1．python简介

Python 是一种面向对象的解释型计算机程序设计语言，它是纯粹的自由软件， 源代码和解释器CPython遵循 GPL(GNU General Public License)协议。 Python语法简洁清晰，特色之一是强制用空白符(white space)作为语句缩进。 Python具有丰富和强大的库。它常被昵称为胶水语言，能够把用其他语言制作的各种模块（尤其是C/C++）很轻松地联结在一起。常见的一种应用情形是，使用Python快速生成程序的原型（有时甚至是程序的最终界面），然后对其中有特别要求的部分，用更合适的语言改写，比如3D游戏中的图形渲染模块，性能要求特别高，就可以用C/C++重写，而后封装为Python可以调用的扩展类库。需要注意的是在您使用扩展类库时可能需要考虑平台问题，某些可能不提供跨平台的实现。

1.1 python风格

Python在设计上坚持了清晰划一的风格，这使得Python成为一门易读、易维护，并且被大量用户所欢迎的、用途广泛的语言。

设计者开发时总的指导思想是，对于一个特定的问题，只要有一种最好的方法来解决就好了。这在由Tim Peters写的Python格言（称为The Zen of Python）里面表述为：There should be one-- and preferably only one --obvious way to do it. 这正好和Perl语言（另一种功能类似的高级动态语言）的中心思想TMTOWTDI（There's More Than One Way To Do It）完全相反。

Python的作者有意的设计限制性很强的语法，使得不好的编程习惯（例如if语句的下一行不向右缩进）都不能通过编译。其中很重要的一项就是Python的缩进规则。

一个和其他大多数语言（如C）的区别就是，一个模块的界限，完全是由每行的首字符在这一行的位置来决定的（而C语言是用一对花括号{}来明确的定出模块的边界的，与字符的位置毫无关系）。这一点曾经引起过争议。因为自从C这类的语言诞生后，语言的语法含义与字符的排列方式分离开来，曾经被认为是一种程序语言的进步。不过不可否认的是，通过强制程序员们缩进（包括if，for和函数定义等所有需要使用模块的地方），Python确实使得程序更加清晰和美观。

Python在执行时，首先会将.py文件中的源代码编译成Python的byte code（字节码），然后再由Python Virtual Machine（Python虚拟机）来执行这些编译好的byte code。这种机制的基本思想跟Java，.NET是一致的。然而，Python Virtual Machine与Java或.NET的Virtual Machine不同的是，Python的Virtual Machine是一种更高级的Virtual Machine。这里的高级并不是通常意义上的高级，不是说Python的Virtual Machine比Java或.NET的功能更强大，而是说和Java 或.NET相比，Python的Virtual Machine距离真实机器的距离更远。或者可以这么说，Python的Virtual Machine是一种抽象层次更高的Virtual Machine。

基于C的Python编译出的字节码文件，通常是.pyc格式。除此之外，Python还可以以交互模式运行，比如主流操作系统Unix/Linux、Mac、Windows都可以直接在命令模式下直接运行Python交互环境。直接下达操作指令即可实现交互操作。

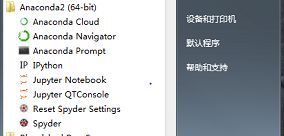
2．python安装

Anaconda是Python的一个开源发行版本，主要面向科学计算。我们可以简单理解为，Anaconda是一个预装了很多我们用的到或用不到的第三方库的Python。而且相比于大家熟悉的pip install命令，Anaconda中增加了conda install命令。当你熟悉了Anaconda以后会发现，conda install会比pip install更方便一些。比如大家经常烦恼的lxml包的问题，在Windows下pip是无法顺利安装的，而conda命令则可以，后面会详细展示。

2.1 安装anaconda

直接双击安装包安装，可以自己选定安装位置（这个你自己决定）。

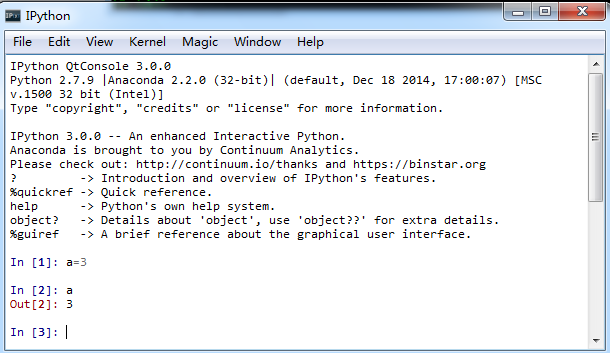
安装完成后，我们可以看到：



* 1. IPython(shell)



