《Python语言程序设计》2020第四次平时作业

平时作业说明：

（1）平时作业只需提供答案，不用抄题。

（2）平时作业答案可打印，也可手写。特别提醒：需写上姓名和学号。

（3）所有平时作业答案都等到期末与课程设计报告一起提交。具体提交办法期末时公布。

知识点：函数、模块、文件操作、异常处理与面向对象程序设计

一、程序填空题

（为便于阅卷，不用抄题，只提交各小题答案）

请将每个小题的答案写在实验报告中提交。（不用抄题，只提交答案即可）

1. 下面程序的功能是从键盘读入一个正整数，将其转换为二进制数，并输出之。程序按照源程序文件方式运行后的输入输出情况如下图所示。请在横线处填写适当的语句或表达式将程序补充完整。



源程序：

def f(n):

if(n==0):

return

else:

f(\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_)

print(n%2,end='')

def main():

x=eval(input('x='))

print('0b',end='')

f(x)

main()

2. 下面程序的功能是从键盘读入一个正整数，将其转换为十六进制数，并输出之。程序按照源程序文件方式运行后的输入输出情况如下图所示。请在横线处填写适当的语句或表达式将程序补充完整。



源程序：

def f(n):

global s

if(n==0):

return

else:

f(n//16)

print(s[\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_],end='')

def main():

global s

s='0123456789ABCDEF'

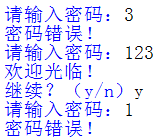
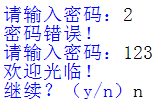
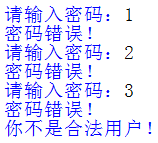
x=eval(input('x='))

print('0x',end='')

f(x)

main()

3. 下面程序的功能是模拟系统登录过程。从键盘读入一个账号的密码，如果输入的密码正确，就显示“欢迎光临！”；如果输入的密码错误，则显示“密码错误！”；如果用户输入的密码连续错误3次，则显示“你不是合法用户！”。程序按照源程序文件方式运行后的输入输出情况如下图所示。请在横线处填写适当的语句或表达式将程序补充完整。



源程序：

def f():

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

if(n>3):

print('你不是合法用户！')

return

else:

k=input('请输入密码：')

if(k=='123'):

print('欢迎光临！')

x=input('继续？（y/n）')

if (x=='y' or x=='Y'):

n=1

f()

else:

return

else:

print('密码错误！')

n=n+1

f()

def main():

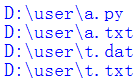
global n

n=1

f()

main()

4. 下面程序的功能是遍历输出当前工作目录中的所有文件和文件夹。程序按照源程序文件方式运行后的输入输出情况如下图所示。请在横线处填写适当的语句或表达式将程序补充完整。



源程序：

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

s=[]

path=os.getcwd()

f=os.walk(path)

for dirpath,dirnames,filenames in f:

for i in dirnames:

s.append(os.path.join(dirpath,i))

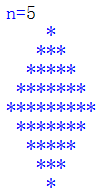
for i in filenames:

s.append(os.path.join(dirpath,i))

for i in s:

print(i)

5. 下面程序的功能是从键盘读入菱形图案的行数（上三角行数），构造该菱形图案，并将其存入文本文件“D:\user\a.txt”中。程序按照源程序文件方式运行后的输入输出情况如下图所示。请在横线处填写适当的语句或表达式将程序补充完整。



源程序：

f=open(r'D:\user\a.txt','w')

n=int(input('n='))

for i in range(n):

s=''

for j in range(n-i-1):

s+=' '

for k in range(2\*i+1):

s+='\*'

f.write(s+'\n')

for i in range(n-1):

s=''

for j in range(i+1):

s+=' '

for k in range(\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_):

s+='\*'

f.write(s+'\n')

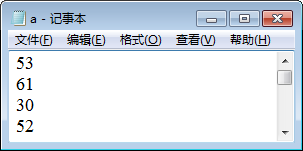
f.close()

with open('D:/user/a.txt','r') as f:

s=f.read()

print(s)

6. 下面程序的功能是随机生成100个0~100的正整数，每个整数占一行，写入“D:\user\a.txt”文件中。程序按照源程序文件方式运行后的输入输出情况如下图所示。请在横线处填写适当的语句或表达式将程序补充完整。



源程序：

import random

with open('d:/user/a.txt', \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_) as f:

for i in range(100):

f.write(str(random.randrange(0,100))+'\n')

