1.举例说明 Python 的可变类型与不可变类型。(36)

不可变类型：数值、字符串、元组；可变类型：列表、字典

2.python 中路径相关操作函数有哪些？相应的功能？(38)

路径的常用操作在 os 模块中，可以实现跨平台。

Getcwd() 返回当前工作目录 listdir(path='')列举指定目录中的文件名

Remove(path)删除文件 basename()去掉目录路径，单独返回文件名

Dirname(path)去掉文件名，返回路径 splitext(path)分离文件名与扩展名

Exist（path）

3.列出 6 种 python 内置函数，说明功能以及应用场景。(45 题目)

https://docs.python.org/2/library/functions.html#map

max()、max() 求最大、最小值

lis=[2,3,4,56,7,4]

max(lis)

b.len() 求长度

lis=[2,3,4,56,7,4]

len(lis)

c.eval() 将字符串 str 当成有效的表达式来求值并返回计算结果。可以求值运算，也可以将其与 raw_input 结合用于容器类对象的输入。

```
a=1
g={'a':20}
eval("a+1",g)
```

```
c=raw_input('输入列表：')
c=eval(c)
```

d.enumerate() 将可迭代对象生成一个迭代对象

```
list1 = ["这", "是", "一个", "测试"]
for i in range (len(list1)):
    print i ,list1[i]
```

```
list1 = ["这", "是", "一个", "测试"]
for index, item in enumerate(list1):
    print index, item
```

e.zip()将任意序列合并转换成元组序列

```
x = [1, 2, 3]
```

```
y = [4, 5, 6]
z = [7, 8, 9]
xyz = zip(x, y, z)
```

f.map() map 将传入的函数依次作用到序列的每个元素，并把结果作为新的 list 返回

```
print map(lambda x: x % 2, range(7))
```

4.简述下面两个类的区别？（43 题）

```
class A:
    Pass
class B(object):
     Pass
```

上面两个分别为旧式类和新式类。python 的新式类是 2.2 版本引进来的，我们可以将之前的类叫做经典类或者旧类。目的是为了统一类(class)和类型(type)。新式类是广度优先的查找算法。旧式类的查找方法是深度优先的。

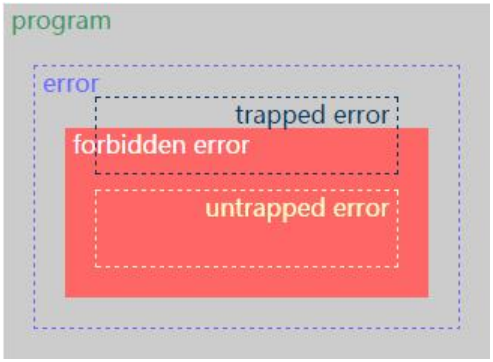
```
95 class A:
96     pass
97
98 class B(object):
99     pass
100
101 a=A()
102 b=B()
103
104 print type(A)
105 print a.__class__
106 print type(a)
107 print '\n'
108 print type(B)
109 print b.__class__
110 print type(b)
```

Problems Console Search

```
<terminated> H:\pybatis\PyBatis\com\test\snippet.py
<type 'classobj'>
__main__.A
<type 'instance'>


<type 'type'>
<class '__main__.B'>
<class '__main__.B'>
```

5.弱类型、强类型、动态类型、静态类型语言的区别是什么？



红色区域外 :well behaved (type soundness)

红色区域内：ill behaved

如果所有程序都是灰的，strongly typed

否则如果存在红色的程序，weakly typed

编译时排除红色程序，statically typed

运行时排除红色程序，dynamically typed

所有程序都在黄框以外，type safe

Program Errors

trapped errors，导致程序终止执行，如除 0，中数组越界访问。

untrapped errors， 出错后继续执行，但可能出现任意行为。如 C 里的缓冲区溢出、Jump 到错误地址。

Forbidden Behaviours

语言设计时，可以定义一组 forbidden behaviors. 它必须包括所有 untrapped errors, 但可能包含 trapped errors.

Well behaved、ill behavedwell behaved: 如果程序执行不可能出现 forbidden behaviors, 则为 well behaved。ill behaved: 否则为 ill behaved...

