2023 年 6 月 GESP Python 四级试卷解析

CCF 编程能力等级认证，英文名 Grade Examination of Software Programming（以下简称 GESP），由中国计算机学会发起并主办，是为青少年计算机和编程学习者提供学业能力验证的平台。GESP 覆盖中小学全学段，符合条件的青少年均可参加认证。GESP 旨在提升青少年计算机和编程教育水平，推广和普及青少年计算机和编程教育。

GESP 考察语言为图形化（Scratch）编程、Python 编程及 C++编程，主要考察学生掌握相关编程知识和操作能力，熟悉编程各项基础知识和理论框架，通过设定不同等级的考试目标，让学生具备编程从简单的程序到复杂程序设计的编程能力，为后期专业化编程学习打下良好基础。

本次为大家带来的是 2023 年 6 月份 Python 四级认证试卷真题解析。

一、单选题（每题 2 分，共 30 分）

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

   【答案】D

   【解析】本题考察基本概念，编译程序和高级语言是计算机编程中的两个概念，高级语言是一种人类易于理解的编程语言，使用高级语言编写的程序通常不会直接在计算机上执行，它们需要通过一个称为编译器的程序将其转换为计算机可以理解的机器语言，也就是 0 和 1 的二进制代码，这个过程被称为编译。

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

【答案】C

【解析】考察排序算法和基础概念，我们通过选择排序的模拟过程或代码可知，选择排序不能保证相同数据在排序后仍然保持排序前位置，是不稳定的，冒泡排序和插入排序能保证相对位置不变，故选择 C 选项。

3. 下面代码执行后输出是（ ）。

```
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 = "#")
```

A. 12#20#220#220

B. 20#10#220#220

C. 20#10#30#220

D. 10#20#30#220

【答案】D

【解析】本题考察自定义函数和局部、全局变量内容，首先是将 20 和 10 传入自定义函数，分别被形参 a 和 b 接收，在函数内部完成的运算不会影响外部的变量，只有返回结果被 d 接收，结果为 220，所以输出出来的是原来是的初始数值，使用#隔开，故选择 D 选项。

4. 下面代码执行后输出是（ ）。

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

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

A. 100, 110

B. 110, 110

C. 100, 100

D. 110, 100

【答案】A

【解析】本题考察自定义函数和变量作用域知识，首先 100 作为实参传入自定义函数，内部运算加 10 后返回被 b 接收，需要注意的是自定义函数内部 a 变化不会影响到外部的 a，所以 a 还是 100，b 是 110，输出的是 100 和 110，故选择 A 选项。

5. 下面代码执行后输出是（ ）。

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

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

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

【答案】B

【解析】本题考察自定义函数中列表作为参数知识点，列表作为参数具有特殊性，相当于传引用，与普通变量不同的是自定义函数内部对列表的改变，会真正的改变列表本身，所以在函数内部对列表 a 追加了一个 10，会影响改变到 a 本身，a 也变成[1,2,3,10]，b 直接接收返回结果，也被赋值为[1,2,3,10]，故选择 B 选项。

6. 下面 Python 代码执行后输出是（ ）。

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

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

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

【答案】D

【解析】本题考察自定义函数中列表作为参数知识点，还有 append 函数是否具有返回值，本题代码和前一题的区别是把 a.append(10)这个操作的返回值用 a 接收，并且返回，被 b 接收，然而 append 追加这个操作是没有返回值的，只有作用效果，所以 a 接收到的是 None，所以在打印输出的时候是被改变的列表 a 和 None，故选择 D 选项。

7. 下面 Python 代码所定义函数 AVG()的时间复杂度是（ ）。

```
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
```

A. $O(1)$

B. $O(n)$

C. $O(n+4)$

D. $O(n^2)$

【答案】B

【解析】本题考察算法的时间复杂度概念，在不同的算法中，若算法语句执行次数为常数，则时间复杂度为 O(1)，观察图中程序可知该算法是线性算法，和问题规模 n 相关，随着问题规模 n 不断增大，时间复杂度不断增大，时间复杂度为 O(n)，故选择 B 选项。

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 1-100 中的每个数，值为其对应的因数，横线处应填入代码是（ ）。

```
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)
```

