1. 单选题
2. Python是一种（ ）的计算机程序设计语言。
3. 面向过程
4. 面向对象
5. 面向数据
6. 面向用户
7. 以下说法错误的是（ ）。
8. Python语言是一种面向对象的语言。
9. Python语言具有可移植性的特点。
10. Python语言是一种编译性语言。
11. Python语言具有可扩展性的特点。
12. 以下哪项不是Python的标准库（ ）。
13. os
14. math
15. random
16. numpy
17. 使用（ ）命令可以查看Python3的版本。
18. python3 -B
19. python3 -V
20. python3 -X
21. python3 -W
22. Python中可以使用（ ）函数实现交互过程。
23. print
24. import
25. input
26. output
27. 在Python中，符号（ ）可以给变量进行赋值。
28. <-
29. =
30. ==
31. !=
32. 在Python中，以下变量名错误的是（ ）。
33. a
34. a\_001
35. 001\_a
36. a001
37. 以下代码的运行结果为（ ）。

a = 123  
print(a)  
a = **'ABC'**print(a)

1. 123

123

1. ABC

ABC

1. 123

ABC

1. 123

'ABC'

1. 以下代码的运行结果为（ ）。

a = **'ABC'**b = a  
a = **'XYZ'**print(b)

1. ABC
2. 'ABC'
3. XYZ
4. 'XYZ'
5. 下列选项中，（ ）不是字符串。
6. “123”
7. ‘123’
8. 123
9. “一2三”
10. 以下代码的运行结果为（ ）。

print(**r'''hello,\n  
world'''**)

1. hello,\n

world

1. hello,world
2. '''hello,world'''
3. r'''hello,\n world'''
4. 以下代码的运行结果为（ ）。

print(**'%.2f'** % 3.1415926)

1. %.2f
2. 3.1415926
3. 3.14
4. %3.1415926
5. 在Python中使用（ ）表示 % 符号。
6. %
7. %%
8. 100%
9. %-%
10. 在Python中使用（ ）表示两个对象相加。
11. +
12. -
13. \*
14. /
15. 在Python中使用（ ）表示两个对象取整除。
16. /
17. %
18. //
19. %%
20. 在Python中使用（ ）表示两个对象不相等。
21. =/
22. /=
23. =！
24. ！=
25. 以下语句等效于（ ）。

c \*\*= a

1. c = c \*\* a
2. c = c \* a
3. c = a \*\* a
4. a = c \*\* c
5. 在Python中，&表示按位（ ）运算符。
6. 与
7. 或
8. 异或
9. 取反
10. （ ）不是Python中的逻辑运算符。
11. and
12. or
13. no
14. not
15. 以下运算符中，优先级最高的是（ ）。
16. &
17. <
18. is not
19. \*\*
20. x的y次方表达式正确的是（ ）。
21. x^y
22. x\*\*y
23. x^^y
24. Python没有提到
25. 22%3表达式输出结果为（ ）。
26. 7
27. 1
28. 0
29. 5
30. 3\*1\*\*3表达式输出结果为（ ）。
31. 27
32. 9
33. 3
34. 1
35. 如果表达式的操作符有相同的优先级，则运算规则是（ ）。
36. 从左到右
37. 从右到左
38. 随便
39. 同时运算
40. 以下代码的运行结果为（ ）。

x = **True**y = **False**z = **False  
if** x **or** y **and** z:  
 print(**"1"**)  
**else**:  
 print(**"2"**)

1. 1
2. 2
3. True
4. 运行出错
5. 下列选项属于列表的是（ ）。
6. (1,2,3)
7. [1,2,3]
8. {1,2,3}
9. {a:1,b:2,c:3}
10. 在Python中，可以使用（ ）函数在列表末尾添加新的对象。
11. add()
12. index()
13. append()
14. pop()
15. 以下代码的运行结果为（ ）。

a = [1,2,3]  
b = a.copy()  
print(b)

1. [1,1,2,2,3,3]
2. [1,2,3][1,2,3]
3. [1,2,3]
4. [6]
5. 以下代码的运行结果为（ ）。

x = []  
n = len(x)  
print(n)

1. 0
2. NA
3. NULL
4. -1
5. （ ）变量是元组类型。
6. 3
7. [3]
8. (3)
9. (3,)
10. a是一个列表，a = ['A', 'b', 'C', 'd', 'E']，若要获取列表a的前三个元素应当采用（ ）。
11. a[3]
12. a[0,1,2]
13. a[:3]
14. a[1,2,3]
15. 以下那种数据结构不含重复元素（ ）
16. 列表
17. 元组
18. 字典
19. 集合
20. 以下不能创建一个字典的语句是（ ）。
21. dic1 = {}
22. dic2 = {123:345}
23. dic3 = {[1,2,3]:'abc'}
24. dic3 = {(1,2,3):'abc'}
25. 以下代码的运行结果是（ ）。

a=(1,2,[**'1,2'**])  
a[2].append(3)  
print(a)

1. (1, 2, ['1,2', '3'])
2. (1, 2, ['1,2', 3])
3. [1, 2, ['1,2', 3]]
4. 报错
5. 以下代码的运行结果可能是（ ）。

print(set(**'121323'**))

1. ('1','2','1','3','2','3')
2. {'121323'}
3. {'1', '2', '3'}
4. ('1', '2', '3')
5. 以下关于字典的描述，错误的是（ ）。
6. 字典是键值对的集合
7. 字典长度是可变的
8. 字典中的键可以对应多个值信息
9. 字典中元素以键信息为索引访问
10. 以下代码的运行结果为（ ）。

x = **True**country\_number = {}  
**def** example(country):  
 **if** country **in** country\_number:  
 country\_number[country] += 1  
 **else**:  
 country\_number[country] = 1  
example(**'中国'**)  
example(**'美国'**)  
example(**'中国'**)  
print(len(country\_number))

1. 0
2. 1
3. 2
4. 3
5. 以下代码的运行结果为（ ）。

a = {}  
a[1] = 1  
a[**'1'**] = 2  
a[1] += 1  
sum = 0  
**for** k **in** a:  
 sum += a[k]  
print(sum)

1. 1
2. 2
3. 3
4. 4
5. Python中，（ ）函数是用于输出内容到终端的。
6. echo
7. output
8. print
9. console.log
10. input()函数的返回结果类型为（ ）
11. 整数
12. 浮点数
13. 字符串
14. 文本
15. 布尔值不可以用（ ）运算。
16. and
17. or
18. not
19. is not
20. 在Python程序中，条件判断的实现用（ ）语句实现。
21. for
22. if
23. while
24. do
25. 代码块的规则不包括（ ）。
26. 当缩进增加时，代码块就开始了。
27. 代码块需要用{}包括起来。
28. 代码块可以包含其他代码块。
29. 缩进减少为零或减少为外面包围代码块的缩进，代码块就结束了。
30. 以下代码的运行结果为（ ）

a = **"1"  
if** a == 1:  
 print(**"苹果"**)  
**else**:  
 print(**"香蕉"**)  
print(**"西瓜"**)

1. 苹果

香蕉

1. 苹果

西瓜

1. 香蕉

西瓜

1. 香蕉
2. if 语句不包括包含以下（ ）部分。
3. if关键字
4. 条件
5. 冒号
6. 大括号
7. else语句不包括包含以下（ ）部分。
8. else关键字
9. 条件
10. 冒号
11. 代码块
12. 以下代码的运行结果为（ ）。

name = **"张三"**age = 18  
**if** name == **'张三'**:  
 print(**'你好, 张三'**)  
**elif** age < 16:  
 print(**'****16岁以下'**)  
**elif** age > 22:  
 print(**'****22岁以上'**)

1. 张三
2. 你好, 张三
3. 16岁以下
4. 22岁以上
5. 以下代码的运行结果为（ ）。

a = 1  
b = 2  
**if** a:  
 a = 3  
 print(b)  
**else**:  
 b = 4  
 print(a)

1. 1
2. 2
3. 3
4. 4
5. 以下代码的运行结果为（ ）。

a = 1  
b = 1  
c = 2  
**if** a == b **and** a == c:  
 print(**"a+b"**)  
**else**:  
 print(**"a-b"**)

1. 2
2. 0
3. a+b
4. a-b
5. 以下代码的运行结果为（ ）。

b1 = [1,2,3]  
b2 = [2,3,4]  
b3 = [x **for** x **in** b1 **if** x **in** b2]  
print (b3)

1. [1,2,3,4]
2. [2]
3. [2,3]
4. 程序报错
5. 以下代码的循环次为（ ）。

n = 1000  
**while** n > 1:  
 print(n)  
 n = n / 2

1. 9
2. 10
3. 11
4. 无限循环
5. 以下代码的运行结果为（ ）。

a = []  
**for** i **in 'apple'**:  
 **if** i == **'p'**:  
 **continue** a.append(i)  
print(**''**.join(a))

1. a,p,p,l,e
2. apple
3. aple
4. ale
5. 以下代码的运行结果为（ ）。

**for** i **in** range(2, 5):  
 print(**"编号-"** + str(i), end=**" "**)

1. 编号-1 编号-2 编号-3 编号-4 编号-5
2. 编号-2 编号-3 编号-4 编号-5
3. 编号-2 编号-3 编号-4
4. 编号-2 编号-5
5. 以下代码的运行结果为（ ）。

**for** i **in** range(10):  
 **if** i == 5:  
 **break** print(i, end=**""**)

1. 12345678910
2. 0123456789
3. 01234
4. 012345
5. 以下代码的运行结果为（ ）。

y = 1  
**for** i **in** range(0, 10, 2):  
 y += i  
print(y)

1. 10
2. 11
3. 21
4. 31
5. 以下代码的运行结果为（ ）。

number = 30  
**if** number % 2 == 0:  
 print(number, **'is even'**)  
**elif** number % 3 == 0:  
 print(number, **'is multiple of 3'**)

1. 30 is even

30 is multiple of 3

1. 30 is multiple of 3
2. 程序出错
3. 30 is even
4. 以下代码的运行结果为（ ）。

x = 1  
y = -1  
z = 1  
**if** x > 0:  
 **if** y > 0: print(**'A'**)  
**elif** z > 0: print(**'B'**)

1. 无输出5,
2. A
3. B
4. 语法错误
5. 如果输入5, -1, 6, 16, 9, 5, 0，以下代码的运行结果为（ ）。

number = int(input(**'请输入整数: '**))  
max = number  
**while** number != 0:  
 number = int(input(**'请输入整数: '**))  
 **if** number > max:  
 max = number  
print(max)

1. 5
2. -1
3. 16
4. 0
5. 下列for语句中，在in后使用不正确的是（ ）。

**for** x **in \_\_\_\_**:  
 print(x)

1. set('str')
2. range(0, 10, 5)
3. [1, 2, 3, 4, 5]
4. (1)
5. 关于以下代码说法正确的是（ ）。

k = 10  
**while** k:  
 k = k-1  
 print(k)

