1. 如何为函数定义keyword-only参数（写出个例子即可）？

**keyword-only参数**

* 在一个星号参数后、或者一个可变位置参数后的形参
* 函数调用时必须使用关键字参数传参
* 调用时如果没有默认值，则必须传递实参，否则将抛出TypeError缺少keyword-only参数异常

def func(\*args, x=1, y, \*\*kwargs):

# def func(\*, x=1, y, \*\*kwargs):

print(x)

print(y)

print(args)

print(kwargs)

func(3, 5, x=3, y=5, b='KeithTt')

1. 什么是LEGB，请解释

我们已经知道了多个命名空间可以独立存在，而且可以在不同的层次上包含相同的变量名。“作用域”定义了Python在哪一个层次上查找某个“变量名”对应的对象。接下来的问题就是：“Python在查找‘名称-对象’映射时，是按照什么顺序对命名空间的不同层次进行查找的？”

答案就是：使用的是LEGB规则，表示的是**Local -> Enclosed -> Global -> Built-in**，其中的箭头方向表示的是搜索顺序。

* *Local* 可能是在一个函数或者类方法内部。
* *Enclosed* 可能是嵌套函数内，比如说 一个函数包裹在另一个函数内部。
* *Global* 代表的是执行脚本自身的最高层次。
* *Built-in* 是Python为自身保留的特殊名称。

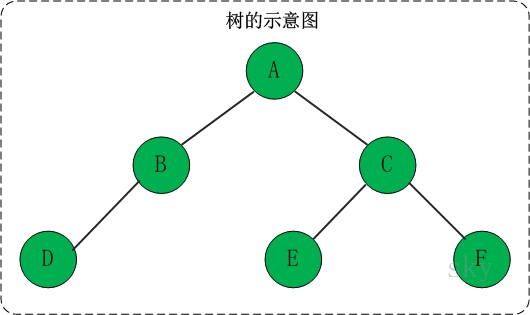
因此，如果某个name:object映射在局部(local)命名空间中没有找到，接下来就会在闭包作用域(enclosed)进行搜索，如果闭包作用域也没有找到，Python就会到全局(global)命名空间中进行查找，最后会在内建(built-in)命名空间搜索（注：如果一个名称在所有命名空间中都没有找到，就会产生一个NameError）。

1. 二叉树的性质

**树的介绍**

1. 树的定义

树是一种数据结构，它是由n（n>=1）个有限节点组成一个具有层次关系的集合。

[](https://images0.cnblogs.com/i/497634/201403/270929194211610.jpg)

把它叫做“树”是因为它看起来像一棵倒挂的树，也就是说它是根朝上，而叶朝下的。它具有以下的特点：  
(01) 每个节点有零个或多个子节点；  
(02) 没有父节点的节点称为根节点；  
(03) 每一个非根节点有且只有一个父节点；  
(04) 除了根节点外，每个子节点可以分为多个不相交的子树。

2. 树的基本术语

若一个结点有子树，那么该结点称为子树根的"双亲"，子树的根是该结点的"孩子"。有相同双亲的结点互为"兄弟"。一个结点的所有子树上的任何结点都是该结点的后裔。从根结点到某个结点的路径上的所有结点都是该结点的祖先。

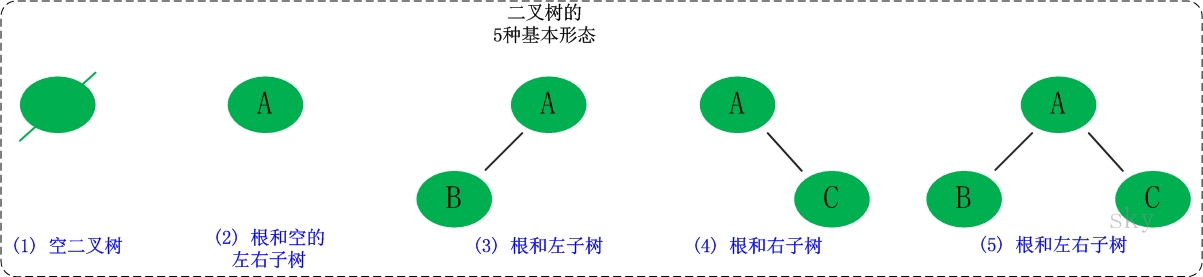
结点的度：结点拥有的子树的数目。  
叶子：度为零的结点。  
分支结点：度不为零的结点。  
树的度：树中结点的最大的度。

