GESP Python 四级认证试卷

2023 年 06 月

（满分：100 分　考试时间：90 分钟）

学校：_____　　姓名：_____

题目	一	二	三	总分
得分				

一、单选题（每题 2 分，共 30 分）

题号	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
答案	D	C	D	A	B	D	B	D	C	D	C	C	A	A	A

1. 高级语言编写的程序需要经过以下（ ）操作，可以生成在计算机上运行的可执行代码。

A. 编辑

B. 保存

C. 调试

D. 编译

2. 排序算法是稳定的（Stable Sorting），就是指排序算法可以保证，在待排序数据中有两个相等记录的关键字 R 和 S（R 出现在 S 之前），在排序后的列表中 R 也一定在 S 前。下面关于排序稳定性的描述，正确的是（ ）。

A. 冒泡排序是不稳定的

B. 插入排序是不稳定的

C. 选择排序是不稳定的

D. 以上都不正确

```
a, b, c = 10, 20, 30
def f(a, b):
    a += 2
    c = a * b
    return c
d = f(b, a)
print(a, b, c, d, sep = "#")
```

3. 下面代码执行后输出是（ ）。

A. 12#20#220#220

B. 20#10#220#220

C. 20#10#30#220

D. 10#20#30#220

```
def ADD(a):
    a = a+10
    return a

a = 100
b = ADD(a)
print(a,b)
```

4. 下面代码执行后输出是（ ）。

A. 100, 110

B. 110, 110

C. 100, 100

D. 110, 100

5. 下面代码执行后输出是（ ）。

A. [1, 2, 3] [1, 2, 3]

B. [1, 2, 3, 10] [1, 2, 3, 10]

C. [1, 2, 3] [1, 2, 3, 10]

D. [1, 2, 3, 10] None

```
def ADD(a):
    a.append(10)
    return a

a = [1, 2, 3]
b = ADD(a)
print(a,b)
```

```
def ADD(a):
    a = a.append(10)
    return a

a = [1, 2, 3]
b = ADD(a)
print(a,b)
```

6. 下面 Python 代码执行后输出是（ ）。

A. [1, 2, 3] [1, 2, 3]

B. [1, 2, 3, 10] [1, 2, 3, 10]

C. [1, 2, 3] [1, 2, 3, 10]

D. [1, 2, 3, 10] None

```
def AVG(Nums):
    #分别保存求和值和成员数量
    Sum,numCount=0,0
    for everyNum in Nums:
        Sum+=everyNum
        numCount+=1
    return Sum/numCount
```

7. 下面 Python 代码所定义函数 AVG()的时间复杂度是（ ）。

A. $O(1)$

B. $O(n)$

C. $O(n+4)$

D. $O(n^2)$

8. 下面 Python 代码的功能是在 dictA 中保存形如{12:[1, 2, 3, 4, 6, 12]}数据，即键为

```
dictA = {} #保存1-100每个数对应的因数。
for i in range(1,100+1):
    for j in range(1,i+1):
        if i % j == 0:
            dictA[i] =
_____________________________
print(dictA)
```

1-100 中的每个数，值为其对应的因数，横线处应填入代码是（ ）。

A. dictA.get(i,[]).append(j)

B. dictA.get(i,list).append(j)

C. dictA.get(i,list())+ (j)

D. dictA.get(i,[])+[j]

9. 下面 Python 代码中的 dictA 是 1-100 所有数及其对应的因数，此处仅列出其中一部分。

```
dictA = {1:(1,),2:(1,2),3:(1,3),4:
(1,2,4),5:(1,5),6:(1,2,3)}
print({_____________________________})
```

在横线处填上合适代码，实现只输出质数对应的因数（ ）。

A. itm for itm in dictA if len(itm) > 2

B. K: V for K, V in dictA if len(V) > 2

C. K: V for K, V in dictA.items() if len(V) > 2

D. K: K.value for K in dictA if len(dictA[K]) > 2

10. 要打开一个已经存在的文件并在末尾处续写新的内容，则打开模式应该设定为（ ）。

A. w

B. w+

C. r+

D. a+

```
try:
    m = int(input())
    n = int(input())
    res = m/n
except ZeroDivisionError:
    print(1, end = "#")
except:
    print(2, end = "#")
else:
    print(3, end = "#")
finally:
    print(4, end = "#")
```

