1.Pytharm设置

1.由于顶部菜单栏的字体太小，并且长时间观看对眼睛不好，所以我们来调大。

2.先点击“File”菜单，接着点击“settings”菜单，这是用于设置Pycharm的选项的菜单。

3.在Appearance右边找到“Override default……”选项，先勾选上，然后就可以调整菜单的字体大小了。

4.如果要修改代码的字体大小，则先找到“editor”选项，然后点击“font”，这个是设置代码字体大小的地方。5.在Size这里输入数字即可，可根据个人情况进行适当调整。然后点击下方的OK就完成了。

2.pytharm的快捷按键



3. PEP 8编写规范

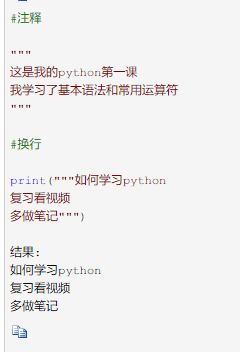
1. 用小写字母拼写，多个单词用下划线连接。
2. 受保护的实例属性用单个下划线开头（后面会讲到）。
3. 私有的实例属性用两个下划线开头（后面会讲到）。

4.range

1. range(101)可以产生一个0到100的整数序列。
2. range(1, 100)可以产生一个1到99的整数序列。
3. range(1, 100, 2)可以产生一个1到99的奇数序列，其中2是步长，即数值序列的增量

5.单双引号的区别

1. 单引号中可以包含双引号，双引号中可以包含单引号
2. 单引号中包含单引号，双引号中包含双引号，只能通过"\"来转义
3. 三引号的注释和换行功能



1. 单引号双引号引号内的回车换行，不是真正的换行
2. 单引号双引号利用"\n"做到真正的换行

6.如何判断一个float实际是int还是float类型

math.ceil(7.5644) 会返回8

7.

8生成式例子

* 1. date = [randint(10,20) for \_ in range(10)]

1. 1
2. 1

9.format用法

format()功能更强大，该函数把字符串当成一个模板，通过传入的参数进行格式化，并且使用大括号‘{}’作为特殊字符代替‘%’

位置匹配

（1）不带编号，即“{}”

（2）带数字编号，可调换顺序，即“{1}”、“{2}”

（3）带关键字，即“{a}”、“{tom}”

复制代码

1 >>> print(’{} {}’.format(‘hello’,‘world’)) # 不带字段

2 hello world

3 >>> print(’{0} {1}’.format(‘hello’,‘world’)) # 带数字编号

4 hello world

5 >>> print(’{0} {1} {0}’.format(‘hello’,‘world’)) # 打乱顺序

6 hello world hello

7 >>> print(’{1} {1} {0}’.format(‘hello’,‘world’))

8 world world hello

9 >>> print(’{a} {tom} {a}’.format(tom=‘hello’,a=‘world’)) # 带关键字

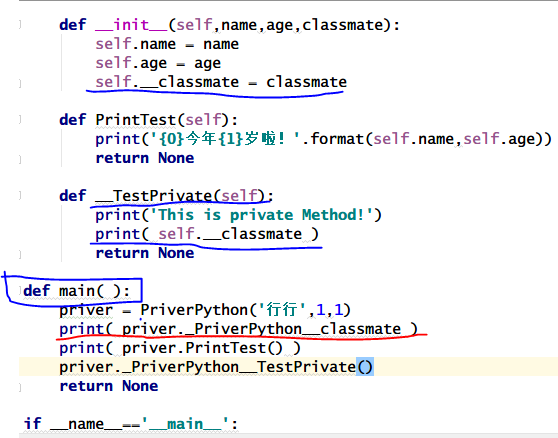
10 world hello world

10.“\_”“\_\_”and“\_\_xxx\_\_()”

1. \_xxx "单下划线 " 开始的成员变量叫做保护变量，意思是只有类实例和子类实例能访问到这些变量，需通过类提供的接口进行访问；不能用'from module import \*'导入
2. \_\_xxx 类中的私有变量/方法名 （Python的函数也是对象，所以成员方法称为成员变量也行得通。）," 双下划线 " 开始的是私有成员，意思是只有类对象自己能访问，连子类对象也不能访问到这个数据。

防止被覆盖，都只能被类中的函数调用

1. \_\_xxx\_\_ 系统定义名字，前后均有一个“双下划线” 代表python里特殊方法专用的标识，如 \_\_init\_\_（）代表类的构造函数



11.

class Student:

pass

class MyStudent(Student):

pass

stu = Student()

print(type(stu)) <class '\_\_main\_\_.Student'>

print(type(Student)) <class 'type'>

print(int.\_\_bases\_\_) (<class 'object'>,)

print(str.\_\_bases\_\_) (<class 'object'>,)

print(Student.\_\_bases\_\_) (<class 'object'>,)

print(MyStudent.\_\_bases\_\_) (<class '\_\_main\_\_.Student'>,)

print(type.\_\_bases\_\_) (<class 'object'>,)

print(object.\_\_bases\_\_) ()

print(type(object)) <class 'type'>

12.@property

正常的给类的属性赋值是class.menthod(参数),但是如果有@属性.setter的修饰符后，可以

class.属性 == value

13.静态类和方法

@staticmethod

def is\_valid(a, b, c):

return a + b > c and b + c > a and a + c > b

使用的时候：

if class.is\_valid(a, b, c):即可

14. \_\_slots\_\_

\_\_slots\_\_ = ('\_name', '\_age', '\_gender')