实验课 04

实验4-1 进制转换

计算机网络协议中的IP数据报中包含源地址字段和目的地址字段，以达到在网络层传输数据的需要。

众所周知，目前我们常用的IPv4地址由4部分十进制数组成（例如：202.120.87.12）。但是这是为了方便人们记忆采取的方法，由于信号的传递都是靠高低电平，所以在数据报中地址字段均以二进制表示，上述IP地址实际上会以11001010011110000101011100001100这32位二进制数存储在相应字段中。

现在我们尝试做一个IPv4地址转换器！

我们尝试做一个将二进制转换为十进制的转换器。可以看到32位的二进制数可按8位一组分成4个组，然后分别对每个组的二进制数转换为十进制数。

一般输入的二进制数是**字符串**的形式，我们可以先将字符串分为4个子串，然后分别转换为十进制数再组合成人们常见的IPv4地址格式。

```
def convert_ip_address_to_dec(address):
    subaddress_list=[]
    for i in range(4):
        subaddress_list.append(address[i*8:i*8+8])
    decaddress_list=[]
    for add in subaddress_list:
        decaddress_list.append(int(add,2))

 result=str(decaddress_list[0])+'.'+str(decaddress_list[1])+'.'+str(decaddress_l
ist[2])+'.'+str(decaddress_list[3])
    return result
```

上述函数中我们利用`int()`方法进行进制转换，可以尝试一下不同进制直接的转换。

```
print(int('1100',2))  # 二进制 -> 十进制
print(int('227',8))  # 八进制 -> 十进制
print(int('3C5F',16))  # 十六进制 -> 十进制
```

现在我们测试一下这个函数

```
address_1='11001010011110000101011100001100'
print(convert_ip_address_to_dec(address_1))
```

实验4-2 Python基础数据结构

Python中的变量不需要声明。每个变量在使用前都必须赋值，变量赋值以后该变量才会被创建。

在Python中，变量就是变量，它没有类型，我们所说的"类型"是变量所指的内存中对象的类型。

Python中的基本数据类型包括`int`,`float`,`bool`,`complex`,`str`,`list`,`set`,`tuple`和`dict`。

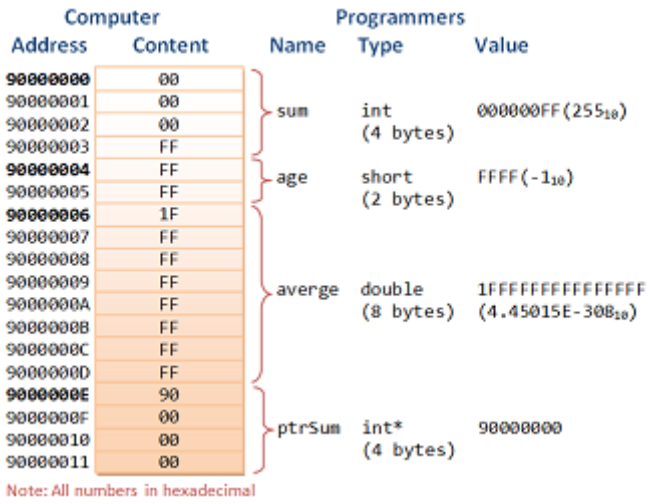
步骤1 查看变量的“类型”

```
num_1 = 6
num_2 = 3.14
num_3 = "Tom"
list_1 = [1,2,3,4,5]
dict_1 = {'a':1 , 'b':2 , 'c':3}

print(type(num_1))
print(type(num_2))
print(type(num_3))
print(type(list_1))
print(type(dict_1))
```

可以看到，使用`type()`方法可以返回变量所指的内存中对象的类型。

步骤2 查看变量的地址

在 C/C++ 语言中，指针用来表示或存储一个存储器地址，这个地址的值直接指向存在该地址的对象的值。



Python中并没有“指针”的概念，不过我们还是可以通过一些方法查看变量在内存中的唯一身份标识。

我们查看一下步骤1中定义的几个变量的内存唯一身份标识。

```
print(id(num_1))
print(id(num_2))
print(id(num_3))
print(id(list_1))
print(id(dict_1))
```