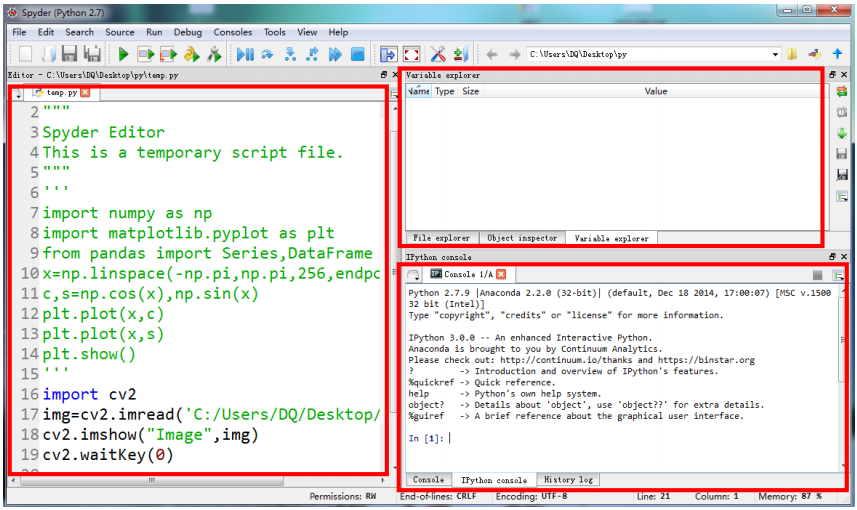
* 1. QTConsole



* 1. Notebook

Spyder的最大优点就是模仿MATLAB的“工作空间”。

直接点击打开：



2.安装包其实安装完anaconda后，基本上就把我们常用的部分包安装好了，但是毕竟只是安装了一小部分（我们可以去官网查看安装了哪些包），还有一部分需要我们自己来安装。

查看已经安装的包我们可以在命令提示符中输入pip list或者用anaconda自带的包管理器conda（conda list）。

2.5 pip 或conda安装

我们可以在命令提示符中输入pip install 包名，或者conda install 包名。conda和pip的用法基本上一致（更新？卸载？还不确定）。但是我们使用conda安装的时候不仅会安装当前你要安装的包还会提示更新其他已经安装过的包，所以我基本上都是直接用pip安装。

2.6 文件安装

pip或conda不能安装的话，我们就下载文件安装，比如exe文件（双击安装）或者whl文件（pip安装）等等。

2.7 python+pycharm

作为基本的python,可以从官网下载安装包.注意目前python2已经不推荐在新工程里使用了.两者差别比较大.

首选安装python,记得其中选择把路径放到path中.

安装pycharm集成开发环境.后续建立工程时就可以选择已经安装的python3解释器了.

3.Python基本语法

Python的设计目标之一是让代码具备高度的可阅读性。它设计时尽量使用其它语言经常使用的标点符号和英文单字，让代码看起来整洁美观。它不像其他的静态语言如C、Pascal那样需要重复书写声明语句，也不像它们的语法那样经常有特殊情况和意外。

3.1缩进

Python语言利用缩进表示语句块的开始和退出（Off-side规则），而非使用花括号或者某种关键字。增加缩进表示语句块的开始，而减少缩进则表示语句块的退出。缩进成为了语法的一部分。例如if语句：

|  |  |
| --- | --- |
| 1  2  3  4 | if age<21:      print("你不能买酒。")      print("不过你能买口香糖。")  print("这句话处於if语句块的外面。") |

根据PEP的规定，必须使用4个空格来表示每级缩进（不清楚4个空格的规定如何，在实际编写中可以自定义空格数，但是要满足每级缩进间空格数相等）。使用Tab字符和其它数目的空格虽然都可以编译通过，但不符合编码规范。支持Tab字符和其它数目的空格仅仅是为兼容很旧的的Python程序和某些有问题的编辑程序。