1. 循环体语句执行1次
2. while循环执行10次
3. 循环体语句一次也不会执行
4. 循环是无限循环
5. 以下代码中while循环结束的条件是（ ）。

n = p = 0  
**while** p != 10 **and** n < 5:  
 p = int(input())  
 n += 1

1. p的值不等于10并且n的值小于5
2. p的值不等于10或者n的值小于5
3. p的值等于10或者n的值大于等于5
4. p的值等于10并且n的值大于等于5
5. 以下for语句中,不能完成1-10的累加功能的是（ ）。
6. for i in(10,9,8,7,6,5,4,3,2,1): sum+=i
7. for i in range(10,-1,-1): sum+=i
8. for i in range(10,0): sum+=i
9. for i in range(1,11): sum+=i
10. 下列说法中正确的是（ ）。
11. break用在while语句中,而continue用在for语句中
12. break能结束循环,而continue只能结束本次循环
13. break用在for语句中,而continue用在while语句中
14. continue能结束循环,而break只能结束本次循环
15. 可以终结一个循环的执行的语句是（ ）。
16. input
17. if
18. exit
19. break
20. 以下关于循环控制语句描述错误的是哪一项（ ）。
21. Python中的for语句可以在任意序列上进行迭代访问,例如列表、字符串和元组。
22. Python中多分支可以用if…elif…else…来表达。
23. Python中循环结构中存在else语句。
24. Python中循环不可以嵌套使用,一个while语句中不能存在for语句。
25. 表达式sum(range(10))的值为（ ）。
26. 45
27. 55
28. 11
29. 46
30. 以下（ ）程序输出"快乐"10次。
31. for a in range(1, 10):

print("快乐")

1. ‌for b in range(0, 10):

print("快乐")

1. ‌for c in range(1, 11):

print("快乐")

1. ‌for d in range(1, 12):

print("快乐")

1. （2）（4）
2. （1）（2）（3）
3. （1）（3）
4. （2）（3）
5. 以下代码的运行结果为（ ）。

**for** i **in** range(3):  
 **for** s **in 'xyzabc'**:  
 **if** s==**'a'**:  
 **break** print(s,end=**''**)

1. xxxyyyzzz
2. xxxyyyzzzaaa
3. xyzxyzxyz
4. xyzaxyzaxyza
5. 以下代码的运行结果为（ ）。

**for** s **in 'xyab'**:  
 **for** i **in** range(3):  
 **if** s == **'a'**:  
 **break** print(s, end=**''**)

1. xyxyxy
2. xxxyyybbb
3. xxxyyy
4. xybxybxyb
5. 关于Python循环结构，以下选项中描述不正确的是（ ）。
6. 遇到break语句后，所有层次的循环都会结束。
7. for循环结构用遍历的方式对字符串或者range()对象中的每个元素进行操作。
8. 所有for循环功能都可以用while循环语句来替代。
9. 使用while循环语句不需要提前知道循环的次数。
10. 以下代码的运行结果为（ ）。

n = 0  
**for** s **in "01234"**:  
 **if** s == **"5"**:  
 **break** n = n + 1  
print(n)

1. 0
2. 1
3. 5
4. 无输出
5. 以下代码的运行结果为（ ）。

n = 0  
**while** n < 10:  
 **if** n == 5:  
 **break** n = n + 2  
print(n)

1. 0
2. 5
3. 10
4. 死循环
5. 以下叙述正确的是（ ）。
6. continue语句可以用来结束整个循环。
7. 只能在循环体内使用break语句。
8. 从多层嵌套循环中退出时，只能使用goto语句。
9. 在循环体内使用break语句或continue语句的作用相同。
10. 哪个选项关于循环结构的描述是错误的（ ）
11. 死循环无法退出，所以没有任何作用。
12. 条件循环和遍历循环结构都是基本的循环结构。
13. 循环是程序根据条件判断结果向后反复执行的一种运行方式。
14. 循环是一种程序的基本控制结构。
15. 关于Python遍历循环，以下选项中描述错误的是
16. 无限循环无法实现遍历循环的功能
17. 遍历循环可以理解为从遍历结构中逐一提取元素，放在循环变量中，对于所提取的每个元素只执行一次语句块
18. 遍历循环可以通过for实现
19. 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
20. 以下内容关于函数描述正确的是（ ）。
21. 函数用于创建对象和方法
22. 函数可以让代码执行得更快
23. 函数是一段代码用于执行特定的任务
24. 以上的说法都是正确的
25. 如果函数没有使用return语句，则函数返回的（ ）。
26. 0
27. None对象
28. 任意的字符串
29. 抛出错误，因为函数必须要有返回值
30. 以下代码的运行结果为（ ）。

**def** example(\*name):  
 print(**'Hello'**, name)  
example(**'XXX'**, **'YYY'**)

1. Hello XXX

Hello YYY

1. Hello ('XXX', 'YYY')
2. Hello XXX
3. 抛出错误，因为函数只能接收一个参数。
4. 以下代码的运行结果为（ ）。

result = **lambda** x: x \* x  
print(result(5))

1. lambda x: x\*x
2. 5
3. 25
4. 5\*5
5. 以下代码的运行结果为（ ）。

**def** example(x):  
 **if** x == 1:  
 **return** 1  
 **else**:  
 **return** x + example(x - 1)  
print(example(4))

1. 10
2. 24
3. 7
4. 1
5. 如果我们需要从math模块中输出pi常量，以下代码正确的是（ ）。
6. print(math.pi)
7. print(pi)
8. from math import pi

print(pi)

1. from math import pi

print(math.pi)

1. 以下哪个符号用于从包中导入模块（ ）。
2. .
3. \*
4. ->
5. ,
6. 以下代码的运行结果为（ ）。

numbers = [3, 3, 6]  
newNumbers = tuple(map(**lambda** x: x, numbers))  
print(newNumbers)

1. [3, 3, 6]
2. (3, 3, 6)
3. [6, 6, 12]
4. (6, 6, 12)
5. 以下代码的运行结果为（ ）。

**def** example(n):  
 **if** n == 1:  
 **return** 1  
 **return** n \* example(n - 1)  
print(example(5))

1. 15
2. 24
3. 120
4. 720
5. 定义函数时函数名后面的一对小括号中给出的参数称为（ ）。
6. 名字参数
7. 形参
8. 实参
9. 类型参数
10. 以下代码的运行结果为（ ）。

**def** example(country=**'中国'**,name):  
 print(**'%s，%s'**%(name,country))  
example(**'美国'**,**'张三'**)

1. 美国，张三
2. 报错
3. 张三，美国
4. 张三，中国
5. 不定长的位置参数在传递给函数时会被封装成（ ）。
6. 集合
7. 字典
8. 元组
9. 列表
10. ‏将一个函数的运算结果返回到函数调用的地方，应使用（ ）。
11. break
12. print
13. return
14. continue
15. 当要使用一个模块中的某些功能时，可以通过（ ）语句将该模块导入。
16. export
17. import
18. include
19. load
20. 关于以下代码，说法错误的是（ ）。

**def** example(a, b):  
 c = a \*\* 2 + b  
 b = a  
 **return** c  
a = 10  
b = 100  
c = example(a, b) + a

1. 该函数名称为func
2. 执行该函数后，变量b的值为100
3. 执行该函数后，变量c的值为200
4. 执行该函数后，变量a的值为10
5. 在Python中，关于全局变量和局部变量，以下选项中描述错误的是（ ）。
6. 全局变量不能和局部变量重名
7. 全局变量一般没有缩进
8. 全局变量在程序执行的全过程有效
9. 一个程序中的变量包含两类：局部变量和全局变量
10. 关于Python的全局变量和局部变量，以下选项中描述错误的是（ ）。
11. 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
12. 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
13. 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
14. 全局变量指在函数外定义的变量，在程序执行全过程有效
15. 关于以下代码，说法错误的是（ ）。

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

1. 代码中n是可选参数
2. s是局部变量
3. example(n)函数功能为求n的阶乘
4. range()函数是Python内置函数
5. 关于Python的lambda函数，以下选项中描述错误的是（ ）。
6. f = lambda x,y:x+y 执行后，f的类型为数字类型
7. lambda函数将函数名作为函数结果返回
8. 可以使用lambda函数定义列表的排序原则
9. lambda用于定义简单的、能够在一行内表示的函数
10. 下列函数定义语句第一行中存在错误的是（ ）。
11. def func(a, b):
12. def func(\*a, b):
13. def func(a, \*b):
14. def func(a, b=1):
15. 关于函数中关键字参数的使用，以下描述中错误的是（ ）。
16. 关键字参数必须位于位置参数之前
17. 关键字参数顺序无限制
18. 关键字参数必须位于位置参数之后
19. 不得重复传入关键字参数
20. 关于lambda表达式，以下描述中错误的是（ ）。
21. lambda不是Python的保留字
22. lambda表达式也称为匿名函数
23. lambda表达式是定义函数的一种方式
24. lambda表达式的返回值是一个函数
25. 以下代码实现的功能为（ ）。

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

1. 接受用户输入的整数n，输出n的阶乘值
2. 接受用户输入的整数n，判断n是否是水仙花数
3. 接受用户输入的整数n，判断n是否是完数并输出结论
4. 接受用户输入的整数n，判断n是否是素数并输出结论
5. 以下代码的运行结果为（ ）。

ls = [**"P"**, **"p"**]  
**def** example(a):  
 ls.append(a)  
 **return**example(**"a"**)  
print(ls)

1. ['P', 'p', 'a']
2. ['a']
3. 出错
4. ['P', 'p']
5. 关于以下代码，说法错误的是（ ）。

n = 1  
**def** example(a, b):  
 c = a\*b  
 **return** c  
s = example(**"Hello"**, 3)  
print(c)

1. 打印输出字符串："HelloHelloHello"
2. c是一个局部变量
3. n是一个全局变量
4. 运行结果是出错，出错类型是NameError: name 'c' is not defined
5. 以下关于Python函数对变量的作用，错误的是（ ）。
6. 简单数据类型在函数内部用global保留字声明后，函数退出后该变量保留
7. 函数内部建立的局部变量，在函数外不可以使用
8. 全局变量在程序执行全过程有效
9. 对于组合数据类型的全局变量，如果在函数内部没有被真实创建的同名变量，则函数内部不可以直接使用并修改全局变量的值
10. 有关递归函数基例的说明，哪个是错误的（ ）
11. 递归函数必须有基例
12. 每个递归函数只能有一个基例
13. 递归函数的基例决定递归的深度
14. 递归函数的基例不再进行递归
15. ​有关函数的说法，哪个是错误的（ ）。
16. 函数是代码逻辑的封装
17. 函数对一段代码的命名
18. 函数是代码功能的一种抽象
19. 函数是计算机对代码执行优化的要求
20. 传递多个参数时各参数由（ ）分隔。
21. 分号；
22. 圆点.
23. 下划线 \_
24. 逗号，
25. 以下代码的运行结果为（ ）。

