' '))
True
>>> print(bool('How are you!'))
True
```

bool 函数对于空的列表、元组和字典返回假，否则返回真。

```
>>> my_silly_list=[]
>>> print(bool(my_silly_list))
False
>>> my_silly_list=['s','i','l']
>>> print(bool(my_silly_list))
True
```

可以用 bool 函数来验证一个值是否已被设置。如，我们叫人们用我们的程序输入他的出生年份，我们的 if 语句可以用 bool 函数来验证输入值：

```
>>> year=input('year of birth')
year of birth
>>> if not bool(year.rstrip()):
        print('you need to enter a value for you year of birth')


you need to enter a value for you year of birth
>>> year=input('year of birth')
year of birth90
>>> if not bool(year.rstrip()):
        print('you need to enter a value for you year of birth')


>>>
```

这个列子的第一行使用 input 来吧别人在键盘上敲得东西保存到变量 year 中。在下一行中直接按回车（不输入任何东西），这样会把回车键的值保存在变量中。在接下来的一行，if 语句把 rstrip 函数的返回值当做布尔值检查（rstrip 函数把字符串结尾的空白和回车删除）。因为在例子里用户没有任何输入，所以 bool 函数返回 False。因为 if 语句使用了 not 关键字，意思就是“如果函数没有返回 true 的话才做这件事情”，所以代码会在下一行打印出：you need to enter a value for you year of birth。

？？？？？？

```
>>> import sys
>>> print(sys.stdin.readline())


>>> if not bool(sys.stdin.readline()):
        print('you need to enter a value...')


>>>
```

（3）dir 函数

dir 是 directory，目录的缩写。dir 函数可以返回关于任何值得相关信息。基本上就是按照字母顺序告诉你那个值上面可以使用的函数都有什么。

dir 函数基本上可以用于任何东西，包括字符串、数字、函数、模块、对象和类。

显示对一个列表值可用的函数：

```
>>> dir(['a','short',1])
['__add__', '__class__', '__contains__', '__delattr__', '__delitem__', '__dir__'
, '__doc__', '__eq__', '__format__', '__ge__', '__getattribute__', '__getitem__'
, '__gt__', '__hash__', '__iadd__', '__imul__', '__init__', '__iter__', '__le__'
, '__len__', '__lt__', '__mul__', '__ne__', '__new__', '__reduce__', '__reduce_e
x__', '__repr__', '__reversed__', '__rmul__', '__setattr__', '__setitem__', '__s
izeof__', '__str__', '__subclasshook__', 'append', 'clear', 'copy', 'count', 'ex
tend', 'index', 'insert', 'pop', 'remove', 'reverse', 'sort']
```

当你想要快速找到在一个变量上可以做些什么的时候，dir 函数很有用。如：对一个包含字符串值得叫 popcorn 的变量调用 dir 函数，就会得到一系列 string 类所提供的函数（所有字符串都属于 string 类，即 str 类）。

```
>>> popcorn='I love popcorn!'
>>> dir(popcorn)
['__add__', '__class__', '__contains__', '__delattr__', '__dir__', '__doc__', '_
_eq__', '__format__', '__ge__', '__getattribute__', '__getitem__', '__getnewargs
__', '__gt__', '__hash__', '__init__', '__iter__', '__le__', '__len__', '__lt__'
, '__mod__', '__mul__', '__ne__', '__new__', '__reduce__', '__reduce_ex__', '__r
epr__', '__rmod__', '__rmul__', '__setattr__', '__sizeof__', '__str__', '__subcl
asshook__', 'capitalize', 'casefold', 'center', 'count', 'encode', 'endswith', '
expandtabs', 'find', 'format', 'format_map', 'index', 'isalnum', 'isalpha', 'isd
ecimal', 'isdigit', 'isidentifier', 'islower', 'isnumeric', 'isprintable', 'issp
ace', 'istitle', 'isupper', 'join', 'ljust', 'lower', 'lstrip', 'maketrans', 'pa
rtition', 'replace', 'rfind', 'rindex', 'rjust', 'rpartition', 'rsplit', 'rstrip
', 'split', 'splitlines', 'startswith', 'strip', 'swapcase', 'title', 'translate
', 'upper', 'zfill']
>>>
```

然后可以用 help 得到列表中某个函数的简单描述。

```
>>> help(popcorn.upper)
Help on built-in function upper:

upper(...) method of builtins.str instance
    S.upper() -> str

    Return a copy of S converted to uppercase.

>>>
```

返回信息中省略号意味着 upper 是一个 string 类内建的函数并且没有参数，下一行的箭头意思是这个函数返回一个字符串（str）。最后一行给出了这个函数简要的简绍。

（4）eval 函数

eval 是 evaluate“估值”的缩写，eval 函数把一个字符串作为参数并返回它作为一个 Python 表达式的结果。

（5）exec 函数

与 eval 函数相似，不同点在于 eval 函数返回一个值，而 exec 函数不会，而且 exec 函数适用于拆分成多行的表达式。

（6）float 函数

该函数可以吧字符串或者数字转换成“浮点数”，也就是一个带有小数点的数字（也叫实数），如数字 10.0、10.1 等都叫浮点数。

```
>>> float(12)
12.0
>>> float('12')
12.0
>>>
```

第一行是数字 12，第三行是字符串。

可以用 float 把程序中的输入转换成恰当的数字，尤其是在需要把某人的输入与其他值做比较的时候。

```
>>> your_age=input('Enter your age:')
Enter your age:20
>>> age=float(your_age)
>>> if age>13:
        print('You are %s years too old'% (age-13))