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

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

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

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

【答案】D

【解析】本题考察字典和双层循环知识点，首先是外层循环的循环变量 i，作为字典的键，通过内层循环和 if 语句，找到 i 的因子，通过 get 函数获取 i 对应的值，后面的空列表含义为：如果指定的键的值不存在，返回该默认值。用获取到的列表一直和最新的因子 j 做列表相加，更新改键的值，也就是外层循环内循环一次，就能找打一个数字的所有因子，故选择 D 选项。

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。在横线处填上合适代码，实现只输出质数对应的因数（ ）。

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

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

【答案】C

【解析】本题考察字典键值对的操作，C 选项是正确的，通过 items()函数遍历字典，k 遍历出所有的键，v 遍历所有的值，这里我们找质数，质数只有 1 和本身所有只有两个值，len(v)==2 的就是我们要找的！A 选项直接对字典名遍历时，获取到的是字典的键，这里键是整数获取不到 len()的值程序报错，B 选项用 k,v 两个值直接遍历字典程序会报错，D 选项调用 k.value 会报错，k 是整数没有 value 这个属性，答案选择 C。

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+

【答案】D

【解析】本题考察 python 中的文件操作模式，为概念性题目，其中 w 为只写模式、w+为写读模式、r+为可读可写模式，a+为追加读写模式，故选择 D 选项。

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

```
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 = "#")
```

A. 2#

B. 1#4#

C. 2#4#

D. 2#3#4#

【答案】C

【解析】本题考察 python 中的异常处理，各个语句代码块什么情况下执行需要理清思路，首先本题输入 3.14 会报错，错误类型不是 0 作为除数，所以代码执行 except 代码块，输出 2#。try 代码块有错，else 代码块不会执行。最后的 finally 代码块不受限制会执行，输出 4#，故最终打印结果为 2#4#，选择 C 选项。

12. 有关下面 Python 代码的说法正确的是（ ）。

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

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*_

【答案】C

【解析】本题考察 map 函数用法和带有*的变量，本格式中假设输入 1，2，3，4，5 的话，经过 split 函数切割后，结果为：["1", "2", "3", "4", "5"]，经过 int 转换后结果为：[1, 2, 3, 4, 5]，但是结果是一个可迭代数列，其中前两个变量会被 m 和 n 接收，其余数字会被*_这个变量压缩接收，效果如同学们学过的带*的自定义函数参数，在输出的时候被解压缩输出，A 选项因为浮点型不能被 int 转换，所以会报错。B 选项不会报错，数据再多一些也不会报错。不存在语法错误，D 选项错误。故选择 C 选项。

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（ ）。

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

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

A. rb wb

B. r w

C. r+ w+

D. a+ a+

【答案】A

【解析】本题考察文件操作，读写模式知识点，题目描述为图片文件，图片文件为二进制格式，所以在复制过程中，读取和写入过程应使用 rb 和 wb 模式，故选择 A 选项。

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。要求实现第 1 个按升序第 2 个数按降序，在横线处填上合适代码（ ）。

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

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

【答案】A

【解析】本题考察列表排序函数 sort，sort 函数直接对列表元素进行排序，当指定 key 参数时，列表.sort 函数会使用该参数指定的函数对列表中的每个元素进行处理，并根据处理后的结果进行排序。排序将根据处理后的结果来确定元素的顺序，而不是直接比较原始的元素本身。而 lambda 函数表示创建匿名函数，没有函数名，只有参数列表和函数体，先指定元组中的第 0 位 x[0]，再指定 x[1]，因为数字越大，加上负号之后对应的数字越小，相当于也是从小打大，所以写法-x[1]，故选择 A 选项。

15. 有关下面 Python 代码的说法，正确的是（ ）。

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

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

【答案】A

【解析】本题考察自定义函数的定义、参数传递等知识，首先由形式参数的书写顺序可知函数定义错误，因为带有默认值的形参需要写在没默认值参数的后面，交换顺序后 b 接收的是字符串 1，a 接收的是 10，数字和字符串乘法，所以结果为 11，B 选项错误。需要改正才能出结果，C 选项错误，D 选项也错误。故选择 A 选项。

二、判断题（每题 2 分，共 20 分）

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。

【答案】错误

【解析】本题考察域名相关概念，顶级域名、二级域名、三级域名等叫法，本题域名中.cn 为顶级域名，而不是 gesp，故说法错误。

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。

【答案】正确

【解析】本题考察递推的概念，斐波那契数列是我们在学习递推知识点时经典题目，由前面的已知项逐步推出后面的项，体现了递推思想，故说法正确。

3. Python 列表的 sort()函数是稳定排序。

【答案】正确

【解析】本题考察 sort 函数相关知识和算法稳定概念，sort 函数的内部实现机制为 Timsort，是结合了合并排序和插入排序而得出的排序算法，该算法是稳定的，不改变相同元素的前后关系，故说法正确。