**def** example(x):  
 **if** x == 1:  
 **return** 1  
 **else**:  
 **return** x - example(x - 1)  
print(example(5))

1. 1
2. 2
3. 3
4. 4
5. Python文件只读打开模式是（ ）。
6. 'b'
7. 'w'
8. 'x'
9. 'r'
10. 不是Python对文件的打开模式的是（ ）。
11. '+'
12. 'w'
13. 'c'
14. 'r'
15. 关于Python文件打开模式的描述，以下选项中描述错误的是（ ）。
16. 追加写模式 'a'
17. 创建写模式 'n'
18. 覆盖写模式 'w'
19. 只读模式 'r'
20. 以下选项中不是 Python 对文件的操作方法的是（ ）。
21. writetext
22. seek
23. write
24. writelines
25. 以下选项中，对文件的描述错误的是（ ）。
26. 文件是一个存储在辅助存储器上的数据序列
27. 文本文件不能用二进制文件方式读入
28. 文本文件和二进制文件都是文件
29. 文件中可以包含任何数据内容
30. os.path模块检查文件是否存在的函数是（ ）。
31. isdir(path)
32. exists(path)
33. isfile(path)
34. splitext(path)
35. 文件指针可移动位置。seek(5,1)表达的含义是（ ）。
36. 从文件当前位置，向后移动5个字节
37. 从文件末尾起，向前移动5个字节
38. 从文件当前位置，向前移动5个字节
39. 从文件开头起，向后移动5个字节
40. 关于代码with open('abc.txt'','r+') as f :的描述，错误的选项是（ ）。
41. 返回文件指针f
42. 以读写方式打开文件
43. 程序自动执行close()函数，关闭文件
44. 执行代码后，abc.txt文件未关闭，必须通过close()函数关闭
45. 在读写文件之前，必须通过以下哪个方法创建文件对象（ ）。
46. create
47. folder
48. File
49. open
50. 以下选项中，不是Python中文件操作的相关函数是（ ）。
51. read()
52. write()
53. open()
54. load()
55. 文件apple.txt在当前程序所在目录内，其内容是一段文本: good，以下代码的运行结果为（ ）。

txt = open(**"apple.txt"**, **"r"**)  
print(txt)  
txt.close()

1. apple.txt
2. apple
3. good
4. 以上答案都不对
5. 以下程序输出到文件text.csv里的结果是（ ）。

a = open(**"text.csv"**, **'w'**)  
b = [11, 12, 13]  
a.write(**","**.join(str(b)))  
a.close()

1. 11,12,13
2. [11,12,13]
3. ,1,1,,, ,1,2,,, ,1,3,
4. [,1,1,,, ,1,2,,, ,1,3]
5. 设city.csv文件内容如下：‍

A,B,C,D

E,F,G

以下代码的运行结果为（ ）。

a = open(**"字母.csv"**, **"r"**)  
b = a.read().split(**","**)  
a.close()  
print(b)

1. ['A', 'B', 'C', 'D', '\n', 'E', 'F', 'G']
2. ['A', 'B', 'C', 'D\nE', 'F', 'G']
3. ['A, B, C, D, E, F, G']
4. ['A', 'B', 'C', 'D', 'E', 'F', 'G']
5. 关于以下代码，选项中描述错误的是（ ）。

fname = input(**"请输入要写入的文件："**)  
a = open(fname, **"w+"**)  
b = [**"A"**, **"B"**, **"C"**, **"D"**]  
a.writelines(b)  
a.seek(0)  
**for** line **in** a:  
 print(line)  
a.close()

1. a.writelines(b)将b列表中的全部字符串写入文件
2. a.seek(0)这行代码如果省略，也能打印输出文件内容
3. 执行代码时，从键盘键入"c.txt"，则c.txt被创建
4. 代码主要功能为向文件写入一个列表中的字符串，并打印输出结果
5. 关于以下代码的描述，错误的选项是（ ）。

**with** open(**'abc.txt'**, **'r+'**) **as** f:  
 lines = f.readlines()  
**for** item **in** lines:  
 print(item)

1. lines是列表类型
2. 执行代码后，abc.txt文件未关闭，必须通过close()函数关闭
3. item是字符串类型
4. 程序功能是打印输出abc.txt文件内容
5. 有一个文件记录了1000个人的高考成绩总分，每一行信息长度是20个字节，要想只读最后10行的内容，不可能用到的函数是（ ）。
6. write()
7. open()
8. readline()
9. seek()
10. 打开一个已有文件，然后在文件末尾添加信息，正确的打开方式为（ ）。
11. 'a'
12. 'w'
13. 'w+'
14. 'r'
15. 假设file是文本文件对象，下列选项中，哪个用于读取一行内容？（ ）
16. file.read()
17. file.readlines()
18. file.readline()
19. file.read(200)
20. 下列语句打开文件的位置应该在（ ）。

f = open(**'ABC.txt'**, **'w'**)

1. Python安装目录下
2. 与源文件在相同的目录下
3. C盘根目录下
4. D盘根目录下
5. 构造方法的作用是（ ）。
6. 对象的初始化
7. 类的初始化
8. 一般成员方法
9. 对象的建立
10. 构造方法是类的一个特殊方法，Python中它的名称为（ ）。
11. init
12. \_\_init\_\_
13. \_construct
14. 与类同名
15. ‌Python中定义私有属性的方法是（ ）。
16. 使用\_\_XX\_\_定义属性名
17. 使用\_\_XX定义属性名
18. 使用public关键字
19. 使用private关键字
20. 以下C类继承A类和B类的格式中，正确的是（ ）。
21. class C A, B:
22. class C A and B:
23. class C (A: B):
24. class C (A, B):
25. 下列选项中，不属于面向对象程序设计的三个特征的是（ ）。
26. 封装
27. 继承
28. 抽象
29. 多态
30. 以下代码的运行结果为（ ）。

**class** Student:  
 **def** \_\_init\_\_(self, n=**"AAA"**):  
 self.name = n  
 **def** show(self):  
 print(self.name)  
s = Student(**"BBB"**)  
s.show()

1. AAA
2. BBB
3. None
4. 错误
5. 以下代码的运行结果为（ ）。

**class** Student:  
 **def** \_\_init\_\_(self, n=**"AAA"**):  
 self.name = n  
 @classmethod  
 **def** show(cls):  
 s = cls()  
 print(s.name)  
s = Student(**"BBB"**)  
s.show()

1. AAA
2. BBB
3. None
4. 错误
5. 以下代码的运行结果为（ ）。

**class** Student:  
 **def** \_\_init\_\_(self, n=**"AAA"**):  
 self.name = n  
 @classmethod  
 **def** show(cls):  
 s = cls()  
 print(s.name)  
s = Student(**"BBB"**)  
Student.show(s)

1. AAA
2. BBB
3. None
4. 错误
5. 在A类的某个方法中，“super()”等价于（ ）。
6. super(self)
7. super(A,self)
8. super
9. super(A)
10. 在执行同样代码的情况下，系统会根据对象实际所属的类去调用相应类中的方法，这个特性是类的（ ）。
11. 继承性
12. 封装性
13. 多态性
14. 自适应性
15. 下列选项中，描述正确的是（ ）。
16. 子类会继承父类中定义的所有属性和方法，在子类中不允许增加新的属性和方法
17. 父类会继承子类中定义的所有属性和方法，也可以在父类中增加新的属性和方法
18. 父类会继承子类中定义的所有属性和方法，在父类中不允许增加新的属性和方法
19. 子类会继承父类中定义的所有属性和方法，也可以在子类中增加新的属性和方法
20. 已知stu1和stu2是Student类的两个对象，则执行“stu1>stu2”时会自动执行Student类的（ ）方法。
21. \_\_ge\_\_
22. \_\_lt\_\_
23. \_\_le\_\_
24. \_\_gt\_\_
25. 析构方法的方法名是（ ）。
26. \_\_destruct\_\_
27. \_\_end\_\_
28. \_\_term\_\_
29. \_\_del\_\_
30. 我们使用对象名调用类中的一个普通方法，则该方法至少有（ ）个参数。
31. 2
32. 1
33. 3
34. 0
35. 以下代码的运行结果为（ ）。

**class** Person:  
 **def** \_\_init\_\_(self, id):  
 self.id = id  
tom = Person(123)  
tom.\_\_dict\_\_[**'age'**] = 20  
print(tom.age + len(tom.\_\_dict\_\_))

1. 21
2. 22
3. 23
4. 24
5. 关于以下代码，说法正确的是（ ）。

**class** parent:  
 **def** \_\_init\_\_(self, param):  
 self.v1 = param  
**class** child(parent):  
 **def** \_\_init\_\_(self, param):  
 parent.\_\_init\_\_(self, param)  
 self.v2 = param  
odj = child(100)

1. print(odj.v1)值为0。
2. print(odj.v1==odj.v2)结果为False。
3. print(odj.v1)值为100。
4. print(odj.v1==odj.v2)会报错。
5. ​在一个方法的定义中，可以通过表达式（ ）访问实例变量x。
6. x
7. self.get(x)
8. self[x]
9. self.x
10. 假设a为类A的对象且包含一个私有数据成员“\_\_value”，那么在类的外部通过对象a直接将其私有数据成员“\_\_value”的值设置为3的语句可以写作（ ）。
11. a.\_\_value=3
12. a.\_A\_\_value=3
13. A\_\_value=3
14. A.\_\_value=3
15. 下列程序可用于查看实例对象a的成员的是（ ）。
16. help(a)
17. print(a)
18. dir(a)
19. a?
20. 下面哪项特性不是面向对象程序设计的主要特征（ ）。
21. 封装
22. 函数
23. 多态
24. 继承
25. 下列哪个说法是正确的（ ）
26. 类方法只能通过类名来调用
27. 面向对象的编程是以函数为核心的
28. \_\_init\_\_方法构造的是实例方法
29. 多态是指是指不同对象收到不同消息时产生相同的操作行为
30. 对于if语句序列的两条语句，如果第一条语句前面有4个空格、第二条语句前面有1个制表符，则运行时会产生（ ）异常。
31. IndexError
32. TabError
33. SyntaxError
34. IndentationError
35. ‏除了系统遇到错误产生异常外，我们也可以使用（ ）产生异常。
36. error
37. raise
38. exception
39. except
40. 以下代码的运行结果为（ ）。

**try**:  
 print(**'try...'**)  
 r = 10 / 0  
 print(**'result:'**, r)  
**except** ZeroDivisionError **as** e:  
 print(**'except:'**, e)  
**finally**:  
 print(**'finally...'**)  
print(**'END'**)

1. try...
2. try...

except: division by zero

1. try...

except: division by zero

finally...

1. try...

except: division by zero

finally...

