1 Hint

1.本章所有代码题，将其拷贝到Thonny中，run起来后，查看其结果，明白结果的由来。学有余力者可研究每行代码的含义，如果没有兴趣，可忽略研究代码。

2 语法补充

`def` 为Python中定义函数的标志，以4-3为例， `bubble_sort` 表示函数名称， `return` 表示返回值， `len()` 表示计算 `list` 的长度

```
def bubble_sort(nums):
    # 这个循环负责设置冒泡排序进行的次数
    for i in range(len(nums) - 1):
        for j in range(len(nums) - i - 1):  # j为列表下标
            if nums[j] > nums[j + 1]:
                nums[j], nums[j + 1] = nums[j + 1], nums[j]
        if i==2:print(nums[5])
    return nums

bubble_sort([45, 32, 8, 33, 12, 22, 19, 97])
```

3 填空题

本部分对题目中所有的代码进行解释

3.1 4-3

下面代码实现了冒泡排序算法

```
def bubble_sort(nums):
    # 这个循环负责设置冒泡排序进行的次数，因为是从小到大排序，所以第(i+1)次就是代表计算出第(i+1)大的数
    for i in range(len(nums) - 1):
        for j in range(len(nums) - i - 1):  # j为列表下标
            if nums[j] > nums[j + 1]:# 当前元素比后面元素大，进行交换
                nums[j], nums[j + 1] = nums[j + 1], nums[j] #交换位置
        if i==2:print(nums[5]) #输出第i+1大的数，这里输出第3大的数，第三大的数在数组nums中的位置为5，
    return nums

bubble_sort([45, 32, 8, 33, 12, 22, 19, 97])
```

3.2 4-4

该题代码与上一道基本一致，

```
def bubble_sort(nums):
    # 这个循环负责设置冒泡排序进行的次数，
    rst=0
    for i in range(len(nums) - 1):
        for j in range(len(nums) - i - 1):  # j为列表下标
            if nums[j] > nums[j + 1]:
                nums[j], nums[j + 1] = nums[j + 1], nums[j]
        if i==0 or i==3: #如果是第1大的数或第4大的数
            rst=rst+nums[5] #将当前的nums[5]和rst相加，注意这里的nums[5]是变动的
    return rst

print(bubble_sort([45, 32, 8, 33, 12, 22, 19, 97])) #rst=19+33=52
```

3.3 4-5

下面代码实现了选择排序算法

```
#从小到大排序
def selectionSort(arr):
    for i in range(len(arr) - 1):#i+1表示第i+1小的数，i=0，表示第1小的数
        # 记录最小数的索引
        minIndex = i
        for j in range(i + 1, len(arr)): #
            if arr[j] < arr[minIndex]:#如果后续数字有比初始最小数小，更新最小值索引
                minIndex = j
        # i 不是最小数时，将 i 和最小数进行交换
        if i != minIndex:
            arr[i], arr[minIndex] = arr[minIndex], arr[i] #交换
        if i==2:print(arr[5])#i==2表示第3小的数，当算出第3小的数时，输出此是数组中第6个数(arr[5]),
        #即输出100
    return arr

selectionSort([33,6,9,7,-1,100,-32])
```

3.4 4-6

一个二进制转换程序，将输入的整数转换为2进制数

```
n=int(input())
lst=[]#产生一个空的list
while n>0:
    lst.insert(0,n%2)#insert函数，每次将n%2的值，插入在lst的0号位置。n%2代表计算余数，例如16%2的余数为
    n=n//2#n与2进行整除，注意n//2和n/2是不一样的操作，n/2和实际我们算术运算除法一样
           #，而n//2代表整除，例如17//2=8，而不是8.5
print(lst[3]) #输出lst中的第4个元素，当输入为16时，也就是0
```