7. 下面程序的功能是随机生成100个0~100的正整数，每个整数占一行，写入“D:\user\a.txt”文件中。然后统计其中奇数的个数。程序按照源程序文件方式运行后的输入输出情况如下图所示。请在横线处填写适当的语句或表达式将程序补充完整。



源程序：

import random

with open('d:/user/a.txt','w') as f:

for i in range(100):

f.write(str(random.randrange(0,100))+'\n')

n=0

with open('d:/user/a.txt','r') as f:

for i in f:

t=int(i)

if(\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_):

n=n+1

print(n)

8. 下面程序的功能是文本文件的复制，即将已经存在的“D:\user\a.txt”文件内容复制到文件“D:\user\b.txt”中。请在横线处填写适当的语句或表达式将程序补充完整。

源程序：

f1=open('d:/user/a.txt','r')

f2=open('d:/user/b.txt','w')

s=f1.read()

f2.write(\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_)

f1.close()

f2.close()

9. 下面程序的功能是从键盘读入一个字符串，将其转换为整数。程序按照源程序文件方式运行后的输入输出情况如下图所示。请在横线处填写适当的语句或表达式将程序补充完整。



源程序：

import sys

x=input('x=')

try:

y=int(x)

except Exception as \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_:

print(s)

sys.exit(0)

else:

print(y)

10. 下面程序的功能是从键盘读入一个整数，对除法运算进行分类异常处理。程序按照源程序文件方式运行后的输入输出情况如下图所示。请在横线处填写适当的语句或表达式将程序补充完整。



源程序：

try:

x=eval(input('x='))

y=12/x

except (NameError,ZeroDivisionError) as s:

print(s)

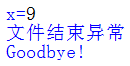
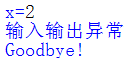
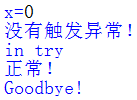
except:

print(\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_)

else:

print(y)

11. 下面程序的功能是从键盘读入一个整数，测试各种异常处理场景。程序按照源程序文件方式运行后的输入输出情况如下图所示。请在横线处填写适当的语句或表达式将程序补充完整。





源程序：

try:

x=eval(input('x='))

if(x==1):raise AttributeError('对象属性异常')

elif(x==2):raise IOError('输入输出异常')

elif(x==3):raise IndexError('索引异常')

elif(x==4):raise KeyError('关键字异常')

elif(x==5):raise NameError('名字异常')

elif(x==6):raise TypeError('类型异常')

elif(x==7):raise ValueError('值异常')

elif(x==8):raise ZeroDivisionError('零除异常')

elif(x==9):raise EOFError('文件结束异常')

else:print('没有触发异常！')

print('in try')

except (AttributeError,IOError,IndexError,KeyError,NameError,\

TypeError,ValueError,ZeroDivisionError,EOFError) as s:

print(s)

except:

print('其他异常！')

else:

print('正常！')

finally:

print(\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_)

12. 下面程序的功能是从键盘读入一个整数，测试自定义零除异常类的异常处理。程序按照源程序文件方式运行后的输入输出情况如下图所示。请在横线处填写适当的语句或表达式将程序补充完整。



源程序：

class MyException(\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_):

def \_\_init\_\_(self,data):

self.data=data

def main():

try:

x=eval(input('x='))

if x==0:

raise MyException('不能为0哦！')

y=100/x

print(y)

except MyException as s:

print(s)

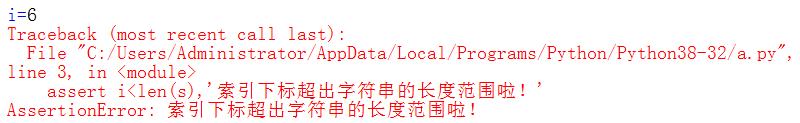
finally:

print('Goodbye!')

main()

13. 下面程序的功能是从键盘读入一个整数，根据该整数，从字符串中提取指定字符。程序按照源程序文件方式运行后的输入输出情况如下图所示。请在横线处填写适当的语句或表达式将程序补充完整。





源程序：

s='uestc'

i=eval(input('i='))

assert i<\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_,'索引下标超出字符串的长度范围啦！'

print('s[{}]={}'.format(i,s[i]))

14. 下面程序的功能是从键盘读入一个整数，根据该整数，从字符串中提取指定字符。程序按照源程序文件方式运行后的输入输出情况如下图所示。请在横线处填写适当的语句或表达式将程序补充完整。





源程序：

s='uestc'

try:

i=eval(input('i='))

assert i<len(s),'索引下标超出字符串的长度范围啦！'

print('s[{}]={}'.format(i,s[i]))

except \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ as e:

print('这是断言错误！',e)

except:

print('其他错误！')