可以看到，使用`id()`方法可以返回变量在内存中的唯一身份标识，可理解为内存地址。

在CPython中使用`id()`方法可以返回变量在内存中的真正地址。

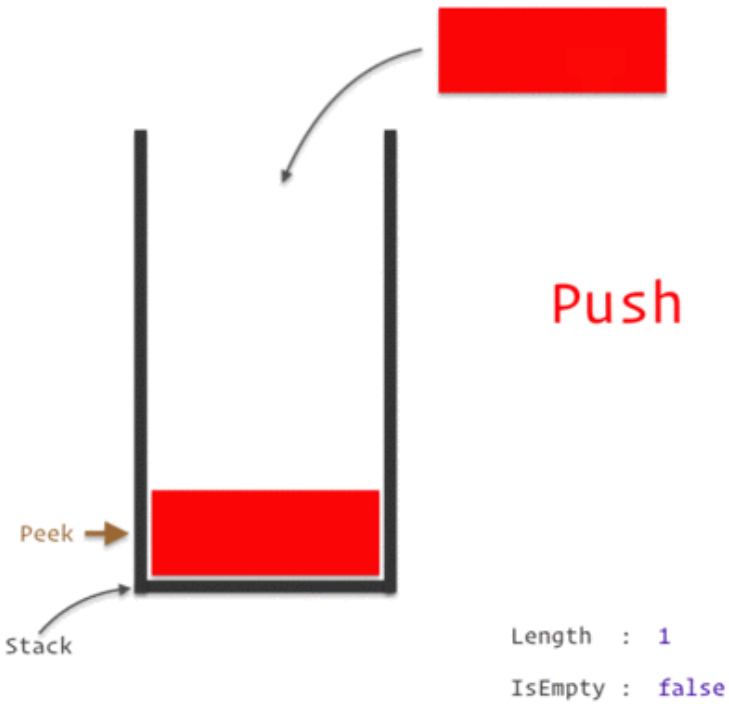
步骤3 判断对象的类型

有的时候我们会写大量的复杂代码，需要了解其中某个对象的类型，我们可以使用`isinstance()`方法来判断某个对象是否为一个已知的类型。

```
a = 2
print(isinstance(a,int))

b = [1,2,3]
print(isinstance(b,tuple))
```

实验4-3 Python高级数据结构示例1：堆栈

堆栈(stack)是计算机科学中的一种抽象数据类型，只允许在有序的线性集合的一端（称为栈顶,top）进行加入数据（push）和移除数据（pop）的运算。因而按照后进先出（LIFO, Last In First Out）的原理运作。



我们用Python实现一个堆栈类，并进行测试。

```
class Stack():
    def __init__(self):
        self.__items = []
    def size(self):  # 返回堆栈长度
```

```
        return len(self.__items)
    def isempty(self):  # 返回堆栈是否为空
        if len(self.__items)==0:
            return True
        else:
            return False
    def push(self,element):  # 压入堆栈
        self.__items.append(element)
    def pop(self):  # 弹出堆栈，注意需要处理堆栈为空的情况
        try:
            return self.__items.pop()
        except:
            print('ERROR: Stack is empty now!')
    def peek(self):  # 返回栈顶元素，注意需要处理堆栈为空的情况
        try:
            return self.__items[-1]
        except:
            print('ERROR: Stack is empty now!')
```

```
s = Stack()
s.push(1)  # 堆栈目前为 [1]
print(s.pop()) # 堆栈目前为 [ ]
print(s.pop()) # 错误，堆栈为空

print('***********')

s.push(3.5)  # 堆栈目前为[3.5]
s.push(2.7)  # 堆栈目前为[3.5, 2.7]
print(s.peek())
print(s.size())
print(s.isempty())
```

实验4-4 Python高级数据结构示例2：树

在数据结构中，树(tree)是由n（n>0）个有限节点组成一个具有层次关系的集合。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：

