人生苦短，我学 Python

矿大 Python 理论题库 v1.1 正式版

排版编校 胡椒 供题 矿大 python 研学群（464456244）全体小可爱

目录

1 Python 语言简介与基础

Python 语言的主网站网址 **A**

(A)https://www.python.org/
(B)https://www.pythonl23.org/
(C)https://pypi.python.org/pypi
(D)https://www.pythonl23.io/

Python 为源文件指定系统默认字符编码的声明是 **A**

(A)#coding:utf-8
(B)#coding:cp936
(C)#coding:GBK
(D)#coding:GB2312

不属于 IPO 模式的一部分是 **B**

(A)Input (B)Program
(C)Process (D)Output

下列选项中，不属于 Python 语言特点的是 C 面向对象的脚本语言

(A)简单易学
(B)开源
(C)面对过程
(D)可移植性

关于 Python 程序格式框架的描述，以下选项中错误的是 **A**

(A)Python 语言不采用严格的缩进来表明程序框架
(B)Python 语言的缩进可以采用 Tab 键实现
(C)Python 单层缩进代码属于之前最邻近的
一行非缩进代码，多层缩进代码根据缩进关系决定
(D)判断、循环、函数等语法形式能够通过缩进
包含一批 Python 代码，进而表达对应的语义

关于 Python 语言的注释，以下选项中描述错误的是 **C**

(A)Python 语言有两种注释方式：单行注释多行注释
(B)Python 语言的单行注释以#开头
(C)Python 语言的单行注释以'开头
(D)Python 语言的多行注释以'''（三个单引)开头和结尾

以下表达式输出结果为 11 的选项是 D

(A)print("1+1") (B)print(1+1)
(C)print(eval("1+1"))
(D)print(eval("1"+"1"))

关于 Python 赋值语句，以下选项中不合法的是 C

(A)x=1;y=1 (B)x=y=1

(C)x=(y=1) (D)x,y=y,x

以下哪项是无效声明 B

abc = 1,000,000 abc 是元组(1,0,0)

a b c = 1000 2000 3000

a,b,c = 1000,0000,0000 a 为 1000，b 和 c 为 0

a_b_c = 1,000,000 a_b_c 是元组(1,0,0)

以下选项中不符合 Python 语言变量命名规则的是 B

(A)I

(B)3_1

(C)_AI

(D)TempStr

给标识符关联名字的过程是 B

(A)生成语句 (B)命名

(C)赋值语句 (D)表达

给出如下代码：如下描述错误的是 B

```
for i in range (10):
    print(chr(ord("!")+i),end="")
```

(A)ord("!")返回"!"对应的 Unicode 编码

(B)系统报错

(C)chr(x)函数返回 Unicode 编码对应的字符

(D)输出结果为!"#$%&'()*

下面程序的输出结果第一行是 1

输出结果最后一行是 4

```
num = 4321
while num != 0:
    print(num % 10)
    num = num // 10
```

下面代码的执行结果是 Plguba

```
d = {}
for i in range(26):
    d[chr(i+ord("a"))] = chr((i+13) % 26 + ord("a"))
for c in "Python":
    print(d.get(c, c), end="")
```

关于 Python 字符编码，以下选项中描述错误的是 **D**

(A)Python 可以处理可字符编码文本
(B)Python 默认采用 Unicode 字符编码
(C)ord(x)和 chr(x)是一对函数
(D)chr(x)将字符转换为 Unicode 编码

下面代码的执行结果是 **56**

```
x = 2
x *= 3 + 5**2
print(x)
```

下面代码的执行结果是 **3.14 3**

```
x=3.1415926
print(round(x,2),round(x))
```

下面代码的执行结果是 **257**

```
x=0x0101
print(x)
```

下面代码的执行结果是 **A**

```
a=10.99
print(complex(a))
```

(A)(10.99+0j)　(B)0.99
(C)10.99　(D)10.99+0j

type(1+0xf*3.14)结果是 **D**

(A)<class 'int'>　(B)<class 'long'>
(C)<class 'str'>　(D)<class 'float'>

关于 Python 的复数类型，以下选项中描述错误的是 **B**

(A)复数类型表示数学中的复数
(B)对于复数 z,可以用 z.imag 获得实数部分
(C)对于复数 z,可以用 z.real 获得实数部分
(D)复数的虚数部分通过后缀"J"或"j"来表示

关于 Python 的浮点数类型，以下选项中描述错误的是 **C**

(A)浮点数类型与数学中实数的概念一致，表示带有小数的数值
(B)sys.float.info 可以详细列出
　Python 解释器所运行系统的浮点数各项参数
(C)Python 语言的浮点数可以不带小数部分
(D)浮点数表示方法:十进制表示和科学计数法

下列选项中输出结果是 True 的是 **A**

(A)isinstance(255,int)
(B)chr(10).isnumeric()
(C)'Python'.islower()
(D)chr(13).isprintable

isinstance() 函数来判断一个对象是否是一个已知的类型，类似 type()
isinstance() 与 type() 区别:
type() 不会认为子类是一种父类类型，不考虑继承关系。
isinstance() 会认为子类是一种父类类型，考虑继承关系。
如果要判断两个类型是否相同推荐使用 isinstance()
isnumeric() 方法检测字符串是否只由数字组成。这种方法是只针对 unicode 对象。
语法：str.isnumeric()
如果字符串中只包含数字字符，则返回 True，否则返回 False
isprintable() 方法，如果所有字符都是可打印的，则返回 True，否则返回 False。
不可打印的字符可以是回车和换行符。这里 chr(13)是'\r'

```
while True:
    guess=eval(input())
    if guess==0x452//2:
        break
```

作为输入能够结束程序运行的是 **B**

(A)break
(B)553
(C)0x452
(D)'0x452//2'