层次：根结点的层次为1，其余结点的层次等于该结点的双亲结点的层次加1。  
树的高度：树中结点的最大层次。  
无序树：如果树中结点的各子树之间的次序是不重要的，可以交换位置。  
有序树：如果树中结点的各子树之间的次序是重要的, 不可以交换位置。  
森林：0个或多个不相交的树组成。对森林加上一个根，森林即成为树；删去根，树即成为森林。

**二叉树的介绍**

1. 二叉树的定义

二叉树是每个节点最多有两个子树的树结构。它有五种基本形态：二叉树可以是空集；根可以有空的左子树或右子树；或者左、右子树皆为空。

[](https://images0.cnblogs.com/i/497634/201403/270929530778327.jpg)

2. 二叉树的性质

二叉树有以下几个性质：TODO(上标和下标)  
性质1：二叉树第i层上的结点数目最多为 2{i-1}(i≥1)。  
性质2：深度为k的二叉树至多有2{k}-1个结点(k≥1)。  
性质3：包含n个结点的二叉树的高度至少为log2 (n+1)。  
性质4：在任意一棵二叉树中，若终端结点的个数为n0，度为2的结点数为n2，则n0=n2+1。

2.1 性质1：二叉树第i层上的结点数目最多为 2{i-1}(i≥1)

证明：下面用"数学归纳法"进行证明。  
        (01) 当i=1时，第i层的节点数目为2{i-1}=2{0}=1。因为第1层上只有一个根结点，所以命题成立。  
        (02) 假设当i>1，第i层的节点数目为2{i-1}。这个是根据(01)推断出来的！  
               下面根据这个假设，推断出"第(i+1)层的节点数目为2{i}"即可。  
                由于二叉树的每个结点至多有两个孩子，故"第(i+1)层上的结点数目" 最多是 "第i层的结点数目的2倍"。即，第(i+1)层上的结点数目最大值=2×2{i-1}=2{i}。  
                故假设成立，原命题得证！

2.2 性质2：深度为k的二叉树至多有2{k}-1个结点(k≥1)

证明：在具有相同深度的二叉树中，当每一层都含有最大结点数时，其树中结点数最多。利用"性质1"可知，深度为k的二叉树的结点数至多为：  
           20+21+…+2k-1=2k-1  
           故原命题得证！

2.3 性质3：包含n个结点的二叉树的高度至少为log2 (n+1)

证明：根据"性质2"可知，高度为h的二叉树最多有2{h}–1个结点。反之，对于包含n个节点的二叉树的高度至少为log2(n+1)。

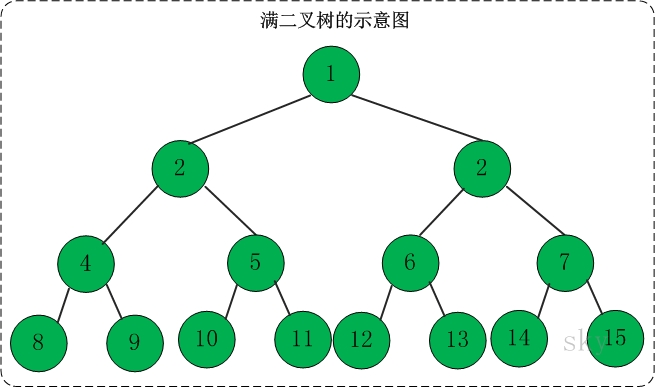
2.4 性质4：在任意一棵二叉树中，若终端结点的个数为n0，度为2的结点数为n2，则n0=n2+1

