Python基础DAY06

一、驻留机制

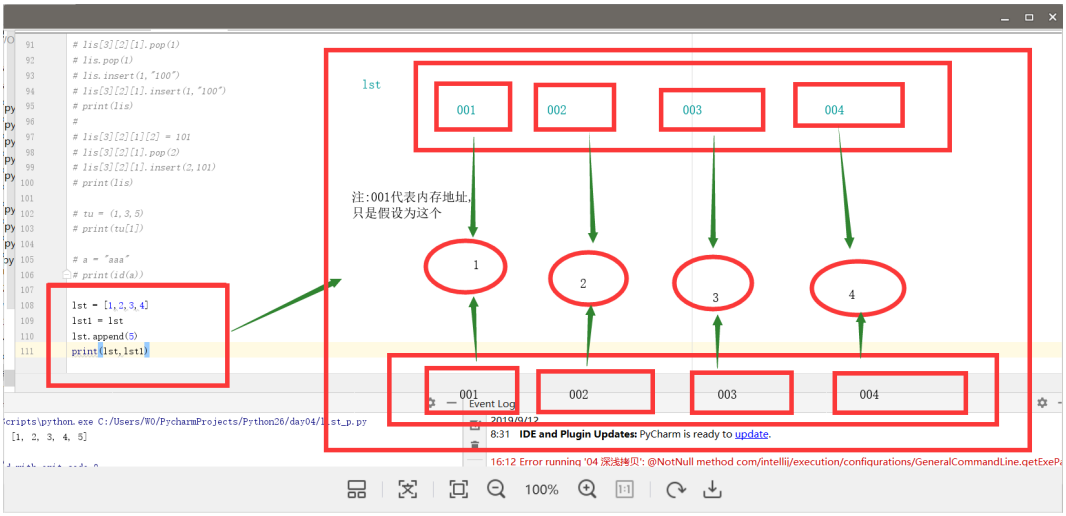
- 驻留机制是为了节省内存,提高效率,将一部分内容相同的变量都指向同一地址,由小数据池和代码块组成
- 小数据池:
  - 要求:
    - 整数:-5 - 256
    - 字符串:只能由字母和数字组成,不能有中文或特殊符号
    - 可以做乘法运算,但在python3.6中乘法运算后的长度不能超过20,python3.7中不能超过4096
- 代码块:一个py文件,一个模块,一个类,一个函数,终端下的每一行代码
  - 要求:
    - 整数:-5 - 正无穷
    - 字符串:任意
    - 乘法运算时不能有中文和特殊符号,且运算后的长度不能超过20
- == 和 is的区别: == 是判断 == 两边的值是否相同,is是判断两者是不是同一个东西,is是通过内存地址来判断的
- 驻留机制本质上就是一个规则

二、深浅copy

- 赋值:不同变量指向同一地址,一变都变
  - ```
    lst = [1,2,3,4]
    lst1 = lst
    lst.append(5)
    print(lst,lst1)

    输出结果:
    [1, 2, 3, 4, 5] [1, 2, 3, 4, 5]

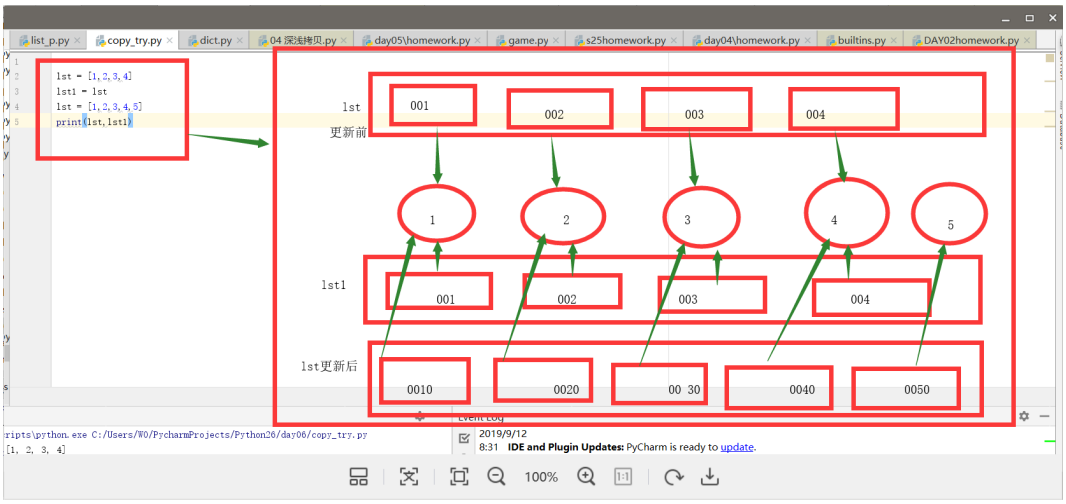
    #  由于lst被赋值给lst1,所以lst变化lst1也会发生同样的变化
    ```
  -
    ```



-
```
lst = [1,2,3,4]
lst1 = lst
lst = [1,2,3,4,5]
print(lst,lst1)

输出结果:
[1, 2, 3, 4, 5] [1, 2, 3, 4]

# 这个代码与上面的不同在于lst被重新赋值,即lst的地址被改变,但是lst1的地址没有变,所以lst1不发生改变
```