3.5 4-7

下面的代码是在计算下面函数

f(n)=f(n-1)+f(n-2)+2,n>=3

当f(1)=3,f(2)=4时，

f(5)=f(4)+(f3)+2

f(4)=f(3)+f(2)+2

f(3)=f(2)+f(1)+2

由于f(4),f(5)依赖于f(3)，目前只有f(3)能够算出，f(3)=9，f(4)=15，f(5)=26

```
def f(n):
    result=[3,4]
    for i in range(n-2):
        result.append(result[-2]+result[-1]+2) #result[-1]代表result的倒数第1个元素，
                                    #同理-2代表第2个，append为在list的最末尾添加一个元素
    return result[-1]
print(f(5))  #输出26
```

3.6 4-8

下面代码是在完成输入一个数，找到这个数要插入的位置
输入101，通过列表a我们能够看到，它应该被插入在100的后面，100的位置是6，则101是7，也就是插入到第7个位置

```
a=[5,16,39,45,51,98,100,202,226,321,368,444,501]
x = int(input())
found = -1
left = 0                        #第一个元素下标
right = len(a)-1                #最后一个元素下标
while left<=right:
    mid = (left + right) // 2
    if a[mid] > x:
        right = mid - 1
    elif a[mid] < x:
        left = mid + 1
    else:                       # a[mid]==x
        found = mid
        break
print(left)
```

3.7 4-9

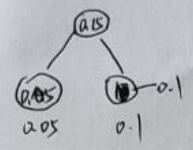
哈夫曼树的建立规则：每次选出两个权值最小结点，同时计算出两个结点的和，计算出的结点为新的结点。
左0右1还是左1右0，对结果没有影响，哈夫曼树算法的目的是做数据压缩，因此无论前面哪种编码方式，数据压缩比（平均位数）均一致。