证明：因为二叉树中所有结点的度数均不大于2，所以结点总数(记为n)="0度结点数(n0)" + "1度结点数(n1)" + "2度结点数(n2)"。由此，得到等式一。  
         (等式一) n=n0+n1+n2  
　     另一方面，0度结点没有孩子，1度结点有一个孩子，2度结点有两个孩子，故二叉树中孩子结点总数是：n1+2n2。此外，只有根不是任何结点的孩子。故二叉树中的结点总数又可表示为等式二。  
         (等式二) n=n1+2n2+1  
        由(等式一)和(等式二)计算得到：n0=n2+1。原命题得证！

3. 满二叉树，完全二叉树和二叉查找树

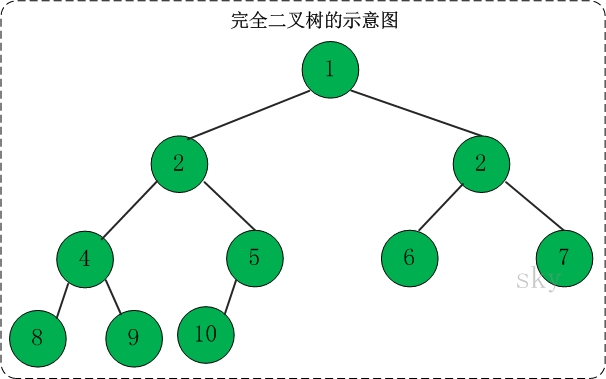
3.1 满二叉树

定义：高度为h，并且由2{h} –1个结点的二叉树，被称为满二叉树。

[](https://images0.cnblogs.com/i/497634/201403/270930282184259.jpg)

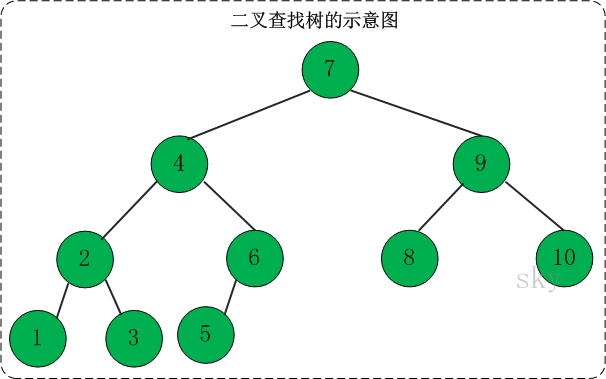
3.2 完全二叉树

定义：一棵二叉树中，只有最下面两层结点的度可以小于2，并且最下一层的叶结点集中在靠左的若干位置上。这样的二叉树称为完全二叉树。  
特点：叶子结点只能出现在最下层和次下层，且最下层的叶子结点集中在树的左部。显然，一棵满二叉树必定是一棵完全二叉树，而完全二叉树未必是满二叉树。

[](https://images0.cnblogs.com/i/497634/201403/270931211084932.jpg)

3.3 二叉查找树

定义：二叉查找树(Binary Search Tree)，又被称为二叉搜索树。设x为二叉查找树中的一个结点，x节点包含关键字key，节点x的key值记为key[x]。如果y是x的左子树中的一个结点，则key[y] <= key[x]；如果y是x的右子树的一个结点，则key[y] >= key[x]。

[](https://images0.cnblogs.com/i/497634/201403/270932052801072.jpg)

在二叉查找树中：  
(01) 若任意节点的左子树不空，则左子树上所有结点的值均小于它的根结点的值；  
(02) 任意节点的右子树不空，则右子树上所有结点的值均大于它的根结点的值；  
(03) 任意节点的左、右子树也分别为二叉查找树。  
(04) 没有键值相等的节点（no duplicate nodes）。

4. 使用本周学习的技术改造第二周的计算器实现，其目标如下：

1. 运行后提示让用户输入一个数字

2. 提示输入操作符（+ - \* /）

3. 再次提示输入一个数字

4. 打印计算结果

5. 在不退出程序的前提下，可以允许用户继续输入新一组数据计

6. 尽可能改善用户体验（新需求）