15. 下面程序的功能是从键盘读入一个整数，根据该整数，从字符串中提取指定字符。程序按照源程序文件方式运行后的输入输出情况如下图所示。请在横线处填写适当的语句或表达式将程序补充完整。



源程序：

s='uestc'

try:

i=eval(input('i='))

print('s[{}]={}'.format(i,s[i]))

except \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ as e:

print(e)

except:

print('其他错误！')

16. 下面程序的功能是从键盘读入一个字符串关键字，根据该字符串关键字，从字典中提取指定词条。程序按照源程序文件方式运行后的输入输出情况如下图所示。请在横线处填写适当的语句或表达式将程序补充完整。



源程序：

d=dict(a=2,b=3,c=4)

try:

key=input('key=')

print("d['{}']={}".format(key,d[key]))

except \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ as e:

print('字典中没有这个词条：',e)

except:

print('其他错误！')

17. 下面程序的功能是从键盘读入一个文件名，然后打开该文件，读取其内容，再关闭该文件。假设D:\user\b.txt文件的内容为“我是b.txt文件！”。程序按照源程序文件方式运行后的输入输出情况如下图所示。请在横线处填写适当的语句或表达式将程序补充完整。





源程序：

d=dict(a=2,b=3,c=4)

try:

fn=input('fileName=')

f=open('D:/user/'+fn,'r')

print(f.read())

f.close()

except \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ as e:

print('没有找到这个文件：',e)

except:

print('其他错误！')

18. 下面程序的功能是从键盘读入一个变量名，输出其值。程序按照源程序文件方式运行后的输入输出情况如下图所示。请在横线处填写适当的语句或表达式将程序补充完整。





源程序：

x,y,z=2,3,4

try:

n=input('name=')

print(eval(n))

except \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ as e:

print('没有找到这个变量：',e)

except:

print('其他错误！')

19. 下面程序的功能是从键盘读入一个数据作为函数调用的实在参数。程序按照源程序文件方式运行后的输入输出情况如下图所示。请在横线处填写适当的语句或表达式将程序补充完整。





源程序：

def f(s):

s[2]=3

return s[2]

try:

x=eval(input('x='))

y=f(x)

print(y)

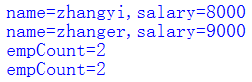
except \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ as e:

print('类型不匹配：',e)

except:

print('其他错误！')

20. 下面程序的功能是统计员工数量和显示员工信息。程序按照源程序文件方式运行后的输入输出情况如下图所示。请在横线处填写适当的语句或表达式将程序补充完整。



源程序：

class Employee:

empCount=\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

def \_\_init\_\_(self,name,salary):

self.name=name

self.salary=salary

Employee.empCount+=1

def displayCount(self):

print("empCount={}".format(Employee.empCount))

def displayEmployee(self):

print("name={},salary={}".format(self.name,self.salary))

def main():

emp1=Employee("zhangyi",8000)

emp2=Employee("zhanger",9000)

emp1.displayEmployee()

emp2.displayEmployee()

print("empCount={}".format(Employee.empCount))

emp1.displayCount()

main()

21. 下面程序的功能是创建父类Person，并派生出子类Man。程序按照源程序文件方式运行后的输入输出情况如下图所示。请在横线处填写适当的语句或表达式将程序补充完整。



源程序：

class Person:

def \_\_init\_\_(self,name,age):

self.name=name

self.age=age

def print\_age(self):

print(self.name,".age=",self.age)

class Man(\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_):

work="Teacher"

def print\_age(self):

print("{}'s age is {}".format(self.name,self.age))

def print\_work(self):

print("{}'s work is {}".format(self.name,self.work))

def main():

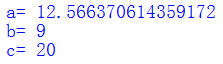
a=Man("Bob",30)

a.print\_age()

a.print\_work()

main()

22. 下面程序的功能是求圆、正方形和矩形的面积。程序按照源程序文件方式运行后的输入输出情况如下图所示。请在横线处填写适当的语句或表达式将程序补充完整。



源程序：

import math

class Circle:

def \_\_init\_\_(self,r):

self.r=r

def getArea(self):

return math.pi\*self.r\*\*2

class Square:

def \_\_init\_\_(self,size):

self.size=size

def getArea(self):

return self.size\*self.size

class Rectangle:

def \_\_init\_\_(self,w,h):

self.w=w

self.h=h

def getArea(self):

return self.w\*self.h

def main():

a=Circle(2)

b=Square(3)

c=Rectangle(4, \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_)

print("a=",a.getArea())