- 每个节点都只有有限个子节点或无子节点；
- 没有父节点的节点称为根节点；
- 每一个非根节点有且只有一个父节点；
- 除了根节点外，每个子节点可以分为多个不相交的子树；
- 树里面没有环路。

在C语言中，我们经常利用结构体定义树节点，例如定义一个二叉树结点：

```
struct TreeNode
{
    int data;
    struct TreeNode* LNode;
    struct TreeNode* RNode;
};
```

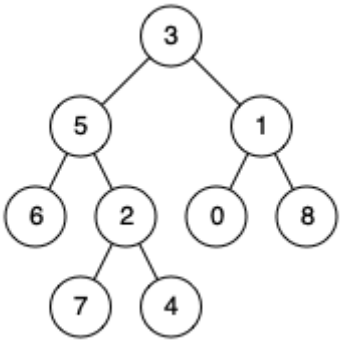
结果我们实际运用时被烦人的指针弄晕了……

在Python中，我们可以创建一个类构造一棵树，现在我们定义一个二叉树类，并测试相关这个类方法。

```
class BinaryTree:
    def __init__(self,data=None,left=None,right=None):  # 如果创建节点对象时left或
right参数为空，则默认该节点没有左或右子树
        self.data=data
        self.left=left
        self.right=right
    def preorder(self):  # 前序遍历
        print(self.data,end=' ')
        if self.left != None:
            self.left.preorder()
        if self.right != None:
            self.right.preorder()
    def midorder(self):  # 中序遍历
        if self.left != None:
            self.left.midorder()
        print(self.data,end=' ')
        if self.right != None:
            self.right.midorder()
    def postorder(self):  # 后序遍历
        if self.left != None:
            self.left.preorder()
        if self.right != None:
            self.right.preorder()
        print(self.data,end=' ')
    def height(self):
        if self.data is None:  # 空的树高度为0, 只有root节点的树高度为1
            return 0
        elif self.left is None and self.right is None:
            return 1
        elif self.left is None and self.right is not None:
            return 1 + self.right.height()
        elif self.left is not None and self.right is None:
            return 1 + self.left.height()
        else:
            return 1 + max(self.left.height(), self.right.height())
```

上述定义地类包含先序、中序和后续遍历，获得树的高度。

我们以这棵树作为测试，为了上层能直接添加左右子树，我们从树的底部开始逐步创建完整的树。




```
layer3_2 = BinaryTree(2,BinaryTree(7),BinaryTree(4))
layer2_5 = BinaryTree(5,BinaryTree(6),layer3_2)
layer2_1 = BinaryTree(1,BinaryTree(0),BinaryTree(8))
layer1_3 = BinaryTree(3,layer2_5,layer2_1)
```

```
layer1_3.preorder()
print()
layer1_3.midorder()
print()
layer1_3.postorder()
print()
print(layer1_3.height())
```