END

1. 以下代码的运行结果为（ ）。

**import** numpy **as** np  
a = np.matrix([[1, 0], [0, 1]])  
b = np.dot(a, a.T)  
print(b)

1. [[1 1]

[1 1]]

1. [[1 0]

[1 1]]

1. [[1 0]

[0 1]]

1. [[1 1]

[0 1]]

1. 下列不属于Python第三方库的是（ ）。
2. pygame
3. matplotlib
4. numpy
5. random
6. 下列关于NumPy的说法中，错误的是（ ）。
7. NumPy可以处理傅立叶变换和用于图形操作的例程
8. NumPy能进行数组（限一维数组）的算术和逻辑运算
9. NumPy拥有线性代数和随机数生成的内置函数
10. NumPy系统主要用于数学/科学计算，提供了许多高级的数值编程工具
11. 已知import numpy as np，无法使用numpy创建数组的操作是（ ）。
12. np.ones(8)
13. np.zeros(8)
14. np.empty(8)
15. np.arrays(8)
16. 以下关于Python数值运算描述错误的是（ ）。
17. Python内置支持复数运算，可以使用j或者J来表示。
18. Python支持+=、%=这样的增量赋值操作符。
19. 在Python 3.x中，10 / 3 == 3的判别结果是True。
20. %运算符表示运算对象取余。
21. 以下关于字符串处理的描述错误的是（ ）。
22. "\"符号可以被用来表示转义符号，例如'doesn\'t'和"doesn't"都表示字符串doesn't。
23. 被单引号('...')或者双引号("...")包围的都是字符串。
24. 如果字符串赋值的时候，内部有单引号或双引号时如果不想使用转义字符常常可以使用r运算符来表示。
25. 打印语句print('C:\file\time')的输出结果是C:\file\time。
26. 结构化程序设计的三种基本的控制结构不包括（ ）。
27. 选择
28. 顺序
29. 递归
30. 循环
31. 以下关于列表表述中错误的是（ ）。
32. 可以用“==”比较两个列表是否相同。
33. Python的列表可以嵌套，形成二维列表。
34. 列表与字符串一样都是不可变的。
35. “+”号可以用来拼接两个列表。
36. 在Python中，以下关于函数的描述错误的是（ ）。
37. 在Python中，默认参数的值可以修改。
38. 在Python中，函数的return语句可以以元组tuple的方式返回多个值。
39. 在Python中，example为一个函数，执行map(example, lst)则可将example函数映射到lst中的每一个元素上。
40. 在Python的函数中，有且只能有一个return语句。
41. 以下关于Python的说法中正确的是（ ）。
42. Python中函数的返回值如果多于1个，则系统默认将它们处理成一个字典。
43. 递归调用语句不允许出现在循环结构中。
44. 在Python中，一个算法的递归实现往往可以用循环实现等价表示，但是大多数情况下递归表达的效率要更高一些。
45. 可以在函数参数名前面加上星号\*，这样用户所有传来的参数都被收集起来然后使用，星号在这里的作用是收集其余的位置参数，这样就实现了变长参数。
46. 以下关于Python自带数据结构的运算结果中错误的是（ ）。
47. l = [1, 2, 3, 4]; l.reverse(); 则l[1]为3。
48. l = [1, 2, 3, 4]; l.pop(1); 则l结果为[1, 3, 4]。
49. l = [1, 2, 3, 4]; l.pop(); 则l.index(3) 结果为2。
50. l = [1, 2, 3, 4]; l.insert(2, -1); 则l为 [1, 2, -1, 4]。
51. 以下关于Python自带数据结构的运算结果中正确的是（ ）。
52. l = [2, 1, 3, 5, 4]; l.remove(3); l.sort(); 则运算之后l为[1, 2, 4, 5]。
53. l = [1, 2, 3, 4, 5]; del l[2:4]; 则运算之后l为[1, 2, 3]。
54. basket = ['apple', 'banana', 'apple', 'orange'] ; fruit = set(basket);

fruit2 = set(['apple', 'melo']); len(fruit | fruit2)的结果是5。

1. basket = ['apple', 'banana', 'apple', 'orange'] ; fruit = set(basket); len(fruit) 的运算结果是4。
2. 以下关于Python自带数据结构的运算结果中正确的是（ ）。
3. scores = {'Jack': 90, 'Mike': 80, 'Jay': 85, 'Bill': 60}; del scores['Bill']; len(scores)的结果是6。
4. scores = {'Jack': 90, 'Mike': 80, 'Jay': 85, 'Bill': 60}; 无法执行命令 scores['Bill']=90; 因为原来的字典中已经有了一个值为90了。
5. scores = {'Jack': 90, 'Mike': 80, 'Jay': 85, 'Bill': 60}; s = dict(Jack=90, Mike=80, Jay=85, Bill=60); 则scores == s 判断结果是False。
6. scores = {'Jack': 90, 'Mike': 80, 'Jay': 85, 'Bill': 60}; sorted(scores.keys())的运算结果是 ['Bill', 'Jack', 'Jay', 'Mike']。
7. 以下描述中错误的是（ ）。
8. 打开一个文本文件，再对其进行读取，f = open('test.txt', 'r+'); f.read() ;read函数内不带参数是读取整个文本文件内容并返回。
9. Python中的同一种序列互相之间可以进行比较，按照字典序进行排序，例如 (1, 2, 3, 4) < (1, 2, 4) 的运算结果是True。
10. 字符串的格式化，'{:.2f}'.format(math.pi) 与 '%.2f' % math.pi 在运算结果上相同。
11. 打开一个文本文件，f = open('test.txt', 'r+'); 则 f.seek(5) 表示在打开的文件里寻找数字5。
12. 以下不能作为字典的key的是（ ）。
13. 'num'
14. tupleA = ('sum')
15. 123
16. listA = ['className']
17. 面向对象思想的程序设计中通常的使用顺序是（ ）。
18. 定义类—创建实例—通过实例使用属性或方法
19. 定义类—通过实例使用属性或方法—创建实例
20. 创建实例—通过实例使用属性或方法—定义类
21. 创建实例—定义类—通过实例使用属性或方法
22. 对于一个列表aList和一个元组bTuple，以下函数和对象方法使用错误的选项是（ ）。
23. sorted(aList)
24. bTuple.sort()
25. sorted(bTuple)
26. aList.sort()
27. 定义函数如下：

**def** example(\*numbers):   
 s = 1  
 **for** n **in** numbers:   
 s = s \* n + n   
 **return** s

以下（ ）的输入可以得到21的结果？

1. example([3, 2, 1])
2. example([3, 3])
3. nums = [1, 2, 3]; example(nums)
4. nums = (3, 3); example(\*nums)
5. 以下不属于无监督学习的算法是（ ）。
6. PCA
7. KMeans
8. KNN
9. DBSCAN
10. ‍以下说法正确的是（ ）。
11. 决策树只能用于二分类问题
12. 监督学习与无监督学习的主要区别是，监督学习的训练样本无标签
13. 分类任务的评价指标精确率和准确率是同一个概念
14. 神经网络可以用于多分类问题
15. Python3.x环境创建Python脚本文件时，需要将文件编码格式设置为（ ）。
16. ANSI
17. GBK
18. UTF-8
19. UNICODE
20. 下列选项中，不能创建字典对象的语句是（ ）。
21. dict('one':1, 'two':2, 'three':3)
22. {'one':1, 'two':2, 'three':3}
23. dict(zip(['one','two','three'], [1,2,3]))
24. dict([('one',1), ('two',2), ('three',3)])
25. 下面选项中，描述错误的是（ ）。
26. 如果要创建的目录已经存在，则os.makedirs函数不会报错
27. 如果要删除的目录存在但目录不为空，则os.rmdir函数会报错
28. 如果要创建的目录已经存在，则os.mkdir函数会报错
29. 如果要删除的目录不存在，则os.rmdir函数会报错
30. 用if语句统计符合报名条件的人数，条件是男性年龄大于18岁小于60岁，下列正确的语句是（ ）。
31. if sex=='男' and age<60 or age>18: n+=1
32. if sex=='男' or (age<60 or age>18): n+=1
33. if sex=='男' and (age<60 and age>18): n+=1
34. if sex=='男' or age<60 and age>18: n+=1
35. 闰年的判定条件是能被400整除，或者能被4整除但不能被100整除，正确的Python表达式为（ ）。
36. year%400==0 and year%4==0 and year % 100!=0
37. (year//400==0) or (year//4==0 and year//100!=0)
38. (year%400==0) or (year%4==0 and year % 100!=0)
39. year//400==0 or year//4==0 and year//100!=0
40. 以下for语句中,不能完成1-10的累加功能的是（ ）。
41. for i in range(10,-1,-1): sum+=i
42. for i in range(10,0): sum+=i
43. for i in range(1,11): sum+=i
44. for i in(10,9,8,7,6,5,4,3,2,1): sum+=i
45. 关于break语句与continue语句的说法中，以下选项中不正确的是（ ）。
46. continue结束整个循环过程，不再判断循环的执行条件
47. continue用来结束当前当次语句，但不跳出当前的循环体
48. Python通过for、while等保留字构建循环结构
49. 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
50. ‍以下关于Python程序的基本语法元素，错误的描述是（ ）。
51. Python语言只能用4个空格的缩进来实现程序的强制可读性
52. 变量是由用户定义的用来保存和表示数据的一种语法元素
53. 变量标识符是一个字符串，长度是没有限制的
54. 变量的命名规则之一是名字的首位不能是数字
55. 关于基本输入输出函数的描述，错误的选项是（ ）。
56. eval()函数的参数是“3\*4”的时候，返回的值是整数“12”
57. print()函数的参数可以是一个函数，执行结果是显示函数返回的值
58. 当print()函数输出多个变量的时候，可以用逗号分隔多个变量名
59. 当用户输入一个整数“6”的时候，input()函数返回的也是整数“6”
60. ‌一般整数用十进制表示，若用其他进制表示一个数，错误的描述选项是（ ）。
61. 1010b表示一个二进制数
62. 0b1010表示一个二进制数
63. 0o1010表示一个八进制数
64. 0x1010表示一个十六进制数
65. ‎关于Python数据类型的描述,正确的选项是（ ）。
66. 函数lower(x)是将字符串x的首字母小写
67. 内置函数divmod(x,y)的运算结果是两个整数：x除y的整数商及余数
68. 函数ord(x)是返回字符串x对应的Unicode编码
69. 运算符+、-、\*、/ 等跟赋值符号 = 相连，形成增强赋值操作符
70. 关于打开文件函数open(<文件路径名>,<打开模式>)中打开模式的描述，正确的选项是（ ）。
71. ’r’表示只读模式打开文件，如果文件不存在，就会返回异常
72. ‘a’表示追加模式打开文件，如果文件不存在，就返回异常
73. ‘w’表示写模式打开文件，如果文件存在，就会在文件尾继续写
74. ‘b’表示二进制文件模式打开文件，可以单独作为open函数的参数
75. 以下正确的描述是（ ）。
76. 函数的定义和函数的调用均可嵌套
77. 函数的定义和函数的调用均不可以嵌套
78. 函数的定义不可以嵌套，但函数的调用可以嵌套
79. 函数的定义可以嵌套，但函数的调用不可以嵌套
80. 关于列表数据结构，下面描述正确的是（ ）。
81. 不支持 in 运算符
82. 可以不按顺序查找元素
83. 必须按顺序插入元素
84. 所有元素类型必须相同
85. 关于递归的说法正确的是（ ），
86. 递归编写的程序代码通常比非递归的少，所以递归就是比非递归好
87. 在能够使用递归函数的时候，尽量使用递归，因为它可以使得程序变得简洁，易于理解
88. 递归是一种程序调用自身的现象
89. 递归函数的执行效率优于非递归函数
90. 以下哪个选项可以创建一个 4x4 的单位矩阵（ ）。
91. np.range(4,4)
92. np.eye[4]
93. np.zeros(4)
94. np.eye(4)
95. 关于pandas库的DataFrame对象，哪个说法是正确的（ ）。
96. DataFrame是二维带索引的数组，索引可自定义
97. DataFrame与二维ndarray类型在数据运算上方法一致
98. DataFrame只能表示二维数据
99. DataFrame由2个Series组成
100. 下面两段代码，哪个说法不正确（ ）。