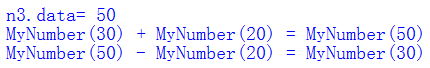
print("b=",b.getArea())

print("c=",c.getArea())

main()

23. 下面程序的功能是实现“+”和“\*\*”的运算符重载。程序按照源程序文件方式运行后的输入输出情况如下图所示。请在横线处填写适当的语句或表达式将程序补充完整。

提示：Python的运算符是通过调用对象的特殊方法来实现的。重写运算符所对应的特殊方法，就可以实现运算符的重载。详情参加本作业文档最后面的附录。



源程序：

class MyNumber:

def \_\_init\_\_(self,value):

self.data=\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

def \_\_repr\_\_(self): # 对应于内置函数str()，用于显示数据

return "MyNumber({})".format(self.data)

def \_\_add\_\_(self,other):

v=self.data+other.data

return MyNumber(v)

def \_\_sub\_\_(self,other):

v=self.data-other.data

return MyNumber(v)

def main():

n1= MyNumber(30)

n2=MyNumber(20)

n3=n1+n2 # 等效为n3=n1.\_\_add\_\_(n2)

n4=n3-n2 # 等效为n4=n3.\_\_sub\_\_(n2)

print("n3.data=",n3.data)

print(n1,"+",n2,"=",n3)

print(n3,"-",n2,"=",n4)

main()

24. 编写一段程序实现下面的功能：定义一个Rectangle类，属性为左上角和右下角的平面坐标，编写方法，实现根据坐标计算矩形面积的功能。

25. 编写一段程序实现下面的功能：定义一个三维空间的点类Point，其属性是三维空间的坐标，再由它派生出一个直线类Line，能求直线的长度。

二、单项选择题

说明：为便于阅卷，请按下表格式给出单选题的答案。

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 题号 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| 答案 |  |  |  |  |  |  |  |  |  |  |
| 题号 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 |
| 答案 |  |  |  |  |  |  |  |  |  |  |
| 题号 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 |
| 答案 |  |  |  |  |  |  |  |  |  |  |
| 题号 | 31 | 32 | 33 | 34 | 35 | 36 | 37 | 38 | 39 | 40 |
| 答案 |  |  |  |  |  |  |  |  |  |  |
| 题号 | 41 | 42 | 43 | 44 | 45 | 46 | 47 | 48 | 49 | 50 |
| 答案 |  |  |  |  |  |  |  |  |  |  |
| 题号 | 51 | 52 | 53 | 54 | 55 | 56 | 57 | 58 | 59 | 60 |
| 答案 |  |  |  |  |  |  |  |  |  |  |
| 题号 | 61 | 62 | 63 | 64 | 65 | 66 | 67 | 68 | 69 | 70 |
| 答案 |  |  |  |  |  |  |  |  |  |  |
| 题号 | 71 | 72 | 73 | 74 | 75 | 76 | 77 | 78 | 79 | 80 |
| 答案 |  |  |  |  |  |  |  |  |  |  |
| 题号 | 81 | 82 | 83 | 84 | 85 | 86 | 87 | 88 | 89 | 90 |
| 答案 |  |  |  |  |  |  |  |  |  |  |
| 题号 | 91 | 92 | 93 | 94 | 95 | 96 | 97 | 98 | 99 | 100 |
| 答案 |  |  |  |  |  |  |  |  |  |  |

单项选择题：

1. 按照源程序文件方式运行下面程序段后的输出结果是（ ）

def f():

pass

print(f())

A. 无任何输出 B. None C. pass D. 出错

2. 一个函数中可以有多少个return语句？（ ）

A. 0 B. 1 C. 2 D. 任意

3. 按照源程序文件方式运行下面程序段后的输出结果是（ ）

f()

def f():

print('ok')

A. 无任何输出 B. None C. ok D. 出错

4. 按照源程序文件方式运行下面程序段后的输出结果是（ ）

def f(x,y):

return x\*\*2+y

print(f(2,3))

A. 2 B. 3 C. 7 D. 11

5. 按照源程序文件方式运行下面程序段后的输出结果是（ ）

def f(x,y):

return x\*\*2

return y\*\*2

return x+y

print(f(2,3))

A. 4 B. 9 C. 5 D. 出错

6. 按照源程序文件方式运行下面程序段后的输出结果是（ ）

def f():

print('AA',end='')

return

print('BB',end='')

return

print(f())