3.2控制语句

if语句，当条件成立时运行语句块。经常与else, elif(相当于else if) 配合使用。(tutorial.pdf page 18)

example :

a=1, b=2

if a>b :

print a

else:

print b

for语句，遍历列表、字符串、字典、集合等[迭代器](http://baike.baidu.com/item/%E8%BF%AD%E4%BB%A3%E5%99%A8" \t "_blank)，依次处理迭代器中的每个元素。(tutorial.pdf page 20)

example:

for i in range(10):

print i

while语句，当条件为真时，循环运行语句块。(tutorial.pdf page 16)

example: 输出0到9

x =0

while x <10:

print (x)

x+=1

try语句。与except,finally配合使用处理在程序运行中出现的异常情况。(tutorial.pdf page 62)

example:

try:

print(a)

except Exception as e:

print(e)

print error

class语句。用于定义[类型](http://baike.baidu.com/item/%E7%B1%BB%E5%9E%8B/6737759" \t "_blank)。(tutorial.pdf page 69)

example:

class Dog(object):

def \_\_init\_\_(self):

self.type = ‘dog’

def call():

print(‘woff’)

def语句。用于定义函数和类型的方法。(tutorial.pdf page 23)

example:

def show():

print(‘这是你要展示的东西’)

pass语句。表示此行为空，不运行任何操作。(tutorial.pdf page 22)

assert语句。用于程序调试阶段时测试运行条件是否满足。(tutorial.pdf page 46)

with语句。Python2.6以后定义的语法，在一个场景中运行语句块。比如，运行语句块前加密，然后在语句块运行退出后解密。(tutorial.pdf page 67)

yield语句。在迭代器函数内使用，用于返回一个元素。自从Python 2.5版本以后。这个语句变成一个运算符。(tutorial.pdf page 81)

raise语句。制造一个错误。(tutorial.pdf page 63)

import语句。导入一个模块或包。(tutorial.pdf page 43)

example : import math

print(math.sqrt(3))

from import语句。从包导入模块或从模块导入某个对象。(tutorial.pdf page 44)

>>> from math import sqrt

>>> sqrt(256)

16.0

import as语句。将导入的对象赋值给一个变量。(tutorial.pdf page 45)

in语句。判断一个对象是否在一个字符串/列表/元组里。(tutorial.pdf page 19)

1 in [1,2,3,4,5]

3.3 表达式

Python的表达式写法与C/C++类似。只是在某些写法有所差别。

主要的算术运算符与C/C++类似。+, -, \*, /, //, \*\*, ~, %分别表示加法或者取正、减法或者取负、乘法、除法、整除、乘方、取补、取模。>>, <<表示右移和左移。&, |, ^表示二进制的AND, OR, XOR运算。>, <, ==, !=, <=, >=用于比较两个表达式的值，分别表示大于、小于、等于、不等于、小于等于、大于等于。在这些运算符里面，~, |, ^, &, <<, >>必须应用于整数。

>>> 42

42

>>> -1 - -1

0

>>> 1/2 + 1/4 + 1/8 + 1/16 + 1/32 + 1/64 + 1/128

0.9921875

>>> max(7.5, 9.5)

9.5

>>> from operator import add, sub, mul

>>> add(14, 28) 42

>>> sub(100, mul(7, add(8, 4)))

16

Python使用and, or, not表示逻辑运算。

is, is not用于比较两个变量是否是同一个对象。in, not in用于判断一个对象是否属于另外一个对象。

Python支持“列表推导式”（list comprehension），

比如计算0-9的平方和:

>>> sum(x \* x for x in range(10))

285

3.4对象的方法

对象的方法是指绑定到对象的函数。调用对象方法的语法是instance.method(arguments)。它等价于调用Class.method(instance, arguments)。当定义对象方法时，必须显式地定义第一个参数，一般该参数名都使用self，用于访问对象的内部数据。这里的self相当于C++, Java里面的this变量，但是我们还可以使用任何其它合法的参数名，比如this 和 mine 等，self与C++,Java里面的this不完全一样，它可以被看作是一个习惯性的用法，我们传入任何其它的合法名称都行，比如：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16 | class Fish():      def eat(self,food):          if food is not None:          self.hungry=False    self.a = 1  class User:      def\_\_init\_\_(myself,name):          myself.name=name    #构造Fish的实例：  f=Fish()  #以下两种调用形式是等价的：  Fish.eat(f,"earthworm")  f.eat("earthworm")  u=User('username')  print(u.name) |
|  |  |