Python有一个第三方库AnyTree，可以方便的创建一棵树并对其操作，我们无需过多关注内部实现细节。

实验4-5 正则表达式初步（选做）

正则表达式使用单个字符串来描述、匹配一系列符合某个句法规则的字符串。在很多文本编辑器里，正则表达式通常被用来检索、替换那些符合某个模式的文本。

Python自1.5版本起增加了`re`模块，它提供Perl风格的正则表达式模式。`re`模块使Python语言拥有全部的正则表达式功能。

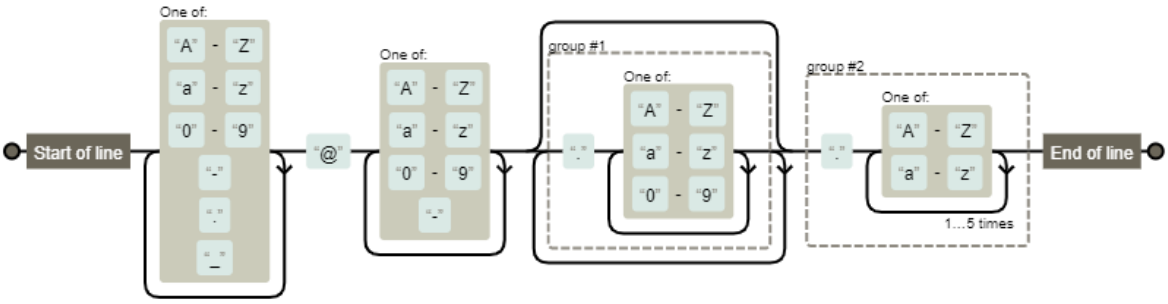
我们来尝试使用`re`库中的一些方法。

步骤1 `match()`方法

当我们注册一个论坛时需要提供自己的邮箱地址作为用户名，为了防止恶意注册，网站都会设置一个检查机制检验输入的电子邮箱地址是否合法。

可以利用该正则表达式匹配电子邮箱地址：`^[A-Za-z0-9._]+@[A-Za-z0-9-]+(\.[A-Za-z0-9]+)*(\.[A-Za-z]{2,6})$`

利用Regexper可视化这个正则表达式如图所示




```
import re
p = '^[A-Za-z0-9._]+@[A-Za-z0-9-]+(\.[A-Za-z0-9]+)*(\.[A-Za-z]{2,6})$'
s1 = 'tom@gmail.com'
s2 = 'xiaoming_wang123@dase.ecnu.edu.cn'
s3 = '113@s#h.xyz'
print(re.match(p,s1))
print(re.match(p,s2))
print(re.match(p,s3))
```