2 Python 控制结构

关于 Python 循环结构，以下选项中描述错误的是 A

(A)每个 continue 语句只有能力跳出当前层次的循环

(B)break 用来跳出最内层 for 或 while 循环
脱离该循环后程序从循环代码后继续执行

(C)遍历循环中的遍历结构可以是字符串、文件、
组合数据类型和 range()函数等

(D)Python 通过 for、while 等保留字提供
遍历循环和无限循环结构

下面代码输出结果是 **PYHON**

```
for s in "PYTHON":
   if s=="T":
      continue
   print(s,end="")
```

下面代码输出结果是 **Hello**

```
for s in "HelloWorld":
   if s=="W":
      break
   print(s,end="")
```

设计程序，计算阶乘 **[1] ==**

[2] n*Factorial(n-1)

[3] Factorial(n)

```
def Factorial(n):
    if n [1]1:
        fn=1
    else:
        fn=[2]
    return fn
n=int(input('请输入正整数：'))
print('结果为：',[3])
```

实现多路分支的最佳控制结构是 **D**

(A)if　(B)if-else

(C)try　(D)if-elif-else

关于分支结构，以下选项中描述不正确的是 **B if 和 else**

(A) if 语句中条件部分可以使用任何
能够产生 True 和 False 的语句和函数

(B) 二分支结构有一种紧凑形式，使用保留字 if 和 elif 实现

(C) 多分支结构用于设置多个判断条件以及对应的多条执行路径

(D) if 语句中语句块执行与否依赖于条件判断

下面程序的输出结果是 **6**

```
sum=0
for i in range(2,11):
    if i%2==0:
        sum+=i
    else:
        sum-=i
print(sum)
```

阅读下列程序，第一行的结果是$ $ *
第二行输出的结果是:【1】 **1 $ ***
最后一行输出的结果是:【2】 **2 *****

```
n=3
for i in range(0,n):
    for j in range(0,2-i):
        print('$',end=' ')
    for j in range(2*i+1):
        print('*',end='')
    print()
```

下面程序输出结果第一行是 **1 700.0/700./700**
输出结果第二行是 **2 4**

```
m=100
n=28
while (m!=n):
    if m>n:
        m=m-n
    else:
        n=n-m
print((100*28)/n)
print(m)
```

编写程序: **[1] random**
读取 7 个 1-50 的整数值 **[2] 7 或 0,7**
每读取一个值，程序打印出该值个数个*号 **[3] 1,50**
[4] m

```
import [1]
for i in range([2]):
    m=random.randint([3])
    print(m,'\n')
    for j in range([4]):
        print('*',end='')
    print('\n')
```

编写程序： **[1] num**
有 1,2,3,4 个数字，能组成多少个互不相同 **[2] c**
且无重复数字的三位数，分别是哪些数。 **[3] b!=c 或者 c!=b**
[4] sum

```
def main():
    num=[1,2,3,4]
    sum=0
    for a in [1]:
        for b in num:
            for [2] in num:
                if (a!=b) and ([3]) and (c!=a):
                    sum+=1
                    print(a,b,c)
    print('sum is',[4])
if __name__=='__main__':
    main()
```

下面代码的输出结果是 **A**

```
for a in ["torch","soap","bath"]:
    print(a)
(A)torch
    soap
    bath
(B)torch,soap,bath
(C)torch,soap,bath,
(C)torch soap bath
```

Python 异常处理中不会用到的关键字是 **C**

(A)try (B)finally (C)if (D)else

当用户输入 abc 时，输出结果是 **没有输出(函数没有被调用)**

```
try:
    n=0
    n=input('请输入一个整数：')
    def pow10(n):
        return n**10
except:
    print('程序执行错误')
```

用于异常处理结构中用来捕获特定类型异常的是 **D**

(A)def (B)pass (C)while (D)except

何时执行 try-except-else 的 else 部分 C

(A)总是 (B)发生异常时 (C)没有异常发生时 (D)没有 else 语法

3 字符串与组合数据类型

输出的结果是 **beh**

```
s= "abcdefghijklmn"
print(s[1:10:3])
```

阅读下列程序，第一行的结果是【1】 **1** **birth**
第二行输出的结果是:【2】 **2** **happy Birthday**

```
name='happy birthday'
print(name[6:11])
name1=name.replace(name[6],'B')
print(name1)
```

下面代码的输出结果是 **kelly**

```
li=["hello",'se',[["m","n"],["h","kelly"],'all'],123,456]
print(li[2][1][1])
```

下面代码的输出结果是 **1~~2~~3~~4**

```
L = [1,2,3,4]
s1 = '~~'.join(str(n) for n in L)
print (s1,'\n')
```

L.reverse()和 L[-1:-1-len(L):-1]的主要区别是 **D**

(A)L.reverse()和 L[-1:-1-len(L):-1]
　都将列表的所有元素反转排列,没有区别
(B)L.reverse()和 L[-1:-1-len(L):-1]
　都不会改变列表 L 原来内容
(C)L.reverse()不会改变列表 L 的内容,
　而 L[-1:-1-len(L):-1]会改变列表 L 原来内容
(D)L.reverse()会改变列表 L 的内容,
　而 L[-1:-1-len(L):-1]产生一个新列表,
　不会改变列表 L 原来内容

对于序列 s,能够返回序列 s 中第 i 到 j 以 h 为步长的元素子序列的表达是 **C**

(A)s[i,j,k]
(B)s[i;j;k]
(C)s[i:j:k]
(D)s(i,j,k)

字典 d={'abc ':123, 'def':456, ' ghi' :789} ,len(d)的结果是 **3**

输出的结果是 **A**

```
list1=[i*2 for i in 'Python']
print(list1)
```

(A)['PP', 'yy', 'tt', 'hh', 'oo', 'nn']
(B)Python Python
(C)错误
(D)[2,4,6,8,10,12]

