第 1 周

尽量不用计算机，快速回答下列问题。
1.
x=123.45
那么 id(x)是 x 的地址还是 123.45 的地址？
答案：是 123.45 的地址

2.执行下面三句话后
x=123.45
y=x
x=123.456
那么 y 的值现在是多少？
答案：123.45
注意：在 shell 下是不能三句话一起执行的。只能分三次输入指令。如果存成.py，那是可以 F5 批量执行的

3.判断浮点变量 x 和 y 是否相等，程序语句如何写？
答案：abs(x-y) < 1e-6
浮点值不像整数，在 python 中经常是不精确的，不能判断直接相等。应该用判断两者是否足够接近来代替。

如何让变量 x 里的浮点值保留 2 位小数？要求总是正常四舍五入的，非银行家圆整。
答案：x=int(x*100+0.5)/100 或 x=round(x,2)

第 2 周

执行以下语句
x=input()
y=input()
print(x<y,x<=y,sep='')
若用户键盘输入：9（回车）11（回车）
输出结果是?
答案：
FalseFalse
input 的结果是字符串，字符串比较按照字典顺序。常用字符的编码大小：空格<0～9<A～Z<a～z。sep 设为空字符串，意为无分隔符。

第 3 周

以下问题尽量不要使用电脑来得到结果。
已知 x=[1, 2, 3]
求下面各语句执行后 x 的内容
x.append([4])
正确答案：[1,2,3,[4]]
下一题
接着上面的语句
x.extend([5])
正确答案：[1,2,3,[4],5]
下一题
x.insert(6,6)
正确答案：[1, 2, 3, [4], 5, 6]
注意：指定位置太大没用，6 实际插在末尾 5 号位置。不会报错，也不会在 5 号位置产生一个空档。
下一题
x.pop(5)
此时 x 的内容？
不是问 pop 返回值
正确答案：[1, 2, 3, [4], 5]
下一题
x.remove(4)
正确答案：报错。
那么，要删掉二级列表[4]该怎么做？
x.remove([4])
指定位置 pop 或 delete 也可以，使用切片删除也可以。
接着上面语句(回到出错之前)，以下语句输出的结果是？
print(x[0:6:3])
正确答案：[1,[4]]
print(x[6:0])
正确答案：[]
print(x[6:0:-1])
正确答案：[5,[4],3,2]
print(x[6:-1:-1])
正确答案：[]。题目中前一个-1 是指最后一个元素位置，理解为“比 0 小，终点可以到 0 号元素”是错的。
print(x[6::-1])
正确答案：[5,[4],3,2,1]

第 4 周

以下问题假设已经执行过
```
import math
import random
```
利用列表推导，生成 100 以内素数的语句（尽量默写）
注意：不要漏了 math.
参考答案：
```
[ p for p in range(2, 100) if 0 not in [ p%d for d in range(2,
int(math.sqrt(p))+1)] ]
```
下一题
利用列表推导，生成[1,100]以内所有不能被 3 整除的数
参考答案：
```
[x for x in range(1,101) if x%3!=0]
```
注意：是 101，不是 100。!=0 也可以省略。
下一题
利用列表推导，生成 100 个[1,100]以内随机整数
参考答案：
```
[ random.randint(1,100) for i in range(100)]
```
注意：randint(a,b)可以产生 b
下一题
利用列表推导，生成 50 个[1,100]以内随机偶数
有一些同学用 if 过滤条件，是不对的。保留了第偶数次随机生成的数字，并不是随机偶数
参考答案：
```
[ 2*random.randint(1,50) for i in range(50)]
```
或者
```
[ random.choice(range(2,101,2)) for i in range(50)]
```
或者
```
[ random.randrange(2, 101, 2) for i in range(50)]
```
最后一个 randrange 函数没讲过，但很好用。
以下是抽查环节，答对者平时成绩加 1～3 分（视题目难度）
抽到的同学先回复“在”，再开始回答
x = [3, 4.0, 1.5e1, '11']，对其排序的结果是？
正确答案：报错，字符串和浮点数无法比较大小
下一题
执行以下语句
```
x = (3, 4.0, 1.5e1, 11)
x.sort()
print(x)
```
的结果是？
正确答案：报错，元组是不可变对象，无法原地排序修改
下一题