如果一个字符串符合正则表达式的匹配模式，则`match()`方法返回一个对象，包含匹配位置属性`span`和匹配字符串`match`。

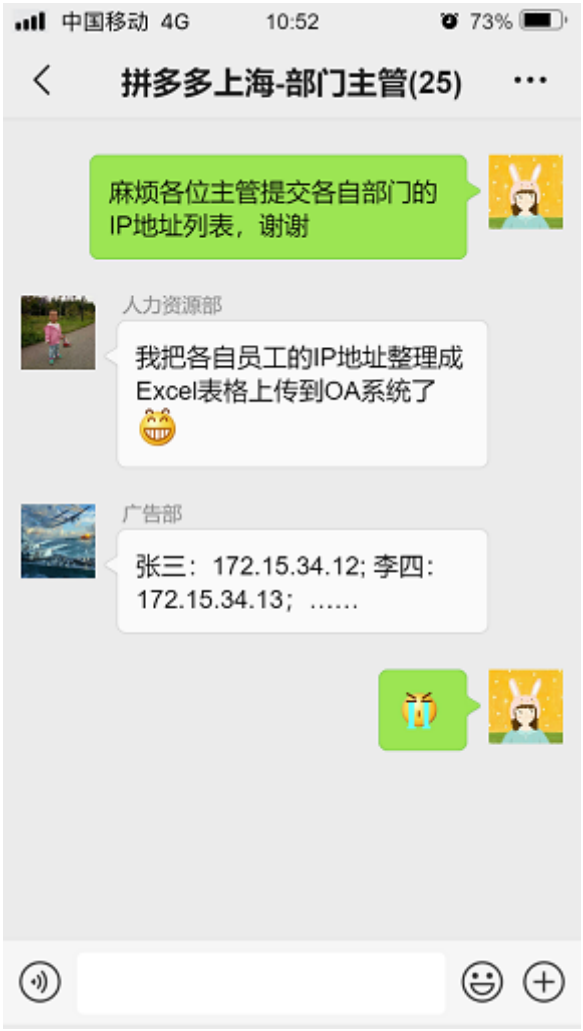
如果不符合，则返回空对象None。

步骤2 `search()`方法

网管有一天在群中发布消息

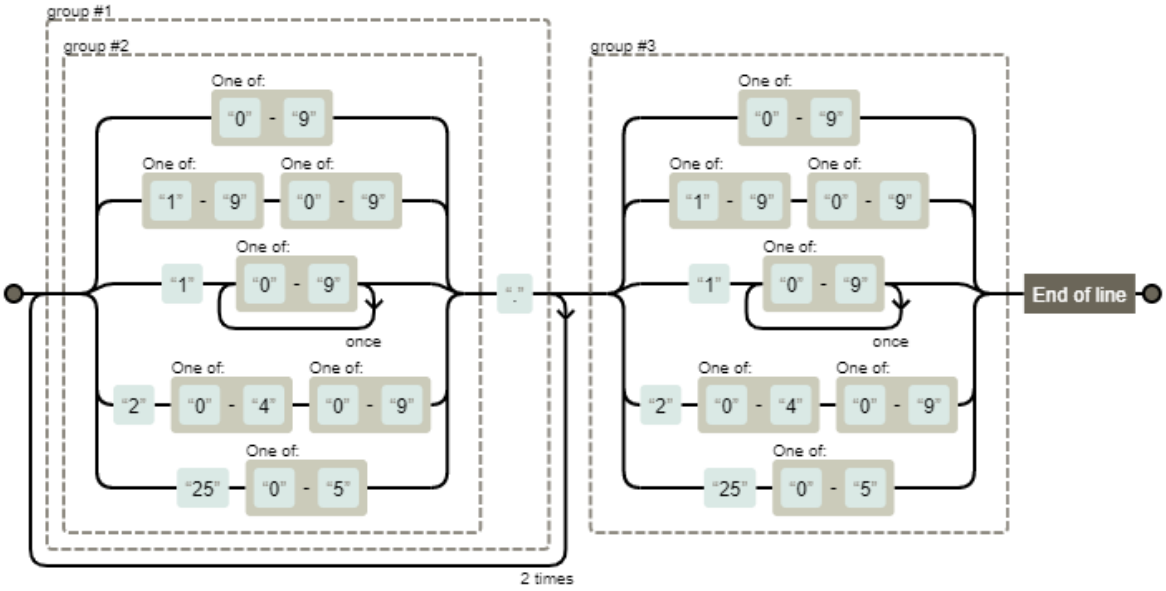
> *技术部小张:* 请全体员工把自己办公室的电脑IP地址提交给各自部门主管，各自部门主管再提交至技术部小张。

后来小张收到了各自部门主管提交的IP地址列表。有些主管贴心地把姓名和IP地址分成两列存为Excel文件，但是一些主管懒得处理，直接把各自员工地聊天记录**粘贴在一起提交了**……



小张也懒得处理，让新来的小李把广告部的IP地址整理好，结果小李最后生成了['张三：172.15.34.12','李四：172.15.34.13',...]

有的员工眼花可能上报了错误的IP地址，我们需要通过正则表达式找出不符合规范的IP地址。




```
import re
ip_list=['张三：172.15.34.12','李四：172.15.34.13','王五：172.15.34.14','小明：
172.15.334.15']
p = '(([0-9]|[1-9][0-9]|1[0-9]{2}|2[0-4][0-9]|25[0-5])\.){3}([0-9]|[1-9][0-
9]|1[0-9]{2}|2[0-4][0-9]|25[0-5])$'

for ip in ip_list:
    if re.search(p,ip) == None:
        print(ip+' is ERROR!!!')
    else:
        print('OK')
```

可以看到，小明提交的IP地址有明显错误，如果字符串中没有符合正则表达式的子串，则`search()`方法返回None

注意：`match()`方法和`search()`方法的区别

- `re.match`从字符串的第一个字符开始匹配，如果字符串一旦不符合正则表达式，则匹配失败，函数返回None；
- 而`re.search`匹配整个字符串，直到找到一个匹配，若字符串没有符合的字串，则返回None。

实验练习04

1. 请编写一个函数，利用***辗转除2取余法（不要使用`int`的直接转换）***实现将32位二进制的IPv4地址转换为我们常见的十进制IPv4地址，并进行测试。