强类型 strongly typed: 如果一种语言的所有程序都是 well behaved——即不可能出现 forbidden behaviors，则该语言为 strongly typed。

弱类型 weakly typed: 否则为 weakly typed。比如 C 语言的缓冲区溢出，属于 trapped errors，即属于 forbidden behaviors..

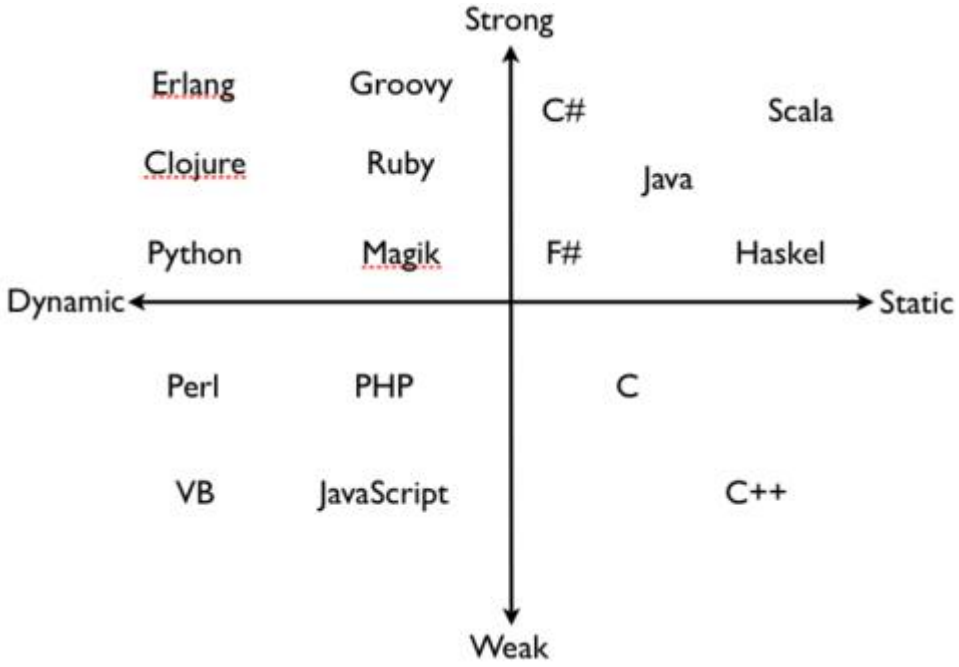
弱类型语言 0, 1,，类型检查更不严格，如偏向于容忍隐式类型转换。譬如说 C 语言的 int 可以变成 double。这样的结果是：容易产生 forbidden behaviours，所以是弱类型的。Python 则不接受这样的隐式转换，例如数值+字符串;而弱类型语言则可能接受这样的转换。

静态类型 statically: 如果在编译时拒绝 ill behaved 程序，则是 statically typed;

动态类型 dynamicl1y: 如果在运行时拒绝 ill behaviors, 则是 dynamicl1y typed。

静态类型指的是编译器在 compile time 执行类型检查，动态类型指的是编译器（虚拟机）在 runtime 执行类型检查。简单地说，在声明了一个变量之后，不能改变它的类型的语言，是静态语言;能够随时改变它的类型的语言，是动态语言。因为动态语言的特性，一般需要运行时虚拟机支持。

https://www.zhihu.com/question/19918532



4.python 什么情况下会出现循环引用，以及如何避免它？（49 题）

当 python 的对象间相互引用、自引用时，模块间互相导入，会出现循环引用。

例如 a=[1,2,3,4] b=a b.append(a) 此时 b 只是 a 的别名，出现循环引用

解决办法：对一单一容器可以利用浅拷贝的方式;复杂容器用深拷贝的方法

循环 import 模块 A 和模块 B 之间互相 import,出现循环引用。解决办法：

http://blog.csdn.net/daijiguo/article/details/52423415

a.延迟导入(lazy import)