输出的结果是 **5（非 0 即 True）**

```
if 2:
    print(5)
else:
    prlnt(6)
```

输出的结果是 **{1:'aa',2:'dd',3:'cc'}**

```
l1=[1,2,3,2]
l2=['aa','bb','cc','dd','ee']
d={}
for index in range(len(l1)):
    d[l1[index]]=l2[index]
print(d)
```

输出的结果是 **C**

```
s=1
while(s<=1):
    print('计数: ',s)
    s=s+1
```

(A)计数: 0
　　计数: 1
(B)出错　(C)计数: 1　(D)计数: 0

ls=[[1,2,3],[4,5,6],[7,8,9]]，以下能获取元素 5 的是 **A**

(A)ls[1][1]　　(B)ls[-2][-1]
(C)ls[-1][-1]　　(D)ls[4]

阅读下列程序，第一行的结果是【1】 **1 Alice**
第二行输出的结果是:【2】 **2 Bob**

```
names1=['Amy','Bob','Charlie','Daling']
names2=names1
names3=namesl[:]
names2[0]='Alice'
names3[1]='Ben'
for name in names1:
    print(name)
```

关于 Python 组合数据类型错误的是 **D**

(A)Python 的 str、tuple
和 list 类型都属于序列类型
(B)Python 组合数据类型能够将多个同类型
或不同类型的数据组织起来
通过单一的表示使数据操作更有序、更容易
(C)组合数据类型可以分为 3 类
序列类型、集合类型和映射类型
(D)序列类型是二维元素向量
元素之间存在先后关系，通过序号访问

关于 Python 的元组类型，以下选项中描述错误的是 **C**

(A)元组一旦创建就不能被修改
(B)元组可以作为另一个元组的元素
可以采多级索引获取信息
(C)元组中元素不可以是不同类型
(D)Python 中元组采用逗号和圆括号（可选）来表示

元组变量 t= ("cat","dog ","tiger","human") **A**

t[::-1]的结果是
(A)('human', 'tiger', 'dog', 'cat')
(B)['human', 'tiger', 'dog', 'cat']
(C){'human', 'tiger', 'dog', 'cat'}
(D)运行出错

关于 Python 字符串，以下选项中描述错误的是 **B type()**

(A)字符串可以保存在变量中，也可以单独存在
(B)可以使用 datatype()测试字符串的类型
(C)输出带有引号的字符串，可以使用转义字符\
(D)字符串是一个字符序列，字符串中的编号叫索引

tuple(range(2,10,2))的返回结果是() C

(A)[2, 4, 6, 8]
(B)[2, 4, 6, 8, 10]
(C)(2, 4, 6, 8)
(D)(2, 4, 6, 8, 10)

不是建立字典的方式是 **D 列表可变**

(A)d={1:[1,2],3:[3,4]}
(B)d={'Alice':1,'Bob':3}
(C)d={(1,2):1,(3,4):3}
(D)d={[1,2]:1,[3,4]:3}

给定字典 d，以下选项中对 d.get(x,y)的描述正确的是 **D**

(A)返回字典 d 中键值对为 x:y 的值
(B)返回字典 d 中值为 y 的值，如果不存在，则返回 x
(C)返回字典 d 中键为 y 的值，如果不存在，则返回 y
(D)返回字典 d 中键为 x 的值，如果不存在，则返回 y

以下关于 Python 字符串的描述中，错误的是 C

A.字符串是字符的序列，可以按照单个字符或者字符片段进行索引
B.字符串包括两种序号体系：正向递增和反向递减
C.字符串提供区间访问，[N:M]表示字符串中从包括 N 到包括 M 的索引子字符串
D.字符串是用一对双引号或者单引号括起来的零个或者多个字符

表达式",".join(ls)中 ls 是列表类型 D

以下选项中对其功能的描述正确的是
(A)将逗号字符串增加到列表 ls 中
(B)在列表 ls 每个元素后增加一个逗号
(C)将列表所有元素连接成一个字符串，每个元素后增加一个逗号
(D)将列表所有元素连接成一个字符串，元素之间增加一个逗号

以下关于列表和字符串的描述，错误的是 D

(A)列表使用正向递增序号和反向递减序号的索引体系
(B)列表是一个可以修改数据项的序列类型
(C)字符和列表均支持成员关系操作符（in）和长度计算函数（len()）
(D)字符串是单一字符的无序组合

4 函数

关于函数，以下选项中描述错误的是 **B def**

(A)函数是一段具有特定功能的、可重用的语句组
(B)Python 使用 del 保留字定义一个函数
(C)使用函数的主要目的是降低编程难度和代码重用
(D)函数能完成特定的功能，对函数的使用不需要了解
函数内部实现原理，只要了解函数的输入输出方式即可

关于函数的参数，以下选项中描述错误的是 **C 在最后**

(A)定义函数时，如果有些参数存在默认值
可以在定义函数时直接为这些参数指定默认值
(B)一个元组可以传递给带有星号的可变参数
(C)可选参数可以定义在非可选参数的前面
(D)在定义函数时，可以设计可变数量参数
通过在参数前增加星号*实现

Python 允许调用函数时通过名字来传入参数值，参数名不是无意义的
调用函数时，支持两种方式为参数指定值
1.位置参数：必须按顺序为每个参数指定参数值
2.关键字参数（命名参数）：按参数名为参数指定参数值，可读性更高
两种传参方式可以混合使用，但使用时关键字参数必须位于位置参数之后

关于函数，以下选项中描述错误的是 **A 不需要**

(A)函数使用时需要了解函数内部实现细节
(B)函数主要通过接口与外界通信，传递信息
(C)函数在需要时被调用，其代码被执行
(D)函数具有特定功能的可重用代码片段
实现解决某个特定问题的算法