> 提示：可先将32位二进制字符串分为4组

2. 请创建一个类模拟实现数据结构中的“**队列**”，类中应包含入队、出队、取队首/队尾元素等方法，并进行测试。
3. 请在实验4-4的基础上实现二叉树的层序遍历的函数。

> 提示：需要用到队列

4. 请在实验4-4的基础上实现输出叶子节点的函数。
5. ***（选做）***目前中国内地三大运营商的手机号段分布如下：

> 中国电信：133、153、173、177、180、181、189、191、193、199

> 中国联通：130、131、132、155、156、166、175、176、185、186
>
> 中国移动：134(但第4位不含9)、135、136、137、138、139、147、150、151、152、157、158、159、172、178、182、183、184、187、188、198

中国内地的手机号长度为11位，请设计一个正则表达式，并在Python中测试某个手机号是否为合法的手机号。

如何提交作业

步骤1 检查计算中是否安装Git

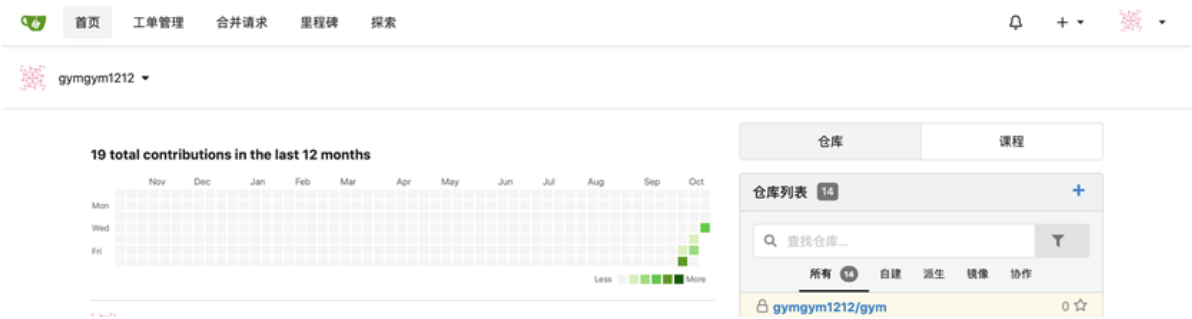
按 Win + R 键，然后输入`cmd`后回车打开命令提示符，再输入`git`回车，若有相关提示信息，则说明计算机中已安装Git。

```
guyeming@guyemingdeMacBook-Pro bin % git
usage: git [--version] [--help] [-C <path>] [-c <name>=<value>]
           [--exec-path[=<path>]] [--html-path] [--man-path] [--info-path]
           [-p | --paginate | -P | --no-pager] [--no-replace-objects] [--bare]
           [--git-dir=<path>] [--work-tree=<path>] [--namespace=<name>]
           <command> [<args>]

These are common Git commands used in various situations:

start a working area (see also: git help tutorial)
   clone     Clone a repository into a new directory
   init      Create an empty Git repository or reinitialize an existing one

work on the current change (see also: git help everyday)
   add       Add file contents to the index
```