执行以下语句
```
x = (3, 4.0, 1.5e1, 11)
x=sorted(x)
print(x)
```
的结果是？
正确答案：[3,4.0,11,15.0]
注意：sorted 总是得到列表
下一题
执行以下语句
```
x = [3, 4.0, 1.5e1, 11]
print(x.sort())
```
的结果是？
正确答案：None
注意：原地修改的函数大多没有返回值
下一题
执行以下语句
```
x = [3, 4.0, 1.5e1, 11]
x.sort(reversed = True))
print(x)
```
的结果是？
正确答案：错误。正确写法 x.sort(reverse = True)
最后一题
执行以下语句
```
x = [3, 4.0, 1.5e1, 11]
print(sorted(x, reverse = True))
print(x)
```
的结果是？
正确答案：
[15.0, 11, 4.0, 3]
[3, 4.0, 15.0, 11]

第 5 周

执行以下语句：
x = {1:100, 2:200}
a = input('Key:')
b = input('Value:')
假设用户键盘分别输入 1 和 123。执行下述语句的输出结果是？
print(x[a])
正确答案：报错。注意 input 的结果是字符串
回到出错之前
print(x.get(a))
正确答案：None
print(x.get(a,'Wrong key'))
正确答案：Wrong key（注意：print 字符串不输出定界符引号）
最后一题
x[a] = b
print(x)
正确答案：{1: 100, 2: 200, '1': '123'}（注意：三个元素的顺序可能与真实打印结果不同，因为字典无序）
这部分的内容以字典用法为重点，特别是访问字典元素要考虑元素是否存在。

第 6 周

已经执行如下代码完成键盘输入三个不同大小的浮点数，请输出最大的一个。必须使用单分支来完成。

```
a = float(input('The 1st float is:'))

b = float(input('The 2nd float is:'))

c = float(input('The 3rd float is:'))
```

注意：不能使用 max 函数，否则就没必要用分支语句了。

参考答案：

```
if b > a:

    a = b

if c > a:

    a = c

print('The max is:', a)
```

下一题。同样的要求，必须使用多分支来完成。
参考答案：

```
if a > b and a > c:

    m = a

elif b > c:

    m = b

else:

    m = c

print('The max is:', m)
```

或者

```
if b > a and b > c:
    a = b
elif c > a and c > b:
    a = c
print('The max is:', a)
```

下一题。同样的要求，必须使用嵌套分支来完成。
参考答案：

```
if a > b:
    if a > c:
        m = a
    else:
        m = c
else:
    if b > c:
        m = b
    else:
        m = c
print('The max is:', m)
```

或者

```
if a > b and a > c:
    m = a
else:
```

```
    if b > c:
        m = b
    else:
        m = c
print('The max is:', m)
```

最后一题是拓展题，不要求必须掌握。
这个例子参考答案中的单分支、多分支、嵌套分支解法，哪种执行效率最高？
执行效率高是指计算简单、速度快。

这题确实比较难。不能只定性讨论，需要定量分析才能明确答案。

答案解析：

解法	平均比较次数（考虑短路）	平均赋值次数	平均逻辑与次数
单分支	2	1/3*0+1/3*1+1/6*1+1/6*2=5/6	0
多分支 1	1/3*2+1/3*2+1/6*3+1/6*2=13/6	1	1
多分支 2	1/3*2+1/3*2+1/6*4+1/6*3=15/6	1/3*0+1/3*1+1/3*1=2/3	1/3*2+1/3*1+1/6*2+1/6*2=4/3
嵌套分支 1	2	1	0
嵌套分支 2	1/3*2+1/3*2+1/6*3+1/6*2=13/6	1	1