关于函数的关键字参数使用限制，描述错误的是 **A**

(A)关键字参数必须位于位置参数之前
(B)关键字参数顺序无限制
(C)不得重复提供实际参数
(D)关键字参数必须位于位置参数之后

关于全局、局部变量，描述错误的是 **D**

(A)全局变量指在函数之外定义的变量
一般没有缩进，在程序执行全过程有效
(B)简单数据类型变量无论是否与全局变量重名
仅在函数内部创建和使用，函数退出后变量被释放
(C)使用 global 保留字声明简单数据类型变量后
该变量作为全局变量使用
(D)局部变量指在函数内部使用的变量，当函数退出时
变量依然存在，下次函数调用可以继续使用

关于函数局部变量和全局变量的使用规则，以下选项中描述错误的是 **D**

(A)对于基本数据类型的变量，无论是否重名，局部变量与全局变量不同
(B)可以通过 global 保留字在函数内部声明全局变量
(C)对于组合数据类型的变量，如果局部变量未真实创建，则是全局变量
(D)return 不可以传递任意多个函数局部变量返回值

关于递归函数的描述，以下选项中正确的是 C

(A)包含一个循环结构
(B)函数比较复杂
(C)函数内部包含对本函数的再次调用
(D)函数名称作为返回值

关于递归函数基例的说明，以下选项中错误的是 C 至少一个

(A)递归函数必须有基例
(B)递归函数的基例不再进行递归
(C)每个递归函数都只能有一个基例
(D)递归函数的基例决定递归的深度

以下选项中，不属于函数的作用的是 A

(A)提高代码执行速度
(B)复用代码
(C)增强代码可读性
(D)降低编程复杂度

假设函数中不包括 global 保留字 D

对于改变参数值的方法，以下选项中错误的是

(A)参数是列表类型时，改变原参数的值
(B)参数是整数类型时，不改变原参数的值
(C)参数是组合类型（可变对象）时，改变原参数的值
(D)参数的值是否改变与函数中对象的操作有关，与参数类型无关

python 里有可变对象和不可变对象之分。只有传入的是不可变对象时，值才不发生改变，若是可变对象，值的传入需要注意。不可变对象：Number，String，Tuple，bool。可变对象：List，Set，Dictionary 是可以改变内部的元素。

在 python 中，关于函数的描述，以下选项中正确的是 D

(A)一个函数中只允许有一条 return 语句
(B)Python 中，def 和 return 是函数必须使用的保留字
(C)Python 函数定义中没有对参数指定类型
　　这说明参数在函数中可以当作任意类型使用
(D)函数 eval()可以用于数值表达式求值，例如 eval('2*3+1')

给出如下代码 A c 值为 210

```
def func(a,b):
    c = a **2 + b                    c=100+100
    b =a                                 b=10
    return c                         函数返回 200
a=10
b=100
c=func(a,b)+a                        主函数，c=200+10
```

以下选项中描述错误的是

(A)执行该函数后，变量 c 的值为 200

(B)该函数名称为 func

(C)执行该函数后，变量 b 的值为 100

(D)执行该函数后，变量 a 的值为 10

在 Python 中，关于全局变量和局部变量 D

以下选项中描述不正确的是

(A)一个程序中的变量包含两类：全局变量和局部变量

(B)全局变量一般没有缩进

(C)全局变量在程序执行的全过程有效

(D)全局变量不能和局部变量重名

> 全局变量可以和局部变量重名，全局变量指的是在函数之外定义的变量，在程序执行全过程有效。全局变量在函数内部使用时，需要提前使用保留字 global 声明。局部变量指在函数内部定义的变量，仅在函数内部有效，当函数退出时，变量将不再存在。例如：
>
> ```
> n=2
> def multiply(x,y=10):
>     n=x*y      #局部变量 n，和第一行代码 n 不同。
>     return n    #返回的 n 也是局部变量 n
> s=multiply(99,2)
> print(s)#输出 198
> print(n)#输出 2
> ```
>
> 上一题 c=210 也是如此。
>
> 因此全局变量可以和局部变量重名。

以下选项中，对于递归程序的描述错误的是 B

(A)书写简单 (B)执行效率高 (C)一定要有基例

(D)递归程序都可以有非递归编写方法

下面代码的输出结果是 B 相当于：def fun(x, y):

```
f = lambda x,y:y+x                       return x+y
f(10,10)                           print(fun(10, 10))
```

(A)10 (B)20

(C)10,10 (D)100

关于形参和实参的描述，以下选项中正确的是 **C**

(A)函数定义中参数列表里面的参数是实际参数，简称实参
(B)参数列表中给出要传入函数内部的参数
　　这类参数称为形式参数，简称形参
(C)程序在调用时，将实参复制给函数的形参
(D)程序在调用时，将形参复制给函数的实参

函数定义中参数列表里面的参数是形参，参数列表中给出要传入函数内部的参数，这类参数称为实参；程序在调用时，将实参复制给函数的形参。例如：

```
def greet_user(username):    #username 是形参
    print("hello," + username + "!" )
greet_user('python')    #'python'是形参
```

关于 lambda 函数，以下选项中描述错误的是 D

(A)lambda 函数也称为匿名函数
(B)lambda 函数将函数名作为函数结果返回
(C)定义了一种特殊的函数
(D)lambda 不是 Python 的保留字

关于 lambda 函数，以下选项中描述错误的是 B function 类型

(A)lambda 函数将函数名作为函数结果返回
(B)f=lambda x,y:x+y 执行后
　　f 的类型为数字类型
(C)lambda 用于定义简单的、能够在一行内表示的函数
(D)可以使用 lambda 函数定义列表的排序原则