You are 7.0 years too old
>>>
```

（7）int 函数

该函数可以用来吧字符串或者数字转换成整数。浮点数转换成整数会丢失小数点后面的数字，带有浮点数的字符串不能转换成整数。

（8）len 函数:

len 函数返回一个对象的长度，对于字符串则返回字符串中的字符个数。

当用在列表、元组和字典时，len 函数返回它们中元素的个数。

在循环中 len 函数最为有用。例如，可以用下面的代码

显示列表中元素的索引位置。

```
>>> list=['a','b','c','d','e']
>>> length=len(list)
>>> for x in range(0,length):
        print('the letter at index %s in %s'%(x,list[x]))


the letter at index 0 in a
the letter at index 1 in b
the letter at index 2 in c
the letter at index 3 in d
the letter at index 4 in e
>>>
```

（9）max 和 min 函数

用 max 函数可以返回列表、元组或字符串中最大的元素。对于字符串要由逗号或空格分隔。

```
>>> strings='s t r i n g S T R I N G'
>>> print(max(strings))
t
>>>
```

（这个例子表明，字母是按照字母表的顺序排列的，并且小写字母排在大写字母之后。）

使用 max 函数不一定非要使用列表、元组或字符串，也可以直接调用 max 函数，把要比较的元素作为参数写在括号中。

```
>>> print(max(10,20,30,45))
45
>>>
```

min 函数和 max 用法相同，但返回最小元素。

（10）range 函数

函数 range 主要用在 for 循环中，用来让一段代码循

环执行指定数字的次数。

函数 range 返回一个叫“迭代器”的特殊对象，它能重复一个动作很多次。可以把迭代器抓换成列表（使用 list）。然后如果你打印对 range 调用的返回值，可以看到它所包含的数字。

range 函数有俩个或三个参数，第三个参数叫“步长”，如果省略时，则步长是 1。步长可有有正负。如：

？？？？

（11）sum 函数

函数 sum 把列表中的元素加在一起并返回这个总和。

2、使用文件

（1）创建测试文件

（2）在 python 中打开文件

Python 的内建函数 open 可以用来在 shell 程序中打开文件，并显示它的内容。

在 windows 系统中，可以用下面的代码打开 text.txt：

在第一行中，我们使用了 open，它会返回一个文件对象，这个对象带有操作文件的函数。这里 open 函数的参数是一个字符串，告诉 python 到哪里去找到这个文件。在第二行中我们使用文件对象提供的 read 函数来读取文件中的内容，并把他保存到变量 text 中。在最后一

```
>>> test_file=open('E:\\test.txt')
>>> text=test_file.read()
>>> print(text)
ssssssssssssssssssssssssssssssssssssjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjjhh
hhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhh
>>> 
```

行，我们把变量的内容打印出来以显示文件内容。

（3）写入到文件

Open 所返回的文件对象不只有 read 函数，我们还可以用它来创建一个新的空文件。在调用函数时要用到字符串‘w’，‘w’这个参数告诉 python 我们想要向文件中写入，而不是读取。

现在我们可以用 write 函数向新文件增加信息了。回车之后出现的数字是字符串的长度。

最后，我们需要用 close 函数告诉 Python，我们对这个文件写入完成了。

```
>>> test_file=open('E:\\file.txt','w')
>>> test_file.write('this is my file.')
16
>>> test_file.close()
>>> 
```

Python 关键字（共 29 个）

1、and 用在一个语句中，连接俩个表达式，俩个表达式同为真时结果为真。

2、as 在引入模块式可以用来给模块重新命名。

3、assert（断言）用来申明一段代码必须为真，用于高级编程。

4、break 用于让某段代码的运行停止，在 for 循环中直接退出循环。与 continue 有区别。

5、class 用于定义一种类型的对象。

类可以有一个__init__函数，它可以执行类的对象创建时所要执行的所用任务。

6、continue 用于在一个循环中直接跳到下一次。与 break 不同，它不跳出循环，只是从下一个元素继续执行。

7、def 用来定义函数

8、del 用来删除，如删除列表中的某个元素，删除后可以用 append 函数在列表末尾添加。

9、elif 是 if 语句的一部分

10、else 是 if 语句的一部分

11、except

12、finally 用于确保有错误时某段代码一定执行(通常是清理工作)。

13、for 在一定范围内的循环，循环次数已知。

14、from 在引入模块时可以用 from 来只引入模块中的某部分。

当有多个部分要引用时可以用*代替，表示引用了模块中的全部，之后再调用模块中的某个函数时就不用再使用模块名，可以通过函数名直接调用。

15、global 涉及变量作用域，当一个变量在赋值前使用 global 关键字，该变量就被定义为全局变量，在任何地方都可见。

16、if 用来做判断，如果条件为真，这执行...。

17、import 用来引入模块

18、in 用来判断某元素是否属于某个列表、元组或字典。

19、is 用来判断俩个东西是否相等，和==相像但有本质的不同

20、lambda

21、not 表示否定，多用在 if 语句中

22、or 连接俩个表达式，当俩个表达式至少有一个为真时结果为真

23、pass 当遇到 for 循环、if 语句等，如果不想添加具体信息代码，可以用 pass 关键字代替。

24、raise

25、return 用来在函数中返回一个值，这个返回值可以用来给另外一个变量赋值，或者把它打印出来。？？？？？

26、try

27、while 循环次数不确定，当表达式为真时会一直循环。

28、with

29、yield