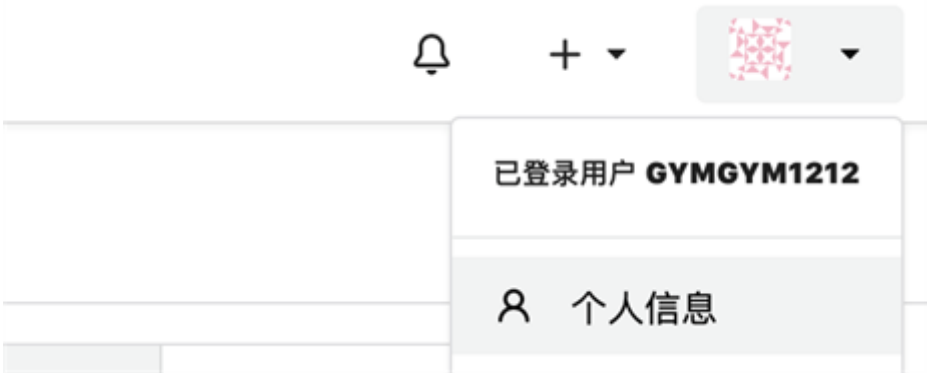
否则请点击该链接下载Git安装包，并进行安装。

步骤2 配置个人信息

登录水杉在线，然后点击水杉码园，系统会自动为你在水杉码园中创立账户。



点击页面右上角的头像，然后点击个人信息。



查看水杉码园账户对应的邮箱，稍后会使用到该字段。

打开命令提示符或PowerShell，输入以下命令以设置你的本地Git姓名和邮箱信息，**请确保邮箱与上图中你所看到的邮箱地址相同。**

```
git config --global user.name "Your Name"
```

```
git config --global user.email "369713635@qq.com"
```

设置完成后可以通过以下命令查看你的姓名和邮箱信息

```
git config user.name
```

```
git config user.email
```

步骤3 生成SSH密钥

在命令行中输入 `ssh-keygen -t rsa -C '你的邮箱地址'` 后回车，如下图

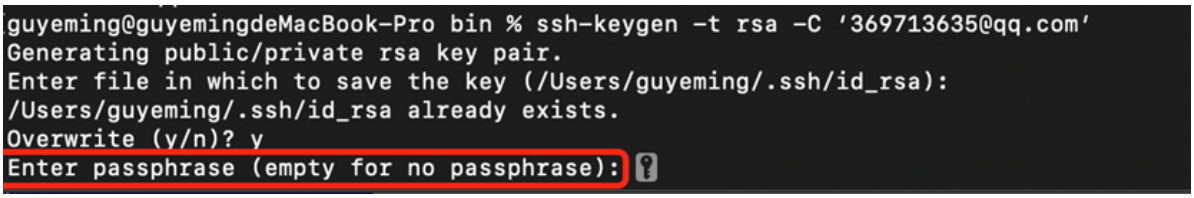
```
guyeming@guyemingdeMacBook-Pro bin % ssh-keygen -t rsa -C '369713635@qq.com'
Generating public/private rsa key pair.
Enter file in which to save the key (/Users/guyeming/.ssh/id_rsa):
```

红框中是提示你要将密钥存放在哪里，括号里的是默认的存放路径，**请记住这个路径**，我们之后需要这个路径找到密钥。

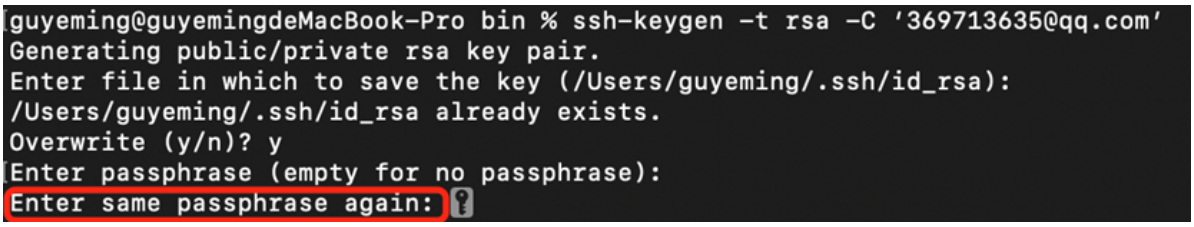
接着直接按回车，如果你之前也按照默认的路径生成过SSH密钥，会出现如下图中的情况，询问你是否要覆盖。

```
guyeming@guyemingdeMacBook-Pro bin % ssh-keygen -t rsa -C '369713635@qq.com'
Generating public/private rsa key pair.
Enter file in which to save the key (/Users/guyeming/.ssh/id_rsa):
/Users/guyeming/.ssh/id_rsa already exists.
Overwrite (y/n)?
```