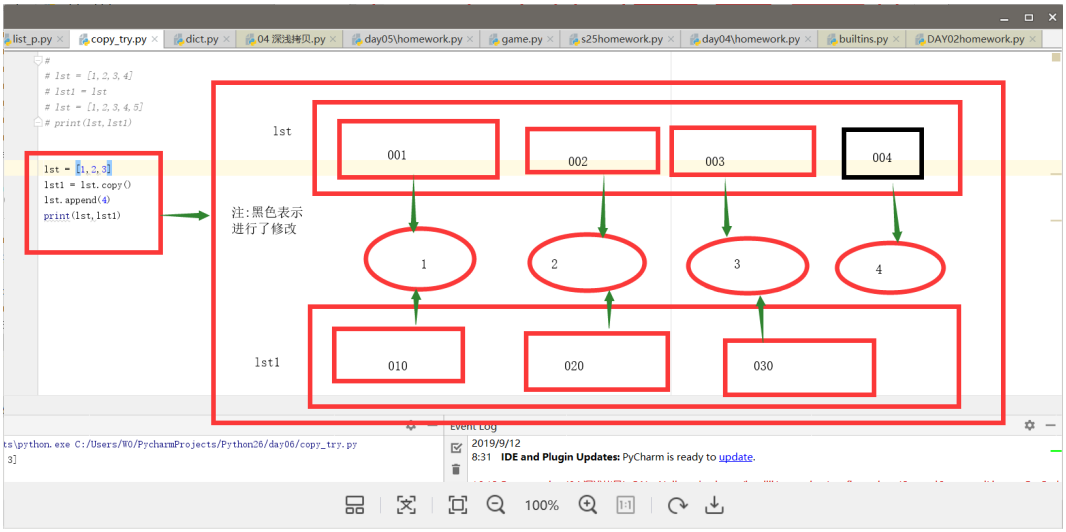
- 

-

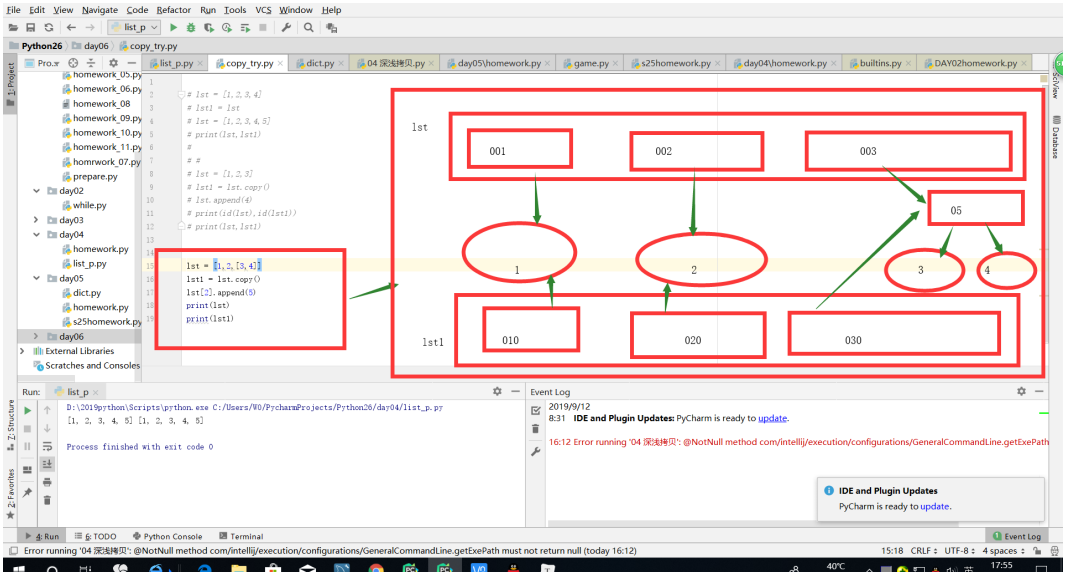
```
lst = [1,2,3]
lst1 = lst.copy()
lst.append(4)
print(id(lst),id(lst1))
print(lst,lst1)

输出结果:
27411920 27413080
[1, 2, 3, 4] [1, 2, 3]
# 由于copy是浅copy只会拷贝第一层的地址,所以实质上是开辟了一个新地址,原列表中添加或删除元素是不会影响新列表的
```