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。

【答案】正确

【解析】本题考察排序算法中的冒泡排序和空间复杂度概念，通过分析冒泡排序的原理，可知冒泡排序在排序的过程中，不需要占用很多额外空间，只在交换元素的时候需要临时变量存储，这里需要的额外空间开销是常量级的，因此冒泡排序的空间复杂度为 O(1)，故说法正确。

5. {1}+{1}在 Python 中是合法的表达式。

【答案】错误

【解析】本题考察集合和运算符，集合之间可以做的操作有交集、差集、并集，补集、等，其中没有使用运算符+的，题目中的写法会报错，故说法错误。

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，可能存在风险。

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

【答案】错误

【解析】本题考察文件操作知识点，代码可以输出前 10 个字符，但是代码中采用的是 with open 方法，会自动关闭，不存在风险，故题目中说法错误。

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。

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

【答案】错误

【解析】本题考察文本长度和所占字节数不是同一个概念，一个字符不等于一个字节，故说法错误。

8. 在 Python 中，文本文件不可以二进制方式读写。

【答案】错误

【解析】本题考察文件操作知识点，文本文件可以使用二进制方式读写，只需要在代码中采用正确的二进制读写方式即可，故说法错误。所有文件都可以按二进制读写。

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。

【答案】正确

【解析】本题考察 python 中的异常处理，如果 try 代码块正常执行，不会执行 except 块，而后的 else 和 finally 块都会被执行，故说法正确。

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、str 都不会报错。

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

【答案】正确

【解析】本题考察自定义函数和数值据之间的运算，因为整型、浮点型、元组、列表、字符串都是可以两个同类型之间执行运算符+的，所以不会报错，故说法正确。

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")
```

【解析】本题考察数位拆分运算和循环的综合应用，首先接收 N 作为循环次数，在循环内写输入，分别接收 N 个数到 M 中，接收到之后，在当次循环内就处理并输出。在一次循环内 tnt 用于存储当前 M 的数位相加之和，最后判断是否为 8 的倍数，决定输出结果，loc 为处于 M 的第几位，首先是偶数位很好理解，因为偶数位不需要任何操作，走 else 直接取出个位，累加在 tnt 中。如果是奇数位则需要判断*7 之后是否大于 9，如果大于 9 就一直用 while 做个位+十位操作，直到不大于 9 为止，然后也累加到 tnt 中，最后将 M 的个位去除掉，loc 继+1，检测后面的数位，直到将当前的 M 的各位累加完成，输出对应的 T 或者 F，然后开始执行下一次 for 循环，直至将所有数字都判断完，分别输出是否为幸运数。

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)
```

【解析】本题考察进制转换、排序、字典及其函数等，首先根据输入的第一个数表示后面有多少行数据，所以先接收 N，后面 for 循环分别接收每组数据存放在列表中。遍历这一组数据，采用切片，把数据每两个十六进制字符分为一组，再以这两个十六进制字符为字典的键，以出现的次数作为值，用字典存储起来，这样就将每种灰阶出现的次数全部统计好了，然后使用 sorted 进行排序由大到小排序参数 reverse=True，把结果再切片，取前 16 个，所以写法是[:16]，即从下标 0 到下标 15，然后在排序的过程中遵循先按着出现次数由大到小，如果出现次数相同，则以灰阶自身的值由小到大为序，所以函数内部有-int(x[0], 16)写法，这样出现次数最多的 16 种灰阶就已经排序存储好了，最后遍历输出 top16，就可以成功打印输出结果的第一行（由大到小出现次数最多的 16 种灰阶）。接下来遍历前面输入的每组数据都存入的列表 rawData，将字典和两个一组十六进制数据为参数传入自定义行数

composePixel，并且将返回值存入列表 newData，最后遍历输出。在自定义函数中先检测传入进来的两个十六进制字符是否在字典中，如果在直接返回，如果不在，需要找到距离它本身最近的十六种灰阶之一进行输出，找的方法就是循环遍历字典，那一个灰阶和他相减后的绝对值最小（采用 abs 函数）就用 nearestKey 存储对应的值，最终作为返回值被 tem 接收，存入 newData。