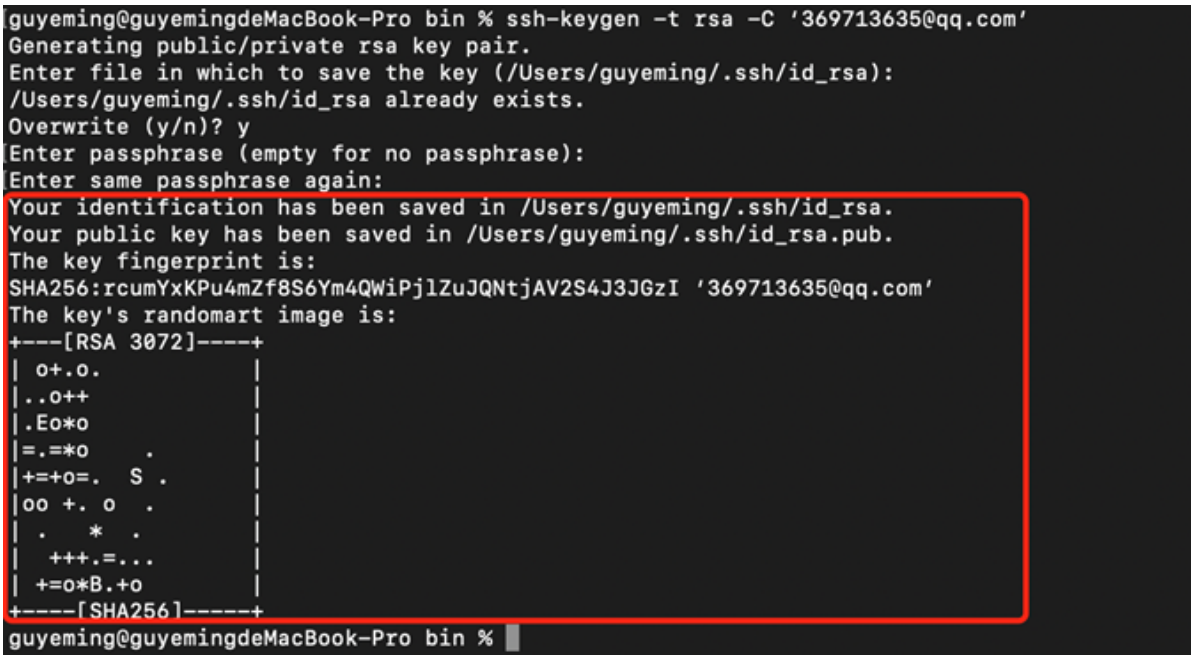
输入y覆盖原有的密钥，或者之前没有生成过密钥，就会出现如下图中情况：



直接回车不用管，如下：



再直接回车不用管，如下：



可以看到你的密钥已经创建成功。

步骤4 上传SSH公钥至码园账户

登陆水杉码园后，在右上角头像的下拉框中点击设置

在标签中选择SSH/GPG密钥

在管理SSH密钥卡片右侧点击增加密钥，打开你的id_rsa.pub文件 *（这个文件在步骤3让你记住的路径下，用记事本打开）*，将里面所有的内容复制到密钥内容一栏中，然后给这个密钥取一个名字，比如：我的MacBook。点击绿色的增加密钥按钮后，就成功添加了SSH密钥。

步骤5 将远程仓库克隆至本地

请在你的电脑中创建一个文件夹，用作本地仓库，以后将提交的作业放入该文件夹中。

在命令行中输入`git clone root@gitea.shuishan.net.cn:DaSE_IntroCourse/你的学号.git`然后再按下回车键，开始克隆远程仓库至本地。

步骤6 放入作业文件并提交

本次作业请将每一道题的代码写入py源代码文件，并分别命名为`lab04-1.py`,`lab04-2.py`

然后放入该文件夹中。

在命令行中输入`git add .`**（不要忘记add后有一个空格和英文句号）**，将代码文件加入追踪文件的清单中。

然后再输入`git commit -m 'Lab04'`，向本地仓库提交刚才加入的被追踪文件，顺带写上一些本次提交的小注释。

最后向远程仓库推送，输入`git push`，向远程仓库同步本地仓库，若第一次连接，会提示信息，输入`yes`或`y`即可。