**import** numpy **as** np  
a = np.array([2, 4, 6, 8, 9])  
  
**import** pandas **as** pd  
b = pd.Series([2, 4, 6, 8, 9])

1. a参与运算的执行速度比b快
2. a和b表达同样的数据内容
3. a和b是不同的数据类型，不能直接运算
4. a和b都是一维数据
5. 以下哪一个步骤不属于数据清洗（ ）。
6. 去重
7. 删除缺失值
8. 异常值检测
9. 数据合并
10. 以下哪个代码表示添加图例（ ）。
11. plt.legend()
12. plt.show()
13. plt.title()
14. plt.figure()
15. 以下说法错误的是（ ）。
16. lines.markersize 表示点的数量
17. lines.linewidth 表示线条宽度
18. lines.marker 表示线条上点的形状
19. lines.linestyle 表示线条样式
20. 一般说，numpy-matplotlib-pandas是数据分析和展示的一条学习路径，哪个是对这三个库不正确的说明（ ）
21. matplotlib支持多种数据展示，使用pyplot子库即可
22. numpy底层采用C实现，因此，运行速度很快
23. pandas也包含一些数据展示函数，可不用matplotlib
24. pandas仅支持一维和二维数据分析，多维数据分析要用numpy
25. 以下哪个选项可以创建一个范围在(0,10)之间，长度为16的等差数列（ ）。
26. np.linspace(0,16,10)
27. np.random(0,10,16)
28. np.linspace(0,10,16)
29. np.randint(0,16,10)
30. 下面哪个对matplotlib库的描述，哪个不正确（ ）。
31. matplotlib库是基于numpy库设计实现的
32. matplotlib库展示的数据图无法输出为文件
33. matplotlib.pyplot是绘图的一种“快捷方式”
34. matplotlib是Python第三方数据展示库
35. 关于 Python 序列类型的通用操作符和函数，以下选项中描述错误的是（ ）。
36. 如果 s 是一个序列，s = [1,“kate”,True]，s[3] 返回 True
37. 如果 x 是 s 的元素，x in s 返回 True
38. 如果 x 不是 s 的元素，x not in s 返回 True
39. 如果 s 是一个序列，s = [1,“kate”,True]，s[–1] 返回 True
40. 判断题
41. Python语言是用来编写程序的高级编程语言。（√）
42. 程序设计是指设计、编制、调试程序的方法和过程。（√）
43. 通过命令行窗口可以进入Python并在交互式解释器中编写代码。（√）
44. IPython是基于CPython之上的一个交互式解释器。（√）
45. 计算机的信息数据是用二进制数来处理的。（√）
46. Python中的变量需要声明才能使用。（×）
47. Python中的变量名可以以“@”符号开头。（×）
48. Python中的变量名必须以小写字母开头，否则会报错。（×）
49. 变量是指在程序运行过程中值会发生变化的量。（√）
50. 在Python中，变量是一种数据类型。（×）
51. Python的字符串是不可变的数据类型。（√）
52. Python不支持单字符类型，单字符也是作为一个字符串使用的。（√）
53. Python不允许空字符串的存在。（×）
54. 格式化字符串的方法可以使用字符串的format()方法。（√）
55. 按位或运算符的意思是：只要对应的二个二进位有一个为1时，结果位就为1。（√）
56. not in表示如果在指定的序列中没有找到值返回True，否则返回False。（√）
57. Python中的表达式是不存在优先级顺序的。（×）
58. 列表的数据项需要具有相同的类型。（×）
59. 列表是一种有序的集合，可以随时添加和删除其中的元素。（√）
60. 使用len()函数可以获得列表元素的个数。（√）
61. 在Python中，列表的索引是从1开始的。（×）
62. 在Python中可以使用insert()方法删除列表末尾的元素。（×）
63. 元组一旦初始化就不能修改。（√）
64. 列表和元组是Python内置的有序集合，前者可变，后者不可变。（√）
65. 对列表使用sort()方法，默认是降序排列。（×）
66. a为列表，其中每个元素均为int型数值，b为int型，当进行a+b操作时，会对a中每个元素加上b的值。（×）
67. a、b均为字符串，进行a+b操作，可直接将两个字符串进行连接。（√）
68. 凡是用花括号(即{})括起来中间用逗号隔开元素的数据结构都叫字典。（×）
69. 任何不可变对象均可作为字典的键。（√）
70. 切片操作属于浅复制的实现方式之一。（√）
71. 在字典中，要删除一个key，用pop(key)方法，对应的value也会从dict中删除。（√）
72. 字典内部存放的顺序和key放入的顺序是一致的。（×）
73. 集合是一个无序的不重复元素序列。（√）
74. 布尔值只有True、False两种值。（√）
75. and运算是与运算，只要其中有一个为True，and运算结果才是True。（×）
76. not运算是非运算，它是一个单目运算符。（√）
77. 条件语句是通过一条或多条语句的执行结果来决定执行的代码块。（√）
78. 条件语句每个条件后面都要使用冒号。（√）
79. 条件判断从上向下匹配，当满足条件时执行对应的块内语句，后续的elif和else会依次执行。（×）
80. Python中的循环语句有for和do while。（×）
81. Python的for循环可以遍历任何可迭代对象，如一个列表。（√）
82. continue语句只能跳出当前层次的循环。（×）
83. 无限循环while一直保持循环操作,当循环条件不满足时就结束。（√）
84. 循环退出条件设置不当的while循环会导致死循环。（√）
85. Python循环嵌套的内循环控制变量与外循环变量可以同名。（×）
86. 以下代码的输出结果是Sumer。（×）

**for** i **in "****Summer"**:  
 **if** i == **"m"**:  
 **break**print(i)

1. Python中定义一个函数要使用def语句，并依次写出函数名、括号、括号中的参数和冒号。（√）
2. 函数体内部的语句在执行时，一旦执行到return时，函数就执行完毕，并将结果返回。（√）
3. pass是跳出函数的语句。（×）
4. 调用函数时，如果参数个数不对，Python会抛出TypeError。（√）
5. 函数不可返回多个值，如果有多个返回值，需要定义多个函数。（×）
6. 定义函数时，需要确定函数名和参数个数。（√）
7. 函数体内部可以用return随时返回函数结果。（√）
8. 函数默认参数可以使用列表。（×）
9. \*args是可变参数，args接收的是一个列表。（×）
10. \*\*kw是关键字参数，kw接收的是一个字典。（√）
11. 凡是可作用于for循环的对象都是可迭代对象。（√）
12. map将传入的函数依次作用到序列的每个元素，并把结果作为新的迭代器返回。（√）
13. 自定义函数可提高代码的执行速度。（×）
14. 自定义函数可增强代码可读性。（√）
15. 一个自定义函数只能包含一个return语句。（×）
16. Python程序中对一个函数的调用不能出现在该函数的定义之前。（√）
17. Python函数支持可变数量的参数，实参用“ \*参数名 ”表示。（×）
18. 调用函数时，如果实参与函数定义的形参顺序不一致，实参中需指出参数名。（√）
19. ‍Python程序中一定要有一个主函数。（×）
20. 函数可以返回多个参数值，需要形成一个列表来返回。（×）
21. 调用函数之后，实参的值是否改变取决于函数中的操作，与实参变量类型无关。（×）
22. 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。（×）
23. 可以基于多个已有类创建新类。（√）
24. 定义类的同时必须定义好该类的所有属性和方法。（×）
25. 定义一个类后还可以为其动态添加属性和方法。（√）
26. 可以基于一个已有类创建新类。（√）
27. 已知B是A的父类，b是B类的对象，则“A==type(b)”返回结果为True。（×）
28. 使用获取到的父类代理对象可以在子类中调用被重写的父类方法。（√）
29. 方法重写是指子类可以对从父类中继承过来的方法进行重新定义，从而使得子类对象可以表现出与父类对象不同的行为。（√）
30. 如果一个子类有两个或更多父类，则这种继承关系称为多重继承。（√）
31. 构造器\_\_init\_\_在实例化对象时调用。（√）
32. \_\_str\_\_(self)方法用来把字符串转换为对象。（×）
33. 类方法不需要以self作为第一个参数。（√）
34. 使用@staticmethod装饰器修饰的方法称之为类方法。（×）
35. 当父类方法的功能不能满足需求时，可以在子类中重写父类的方法，该过程称之为方法重载。（×）
36. 私有属性只能在类的内部进行调用，无法在类的外部访问。（×）
37. Python3的新式类都继承于object。（√）
38. Python中类的特性能够实现利用属性来控制或调用方法。（×）
39. 对于Python的私有成员而言，类本身和子类均可以访问它。（×）
40. 如果一个except子句捕获多个异常，则多个异常名应写为元组的形式。（√）
41. 已知有语句“assert num==0”，则当num的值为0时会引发AssertionError异常。（×）
42. 如果一个异常没有与任何的 except 匹配，那么这个异常将会传递给上层的 try 中。（√）
43. 一个 try 语句可能包含多个except子句，分别来处理不同的特定的异常。（√）
44. 一个except子句可以同时处理多个异常，这些异常将被放在一个列表。（×）
45. Python的错误其实也是class，所有的错误类型都继承自BaseException。（√）
46. 使用try...except捕获错误可以跨越多层调用。（√）
47. pandas的Series可以看成是一个定长的有序字典。（√）
48. NumPy 的ndarray是一种多维数组对象，可以由序列型对象生成。（√）
49. dtype是一种特殊的对象，其含有将ndarray解释为特定数据类型所需的信息，int64表示有符号的64位整型。（√）
50. pandas的DataFrame是一个表格型数据结构，含有一组无序的列，每列可以是不同值类型（数值、字符串、布尔值等等）。（×）
51. ‌K-means是一种聚类学习算法，K代表用户最初在空间中选定的中心数量。（×）
52. ‌Python中无需定义变量类型，根据“值”确定类型，并以“引用”的方式实现赋值操作。（√）
53. ‍在循环控制语句中，有break，continue和pass等控制流关键词。（√）
54. 在Python的函数参数定义中，可以有一些位置参数和一些关键字参数，其中关键字参数让调用者通过使用参数名区分参数，允许改变参数列表中的参数顺序。（√）
55. 执行range(N1, N2)可生成N2-N1+1个整数。（×）
56. 若要在列表l后插入元素x，可以使用l.append(x)，也可以使用l[len(l):] = [x]。（√）
57. ‎对于与循环语句匹配的else语句，如果循环代码从break处终止跳出循环，则执行该循环的else中的语句。（×）
58. ‍Python中如果某个文件的打开模式是“r+”，则将文件指针移动到文件开头，调用f.write('hahaha')则可将字符串“hahaha”插入到文件的开头。（×）
59. ‎Python中的for语句可以在任意序列上进行迭代访问，例如列表、字符串和元组。（√）
60. Python字典的内置函数中没有append()操作，可以用update()来更新字典内容。（√）
61. Python程序中每条语句以分号结尾。（×）
62. ‎使用“import M1; M2; M3”可以一次导入M1、M2和M3这3个模块。（×）
63. ‌\_\_str\_\_方法的返回值可以是整数。（×）
64. 双分支结构里,if和else后面的冒号可以省略。（×）
65. 在异常处理结构中，不论是否发生异常，finally子句中的代码总是会执行的。（√）
66. 同一个集合中的元素之间不允许重复，如果将重复元素加入同一个元素，将会导致程序出错。（×）
67. Python 变量使用前必须先声明， 并且一旦声明就不能再当前作用域内改变其类型。（×）
68. Python 不允许使用关键字作为变量名， 允许使用内置函数名作为变量名，但这会改变函数名的含义。（√）
69. Python 变量名必须以字母或下划线开头，并且区分字母大小写。（√）
70. 9999\*\*9999 这样的命令在 Python 中无法运行。（×）
71. 3+4j 不是合法的 Python 表达式。（×）
72. Python 代码的注释只有一种方式， 那就是使用#符号。（×）
73. 为了让代码更加紧凑， 编写 Python 程序时应尽量避免加入空格和空行。（×）
74. 生成器推导式比列表推导式具有更高的效率。（√）
75. Python 列表中所有元素必须为相同类型的数据。（×）
76. 使用 Python 列表的方法 insert()为列表插入元素时会改变列表中插入位置之后元素的索引。（√）
77. 无法删除集合中指定位置的元素， 只能删除特定值的元素。（√）
78. 假设 x 是含有 5 个元素的列表，那么切片操作 x[1 0:]是无法执行的，会抛出异常。（×）
79. 计中， 函数和方法是完全一样的， 都必须为所有参数进行传值。（×）
80. 在函数内部没有任何声明的情况下直接为某个变量赋值， 这个变量一定是函数内部的局部变量。（√）
81. 执行循环语句for i in range(1,5):pass后，变量i的值是4。（√）
82. 循环语句for i in range(-3,21-4)的循环次数为5。（×）
83. Python包含了数量众多的模块，通过import语句，可以导入模块，并使用其定义的功能。（√）
84. 填空题
85. Python中的complex表示\_\_\_\_\_\_\_。（复数）
86. 已知x=[3,6，9]，那么执行语句x[1:]=[2]之后，x的值为\_\_\_\_\_\_\_。（[3, 2]）
87. Python安装扩展库常用的是\_\_\_\_\_\_\_工具。（pip）
88. Python是一种\_\_\_\_\_\_\_语言，即可以在一个 Python 提示符 >>> 后直接执行代码。（交互式）
89. 计算机语言有三种类型：机器语言、汇编语言和\_\_\_\_\_\_\_。（高级语言）
90. 高级语言源程序的翻译有两种方式：编译方式、\_\_\_\_\_\_\_方式。（解释）
91. 程序设计过程的3个阶段是编程阶段、编译阶段和\_\_\_\_\_\_\_。（执行阶段）
92. 可以通过交互式解释器、命令行脚本和\_\_\_\_\_\_\_三种方式运行Python。（集成开发环境）
93. 变量本身类型不固定的语言称之为\_\_\_\_\_\_\_。（动态语言）
94. 使用内置函数\_\_\_\_\_\_\_可以把十进制数字转换成对应的字符。（chr()）
95. 使用内置函数\_\_\_\_\_\_\_可以计算字符串的字符数。（len()）
96. Python中浮点数的占位符为\_\_\_\_\_\_\_。（%f）
97. Python中的按位运算符是把数字看作\_\_\_\_\_\_\_进制来进行计算的。（二）
98. \_\_\_\_\_\_\_运算符是判断两个标识符是不是引用自一个对象。（is）
99. not是一种\_\_\_\_\_\_\_运算符。（逻辑）
100. Python代码由\_\_\_\_\_\_\_和语句组成，并由Python解释器负责执行。（表达式）
101. 当列表的索引超出了范围时，Python会报一个\_\_\_\_\_\_\_错误。（IndexError）
102. 如果要取列表最后一个元素，除了计算索引位置外，还可以用\_\_\_\_\_\_\_做索引，直接获取最后一个元素。（-1）
103. 只有1个元素“1”的tuple定义时如何表示？\_\_\_\_\_\_\_ （（1，））
104. 字典使用\_\_\_\_\_\_\_存储数据。（键值对）
105. 在集合中使用\_\_\_\_\_\_\_函数移除指定元素。（remove()）
106. print()函数默认是以\_\_\_\_\_\_\_作为其结束值的。（换行符）
107. \_\_\_\_\_\_\_运算是或运算，只要其中有一个为True，运算结果就是True。（or）
108. Python条件语句中elif是\_\_\_\_\_\_\_的缩写。（else if）
109. \_\_\_\_\_\_\_语句用于跳出当前循环体。（break）
110. 以下代码的运行结果为\_\_\_\_\_\_\_。（3）