以下选项中，对于函数的定义错误的是 D

(A)def vfunc(a, b=2):　(B)def vfunc(a, b):
(C)def vfunc(a, *b):　(D)def vfunc(*a, b):

关于函数的参数，以下选项中描述错误的是 **C**

(A)在定义函数时，如果有些参数存在默认值
　　可以在定义函数时直接为这些参数指定默认值
(B)在定义函数时，可以设计可变数量参数
　　通过在参数前增加星号(*)实现
(C)可选参数可以定义在非可选参数的前面
(D)一个元组可以传递给带有星号的可变参数

关于 return 语句，以下选项中描述正确的是 D

(A)函数中最多只有一个 return 语句
(B)函数必须有一个 return 语句
(C)return 只能返回一个值
(D)函数可以没有 return 语句

关于 Python 中的 lambda 函数，以下选项中描述错误的是　　C 类型依参数确定
(A)lambda 用于定义简单的、能够一行内表示的函数
(B)可以使用 lambda 函数定义列表的排序原则
(C)f=lambda x,y:x+y 执行后，f 的类型为数字类型
(D)lambda 函数将函数名作为函数结果返回

关于函数的返回值，以下选项中描述错误的是　　B
(A)函数可以返回 0 个或多个结果
(B)函数必须有返回值
(C)函数可以有 return，也可以没有
(D)return 可以传递 0 个返回值，也可以传递任意多个返回值

关于函数的目的与意义，以下选项中描述错误的是　　B
(A)程序功能抽象，以支持代码重用
(B)函数能调用未实现的函数
(C)使用时无须了解函数内部实现细节
(D)有助于采用分而治之的策略编写大型复杂程序

关于函数，以下选项中描述错误的是　　C
(A)函数也是数据
(B)函数定义语句可执行
(C)函数名称不可赋给其他变量
(D)一条函数定义一个用户自定义函数对象

关于函数的参数传递，以下选项中描述错误的是　　D
(A)形式参数是函数定义时提供的参数
(B)实际参数是函数调用时提供的参数
(C)参数传递时不构造新数据对象，而是让形式参数和实际参数共享同一对象
(D)函数调用时，需要将形式参数传递给实际参数

关于函数的可变参数，可变参数*args 传入函数时存储的类型是　　B
(A)dict　(B)tuple
(C)list　(D)set

f(n, *args, **kwargs)后面两个是 python 中的可变参数。
*args 表示任何多个无名参数，它是一个 tuple
**kwargs 表示关键字参数，它是一个 dict
同时使用 * args 和 ** kwargs 时，必须 * args 参数列要在 ** kwargs 前，

阅读下列程序，一共打印几行【1】 **1 3**

第一行输出的结果是：【2】 **2 3**

```
def f(a,b):
    if b==0:
        print(a)
    else:
        print(f(b,a%b))
f(9,6)
```

实际输出：
3
None
None
为什么输出不是 333？我理解的话是因为，这个不是 return f(b，a%b)，所以不是两个函数相等这种，不能理解为 f(6,3)和=f(3,0)=3，f(6,3)和 f(3,0)没有实际意义，所以 print 无结果，输出为 None

下面代码实现的功能描述为： A

```
def fact(n):
    if n == 0:
        return 1
    else :
        return n * fact(n-1)
num = eval(input('请输入一个整数：'))
print(fact(abs(int(num))))
```

(A)接受用户输入的整数 N，输出 N 的阶乘值
(B)接受用户输入的整数 N，判断 N 是否是素数并输出结论
(C)接受用户输入的整数 N，判断 N 是否是整数并输出结论
(D)接受用户输入的整数 N，判断 N 是否是水仙花数

下面代码的运行结果是 A

```
def func(num):
    num += 1
a = 10
func(a)
print(a)
```

(A)10 (B)11 (C)出错 (D)int

给出如下代码 A

```
def fact(n):
    s = 1
    for i in range(1,n+1):
        s *= i
    return s
```

以下选项中描述错误的是
(A)代码中 n 是可选参数
(B)fact(n)函数功能为求 n 的阶乘
(C)s 是局部变量
(D)range()函数是 Python 内置函数

给出如下代码 **B**

```
ls = ['car', 'truck']
def funC(a):
    ls.append(a)
    return
funC('bus')
print(ls)
```

['car','truck','bus']

以下选项中描述错误的是
(A)ls.append(a)代码中的 ls 是全局变量
(B)执行代码输出结果为['car', 'truck']
(C)ls.append(a)代码中的 ls 是列表类型变量
(D)funC(a)中 a 为非可选参数

给出如下代码 **D 函数里面建立了新列表**

```
ls = ['car','truck']
def funC(a):
    ls = []
    ls.append(a)
    return
funC('bus')
print(ls)
```

以下选项中描述错误的是
(A)代码函数定义中，ls.append(a)中的 ls 是局部变量
(B)执行代码输出的结果为['car','truck']
(C)ls.append(a)代码中的 ls 是列表类型
(D)执行代码输出结果为['car','truck','bus']

```
import turtle
def drawLine(draw):
    turtle.pendown() if draw else turtle.penup()
    turtle.fd(50)
    turtle.right(90)
drawLine(True)
drawLine(True)
drawLine(True)
drawLine(True)
```

D

以下选项中描述错误的是

(A)运行代码，在 Python Turtle Graphic 中，绘制一个正方形

(B)代码 def drawLine(draw)中的 draw 可取值 True 或者 False

(C)代码 drawLine(True)中 True 替换为-1，运行代码结果不变

(D)代码 drawLine(True)中 True 替换为 0，运行代码结果不变

参数如果需要布尔值，而输入的是数值，那么只需要该数值为非 0 的数，对应为 True，如果输入的数值为 0，则代表 False，因此 D 不正确

```
import turtle
def drawLine(draw):
    turtle.pendown() if draw else turtle.penup()
    turtle.fd(50)
    turtle.right(90)
drawLine(True)
drawLine(0)
drawLine(True)
drawLine(True)
turtle.left(90)
drawLine(0)
drawLine(True)
drawLine(True)
```