6个结点的权值如下

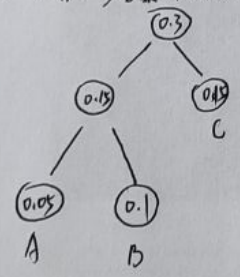
A:0.05　B:0.1　C:0.15　D:0.2　E:0.22　F:0.28

哈夫曼树建立规则：每次选出两个最小结点，同时计算出两个结点的和，计算出的结点为新的结点。

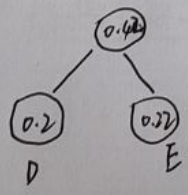
(1) A、B最小，选出A、B



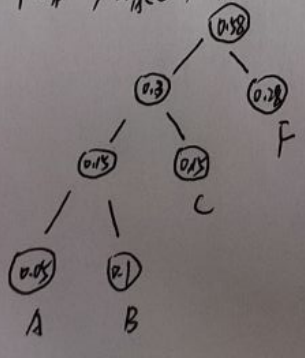
(2) C和步骤(1)建立的结点最小



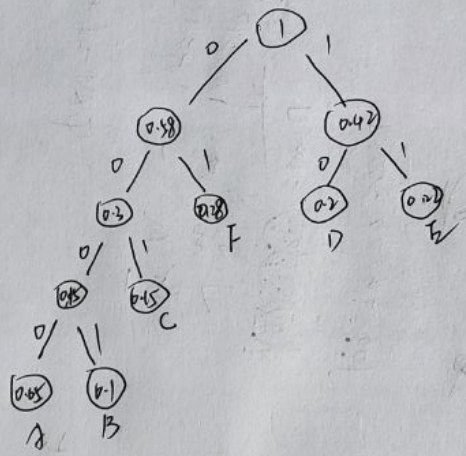
(3) D、E是当前最小两个结点



(4) F和步骤(2)建立的结点最小



(5) 步骤(3)和步骤(4)建立结点最小



A:0000　B:0001　C:001　D:10　E:11　F:01

平均值数：$0.05\times4+0.1\times4+3\times0.15+0.2\times2+2\times0.22+2\times0.28$

$=0.2+0.4+0.45+0.4+0.44+0.56$

$=2.45$

3.8　4-10

下面代码是通过分治算法来求解一个list中的最小和最大值。
假设初始值为[3,8,9,4]，则len=4
执行下面代码时，第一次使用min_max函数
a,b=min_max([3,8,9,4])
m=2，lmin,lmax=min_max(a[:m]) #a[:m]，表示0-m的数字，rmin,rmax=min_max(a[m:])表示从m开始到结束的数
[3,8]被拿出来再次执行min_max，len=2
m=2,lmin,lmax=min_max(a[:m])，lmin=3，lmax=8
[9,4]被拿出来再次执行min_max，len=2，
m=2,lmin,lmax=min_max(a[m:])，rmin=4，rmax=9
最后执行return(min(lmin,rmin),max(lmax,rmax))
min(lmin,rmin)是对左右两边最小值求其最小，因此为3
max(lmax,rmax)是对左右两边最大值求其最大，因此为9
最终结果为3，9

```
def min_max(a):
    if len(a)==1:
        return (a[0],a[0])#长度为1，最大值和最小值均为本身
    elif len(a)==2:
        return (min(a),max(a)) #长度为2，使用min，max函数求最小和最大
    else:
        #对a进行折半，再求其最小和最大
        m=len(a)//2
        lmin,lmax=min_max(a[:m])
        rmin,rmax=min_max(a[m:])
        return (min(lmin,rmin),max(lmax,rmax))#对左右求出的结果，使用min,max分别求最小和最大。

a,b=min_max([3,8,9,4,10,5,1,17,9,-5])
print(a+b) #-5+17=12
```

3.9　4-11

生日计算的原理解释：
这道题目利用的是二进制表示数据的范围，5位二进制最高能表示的是11111，即最大为31，则它能表示的范围是[0,31]，涵盖了一个月的所有天数，.
题目中有5个list，这5个list的作用，就是在控制每一位上是否为1，从而表示出任意一天
假设生日为31号，你可以看到每个list都有31，则说明输入的二进制为11111
s[0]=1 表示生日在列表中

```
lst1[0] if s[0]=="1" else 0 #如果s[0]=='1'，则说明该第1位二进制位应该为lst1[0]=1=2^0，而不是0
lst2[0] if s[1]=="1" else 0 #如果s[1]=='1'，则说明该第2位二进制位应该为lst2[0]=2=2^1，而不是0
lst3[0] if s[2]=="1" else 0 #如果s[2]=='1'，则说明该第2位二进制位应该为lst3[0]=4=2^2，而不是0
lst4[0] if s[3]=="1" else 0 #如果s[3]=='1'，则说明该第2位二进制位应该为lst4[0]=8=2^3，而不是0
lst5[0] if s[4]=="1" else 0 #如果s[4]=='1'，则说明该第2位二进制位应该为lst5[0]=16=2^4，而不是0
```

所以生日为31号，则lst1[0]=1,lst2[0]=2,lst3[0]=4,lst3[0]=8,lst4[0]=16，分别代表2进制转换为十进制的每一位的值

```
lst1=[1,3,5,7,9,11,13,15,17,19,21,23,25,27,29,31]
lst2=[2,3,6,7,10,11,14,15,18,19,22,23,26,27,30,31]
lst3=[4,5,6,7,12,13,14,15,20,21,22,23,28,29,30,31]
lst4=[8,9,10,11,12,13,14,15,24,25,26,27,28,29,30,31]
lst5=[16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31]
s=input()
print((lst1[0] if s[0]=="1" else 0)
        +(lst2[0] if s[1]=="1" else 0)
        +(lst3[0] if s[2]=="1" else 0)
        +(lst4[0] if s[3]=="1" else 0)
        +(lst5[0] if s[4]=="1" else 0))
```

3.10　4-12

3.10.1　简单理解：

重用方框三：重用就是能够该状态能够被再次使用，这里的状态可以简单理解为就是余下重量和剩余可选物品。
因此只要这两个都相同，就可以认为其状态一致，状态一致则可以被重复使用。所以结果为9。