a = 3  
b = 2  
x = a **if** a > b **else** b  
print(x)

1. 对于带有 else 子句的 for 循环和 while 循环，当循环因循环条件不成立而‌自然结束时\_\_\_\_\_\_\_\_(会/不会)执行 else 中的代码。（会）
2. 表达式sum(range(1, 10, 2))的值为\_\_\_\_\_\_。（25）
3. 下面代码的输出结果是\_\_\_\_\_\_。（0,2,4,6,8,）

**for** i **in** range(10):  
 **if** i % 2 != 0:  
 **continue  
 else**:  
 print(i, end=**","**)

1. 下面代码的输出值的个数是\_\_\_\_\_\_个。（12）

age = 23  
start = 2  
**if** age % 2 != 0:  
 start = 1  
**for** x **in** range(start, age + 2, 2):  
 print(x)

1. 在函数内部，可以调用其他函数。如果一个函数在内部调用自身本身，这个函数就是\_\_\_\_\_\_。（递归函数）
2. 如下函数返回两个数的平方和，如果只给一个变量，则另一个变量的默认值为整数10，请补充横线处代码。（a,b=10）

**def** psum(\_\_\_\_\_\_):  
 **return** a \*\* 2 + b \*\* 2

1. 使用\_\_\_\_\_\_关键字声明匿名函数。（lambda）
2. f = lambda y=3, z=2: y\*\*z，则语句 print(f(5))的输出结果为\_\_\_\_\_\_。（25）
3. 已知 g = lambda x, y=3, z=5: x\*y\*z，则语句 print(g(1)) 的输出结果为\_\_\_\_\_\_。（15）
4. 已知函数定义 def func(\*\*p):return sum(p.values())，那么表达式 func(x=1, y=2, z=3) 的值为\_\_\_\_\_\_\_\_。（6）
5. 如果函数中没有return语句或者return语句不带任何返回值，那么该函数的返回值为\_\_\_\_\_\_\_\_。（None）
6. 以下代码的运行结果为\_\_\_\_\_\_\_\_。（3）

**def** foo():  
 m = 1  
 **def** bar():  
 n = 2  
 **return** m + n  
 m = bar()  
 print(m)  
foo()

1. 下面程序打开一个文件，逐行打印，请完善程序。（with）

\_\_\_\_\_open(**"1.py"**, **"r"**) **as** f:  
 **while True**:  
 aLine = f.readline()  
 **if** aLine != **''**:  
 print(aLine)  
 **else**:  
 **break**

1. 如果目录"d:\python"不存在，则创建该目录，请完善程序。（os.path.exists）

**import** os  
**if not** \_\_\_\_\_\_\_\_(**"d:\python"**):  
 os.mkdir(**"d:\python"**)

1. 以下代码的运行结果为\_\_\_\_\_\_\_\_。（100）

**class** Account:  
 **def** \_\_init\_\_(self,id):  
 self.id=id  
 id=888  
acc=Account(100)  
print(acc.id)

1. 以下代码的运行结果为\_\_\_\_\_\_\_\_。（11）

**class** A:  
 **def** \_\_init\_\_(self,a,b,c):  
 self.x=a+b+c  
a=A(6,2,3)  
print(a.x)

1. 类对象支持两种操作：属性引用和\_\_\_\_\_\_\_\_。（实例化）
2. 类的方法与普通的函数只有一个特别的区别——它们必须有一个额外的第一个参数名称, 按照惯例它的名称是\_\_\_\_\_\_\_\_。（self）
3. 在类的内部，使用\_\_\_\_\_\_\_\_关键字来定义一个方法。（def）
4. 由于Python是\_\_\_\_\_\_\_\_语言，根据类创建的实例可以任意绑定属性。（动态）
5. 即便 Python 程序的语法是正确的，在运行它的时候，也有可能发生错误。运行期检测到的错误被称为\_\_\_\_\_\_\_\_。（异常）
6. 请完善以下代码：（reshape）

**import** numpy **as** np  
a = np.array([[1, 2, 3], [4, 5, 6]])  
*# print(a.shape)*b = a.\_\_\_\_\_\_\_\_(3, 2) *# 将数组a变为3行2列*print(b)

1. 请完善以下代码：（figure）

**import** matplotlib.pyplot **as** plt, numpy **as** np  
x = np.linspace(0, 1, 20) *# x是0，0.05,0.1...*y1 = x \* x  
plt.\_\_\_\_\_\_\_\_\_()  
plt.rcParams[**'font.sans-serif'**] = **'SimHei'**plt.plot(x, y1, color=**'green'**, marker=**'o'**)  
plt.xlabel(**'节点数(\*100)'**)  
plt.ylabel(**'能耗(J)'**)  
plt.show()

1. 查看变量类型的Python内置函数是\_\_\_\_\_\_。（type()）
2. 函数可以有多个参数，参数之间使用\_\_\_\_\_\_\_\_分隔。（逗号）
3. 如果想在函数中修改全局变量，需要在变量的前面加上\_\_\_\_\_\_\_\_关键字。（global）
4. 使用readlines方法把整个文件中的内容进行一次性的读取，返回的是一个 \_\_\_\_\_\_\_\_。（列表）
5. Python 语句 x=True; y=False; z=False; print(x or y and z)的程序运行结果是\_\_\_\_\_\_\_\_\_\_。（True）
6. 表达式((2>=2) or (2<2)) and 2的值为多少？\_\_\_\_\_\_\_\_\_（2）
7. 已知列表 x = [1 .0, 2.0, 3.0]， 那么表达式 sum(x)/len(x) 的值为

\_\_\_\_\_\_\_\_\_\_\_。 （2.0）

1. 已知列表 x = [1 , 2, 3] 和 y = [4, 5, 6]， 那么表达式 [(i,j) for i, j in zip(x,y) if i==3] 的值为\_\_\_\_\_\_\_\_\_\_\_\_。 （[(3, 6)]）
2. 表达式 3 not in [1 , 2, 3]的值为\_\_\_\_\_\_\_\_\_\_。 （False）
3. 表达式 [5 for i in range(3)] 的值为\_\_\_\_\_\_\_\_\_。 （[5, 5, 5]）
4. 表达式 'Hello world!'.count('l') 的值为\_\_\_\_\_\_\_\_\_。 （3）
5. 使用上下文管理关键字\_\_\_\_\_\_\_\_\_\_\_可以自动管理文件对象， 不论何种原因结束该关键字中的语句块，都能保证文件被正确关闭。 （with）
6. 表达式 {1 , 2, 3} - {3, 4, 5} 的值为\_\_\_\_\_\_\_\_\_。 （{1 , 2}）
7. 假设 math 标准库已导入， 那么表达式 eval('math.sqrt(4)') 的值为\_\_\_\_\_\_\_\_\_。 （2.0）
8. 代码 print(1 ,2,3,sep=':') 的执行结果为\_\_\_\_\_\_\_\_。 （1:2:3）
9. 表达式 chr(ord('A')+2) 的值为\_\_\_\_\_\_\_\_\_\_。 （'C'）
10. 表达式 list(map(lambda x: x+5, [1 , 2, 3, 4, 5])) 的值为\_\_\_\_\_\_\_\_\_。 （[6, 7, 8, 9, 10]）
11. 表达式 [5 , 6] \* 2 的值为\_\_\_\_\_\_\_\_。 （[5 , 6, 5 , 6]）
12. 已知列表 x = [1,3,5,7]， 那么执行语句 del x[1 ] 之后 x 的值为\_\_\_\_\_\_\_\_\_\_\_\_。 （[1,5,7]）
13. 配伍题
14. 请为名词①-③选择与之匹配的样例，将配对好的a-c填写到括号中。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ① | 列表 |  | a | {‘abc’: 123} |
| ② | 元组 | b | [1,2,3] |
| ③ | 字典 | c | (1,2,3) |