B

以下选项中描述错误的是

(A)运行代码，在 Python Turtle Graphic 中，绘制一个数码管数字 2

(B)代码 drawLine(True)中 True 替换为 0，运行代码结果不变

(C)代码 drawLine(True)中 True 替换为-1，运行代码结果不变

(D)代码 def drawLine(draw)中的 draw 可取值 0、1、-1 等

下面代码的运行结果

```
def func(a, b):
    a *= b
    return a
s = func(5, 2)
print(s)
```

10

下面代码的运行结果是 13

```
def fib(n):
    a,b = 1,1
    for i in range(n-1):
        a,b = b,a+b
    return a
print(fib(7))
```

注意 a,b=b,a+b 是同步赋值
并不等价于先 a=b
再 b=a+b

下面代码的运行结果是 A

```
def hello_world():
    print('ST', end = '*')
def three_hello():
    for i in range(3):
        hello_world()
three_hello()
(A)ST*ST*ST*
(B)STST
(C)ST*
(D)***
```

下面代码的运行结果是 **1 4**

```
counter=1
num=0
def TestVariable():
    global counter
    for i in (1,2,3):
        counter+=1
        num=10
TestVariable()
print(counter)
print(num)
```

2 0

下面代码的运行结果是 20 10

```
def exchange(a,b):
    a,b = b,a
    return(a,b)
x=10
y=20
x,y=exchange(x,y)
print(x,y)
```

下面代码的运行结果是 20

```
MA = lambda x,y:(x>y)*x+(x<y)*y
MI = lambda x,y:(x>y)*y+(x<y)*x
a =10
b=20
print(MA(a,b))
print(MI(a,b))
```

10

x>或<y 是 01 条件

关于下面的代码，以下选项中描述正确的是 B

```
list(range(0,10,2))
```

(A)执行结果为 0，2，4，6，8
(B)按位置参数调用
(C)按关键字参数调用
(D)按可变参数调用

关于下面的代码，以下选项中描述正确的是 A

```
def fact(n,m=1):
    s = 1
    for i in range(1,n+1):
        s *= i
    return s//m
print(fact(m=5,n=10))
```

m=5 覆盖默认的 m=1

(A)参数按照名称传递
(B)按位置参数调用
(C)执行结果为 10886400
(D)按可变参数调用

执行下面代码，运行错误的是 D 位置参数错误

```
def f(x,y=0,z=0):
    pass
```

(A)f(1,y=2,z=3)
(B)f(1,z=3)
(C)f(z=3,x=1,y=2)
(D)f(1,x=1,z=3)

执行下面代码，运行正确的是 C

```
def f(x,y=0,z=0):
    pass
```

(A)f(1,x=1,z=3)
(B)f(x=1,2)
(C)f(x=1,y=2,z=3)
(D)f(1,y=2,t=3)

关于嵌套函数，以下选项中描述或者程序错误的是 D 原因是 B

(A)嵌套函数是在函数内部定义函数
(B)嵌套函数仅供外层函数调用，外层函数之外不得调用
(C)

```
def f():
    print('Outer function f')
    def g():
        print('Inner function g')
    g()
f()
```

(D)

```
def f():
    print('Outer function f')
    def g():
        print('Inner function g')
    g()
f.g()
```

下面代码的执行结果是 D

```
def area(r, pi = 3.14159):
    return pi*r*r
print(area(pi = 3.14, r = 4))
```

(A)出错 (B)无输出 (C)39.4384 (D)50.24

编写程序
输入某年某月某日
判断这一天是这一年的第几天

```
date = input("输入年月日(yyyy-mm-dd):")
y,m,d = (int(i) for i in date.split('-'))
sum=0
special = (1,3,5,7,8,10)
for i in range(1,int([1])):
    if i == 2:
        if y%400==0 or ([2]):
            sum+=29
        else:
            sum+=28
    elif(i in [3]):
        sum+=31
    else:
        sum+=30
sum+=d
print("这一天是一年中的第",sum,"天")
```

[1] m
[2] y%100!=0 and y%4==0
[3] special
[4] sum+=d

下面代码的执行结果是 **D**

```
def greeting(args1, *tupleArgs, **dictArgs):
    print(args1)
    print(tupleArgs)
    print(dictArgs)
names = ['HTY', 'LFF', 'ZH']
info = {'schoolName':'NJRU', 'City':'Nanjing'}
greeting('Hello', *names, **info)
```

(A)出错

(B)无输出

(C)['HTY', 'LFF', 'ZH']

(D)Hello
('HTY', 'LFF', 'ZH')
{'schoolName':'NJRU','City':'Nanjing'}

在函数中，如 greeting('Hello', *names, **info)，可以传递任意数量实参和任意数量的关键字实参，但是前提是传递**的实参是字典形式，传递*的实参是元组的形式。本题对应形参 args1 为'Hello'；tupleArgs 为['HTY', 'LFF', 'ZH']；dictArgs 为{'schoolName':'NJRU','City':'Nanjing'}

5 文件

关于 Python 对文件的处理，以下选项中描述错误的是 **C**

(A)Python 能够以文本和二进制两种方式处理文件
(B)Python 通过解释器内置的 open()函数打开一个文件
(C)当文件以文本方式打开时，读写按照字节流方式
(D)文件使用结束后要用 close()方法关闭，释放文件的使用授权