所以，单分支执行效率最高。注意：只是针对这题而言，并不具有普遍性。

abc 按大小排列共有 6 种情况，按概率分别计算，可以得到上表的定量分析。
单分支如果写的不简练，也可能不如嵌套分支。

第 7 周

如果写代码故意使用死循环的形式，请叙述如何避免真的发生死循环。
参考答案：配合 if...break

如果运行程序发生死循环，应如何解决？
参考答案：Ctrl+C

请叙述循环中执行 break 和 continue 效果上的区别。
参考答案：break 用于结束整个循环，continue 用于结束当前迭代。

while 循环中使用 continue 要避免发生什么错误？
参考答案：注意避免发生死循环。注意涉及循环条件的变量修改不要被 continue 忽略掉。

在例题“水仙花数”中，分解法和组合法哪种执行效率更高些？为什么？
参考答案：
组合法。
分解法使用一重循环，迭代 900 次。组合法使用三层循环嵌套，最内层也是 900 次，但是部分代码可以外提。像百位数乘 100，十位数乘 10，不用做 900 次。另外还有一个原因，计算机做除法差不多比乘法慢一倍。

最后一题
假设正整数 n 已经赋值，请默写判断 n 是否为素数的代码，是输出 1，否输出 0。
参考答案：

```
for k in range(2,n):
    if n%k==0:
        print(0)
        break
else:
    print(1)
```

第 8 周

根据下面的输入输出样例（前两行的冒号后面为输入，其余为输出），利用.find()方法，完成在源字符串中查找目标字符串全部出现位置。

```
Please input source string:asdf qaz ssaad
Please input object string:a
0
6
11
12
```

参考答案：

```
str=input('Please input source string:')
s=input('Please input object string:')
pos=-1
while True:
    pos=str.find(s,pos+1)
    if pos==-1:
        break
    print(pos)
```

思路是：第一次的查找开始位置利用初始化特别处理，后面就一致处理了。

第二题，也是最后一题。

根据下面的输入输出样例（前两行的冒号后面为输入，其余为输出），完成把源字符串中指定字符串（长度不一定是 1）的连续出现缩减为只剩一个。

```
Please input source string:qwer,,asd,zxcv,,,
Please input object string:,
qwer,asd,zxcv,
```

参考答案：

```
str=input('Please input source string:')
s=input('Please input object string:')
s2=s+s
while True:
    if str.find(s2)==-1:
        break
    str=str.replace(s2,s)
print(str)
```

第 9 周

已知字符串 s 中有若干学号，学号是 11 位数字。其中第 3 位数字有特定含义，1 表示博士，2 表示硕士，3 表示本科。请从中得到所有本科生学号，存放到列表 a 中。假设 re 模块已导入。

参考答案：
a=re.findall('\d{2}3\d{8}',s)
\d 是元字符不是转义符，双反斜杠的写法是错的

已知字符串 s 中有若干经纬度信息，其中经纬度格式是：先纬度 N 或 S，跟一个浮点数表示度数，然后是经度 E 或 W，同样跟一个浮点数。浮点数的小数位数不确定，至少 1 位小数。例如：N39.853E118.00。请找出 s 中所有的经纬度信息，存放到列表 a 中。假设 re 模块已导入。

参考答案：
a=re.findall('[NS]\d+\.\d+[EW]\d+\.\d+',s)

最后一题
已知字符串 s 中有若干日期和气温信息，每个信息的形式如：‘2022-04-21 气温：25.6 度’。请提取出其中年月日温度信息，以四元组的形式存放在列表 a 中。例如上面的字符串得到的结果是[('2022', '04', '21', '25.6')]。假设 re 模块已导入。

参考答案：
a=re.findall('(\d{4})-(\d{2})-(\d{2})气温：(\d+\.\d+)度',s)

第 10 周

请说出文件“r”方式打开失败的可能原因。
参考答案：指定位置没有该名字的文件，读写互斥，无访问权限

文件“r+”方式打开失败的可能原因与结果，是否和“r”方式失败一样？
参考答案：不完全一样。如果该文件已经有程序以读方式打开了，可以再次以读方式打开，但不能再用“r+”（既读又写）方式打开，因为读写互斥。另外也可能因为对该文件没有写权限而打开失败。