即把 import 语句写在方法或函数里面，将它的作用域限制在局部。这种方法的缺点就是会有性能问题。

b.将 from xxx import yyy 改成 import xxx;xxx.yyy 来访问的形式

c.组织代码 0，1，

出现循环 import 的问题往往意味着代码的布局有问题。可以合并或者分离竞争资源。合并的话就是都写到一个文件里面去。分离的话就是把需要 import 的资源提取到一个第三方文件去。总之就是将循环变成单向。

4.简述 python 内存管理机制？(50 题)

Python 的内存管理机制可以从三个方面来讲：**（1）内存分配（2）引用计数（3）垃圾回收。**

http://www.cnblogs.com/lifeinsmile/p/5278297.html

http://www.cnblogs.com/vamei/p/3232088.html

http://www.cnblogs.com/CBDoctor/p/3781078.html

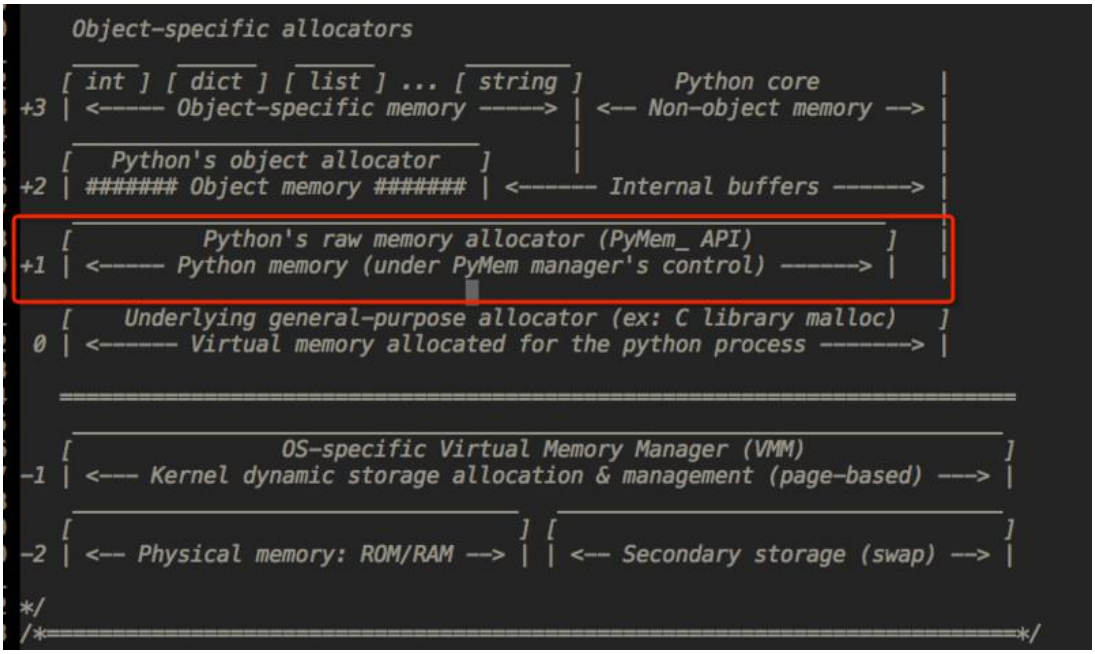
内存分配

Python 解释器承担内存的分配工作，在执行时给对象分配内存。

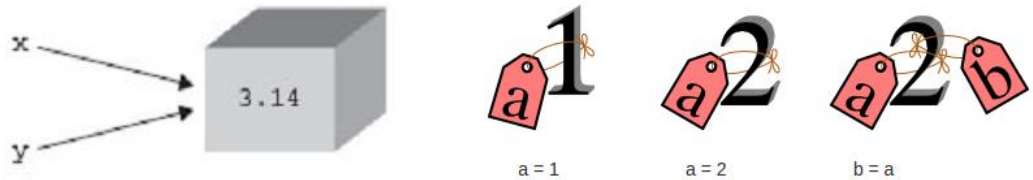
python 的内存机制以金字塔行，-1，-2 层主要有 OS 进行操作;第 0 层是 C 中的 malloc\free 等内存分配和释放函数进行操作；第 1 层和第 2 层是内存池，有 Python 的接口函数 PyMem_Malloc 函数实现;第 3 层是最上层，也就是我们对 Python 对象的直接操作。

如果请求分配的内存在 1~256 字节之间就使用自己的内存管理系统,否则直接使用 malloc.这里还是会调用 malloc 分配内存,但每次会分配一块大小为 256k 的大块内存.经由内存池登记的内存到最后还是会回收到内存池,并不会调用 C 的 free 释放掉.以便下次使用.