文件是一个储存在辅助存储器上的数据序列，可以包含任何数据内容。
文件包括文本文件和二进制文件两种类型
文本文件一般由单一特定编码的字符组成，可被看作是存储在磁盘上的长字符串
二进制文件直接由比特 0 和比特 1 组成，由于没有统一字符编码，只能当做字节流
无论是创建为文本文件还是二进制文件，都可以用“文本文件方式”和“二进制文件方式”打开，但打开后操作方式不同。
以文本文件方式打开时，读写按照字符串方式，采用当前计算机使用的编码或指定编码
以二进制文件方式打开时，读写按照字节流方式

执行如下代码： **B**

```
fname = input("请输入要写入的文件：")
fo = open(fname, “w+”)
ls = [“清明时节雨纷纷，”,“路上行人欲断魂，”,
      “借问酒家何处有？”,“牧童遥指杏花村。”]
fo.writelines(ls)
fo.seek(0)
for line in fo:
    print(line)
fo.close()
```

以下选项中描述错误的是

(A)fo.writelines(ls)将元素全为字符串的 ls 列表写入文件
(B)fo.seek(0)这行代码如果省略，也能打印输出文件内容
(C)代码主要功能为向文件写入一个列表类型，并打印输出结果
(D)执行代码时，从键盘输入“清明.txt”，则清明.txt 被创建

文件打开后，对文件的读写有一个读取指针，当从文件读写内容后，指针随之前进，再次读取的内容从指针的新位置开始。在这里输入完内容后。指针在文件的最后面，所以不 seek 直接对文件进行 for 循环，已经无法从当前指针处读取内容，因此返回结果是空。
seek(0)回到文件开头，seek(2)移动到文件结尾

以下选项中，不是 Python 文件处理.seek()方法的参数是 **C**

(A)0 (B)1 (C)-1 (D)2

以下选项中，不是 Python 文件二进制打开模式的合法组合是 **C**

(A)"b+" (B)"bw" (C)"x+" (D)"bx"

以下选项中，不是 Python 对文件的打开模式的是 **D**

(A)"r" (B)"w" (C)"+" (D)"c"

Python 语句 f=open()，以下选项中对 f 的描述错误的是 **B（生成一个迭代器）**

(A)f 是文件句柄，用来在程序中表达文件

(B)表达式 print(f)执行将报错

(C)f 是一个 Python 内部变量类型

(D)将 f 当作文件对象，f.read()可以读入文件全部信息

给出如下代码： **B**

```
fname=input("请输入要打开的文件:")
fo=open(fname,"r")
for line in fo.readlines():
      print(line)
fo.close()
```

关于上述代码的描述，以下选项中错误的是

(A)用户输入文件路径，以文本文件方式读入文件内容并逐行打印

(B)通过 fo.readlines()方法将文件的全部内容读入一个字典 fo

(C)通过 fo.readlines()方法将文件的全部内容读入一个列表 fo

(D)上述代码可以优化为：

```
fname=input("请输入要打开的文件:")
fo=open(fname,"r")
for line in fo:
      print(line)
fo.close()
```

从键盘输入一个字符串， **[1]'w'/'w+'/'wt'**

将小写字母全部转换成大写字母， **[2]upper()**

然后输出到一个磁盘文件 test 中保存。 **[3]'r'/'r+'/'rt'**

补充程序： **[4]close()**

```
def main():
    fp=open('test.txt',[1])
    string=input('please input a str:\n')
    string=string.[2]
    fp.write(string)
    fp=open('test.txt',[3])
    print(fp.read())
    fp.[4]
if __name__=='__main__':
    main()
```

设 city.csv 文件内容如下： **B**

巴哈马,巴林,孟加拉国,巴巴多斯

俄罗斯,比利时,伯利兹

下面代码的执行结果是：

```
f=open("city.csv","r")
ls=f.read().split(",")
f.close()
print(ls)
```

(A)['巴哈马','巴林','孟加拉国','巴巴多斯','俄罗斯','比利时','伯利兹']

(B)['巴哈马','巴林','孟加拉国','巴巴多斯\n 俄罗斯','比利时','伯利兹']

(C)['巴哈马,巴林,孟加拉国,巴巴多斯,俄罗斯,比利时,伯利兹']

(D)['巴哈马','巴林','孟加拉国','巴巴多斯','\n','俄罗斯','比利时','伯利兹']

```
f=open("city.csv","r")
print(f.read())
```

得到的结果是

巴哈马,巴林,孟加拉国,巴巴多斯

俄罗斯,比利时,伯利兹

其实就应该是：'巴哈马,巴林,孟加拉国,巴巴多斯\n 俄罗斯,比利时,伯利兹'

遇到逗号作为一个分隔

6 科学计算/数据分析/网络爬虫基础

Python 网络爬虫方向的第三方库是 A
A.request B.jieba C.itchat D.time

Python 网络爬虫方向的第三方库是 B
(A)numpy (B)scrapy (C)Arcade (D)FGMK

Python 数据分析方向的第三方库是 C
(A)Bokeh (B)dataswim (C)scipy (D)Gleam

Python 数据可视化方向的第三方库是 A
(A)matplotlib (B)retrying (C)FGMK (D)PyQt5

将 Python 脚本程序转变为可执行程序的第三方库是 D
(A)random (B)pygame (C)PyQt5 (D)PyInstaller

以下选项中，不是 Python 数据分析方向的第三方库是 A
(A)requests (B)numpy (C)scipy (D)pandas

Python 数据分析方向的第三方库是 A
(A)numpy (B)pdfminer (C)beautifulsoup4 (D)time

Python 数据分析方向的第三方库是 D
(A)random (B)PIL (C)Django (D)pandas

关于 requests 的描述，以下选项中正确的是 B
(A)requests 是数据可视化方向的 Python 第三方库
(B)requests 是处理 HTTP 请求的第三方库
(C)requests 是支持多种语言的自然语言处理 Python 第三方库
(D)requests 是一个支持符号计算的 Python 第三方库