11. 下列 Python 代码执行时如果输入 3.14，将输出的是（ ）。

A. 2#

B. 1#4#

C. 2#4#

D. 2#3#4#

```
m, n, *_ = map(int,input().split(","))
print(m + n)
```

12. 有关下面 Python 代码的说法正确的是（ ）。

A. 如果输入 1.414,1.732,2.236 也将被执行，输出为 2

B. 如果输入 1,2,3,4,5 将会因输入过多而报错

C. 如果输入 1,2 也将会执行，输出 3

D. 程序存在语法错误，原因是*_

13. 下面 Python 代码程序用于复制图片文件，文件名仅为示例，文件位置假设正确，横线处分别应填入（ ）。

A. rb wb

B. r w

C. r+ w+

D. a+ a+

```
sourceFile = open("CCF.png",
"________")
sourceData = sourceFile.read()
sourceFile.close()

targetFile = open("CCF_bak.png",
"________")
targetFile.write(sourceData)
targetFile.close()
```

14. 在下面 Python 代码中，lst 数据如代码所示，但仅包含部分数据此处为示例。要求实现

```
lst = [(16,178),(16,182),(17,172),
(15,191),(16,175)]
lst.sort(________________)
print(lst)
```

第 1 个按升序第 2 个数按降序，在横线处填上合适代码（ ）。

A. key = lambda x:(x[0],-x[1])

B. lambda x:(x[0],x[1])

C. key = lambda x:(-x[0],-x[1])

D. 不需要填入代码，按默认排序即可

```
def Fx(a=10,b):
    return a*b
print(Fx("1",2))
```

15. 有关下面 Python 代码的说法，正确的是（ ）。

A. 函数 Fx()定义错误，交换 a 和 b 参数的先后顺序，程序执行后将输出 11

B. 上述代码执行后将输出 2

C. 上述代码执行后将输出 11

D. 在 Fx()函数定义中，a 被定义为整数，因此传入字符串将导致错误，因此第 3 行函数调用后将报错

二、判断题（每题 2 分，共 20 分）

题号	1	2	3	4	5	6	7	8	9	10
答案	X	√	√	√	X	X	X	X	√	√

1. 域名是由一串用点分隔的名字来标识互联网上一个计算机或计算机组的名称，CCF 编程能力等级认证官方网站的域名是 gesp.ccf.org.cn，其中顶级域名是 gesp。

2. 数列 1, 1, 2, 3, 5, 8 … 是以意大利数学家列昂纳多·斐波那契命名的数列，从第三个数开始，每个数是前面两项之和。如果计算该数列的第 n 项（其中 n>3） fib(n)，我们采用如下方法：① 令 fib(1)=fib(2)=1 ②用循环 for i=3 to n 分别计算 f(i) ③输出 fib(n)。这体现了递推的编程思想。

3. Python 列表的 sort()函数是稳定排序。

4. 冒泡排序算法额外空间需求是 $O(1)$，即额外所需空间为常数，与排序数据的数量没有关系。

5. {1}+{1}在 Python 中是合法的表达式。

6. 下面 Python 代码执行后输出了文件 abc.txt 前 10 个字符，但由于没有关闭文件，可能

```
with open("abc.txt","r") as rdFile:
    rdData = rdFile.read()
    print(rdData[:10])
```

存在风险。

7. 如下 Python 代码的第 1 行可被正常执行，该文件内容由中文英文和数字构成。程序执行后输出值与文件所占字节数相同。

8. 在 Python 中，文本文件不可以二进制方式读写。

9. 在与异常处理相关的关键字中，else 所属内容一定是不发生异常时才会被执行。

10. 根据下面 Python 函数定义，调用 Fx()函数时如果两个参数同为 int、float、tuple、list、

```
rdFile = open("xyz.txt","r")
print(len(rdFile.read()))
rdFile.close()
```

str 都不会报错。

```
def Fx(a,b):
    return a+b
```

三、编程题

第 1 题：幸运数

【问题描述】

1. 变化正整数的各个奇数位（从右到左，个位为第 1 位，奇数，16347），变化的规则是乘以 7，如果该奇数位与 7 相乘的结果大于 9 则各位数相加，如相加结果仍然大于 9 则继续各位数相加，直到结果不大于 9；如果该奇数位与 7 相乘的结果不大于 9 则该数为该奇数位变化结果。偶数位不发生变化。各个奇数位变化完毕后，将新数的各位数相加，如果相加之和是 8 的整数倍，则为幸运数；

2. 例如 16347，第 1 位 7 奇数位，乘以 7 结果为 49，大于 9 各位数相加为 13 大于 9 继续各位数相加最后结果为 4；然后变化第 3 位 3，第 5 位 1。最后变化结果为 76344，对于结果 76344 其各位数之和为 24，是 8 的倍数，为幸运数；
3. 首先输入 N，随后输入 N 行正整数。输出 N 行，对应 N 个正整数是否为幸运数，如是则输出 T 否则 F。

【输入描述】

1. 首先输入正整数 N 随后输入 N 行正整数，每个一个数。不考虑输入不合规情形，负数、负整数、非数字构成的字符串等。
2. **特别提示**：常规程序中，输入时好习惯是有提示。考试时由于系统限定，输入时所有 input()函数不可有提示信息。