A. None B. AANone C. AABBNone D. 出错

7. 按照源程序文件方式运行下面程序段后的输出结果是（ ）

def f():

return 2,3,4

def main():

print(f())

return

main()

A. 2,3,4 B. (2,3,4) C. [2,3,4] D. 出错

8. 按照源程序文件方式运行下面程序段后的输出结果是（ ）

def f():

print('AA',end='')

def main():

print(f())

main()

A. 无任何输出 B. None C. AANone D. 出错

9. 按照源程序文件方式运行下面程序段后的输出结果是（ ）

f=lambda:2

def main():

print(f())

return

main()

A. 无任何输出 B. None C. 2 D. 出错

10. 按照源程序文件方式运行下面程序段后的输出结果是（ ）

f=lambda x:(2,x,x\*\*2)

def main():

print(f(3))

return

main()

A. 无任何输出 B. 2,3,9 C. (2, 3, 9) D. 出错

11. 按照源程序文件方式运行下面程序段后的输出结果是（ ）

f,g=lambda x:x\*\*2,lambda y:2\*y

def main():

print('{},{}'.format(f(2),g(3)))

return

main()

A. 无任何输出 B. None C. 4,6 D. 出错

12. 按照源程序文件方式运行下面程序段后的输出结果是（ ）

def g():

return lambda x,y:x+y

def main():

f=g()

print(f(2,3))

return

main()

A. 无任何输出 B. None C. 5 D. 出错

13. 按照源程序文件方式运行下面程序段后的输出结果是（ ）

f=(lambda:2,lambda x:x\*3,lambda y:y\*\*2)

def main():

print(f[0]()+f[1](2)+f[2](3))

return

main()

A. 0 B. 17 C. None D. 出错

14. 按照源程序文件方式运行下面程序段后的输出结果是（ ）

def f(x):

return x(2)

def main():

y=f(lambda x:x\*\*2)

print(y)

return

main()

A. 0 B. 4 C. None D. 出错

15. 按照源程序文件方式运行下面程序段后的输出结果是（ ）

def f(x):

x=3

def main():

x=2

f(x)

print(x)

return

main()

A. None B. 0 C. 2 D. 3

16. 按照源程序文件方式运行下面程序段后的输出结果是（ ）

def f(x):

x[1]=5

def main():

x=[2,3,4]

f(x)

print(x)

return

main()

A. [2,5,4] B. [5,3,4] C. [2,3,4] D. [2,3,5]

17. 按照源程序文件方式运行下面程序段后的输出结果是（ ）

def f(x):

x['a']=4

def main():

x=dict(a=2,b=3)

f(x)

print(x)

return

main()

A. {'a': 4, 'b': 3} B. {'a': 2, 'b': 3} C. {'a': 3, 'b': 4} D. {'a': 2, 'b': 4}

18. 按照源程序文件方式运行下面程序段后的输出结果是（ ）

def f(x,y):

x,y=y,x

def main():

x,y=2,3

f(x,y)

print('{},{}'.format(x,y))

return

main()

A. 3,2 B. 2,3 C. 2,2 D. 3,3

19. 按照源程序文件方式运行下面程序段后的输出结果是（ ）

def f(s):

s[0],s[1]=s[1],s[0]

def main():

s=[2,3]

f(s)

print(s)

return

main()

A. [3, 2] B. [2, 3] C. [2, 2] D. [3, 3]

20. 按照源程序文件方式运行下面程序段后的输出结果是（ ）

def f(x,y=4,z=5):

print(x,y,z,sep=',')

def main():

f(2,3)

return

main()

A. 2,3,5 B. 2,4,5 C. 2,4,3 D. 3,4,5

21. 按照源程序文件方式运行下面程序段后的输出结果是（ ）

def f(x=4,y,z=5):

print(x,y,z,sep=',')

def main():

f(2,3)

return

main()

A. 2,3,5 B. 4,2,5 C. 4,3,5 D. 出错

22. 按照源程序文件方式运行下面程序段后的输出结果是（ ）

def f(x=4,y,z=5):

print(x,y,z,sep=',')

def main():

f(y=2,x=3)

return

main()

A. 3,2,5 B. 4,2,5 C. 2,3,5 D. 出错

23. 按照源程序文件方式运行下面程序段后的输出结果是（ ）

def f(x,y=4,z=5):

print(x,y,z,sep=',')

def main():

f(y=2,x=3)

return

main()

A. 3,2,5 B. 3,4,5 C. 2,3,5 D. 出错

24. 按照源程序文件方式运行下面程序段后的输出结果是（ ）

def f(x,y,\*z):

print(x,y,z,sep=',')

def main():

f(2,3,4,5,6)

return

main()