关于 matplotlib 的描述，以下选项中错误的是 C
(A)matplotlib 主要进行二位图表数据展示，广泛用于科学计算的数据可视化
(B)matplotlib 是提供数据绘图功能的第三方库
(C)matplotlib 是 Python 生态中最流行的开源 Web 应用框架
(D)使用 matplotlib 库可以利用 Python 程序绘制超过 100 种可视化效果

在 python 环境下使用 numpy 函数库，需要做的工作是 A
(A)在 python 环境下单独安装 numpy 才能使用
(B)python 环境已默认安装 numpy 可直接使用
(C)针对不同 python 版本可以选择任何 numpy 安装软件
(D)需要从 python 网站下载 numpy 安装程序

7 jieba/turtle/random/time 库

关于 import 引用，以下选项中描述错误的是 **B**

(A)使用 import turtle 引入 turtle 库
(B)可以使用 from turtle import setup 引入 turtle 库
(C)使用 import turtle as t 引入 turtle 库，取别名为 t
(D)import 保留字用于导入模块或者模块中的对象

关于 jieba 库的函数 **D**

jieba.lcut(x,cut_all=True)，正确的是

(A)精确模式，返回中文文本 x 分词后的列表变量
(B)向分词词典中增加新词 w
(C)搜索引擎模式，返回中文文本 x 分词后的列表变量
(D)全模式，返回中文文本 x 分词后的列表变量

关于 turtle 库中的 setup()函数，以下选项中描述错误的是 **B**

(A)执行下面代码，可以获得一个宽为屏幕 50%，高为屏幕 75%的主窗口

```
import turtle
turtle.setup(0.5,0.75)
```

(B)turtle.setup()函数的作用是设置画笔的尺寸
(C)turtle.setup()函数的定义为

```
turtle.setup(width,height,startx,starty)
```

(D)turtle.setup()函数的作用是设置主窗体的大小和位置

random 库的 seed(a)函数的作用是 **D**

(A)生成一个[0.0,1.0)之间的随机小数
(B)生成一个 k 比特长度的随机整数
(C)生成一个随机整数
(D)设置初始化随机数种子

A

random 库中用于生成随机小数的函数是
(A)random()
(B)randint()
(C)getrandbits()
(D)randrange()

random 库的 random.sample(pop，k)函数的作用是 **B**

(A)从 pop 类型中随机选取 k-1 个元素，以列表类型返回
(B)从 pop 类型中随机选取 k 个元素，以列表类型返回
(C)生成一个随机整数
(D)随机返回一个元素

random.uniform(a,b) **B**

(A)生成一个[a,b]之间的随机整数
(B)生成一个[a,b]之间的随机小数
(C)生成一个[a,b]之间的随机数
(D)生成一个均值为 a，方差为 b 的正态分布

time.ctime()的作用是 **C**

(A)将 struct_time 对象变量 t 转化为时间戳
(B)返回系统当前时间戳对应的本地时间的
struct_time 对象，本地之间经过时区转换
(C)返回系统当前时间戳对应的易读字符串表示
(D)返回系统当前时间戳对应的 struct_time 对象

time 库的 time.mktime(t)函数的作用是 **D**

(A)根据 format 格式打出 struct_time 类变量 t
(B)返回一个代表时间的精确浮点数
两次或多次使用，做差计时
(C)使当前程序暂停执行 secs 秒
(D)将 struct_time 对象变量 t 转化为时间戳

```
##time 库的使用
>>>time.time()#返回系统当前的时间戳
1580886897.2976618

>>> time.ctime()#返回系统当前的时间戳对应的易读字符串表示
'Wed Feb  5 15:15:03 2020'

>>> time.gmtime()#返回系统当前的时间戳的 struct_time 对象
time.struct_time(tm_year=2020, tm_mon=2, tm_mday=5, tm_hour=7,
tm_min=15, tm_sec=40, tm_wday=2, tm_yday=36, tm_isdst=0)

>>> t=time.gmtime()
>>> time.mktime(t)#将 struct_time 对象变量 t 转化为时间戳
1580858773.0

>>>  time.localtime()#返回系统当前的时间戳对应的本地时间的 struct_time
对象，经过本地时区转换
time.struct_time(tm_year=2020, tm_mon=2, tm_mday=5, tm_hour=15,
tm_min=31, tm_sec=49, tm_wday=2, tm_yday=36, tm_isdst=0)

>>> time.strptime("5 Dec 20", "%d %b %y")
time.struct_time(tm_year=2020, tm_mon=12, tm_mday=5, tm_hour=0,
tm_min=0, tm_sec=0, tm_wday=5, tm_yday=340, tm_isdst=-1)

>>>time.sleep(secs)#使当前程序暂停执行 secs 秒

##返回一个代表时间的精确浮点数，两次或多次使用，做差计时
>>> start=time.perf_counter()
2072.2523203
>>> end=time.perf_counter()
2081.0421058
>>>print(end-start)
8.789785500000107

>>>time.strftime(format,t)#根据 format 格式打出 struct_time 类变量 t
>>> time.strftime("%Y %b %d",time.gmtime())
'2020 Feb 05'

%y 两位数的年份表示（00-99）      %Y 四位数的年份表示（000-9999）
%m 月份（01-12）                  %d 月内中的一天（0-31）
%H 24 小时制小时数（0-23）        %I 12 小时制小时数（01-12）
%M 分钟数（00-59）                %S 秒（00-59）
%a 本地简化星期名称               %A 本地完整星期名称
%b 本地简化的月份名称             %B 本地完整的月份名称
%p 本地 A.M.或 P.M.的等价符       %w 星期（0-6），星期天为星期的开始
```