1. （b）
2. （c）
3. （a）
4. 已知如下函数，请为代码①-④选择与之匹配的样例，将配对好的a-d填写到括号中。

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

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ① | print(fib(2)) |  | a | 2 |
| ② | print(fib(3)) | b | 3 |
| ③ | print(fib(4)) | c | 1 |
| ④ | print(fib(5)) | d | 5 |

1. （c）
2. （a）
3. （b）
4. （d）
5. 请为名词①-③选择与之匹配的解释，将配对好的a-c填写到括号中。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ① | 迭代器 |  | a | 对可迭代对象进行迭代的方式或容器 |
| ② | 生成器 | b | 使其他函数在不改变任何代码的前提下增加额外功能 |
| ③ | 装饰器 | c | 在Python中边循环边计算的机制 |

1. （a）
2. （c）
3. （b）
4. 请为函数①-③选择与之匹配的功能，将配对好的a-c填写到括号中。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ① | map() |  | a | 使用function(x,y)函数，将序列缩减成1个元素结果 |
| ② | reduce() | b | 使用返回bool的function对序列过滤，返回满足条件的结果。 |
| ③ | filter() | c | 给序列的每个元素应用一个函数，返回一个迭代器 |

1. （c）
2. （a）
3. （b）
4. 请为open()函数中mode参数①-⑤选择与之匹配的功能，将配对好的a-e填写到括号中。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ① | r |  | a | 打开一个文件用于读写。文件指针将会放在文件的开头。 |
| ② | r+ | b | 打开一个文件只用于写入。如果该文件已存在则打开文件，并从开头开始编辑，即原有内容会被删除。如果该文件不存在，创建新文件。 |
| ③ | w | c | 写模式，新建一个文件，如果该文件已存在则会报错。 |
| ④ | a | d | 以只读方式打开文件。文件的指针将会放在文件的开头。这是默认模式。 |
| ⑤ | x | e | 打开一个文件用于追加。如果该文件已存在，文件指针将会放在文件的结尾。也就是说，新的内容将会被写入到已有内容之后。如果该文件不存在，创建新文件进行写入。 |

1. （d）
2. （a）
3. （b）
4. （e）
5. （c）
6. 请为file对象常用函数①-⑤选择与之匹配的功能，将配对好的a-e填写到括号中。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ① | file.close() |  | a | 将字符串写入文件，返回的是写入的字符长度。 |
| ② | file.read([size]) | b | 关闭文件。关闭后文件不能再进行读写操作。 |
| ③ | file.tell() | c | 刷新文件内部缓冲，把内部缓冲区的数据立刻写入文件, 而不是被动等待输出缓冲区写入。 |
| ④ | file.write(str) | d | 返回文件当前位置。 |
| ⑤ | file.flush() | e | 从文件读取指定的字节数，如果未给定或为负则读取所有。 |

1. （b）
2. （e）
3. （d）
4. （a）
5. （c）
6. 请为异常错误①-⑤选择与之匹配的解释，将配对好的a-e填写到括号中。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ① | NameError |  | a | 当使用序列中不存在的索引时，会引发该异常。 |
| ② | ZeroDivisionError | b | 当解释器发现语法错误时，会引发该异常。 |
| ③ | SyntaxError | c | 尝试访问一个未声明的变量，会引发该异常。 |
| ④ | IndexError | d | 当使用字典中不存在的键访问值时，会引发该异常。 |
| ⑤ | KeyError | e | 当除数为零的时候，会引发该异常。 |

1. （c）
2. （e）
3. （b）
4. （a）
5. （d）
6. 请为第三方库①-⑤选择与之匹配的功能，将配对好的a-e填写到括号中。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ① | Numpy |  | a | 机器学习工具库 |
| ② | Pandas | b | 数据可视化库 |
| ③ | Matplotlib | c | 基于矩阵的数学计算库 |
| ④ | Scikit-learn | d | 基于表格的统计分析库 |
| ⑤ | statsmodels | e | 统计建模和计量经济学工具包 |

1. （c）
2. （d）
3. （b）
4. （a）
5. （e）
6. 简答题
7. 什么是程序设计？

程序设计是给出解决特定问题程序的过程，是软件构造活动中的重要组成部分。程序设计往往以某种程序设计语言为工具，给出这种语言下的程序。程序设计过程应当包括分析、设计、编码、测试、排错等不同阶段。

1. 请简单描述批量化处理数据的过程。

首先从数据库中读取批量数据，经过计算处理，最后以图形方式或数据文件方式输出。

1. 举例说明变量与常量的区别。

变量的概念基本上和初中代数的方程变量是一致的，只是在计算机程序中，变量不仅可以是数字，还可以是任意数据类型。在Python中变量在程序中就是用一个变量名表示了，变量名必须是大小写英文、数字和\_的组合，且不能用数字开头。所谓常量就是不能变的变量，比如常用的数学常数π就是一个常量。在Python中，通常用全部大写的变量名表示常量。

1. 举例说明嵌套循环结构及其应用。

所谓嵌套循环，是指一个循环的循环体是另一个循环。比如for循环里面还有一个for循环，就是嵌套循环。总共的循环次数=外循环次数\*内循环次数。

1. 请简述lambda函数的特点。

lambda 表达式是 Python 中创建匿名函数的一个特殊语法. 或称 lambda 语法本身为 lambda 表达式，而它返回的函数我称之为 lambda 函数。或者称为匿名函数。Python 的 lambda 表达式允许在一行代码中创建一个函数并传递。lamda 可以写出非常简练的代码，但是缺点也非常明显：难于理解，降低了可读性和性能，lambda 表达式只能包含一条语句, 因此某些提高可读性的语言功能不能一起使用。

1. 请简述普通参数、指定参数、默认参数、动态参数的区别。

普通参数是传入函数，没有默认值；如果指定了参数的值，那么它们就是指定参数，可以不按照它们原来的顺序传入；在括号里指定参数的初始值,那么就成为默认参数；动态参数 \*args 是指当我们需要传入多个参数时，可以用\*args代表多个参数，不用分别在括号里指定多个参数，动态参数 \*\*kwargs, 当我们需要传入键值对类型的参数时就可以用\*\*kwargs。

1. 比起面向过程，面向对象的编程模式有什么优点？

使用面向对象思想进行开发有以下优点：

1、易维护

采用面向对象思想设计的结构，可读性高，由于继承的存在，即使改变需求，那么维护也只是在局部模块，所以维护起来是非常方便和较低成本的。

2、质量高

在设计时，可重用现有的，在以前的项目的领域中已被测试过的类使系统满足业务需求并具有较高的质量。

3、效率高

在软件开发时，根据设计的需要对现实世界的事物进行抽象，产生类。使用这样的方法解决问题，接近于日常生活和自然的思考方式，势必提高软件开发的效率和质量。

4、易扩展

由于继承、封装、多态的特性，自然设计出高内聚、低耦合的系统结构，使得系统更灵活、更容易扩展，而且成本较低。

使用面向对象思想进行开发有以下缺点：

1、编程的复杂性远高于面向过程。

如果不了解面向对象，可以立即启动基于设计程序，很容易出现过度设计的问题。在一些扩展性要求较低的场景中，使用面向对象会增加编程难度。

2、无法面向过程的程序设计流水线可以准确预测问题的处理过程和结果。

面向对象的程序一旦开始，对象之间的交互就会解决问题，甚至无法准确预测最终结果。

1. is和==有什么区别？

is：比较的是两个对象的id值是否相等，也就是比较俩对象是否为同一个实例对象。是否指向同一个内存地址。

== ： 比较的两个对象的内容/值是否相等，默认会调用对象的eq()方法。

1. 请简述except 的用法和作用。

try…except…except…[else…][finally…]

执行try 下的语句，如果引发异常，则执行过程会跳到except 语句。对每个except 分支顺序尝试执行，如果引发的异常与except 中的异常组匹配，执行相应的语句。如果所有的except 都不匹配，则异常会传递到下一个调用本代码的最高层try 代码中。try 下的语句正常执行，则执行else 块代码。如果发生异常，就不会执行如果存在finally 语句，最后总是会执行。

1. try-except与try-finally有什么不同？

try-except是失败才跳进 except里执行；try-finally是成功失败都跳进finally语句里执行。

1. 什么是缺省参数？

缺省参数指在调用函数的时候没有传入参数的情况下，调用默认的参数，在调用函数的同时赋值时，所传入的参数会替代默认参数。

\*args是不定长参数，它可以表示输入参数是不确定的，可以是任意多个。

\*\*kwargs是关键字参数，赋值的时候是以键值对的方式，参数可以是任意多对在定义函数的时候

不确定会有多少参数会传入时，就可以使用两个参数。

1. 函数和方法之间的区别是什么？

所在位置不同；定义方式不同；调用方式不同。

1. 请简述函数装饰器的作用。

装饰器本质上是一个callable object，它可以在让其他函数在不需要做任何代码的变动的前提下增加额外的功能。装饰器的返回值也是一个函数的对象，它经常用于有切面需求的场景。比如：插入日志，性能测试，事务处理，缓存。权限的校验等场景，有了装饰器就可以抽离出大量的与函数功能本身无关的雷同代码并发并继续使用。

1. 请简述生成器与迭代器的区别。

迭代器是遵循迭代协议的对象。用户可以使用 iter() 以从任何序列得到迭代器（如 list, tuple,dictionary, set 等）。另一个方法则是创建一个另一种形式的迭代器 —— generator 。要获取下一个元素，则使用成员函数 next()（Python 2）或函数 next() function （Python 3） 。当没有元素时，则引发 StopIteration 此例外。若要实现自己的迭代器，则只要实现 next()（Python 2）或 \_\_next\_\_ ()（Python 3）；

生成器（Generator），只是在需要返回数据的时候使用yield语句。每次next()被调用时，生成器会返回它脱离的位置（它记忆语句最后一次执行的位置和所有的数据值）

区别： 生成器能做到迭代器能做的所有事，而且因为自动创建iter()和next()方法，生成器显得特别简洁，而且生成器也是高效的，使用生成器表达式取代列表解析可以同时节省内存。除了创建和保存程序状态的自动方法，当发生器终结时，还会自动抛出StopIteration异常。

1. Python中的可变对象和不可变对象是什么？有什么区别？

不可变对象，该对象所指向的内存中的值不能被改变。当改变某个变量时候，由于其所指的值不能被改变，相当于把原来的值复制一份后再改变，这会开辟一个新的地址，变量再指向这个新的地址。

可变对象，该对象所指向的内存中的值可以被改变。变量（准确的说是引用）改变后，实际上其所指的值直接发生改变，并没有发生复制行为，也没有开辟出新的地址，通俗点说就是原地改变。

Pyhton中，数值类型(int 和float)，字符串str、元祖tuple都是不可变类型。而列表list、字典dict、集合set是可变类型。

1. 谈谈你对面向对象的理解？

面向对象是相当于面向过程而言的，面向过程语言是一种基于功能分析的，以算法为中心的程序设计方法，而面向对象是一种基于结构分析的，以数据为中心的程序设计思想。在面向对象语言中有一个很重要的东西，叫做类。面向对象有三大特性：封装、继承、多态。