A. 2,3,(4,5,6) B. 2,3,[ 4,5,6] C. 2,3,{4,5,6} D. 2,3,4,5,6

25. 按照源程序文件方式运行下面程序段后的输出结果是（ ）

def f(x=6,y=7,\*\*z):

print(x,y,z,sep=',')

def main():

f(2,3,4,b=5)

return

main()

A. 2,3,{None: 4, 'b': 5} B. 2,3,{'b': 5} C. 3,4,{'b': 5} D. 出错

26. 按照源程序文件方式运行下面程序段后的输出结果是（ ）

def f(x=6,y=7,\*z,\*\*t):

print(x,y,z,t,sep=',')

def main():

f(2,a=4,b=5)

return

main()

A. 2,7,(),{'a': 4, 'b': 5} B. 6,7,(2,),{'a': 4, 'b': 5}

C. 2,7,(4,),{'b': 5} D. 2,7,(4,5),{}

27. 按照源程序文件方式运行下面程序段后的输出结果是（ ）

def f(x=6,y=7,\*z):

print(x,y,z,sep=',')

def main():

x,\*y,z=2,3,4,5

f(x,y,z)

return

main()

A. 2,[3, 4],(5,) B. 2,(3, 4),(5,) C. 2,3,(4,5) D. 2,[3, 4],(5)

28. 按照源程序文件方式运行下面程序段后的输出结果是（ ）

def f(x=6,y=7,\*z):

print(x,y,z,sep=',')

def main():

x,y,\*z=2,3,4,5

f(x,y,z)

return

main()

A. 2,3,([4, 5],) B. 2,3,(4, 5,) C. 2,3,[4, 5] D. 2,3,([4, 5])

29. 按照源程序文件方式运行下面程序段后的输出结果是（ ）

def f(x=6,y=7,\*z):

print(x,y,z,sep=',')

def main():

x,\*y,z=2,3,4,5

f(x,\*y,z)

return

main()

A. 2,[3, 4],(5,) B. 2,3,(4, 5) C. 2,[3],(4, 5) D. 2,[3,4], (5)

30. 按照源程序文件方式运行下面程序段后的输出结果是（ ）

def f(x,\*y):

m=x;

for i in y:

if(i>m):

m=i

return m

def main():

z=f(3,4,2,5,6,8,6,7,1)

print(z)

return

main()

A. 1 B. 6 C. 8 D. 9

31. 按照源程序文件方式运行下面程序段后的输出结果是（ ）

def f(x):

a=2;

x=3

print(a,x,sep='',end='')

def g(x):

a=4;

x=x+5

print(a,x,sep='',end='')

def main():

a,x=2,3

f(a)

g(x)

main()

A. 23412 B. 2348 C. 26412 D. 2328

32. 按照源程序文件方式运行下面程序段后的输出结果是（ ）

def f(x):

a=2;

x=3

print(a,x,sep='',end='')

def g(x):

global a

a=a+4;

x=x+5

print(a,x,sep='',end='')

a,x=2,3

f(a)

g(x)

A. 23412 B. 2348 C. 2368 D. 2328

33. 按照源程序文件方式运行下面程序段后的输出结果是（ ）

x=2

def f(x=4):

print(x,end='')

f(x)

x=3

f(x)

A. 23 B. 32 C. 22 D. 33

34. 按照源程序文件方式运行下面程序段后的输出结果是（ ）

x=2

def f(x=4):

print(x,end='')

def main():

global x

f(x)

x=3

f(x)

main()

A. 23 B. 32 C. 22 D. 33

35. 按照源程序文件方式运行下面程序段后的输出结果是（ ）

s='uestc'

def f(i):

if(i<-len(s)):

return

else:

f(i-1)

print(s[i],end='')

def main():

f(-1)

main()

A. uestc B. ctseu C. None D. 出错

36. 按照源程序文件方式运行下面程序段后的输出结果是（ ）

def f(i):

if(i>=len(s)):

return

else:

f(i+1)

print(s[i],end='')

def main():

global s

s='uestc'

f(0)

main()

A. uestc B. ctseu C. None D. 出错

37. 按照源程序文件方式运行下面程序段后的输出结果是（ ）

def f(y=3):

x=y

print(x,end='')

def g():

nonlocal x

print(x,end='')

x=4

g()

print(x,end='')

def main():

global x

x=2

f(x)

print(x,end='')

main()