【输出描述】

1. 输出 N 行，对应 N 个正整数是否为幸运数，如是则输出 T 否则 F。；
2. **特别提示**：注意输出字母为英文大写，小写或其他将判为错误。

【样例输入 1】

```
2
16347
76344
```

【样例输出 1】

```
T
F
```

【参考程序】

```
#幸运数

N = int(input()) #确定将输入 N 个数

for i in range(N):

    M = int(input()) #输入的数存入 M



    tnt = 0 #各位数相加之和

    loc = 0 #位置
```

```
while M != 0:
    loc += 1

    if loc % 2 == 1:
        tmp = M % 10 * 7

        while tmp > 9:
            tmp = tmp // 10 + tmp % 10

        tnt += tmp

        #print(loc, tmp, tnt)
    else:
        tnt += M % 10
        #print(loc, M%10, tnt)

    M //= 10

if tnt % 8 ==0:
    print("T")
else:
    print("F")
```

第 2 题：图像压缩

【问题描述】

1. 灰度图像有 256 级灰阶，编码 00-FF，对应 0-255，即图像有很多点，每个点取值是 00 到 FF。编程压缩到 16 级灰阶，对应 0-F。
2. 压缩规则：统计出每种灰阶的数量，取数量最多的前 16 种灰阶（如某种灰阶的数量与另外一种灰阶的数量相同，则以灰阶值从小到大为序），分别编号 0-F。其他灰阶转换到最近的 16 种灰阶之一，将某个点灰阶数与 16 种灰阶种的一种相减，绝对值最小即为最近。
3. 输入：多行数据，每行数据等长，每两个字符构成一个点，十六进制；输出：首先连续最多输出 16 种灰阶编码，不足 16 种灰阶就按实际输出；然后各行输出压缩后的编码，每行等长。

【输入描述】

1. 第 1 次输入正整数 n，表示有多少行数据。
2. 随后输入 n 行数据。
3. **特别提示**：常规程序中，输入时好习惯是有提示。考试时由于系统限定，输入时所有 input()函数不可有提示信息。

【输出描述】

1. 首先输出 16 种灰阶编码，共计 32 个字符。不足 16 种按实际输出。输出数量最多的 16 种灰阶，从多到少；如某种灰阶的数量与另外一种灰阶的数量相同，则以灰阶值从小到大为序；
2. **特别提示**：注意输出字母为大写，小写将判为错误，数本身与字母 T 和 F 之前没有空格。

【样例输入 1】

```
10
00FFCFAB00FFAC09071B5CCFAB76
00AFCBAB11FFAB09981D34CFAF56
01BFCEAB00FFAC0907F25FCFBA65
10FBCBAB11FFAB09981DF4CFCA67
00FFCBFB00FFAC0907A25CCFFC76
00FFCBAB1CFFCB09FC1AC4CFCF67
01FCCBAB00FFAC0F071A54CFBA65
10EFCBAB11FFAB09981B34CFCF67
01FFCBAB00FFAC0F071054CFAC76
1000CBAB11FFAB0A981B84CFCF66
```

【样例输出 1】

ABCFFF00CB09AC07101198011B6776FC
321032657CD10E
36409205ACC16D
B41032657FD16D
8F409205ACF14D
324F326570D1FE
3240C245FC411D
BF4032687CD16D
8F409205ACC11D
B240326878D16E
83409205ACE11D

【参考程序】

```
#图像压缩

def composePixel(topDict,srcVal):

    if srcVal in topDict:

        return hex(topDict[srcVal])[2:].upper()

    else:#不在列表中，计算最近距离

        nearest = 255

        nearestKey = "FF"

        for k,v in topDict.items():

            tmpDistance = abs(int(srcVal,16)-int(k,16))

            if tmpDistance < nearest:

                nearest = tmpDistance

                nearestKey = v

        return hex(nearestKey)[2:].upper()


N = int(input())
```

```
dictDot = {}

rawData = []

newData = []

for i in range(N):

    dataLine = input()

    rawData.append(dataLine)

    charCount = len(dataLine)

    groupCount = charCount //2

    for i in range(groupCount):

        twoHex = dataLine[i*2:i*2+2] #两个十六进制字符

        dictDot[twoHex] = dictDot.get(twoHex,0) + 1


top16 = sorted(dictDot.items(), key = lambda x: (x[1],-int(x[0],16)), reverse = True)[:16]


top16Dict = {k:x for x,(k,v) in enumerate(top16)}

for dataLine in rawData:

    groupCount = len(dataLine) // 2

    tmp = ""

    for i in range(groupCount):

        tmp += composePixel(top16Dict,dataLine[i*2:i*2+2])

    newData.append(tmp)


for top in top16:
```

```
        print(top[0],end="")

print()

for everyLine in newData:

        print(everyLine)
```