-

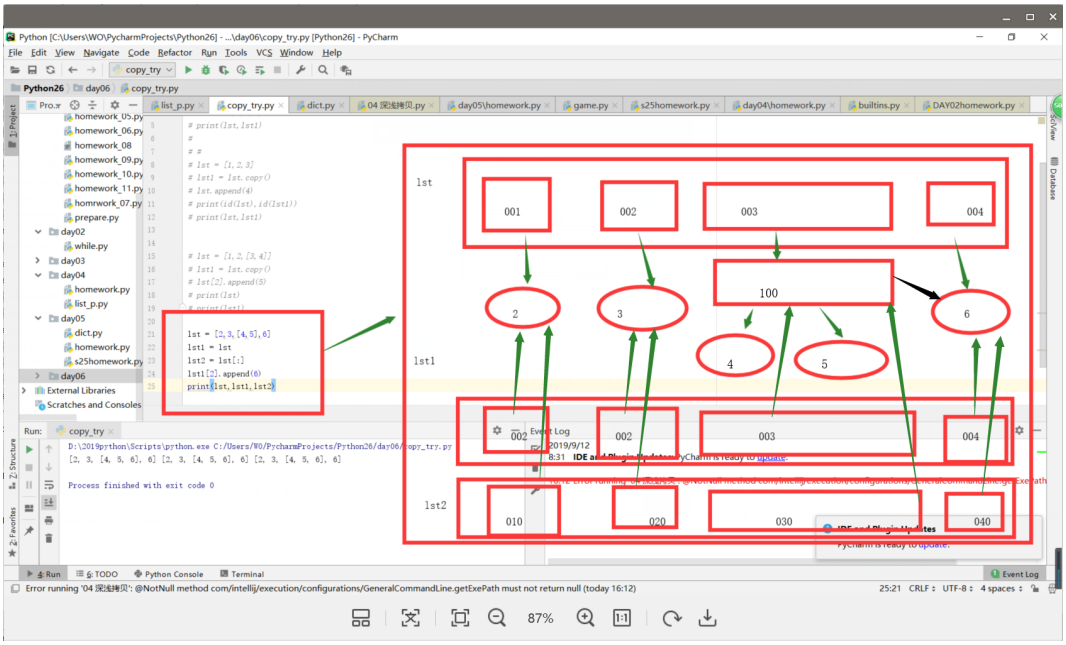


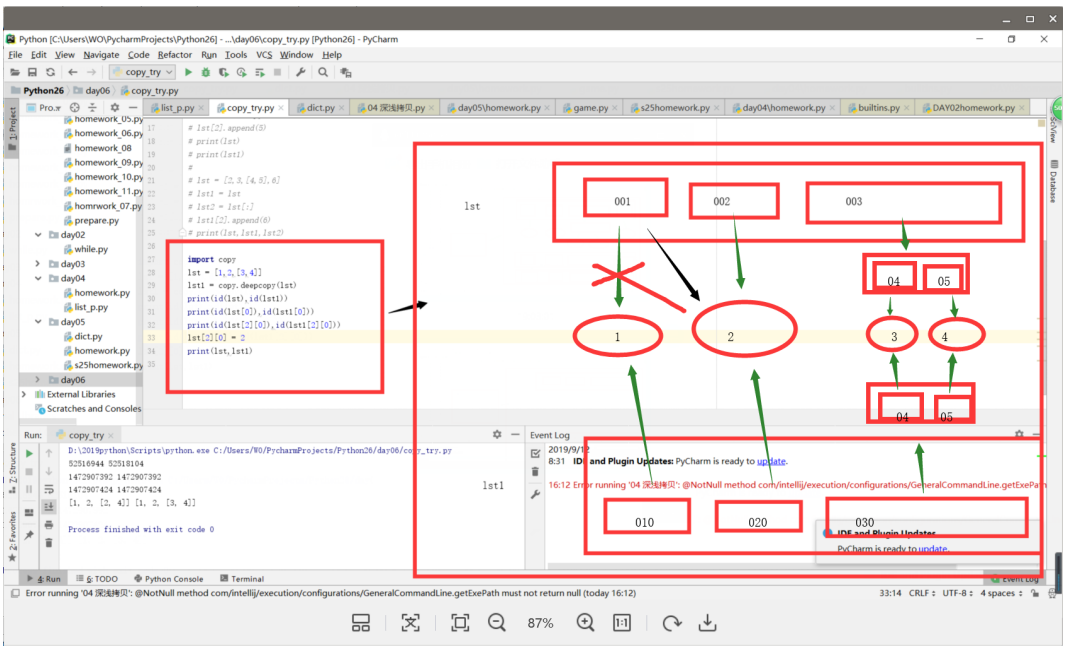

- ```
  lst = [1,2,[3,4]]
  lst1 = lst.copy()
  lst[2].append(5)
  print(lst)
  print(lst1)
  输出结果:
  [1, 2, 3, 4, 5] [1, 2, 3, 4, 5]
  #由于浅copy会拷贝第一层的地址,所以当原列表对当前地址内的内容进行改变时,新列表也会随之改变,而上个代码没有改变的根本原因是原列表是改变了元素的地址
  ```

- 


- ```
  lst = [2,3,[4,5],6]
  lst1 = lst
  lst2 = lst[:]
  lst1[2].append(6)
  print(lst,lst1,lst2)
  输出结果:
  [2, 3, [4, 5, 6], 6] [2, 3, [4, 5, 6], 6] [2, 3, [4, 5, 6], 6]
  #lst和lst1不用多说,都指向一个地址,lst2是因为copy了lst第一层的地址,所以其中列表发生改变时(列表地址不会变)lst2也跟着发生了改变
  ```

-



- 深copy:
  - ```
    import copy
    lst = [1,2,[3,4]]
    lst1=copy.deepcopy(lst)
    print(id(lst),id(lst1))
    print(id(lst[0]),id(lst1[0]))
    print(id(lst[2][0]),id(lst1[2][0]))
    lst[2][0] = 2
    print(lst,lst1)
    输出结果:
    52516944 52518104
    1472907392 1472907392
    1472907424 1472907424
    [1, 2, [2, 4]] [1, 2, [3, 4]]