请说出文件“w”方式打开失败的可能原因。
参考答案：存储介质不可用（优盘拔了，网络断了，磁盘满了），无指定文件夹，读写互斥，无权限。

请说出文件打开方式“w”和“a”的异同。
参考答案：w 和 a 都是只进行写的操作，若文件不存在都会创建新文件。不同点是：w 会清空文件中原来的内容，a 默认在原来文件的最后追加内容。

最后一题
请说出文件打开方式“r+”和“w+”的异同。
参考答案：相同点是如果成功都是把文件指针定位在开头，既能读又能写。区别是：若文件不存在，r+出错，w+会创建文件；若打开成功，w+会清空原有内容。

第 11 周

执行以下语句

```
def f(a):
    a=[1,2]
b=1
f(b)
print(b)
```

的结果是？

答案：1

执行以下语句

```
def f(a):
    a=1
b=[1,2]
f(b)
print(b)
```

的结果是？

答案：[1,2]。虽然实参是可变的，但是函数里形参不是局部修改，形参整体改变不影响实参

最后一题

执行以下语句

```
def f(a=0):
    if a==0:
        a=[]
    else:
        a=[1]
    print(a)
b=0
f(b)
b=1
f(b)
f()
```

的结果是？

答案：

```
[]
[1]
[]
```

第 12 周

执行以下语句

```
a=1
def f():
    print(a)
    a=2
print(a)
```

的结果是？

答案：1。定义函数并不意味着马上就执行了这个函数，缺少调用。

执行以下语句

```
a=1
def f():
    print(a)
    a=2
f()
print(a)
```

的结果是？

答案：出错。因为函数内部有局部变量 a 的定义，所以整个函数体都不在全局 a 的作用域内。函数内的局部 a 先使用后定义，所以出错。

执行以下语句

```
a=1
def f(a):
    print(a)
    a=2
f(a)
print(a)
```

的结果是？

答案：

1

1

形参等同于局部变量。

执行以下语句

```
a=1
def f(a):
    print(a)
    a=2
    return a
a=f(a)
print(a)
```

的结果是？
答案：
1
2
不可变类型的局部变量修改，一般不影响全局的实参，但是可以通过把返回值赋值给实参来修改它。

最后一题
执行以下语句

```
a=1
def f():
    a=2
    print(a)
    return a
a=f()
print(a)
```

的结果是？
答案：
2
2

第 13 周

这周的学习内容是“异常处理”。对于短程序来说，异常处理不是必要的。但是对于交付实际使用的程序来说，是否有周密的异常处理体现了编程者的水平。

对于如下程序结构

```
try:
    try_block
except A:
    block1
except B:
    block2
else:
    block3
finally:
    block4
block5
```

如果在 try_block 中未发生异常，哪几个程序块 block 会被执行？

答案：block3，block4，block5

下一题

如果在 try_block 中发生了异常类型 A，哪几个程序块 block 会被执行？

答案：block1，block4，block5

下一题

如果在 try_block 中发生了异常类型 C，哪几个程序块 block 会被执行？

答案：只有 block4。异常未被捕获，异常对象将继续向外围飞去，被 Python 解释器捕获，程序崩溃进程被杀掉，所以 block5 不会做，但是 finally 子句还是要过一下的。

下一题

对于如下程序结构

```
try:
    try_block
except A:
    block1
except B:
    block2
except:
    block3
else:
    block4
finally:
```

```
    block5
block6
```
如果在 try_block 中发生了异常类型 C，哪几个程序块 block 会被执行？
答案：block3，block5，block6。有了一个能捕获所有类型异常的子句，异常被处理了，6 就会被执行。

最后一题
对于如下程序结构
```
try:
    try_block
except A:
    block1
except B:
    block2
except:
    block3
    raise
else:
    block4
finally:
    block5
block6
```
如果在 try_block 中发生了异常类型 C，哪几个程序块 block 会被执行？
答案：block3，block5。因为 raise 语句二次抛出异常，所以 6 不会被执行。如果这段代码外围没有 try 结构了，那么程序崩溃。