Python 的一个容器对象(container)，比如表、词典等，可以包含多个对象。实际上，容器对象中包含的并不是元素对象本身，是指向各个元素对象的引用。容器对象的引用可能构成很复杂的拓扑结构。我们可以用 objgraph 包来绘制其引用关系。



引用计数



x=3.14;y=x。首先创建了一个对象 3.14，然后将这个浮点数对象的引用赋值给 x，因为 x 是第一个引用。当执行 y=x,又将同样的引用赋值给 y,y 指向 3.14，是引用别名。其等效于有图的贴标签的过程。

在 Python 中，整数和短小的字符，Python 都会缓存这些对象，以便重复使用。当我们创建多个等于 1 的引用时，实际上是让所有这些引用指向同一个对象。

在 python 中，为了对象的回收，每个对象都有一个引用计数。每当增加或者减少对象的引用时，引用计数都会发生变化。sys 包中的 getrefcount()可以查看引用计数。

引用计数增加：

1.对象被创建：x=4

2.另外的别人被创建：y=x

3.被作为参数传递给函数：foo(x)

4.作为容器对象的一个元素：a=[1,x,'33']

引用计数减少：

1.一个本地引用离开了它的作用域。比如上面的 foo(x)函数结束时，x 指向的对象引用减 1。

2.对象的别名被显式的销毁：del x ；或者 del y

3.对象的一个别名被赋值给其他对象：x=789

4.对象从一个窗口对象中移除：myList.remove(x)

5.窗口对象本身被销毁：del myList，或者窗口对象本身离开了作用域。

Python 的一个容器对象(container)，比如表、词典等，可以包含多个对象。实际上，容器对象中包含的并不是元素对象本身，是指向各个元素对象的引用。容器对象的引用可能构成很复杂的拓扑结构。我们可以用 objgraph 包来绘制其引用关系。http://blog.csdn.net/bluehawksky/article/details/50295089

垃圾回收

垃圾收集器是一个引用计算器和一个循环垃圾收集器，与此同时 python 采用了分代回收机制。

引用计数器

当一个对象的引用计数为 0 时，解释器会释放掉这个对象与仅有这个对像可以访问的其他对象。

循环垃圾收集器

```
a,b=[],[]
a.append(b)
```

b.append(a)

这长生了相互引用，当两个对象无其他引用时，实际需要释放掉，但是引用实际值却不为 0。循环垃圾收集器复制引用计算器的副本，采用‘标记-清除’的方式来解决循环引用。

分代回收

Python 将所有的对象分为 0，1，2 三代。所有的新建对象都是 0 代对象。当某一代对象经历过垃圾回收，依然存活，那么它就被归入下一代对象。垃圾回收启动时，一定会扫描所有的 0 代对象。如果 0 代经过一定次数垃圾回收，那么就启动对 0 代和 1 代的扫描清理。当 1 代也经历了一定次数的垃圾回收后，那么会启动对 0，1，2，即对所有对象进行扫描。

这两个次数即上面 get_threshold()返回的(700, 10, 10)返回的两个 10。也就是说，每 10 次 0 代垃圾回收，会配合 1 次 1 代的垃圾回收；而每 10 次 1 代的垃圾回收，才会有 1 次的 0, 1,2 代垃圾回收。

回收启动机制

垃圾回收时，Python 不能进行其它的任务。频繁的垃圾回收将大大降低 Python 的工作效率。如果内存中的对象不多，就没有必要总启动垃圾回收。所以，Python 只会在特定条件下，自动启动垃圾回收。当 Python 运行时，会记录其中分配对象(object allocation)和取消分配对象(object deallocation)的次数。当两者的差值高于某个阈值时，垃圾回收才会启动。可以通过 gc 模块的 get_threshold()方法，查看该阈值。手动启动垃圾回收的方法为使用 gc.collect()。