Python认识一些以“\_\_”开始并以“\_\_”结束的特殊方法名，它们用于实现运算符重载和实现多种特殊功能。

\_\_init\_\_()方法是一种特殊的方法，被称为类的构造函数或初始化方法，当创建了这个类的实例时就会调用该方法

self 代表类的实例。类的方法与普通的函数只有一个特别的区别——它们必须有一个额外的第一个参数名称, 按照惯例它的名称是 self。

3.5 类型

Python采用动态类型系统。在编译的时候，Python不会检查对象是否拥有被调用的方法或者属性，而是直至运行时，才做出检查。所以操作对象时可能会抛出异常。不过，虽然Python采用动态类型系统，它同时也是强类型的。Python禁止没有明确定义的操作，比如数字加字符串。

与其它面向对象语言一样，Python允许程序员定义类型。构造一个对象只需要像函数一样调用类型即可，比如，对于前面定义的Fish类型，使用Fish()。类型本身也是特殊类型type的对象(type类型本身也是type对象)，这种特殊的设计允许对类型进行反射编程。

下面这个列表简要地描述了Python内置数据类型(适用于Python 3.x)：



3.2 python数据类型

List:可以用来存储一串信息的python结构，类似于C中的数组。

<1>列表的格式：

变量A的类型为列表

namesList = ['xiaoWang','xiaoZhang','xiaoHua']

比C语言的数组强大的地方在于列表中的元素可以是不同类型的

testList = [1, 'a']

也可以是多维列表：

>>> pairs = [[10, 20], [30, 40]]

>>> pairs[1]

[30, 40]

>>> pairs[1][0]

30

## <2>打印列表

demo:

namesList = ['xiaoWang','xiaoZhang','xiaoHua']

print(namesList[0])

print(namesList[1])

print(namesList[2])

结果：

xiaoWang

xiaoZhang

xiaoHua

<3>列表的循环

为了更有效率的输出列表的每个数据，可以使用循环来完成

for <name> in <list>:

>>> odds = [1, 3, 5, 7, 9]

>>> for item in odds:

>>> print item

>>> odds = [1, 3, 5, 7, 9]

>>> [x+1 for x in odds]

[2, 4, 6, 8, 10]

<4>列表的常见操作

添加：

通过append可以向列表添加元素

odds.append(9)

删除：

列表元素的常用删除方法有：

* del：根据下标进行删除
* pop：删除最后一个元素
* remove：根据元素的值进行删除

demo:(del)

movieName = ['加勒比海盗','骇客帝国','第一滴血','指环王','霍比特人','速度与激情']

print('------删除之前------')

for tempName in movieName:

print(tempName)

del movieName[2]

print('------删除之后------')

for tempName in movieName:

print(tempName)

结果:

------删除之前------

加勒比海盗

骇客帝国

第一滴血

指环王

霍比特人

速度与激情

------删除之后------

加勒比海盗

骇客帝国

指环王

霍比特人

速度与激情

demo:(pop)

movieName = ['加勒比海盗','骇客帝国','第一滴血','指环王','霍比特人','速度与激情']

print('------删除之前------')

for tempName in movieName:

print(tempName)

movieName.pop()

print('------删除之后------')

for tempName in movieName:

print(tempName)

结果:

------删除之前------

加勒比海盗

骇客帝国

第一滴血

指环王

霍比特人

速度与激情

------删除之后------

加勒比海盗

骇客帝国

第一滴血

指环王

霍比特人

demo:(remove)

movieName = ['加勒比海盗','骇客帝国','第一滴血','指环王','霍比特人','速度与激情']

print('------删除之前------')

for tempName in movieName:

print(tempName)

movieName.remove('指环王')

print('------删除之后------')

for tempName in movieName:

print(tempName)

结果:

------删除之前------

加勒比海盗

骇客帝国

第一滴血

指环王

霍比特人

速度与激情

------删除之后------

加勒比海盗

骇客帝国

第一滴血

霍比特人

速度与激情

修改：

修改元素的时候，要通过下标来确定要修改的是哪个元素，然后才能进行修改

查找：

python中查找的常用方法为：

* in（存在）,如果存在那么结果为true，否则为false
* not in（不存在），如果不存在那么结果为true，否则false