A. 2242 B. 2224 C. 2442 D. 3242

38. 按照源程序文件方式运行下面程序段后的输出结果是（ ）

def f(y=3):

x=y

print(x,end='')

def g():

global x

print(x,end='')

x=4

g()

print(x,end='')

def main():

global x

x=2

f(x)

print(x,end='')

main()

A. 2242 B. 2224 C. 2442 D. 3242

39. 按照源程序文件方式运行下面程序段后的输出结果是（ ）

def f(y=3):

x=y

print(x,end='')

def g():

global x

print(x,end='')

x=4

def h():

print(x,end='')

x=5

g()

print(x,end='')

def main():

global x

x=2

f(x)

print(x,end='')

main()

A. 2242 B. 2224 C. 24422 D. 32422

40. 按照源程序文件方式运行下面程序段后的输出结果是（ ）

def f(y=3):

x=y

print(x,end='')

def g():

global x

print(x,end='')

x=4

def h():

global x

print(x,end='')

x=5

h()

g()

print(x,end='')

def main():

global x

x=2

f(x)

print(x,end='')

main()

A. 22425 B. 22455 C. 24422 D. 32422

41. 按照源程序文件方式运行下面程序段后的输出结果是（ ）

def f(n):

print(n,end='')

if(n<1):

return

else:

f(n-1)

print(n,end='')

def main():

f(3)

main()

A. 3210123 B. 32100123 C. 321123 D. 32123

42. 按照源程序文件方式运行下面程序段后的输出结果是（ ）

def f():

global x

x=4

y=5

def g():

nonlocal y

global x

x=6

y=7

print('{}{}'.format(x,y),end='')

g()

print('{}{}'.format(x,y),end='')

x=2

y=3

f()

print('{}{}'.format(x,y),end='')

A. 674723 B. 674523 C. 676767 D. 676763

43. 要将源程序模块文件搜索路径“D:\user\”加入sys.path中，正确的语句是（ ）

A. import sys;sys.path.append('D:\user\') B. import sys;sys.path.append(r'D:\user\')

C. import sys;sys.path.append("D:\\user\\") D. import sys;sys.path.append(r"D:\user\")

44. 要将源程序模块文件搜索路径“D:\user\”加入sys.path中，错误的语句是（ ）

A. import sys;sys.path.append(r'D:\user\') B. import sys;sys.path.append('D:\\user\\')

C. import sys;sys.path+=["D:\\user\\"] D. import sys;sys.path+=['D:\\user\\']

45. 按照源程序文件方式运行下面程序c.py后的输出结果是（ ）

在D:\user文件夹中建立如下三个Python源程序文件：

# D:\user\a.py

def f():

print('AA',end='')

# D:\user\b.py

def g():

print('BB',end='')

# D:\user\c.py

from a import f

from b import g

def main():

f()

g()

print('CC')

main()

A. AABBCC B. CC C. 无输出 D. 出错

46. 按照源程序文件方式运行下面程序c.py后的输出结果是（ ）

在D:\user\aa文件夹中建立如下a.py源程序文件：

# D:\user\aa\a.py

def f():

print('AA',end='')

在D:\user\bb文件夹中建立如下b.py源程序文件：

# D:\user\bb\b.py

def g():

print('BB',end='')

在D:\user\cc文件夹中建立如下c.py源程序文件：

# D:\user\cc\c.py

import sys

sys.path.append('D:\\user\\aa')

sys.path.append('D:\\user\\bb')

from a import f

from b import g

def main():

f()

g()

print('CC')

main()

A. AABBCC B. CC C. 无输出 D. 出错

47. 按照源程序文件方式运行下面程序c.py后的输出结果是（ ）

在D:\user文件夹中建立如下三个Python源程序文件：

# D:\user\a.py

def f():

print(\_\_name\_\_,end='')

# D:\user\b.py

def g():

print(\_\_name\_\_,end='')

# D:\user\c.py

from a import f

from b import g

def main():

f()

g()

print(\_\_name\_\_,end='')

main()

A. abc B. \_\_main\_\_bc C. a\_\_main\_\_c D. ab\_\_main\_\_

48. 按照源程序文件方式运行下面程序b.py后的输出结果是（ ）

在D:\user文件夹中建立如下三个Python源程序文件：

# D:\user\a.py

def f():

print(\_\_name\_\_,end='')

# D:\user\b.py

import a,c

def g():

a.f()

print(\_\_name\_\_,end='')

c.main()

g()

# D:\user\c.py

def main():

print(\_\_name\_\_,end='')