    #深copy解决了浅copy中的连坐问题,即新列表的改变会导致旧列表也发生改变.现在copy出来的列表完全独立与被copy列表,两者的改变互不影响
    ```
  - 

- 总结:
  - 多个变量指向同一内存地址,如果这个内存地址的数据是:
    ```

        - 不可变的,会开辟新空间
        - 可变的,会在原地修改,地址不变
    - 赋值:不同变量指向同一地址,一变都变
    - 浅copy:列表.copy(),或者列表[:]:会另外开辟空间,只会copy第一层的地址,即当旧列表中元素是不可变类型数据时,他的变化不会影响新列表,当旧列表的元素是可变数据类型时,会导致新列表也发生改变
    - 深copy:需要导入copy模块,copy.deepcopy(列表):另外开辟空间,并且两者之间的改变不会互相影响

三、集合

- 集合概述
    - 关键字:set,声明方式: s = {}
    - 本质上集合就是没有值的字典
    - 作用:去重
    - 特点:去重,可迭代,无序,可变的数据结构
- 集合的操作
    - 声明一集合 s = {1,2,3}
    - 集合的增加:
        - s.update(可迭代对象):迭代添加
        - s.add():添加的元素本质上是个键,所以要符合键的要求
    - 集合的删除:
        - s.pop():随机删除,并返回被删除元素
        - s.remove(元素):删除指定元素
        - s.clear():清空集合
    - 集合的修改:
        - 先删后加
    - 集合的查:
        - 可以用for来遍历,无法取指定的值
- 集合的关系:
    - 声明两集合 s1 = {1,2,3},s2 = {2,3,4}
    - s1 | s2:并集
    - s1 & s2:交集
    - s1 - s2:差集
    - s1 ^ s2:补集(反差集)
    - 超集(父集):
        - 判断s1是不是s2的父集print(s1>s2)