3.11　4-14

一个走迷宫的算法，答案为Thonny下print([5,4] in path)该句输出。
这道题目的是先用m把地图表达出来，1代表边界，0代表能走的区域，之后执行走迷宫算法，走迷宫算法使用到了回溯法。先尝试往下走，如果往下走走不通，那么回溯一步，在走不通的地方尝试往上走，如果还不行，再回溯，再往左走，如果还不行，再回溯，再往右走

```
#走迷宫
from turtle import *
m=[[1,1,1,0,1,1,1,1,1,1],[1,0,0,0,0,0,0,0,1,1],[1,0,1,0,1,1,1,0,0,1],
[1,0,1,0,0,0,0,1,0,1],[1,0,1,0,1,1,0,0,0,1],[1,0,0,1,1,0,1,0,1,1],
[1,1,1,1,0,0,0,0,1,1],[1,0,0,0,0,1,1,1,0,0],[1,0,1,1,0,0,0,0,0,1],
[1,1,1,1,1,1,1,1,1,1]] #地图

def jumpto(x,y):
    up();goto(x,y);down()

def drawBox(x,y,size,blocked):
    color("black");jumpto(x,y)
    if blocked: #边界为黑色，数字1代表边界
        fillcolor("black") #颜色填充函数
        begin_fill()
        for i in range(4):forward(size);right(90)
        end_fill()
    else:#可走部分为白色，数字0代表可走部分
        for i in range(4):
            forward(size/6);up();forward(size/6*4);down()
            forward(size/6);right(90)

def draw_myth():
    global m;reset();speed('fast');size=40
    for i in range(0,len(m)):
        for j in range(0,len(m[i])):
            drawBox(-200+j*size,200-i*size,size,m[i][j])


sta1=0;sta2=3;fsh1=7;fsh2=9; success=0 #sta1,sta2代表入口，fsh1,fsh2代表出口
path=[]
size=40 #每个格子的大小
r=(size-10)/2
global mouse
mouse= Turtle()  #创建一个turtle对象
x=-200+3*size+size/2
y=200-0*size-size/2
mouse.up()
mouse.goto(x,y)
mouse.speed(1) #画笔速度
mouse.down()
from turtle import *
def LabyrinthRat():
    global m #在函数内部对函数外的变量进行操作，就需要在函数内部声明为global
    print("显示迷宫: ")
    for i in range(len(m)):
            print(m[i])
    print("入口: m[%d][%d]: 出口: m[%d][%d]"%(sta1,sta2,fsh1,fsh2))
    if (visit(sta1,sta2,sta1,sta2))==0:
            print("没有找到出口")
```

```
    else:print("显示路径: ")
    for i in range(10):print(m[i])
def visit(i,j,p,q): # 这是走迷宫的算法，先尝试往下走，如果往下走走不通，那么尝试在走不通的地方尝试往上走
    global m
    m[i][j]=2;x=-200+j*size+size/2;y=200-i*size-size/2;
    mouse.pencolor("black");mouse.goto(x,y)
    global success,path
    path.append([i,j])
    if(i==fsh1)and(j==fsh2): success=1
    if(success!=1)and(m[i-1][j]==0): visit(i-1,j,i,j) #往下走
    if(success!=1)and(m[i+1][j]==0): visit(i+1,j,i,j) #往上走
    if(success!=1)and(m[i][j-1]==0): visit(i,j-1,i,j) #往左走
    if(success!=1)and(m[i][j+1]==0): visit(i,j+1,i,j) #往右走
    if success!=1: #如果走到该点，但无法继续走通
        m[i][j]=3; x=-200+q*size+size/2;
        y=200-p*size-size/2;mouse.pencolor("white");
        mouse.goto(x,y)
    return success

tracer(False);draw_myth();tracer(True);LabyrinthRat()
#print(path) #打印path
print([5,4] in path)
#print([5,5] in path)
```