1. 请简述数据工程项目流程。

大数据处理流程主要包括数据收集、数据预处理、数据存储、数据处理与分析、数据展示/数据可视化、数据应用等环节，其中数据质量贯穿于整个大数据流程，每一个数据处理环节都会对大数据质量产生影响作用。 通常，一个好的大数据产品要有大量的数据规模、快速的数据处理、精确的数据分析与预测、优秀的可视化图表以及简练易懂的结果解释，本文将基于以上环节分别分析不同阶段对大数据质量的影响及其关键影响因素。 在数据收集过程中，数据源会影响大数据质量的真实性、完整性数据收集、一致性、准确性和安全性。

1. 实验报告撰写原则有哪些？

1、 科学性：科学性是科学研究成果的生命所在。研究报告的表述必须观点正确、材料可靠，论证要以事实为依据，无论是阐述因果关系，结论的利弊和价值，结论的实用性和可行性，都必须从事实出发。推理要合乎逻辑，不可无根据地臆断。

2、创造性：创造性是衡量研究报告质量水平高低的重要依据。别人没有提出过的理论、概念、教育教学新方案，新的实验方法，别人没有观察到的现象，在实验和调告中第一次获得的新的数据等，都是创造性的研究成果。

3、规范性：研究报告的表述虽无定法，但有常规可循。在撰写研究报告时，要按照一定的格式，不能忽视最基本的规范要求。写作之前要有明确的计划和提纲，要根据研究的结构特点和逻辑顺序，研究课题的任务和内容，来考虑表达的形式和表述的方式。

4、可读性：为了便于传播和交流，研究报告的表述应具有可读性。语言阐述必须精确、通俗，在不损害规范性的前提下，尽可能使用简洁的语言。专门的名词术语，可以用，但不能故弄玄虚。。一般不采用比喻、拟人、夸张等修辞手法；不可把日常概念当作科学概念，不宜采用工作经验总结式的文字。

1. 请给出break和continue的定义和区别。

break:退出整个循环，执行循环体外的语句;是结束整个循环过程，不再判断执行循环的条件是否成立。

continue:退出本次循环，进入下一轮循环;只结束本次循环，而不终止整个循环的执行。

1. 应用题
2. 小明身高1.75，体重80.5kg。请根据BMI公式（体重除以身高的平方）帮小明计算他的BMI指数，并根据BMI指数：

低于18.5：过轻

18.5-25：正常

25-28：过重

28-32：肥胖

高于32：严重肥胖

用if-elif判断并打印结果。

参考代码：

height = 1.75  
weight = 80.5  
bmi = 80.5 / (1.75 \* 1.75)  
print(bmi)  
**if** bmi <= 18.5:  
 print(**"过轻"**)  
**elif** 18.5 < bmi <= 25:  
 print(**"正常"**)  
**elif** 25 < bmi <= 28:  
 print(**"过重"**)  
**elif** 28 < bmi <= 32:  
 print(**"肥胖"**)  
**else**:  
 print(**"严重肥胖"**)

答案是26.285714285714285；过重。

1. 编写一个名为example的函数，他有一个名为number的参数：如果参数是偶数，该函数就打印出number//2的值；如果参数是奇数，该函数就打印3\*number+1旳值。

参考代码：

**def** example(number):  
 **if** number % 2 == 0:  
 print(number//2)  
 **else**:  
 print(3\*number+1)

1. 设计一个函数example，作用为反转一个整数，例如-123 --> -321

参考代码：

**def** example(x):  
 **if** -10<x<10:  
 **return** x  
 str\_x = str(x)  
 **if** str\_x[0] !=**"-"**:  
 str\_x = str\_x[::-1]  
 x = int(str\_x)  
 **else**:  
 str\_x = str\_x[1:][::-1]  
 x = int(str\_x)  
 x = -x  
 **return** x **if** -2147483648<x<2147483647 **else** 0  
print(example(-123))

1. 请写出一个函数满足以下条件：

该函数的输入是一个仅包含数字的list,输出一个新的list，其中每一个元素要满足以下条件：

1、该元素是偶数

2、该元素在原list中是在偶数的位置(index是偶数)

参考代码：

**def** num\_list(num):  
 **return** [i **for** i **in** num **if** i %2 ==0 **and** num.index(i)%2==0]  
num = [0,1,2,3,4,5,6,7,8,9,10]  
result = num\_list(num)  
print(result)

1. 输入三个整数a,b,c,请把这三个数由小到大输出。

参考代码：

L = []  
**for** i **in** range(3):  
 x = int(input(**'integer:\n'**))  
 L.append(x)  
 L.sort()  
print(L)

1. 用\*号输出字母F的图案。（请使用for循环语句）

参考代码：

**for** i **in** range(2):  
 print(**'\*'** \* 10)  
 print(**'\*\n'** \* 2, end=**""**)

答案：

\*\*\*\*\*\*\*\*\*\*

\*

\*

\*\*\*\*\*\*\*\*\*\*

\*

\*

1. 请用for循环语句输出九九乘法表，如下：

1\*1=1

2\*1=2 2\*2=4

3\*1=3 3\*2=6 3\*3=9

4\*1=4 4\*2=8 4\*3=12 4\*4=16

5\*1=5 5\*2=10 5\*3=15 5\*4=20 5\*5=25

6\*1=6 6\*2=12 6\*3=18 6\*4=24 6\*5=30 6\*6=36

7\*1=7 7\*2=14 7\*3=21 7\*4=28 7\*5=35 7\*6=42 7\*7=49

8\*1=8 8\*2=16 8\*3=24 8\*4=32 8\*5=40 8\*6=48 8\*7=56 8\*8=64

9\*1=9 9\*2=18 9\*3=27 9\*4=36 9\*5=45 9\*6=54 9\*7=63 9\*8=72 9\*9=81

参考代码：

**for** m **in** range(1,10):  
 **for** n **in** range(1,m+1):  
 print(**"{0}\*{1}={2}"**.format(m,n,(m\*n)),end=**"\t"**)  
 print()

1. 判断1-200之间有多少个素数，并输出所有素数。

参考代码：

h = 0  
leap = 1  
**from** math **import** sqrt  
**for** m **in** range(1,201):  
 k = int(sqrt(m + 1))  
 **for** i **in** range(2, k + 1):  
 **if** m % i == 0:  
 leap = 0  
 **break  
 if** leap == 1:  
 print(**'%-4d'** % m)  
 h += 1  
 **if** h % 10 == 0:  
 print(**""**,end=**""**)  
 leap = 1  
print(**'The total is %d'**%h)

答案是47。

1. 一个数如果恰好等于它的因子之和，这个数就称为“完全数”。例如6=1+2+3.编程找出1000以内的所有完全数。

参考代码：

**for** n **in** range(1,1000):  
 sum = 0  
 **for** i **in** range(1, n ):  
 **if** n % i == 0:  
 sum = sum + i  
 **if** sum == n:  
 print(n)

答案是6、28、496。

1. 请用一行代码 实现将1-100的整数列表以3为单位分组.

参考代码：

print([[x **for** x **in** range(1, 100)][i:i + 3] **for** i **in** range(0, 100, 3)])

1. 利用递归方法求10的阶乘。

参考代码：

**def** fact(j):  
 s = 0  
 **if** j == 0:  
 s = 1  
 **else**:  
 s = j \* fact(j - 1)  
 **return** s  
**for** i **in** range(11):  
 print(**'%d! = %d'** % (i, fact(i)))

1. 定义一个函数：找出传入的列表或元组的奇数位对应的元素，并返回一个新的列表。

参考代码：

a = input() *# 键盘输入字符串*list1 = list(a) *# 将字符串转为列表***def** ji(list1): *# 定义函数* list2 = list1[::2] *# 利用切片找到奇数* **return** list2 *# 返回一个新列表*ls = ji(list1)  
print(ls)

1. 定义一个函数，接收两个数字参数，返回比较小的数字。

参考代码：

a = int(input())  
b = int(input())  
**def** min1(a, b):  
 **if** a > b:  
 print(b)  
 **elif** a < b:  
 print(a)  
 **else**:  
 print(a, b)  
min1(a, b)

1. 定义一个函数example(n)，判断输入的n是不是素数，是的话返回True，否则返回False。通过键盘输入两个整数X和Y，调用此函数输出两数范围之内素数的个数（包括X和Y）。

参考代码：

**import** math  
**def** example(n):  
 m = int(math.sqrt(n) + 1)  
 **for** i **in** range(2, m):  
 **if** n % i == 0:  
 **return False  
 break  
 else**:  
 **return True  
def** main():  
 n, m = eval(input(**"请输入两个数，来求这两个数之间素数的个数,逗号作为分隔符:"**))  
 count = 0  
 **for** i **in** range(n, m + 1):  
 **if** example(i):  
 count = count + 1  
 print(count)  
main()

1. 使用for循环求解1-100这100个数中所有偶数之和。（提示，可以用if进行判断或使用range构造偶数列表）

参考代码：

s = 0  
**for** i **in** range(2,101,2):  
 s+=i  
print(s)

1. 给出一个成绩grade，需根据成绩进行分档，要求如下：

如果成绩大于60分，输出“及格”；如果成绩大于70分，输出“良”；如果成绩大于80分，输出“好”；如果成绩大于90分，输出“优秀”；如果成绩大于100分，输出“成绩有误，请核实”；否则输出“不及格”。

参考代码：

grade = 150  
**if** 60 <= grade < 70:  
 print(**"及格"**)  
**elif** 70 <= grade < 80:  
 print(**"良"**)  
**elif** 80 <= grade < 90:  
 print(**"好"**)  
**elif** 90 <= grade <= 100:  
 print(**"优秀"**)  
**elif** grade > 100:  
 print (**"成绩有误，请核实"**)  
**else**:  
 print(**"不及格"**)

1. 上楼梯：一个台阶一共有15阶，一次起跳可以跳一阶，也可以跳二阶。问总共有多少中跳法。请给出程序和答案。（提示：采用递归方法）

参考代码：

**def** ladder(n):  
 res = [1, 2]  
 **if** n < 1:  
 **return** 0  
 **if** n == 1:  
 **return** 1  
 **if** n == 2:  
 **return** 2  
 **for** i **in** range(2, n):  
 res.append(res[i - 2] + res[i - 1])  
 **return** res[-1]  
print(ladder(15))

1. 有四个数字：1、2、3、4，能组成多少个互不相同且无重复数字的三位数？各是多少？

参考代码：

count = 0  
**for** i **in** range(1, 5):  
 **for** j **in** range(1, 5):  
 **for** k **in** range(1, 5):  
 **if** i != j **and** i != k **and** j != k:  
 count += 1  
 print(i, j, k)  
print(**"共有%d种排列"** % count)

1. 编写一个程序，它将找到所有这些数字，可被9整除，但不是6的倍数，1949年至2022年(包括在内)。得到的数字应按逗号分隔的顺序打印在一行上。

参考代码：

l = []  
**for** i **in** range(1949, 2023):  
 **if** (i % 9 == 0) **and** (i % 6 != 0):  
 l.append(str(i))  
print(**','**.join(l))

1. 编写一个接受句子并计算字母和数字的程序。假设为程序提供了以下输入：

Hello Python! 12345

然后，输出应该是：

字母11

数字5

参考代码：

print(**'请输入：'**)  
s = input()  
d={**"数字"**:0, **"字母"**:0}  
**for** c **in** s:  
 **if** c.isdigit():  
 d[**"数字"**]+=1  
 **elif** c.isalpha():  
 d[**"字母"**]+=1  
 **else**:  
 **pass**print (**"字母"**, d[**"字母"**])  
print (**"数字"**, d[**"数字"**])