A. abc B. \_\_main\_\_bc C. a\_\_main\_\_c D. ab\_\_main\_\_

49. 按照源程序文件方式运行下面程序a.py后的输出结果是（ ）

在D:\user文件夹中建立如下三个Python源程序文件：

# D:\user\a.py

def f():

print(\_\_name\_\_,end='')

f()

# D:\user\b.py

def g():

print(\_\_name\_\_,end='')

g()

# D:\user\c.py

def main():

print(\_\_name\_\_,end='')

main()

A. a B. \_\_main\_\_ C. a\_\_main\_\_ D. \_\_main\_\_a

50. 按照源程序文件方式运行下面程序a.py后的输出结果是（ ）

在D:\user文件夹中建立如下两个Python源程序文件：

# D:\user\a.py

def f():

print(\_\_name\_\_,end='')

if \_\_name\_\_=='\_\_main\_\_':

f()

import b

b.g()

# D:\user\b.py

def g():

print(\_\_name\_\_,end='')

if \_\_name\_\_=='\_\_main\_\_':

g()

import a

a.f()

A. ab B. \_\_main\_\_a C. \_\_main\_\_b D. \_\_main\_\_\_\_main\_\_

51. 按照源程序文件方式运行下面程序b.py后的输出结果是（ ）

在D:\user文件夹中建立如下两个Python源程序文件：

# D:\user\a.py

def f():

print(\_\_name\_\_,end='')

if \_\_name\_\_=='\_\_main\_\_':

f()

import b

b.g()

# D:\user\b.py

def g():

print(\_\_name\_\_,end='')

if \_\_name\_\_=='\_\_main\_\_':

g()

import a

a.f()

A. ab B. \_\_main\_\_a C. \_\_main\_\_b D. \_\_main\_\_\_\_main\_\_

52. 如果当前源程序模块文件是主模块文件，则其系统变量“\_\_name\_\_”的值为（ ）

A. \_\_main\_\_ B. 其源程序文件的文件根名（不加后面的.py，即模块名）

C. 其源程序文件的路径 D. 其源程序文件所在的文件夹名字

53. 如果当前源程序模块文件是非主模块文件，则其系统变量“\_\_name\_\_”的值为（ ）

A. \_\_main\_\_ B. 其源程序文件的文件根名（不加后面的.py，即模块名）

C. 其源程序文件的路径 D. 其源程序文件所在的文件夹名字

54. 新建一个文本文件，欲对其进行读和写操作，则该文件的正确打开方式是（ ）

A. f=open(r'D:\user\a.txt','r+') B. f=open(r'D:\user\a.txt','w+')

C. f=open(r'D:\user\a.txt','rb+') D. f=open(r'D:\user\a.txt','wb+')

55. 打开一个已经存在的文本文件，欲对其进行读和写操作，则该文件的正确打开方式是（ ）

A. f=open(r'D:\user\a.txt','r+') B. f=open(r'D:\user\a.txt','w+')

C. f=open(r'D:\user\a.txt','rb+') D. f=open(r'D:\user\a.txt','wb+')

56. 打开一个已经存在的二进制文件，欲在其原有内容后面写入一些新内容，则该文件的正确打开方式是（ ）

A. f=open(r'D:\user\a.txt','r+') B. f=open(r'D:\user\a.txt','wb+')

C. f=open(r'D:\user\a.txt','a+') D. f=open(r'D:\user\a.txt','ab+')

57. 下列哪种打开文件的方式会造成原文件内容被清除？（ ）

A. f=open(r'D:\user\a.txt','r+') B. f=open(r'D:\user\a.txt','w+')

C. f=open(r'D:\user\a.txt','a+') D. f=open(r'D:\user\a.txt','ab+')

58. 用方法“f=open(r'D:\user\a.txt','r+')”打开文件，则关闭该文件的方法是（ ）

A. close(f) B. f.close() C. close(f,'r+') D. f.close('r+')

59. 已知文本文件“D:\user\t.txt”的内容如下，按照源程序文件方式运行下面各程序段后，输出结果与其余三项不同的是（ ）

# D:\user\t.txt

AA

BBB

CCCC

A.

f=open(r'D:\user\t.txt')

s=f.read()

print(s)

f.close()

B.

f=open(r'D:\user\t.txt','r')

while True:

s=f.read(2)

if not s:

break

print(s,end='')

f.close()

C.

f=open(r'D:\user\t.txt','r')

while True:

s=f.readline()

if not s:

break

print(s)

f.close()

D.

f=open(r'D:\user\t.txt')

for i in f:

print(i,end='')

f.close()

60. 已知文本文件“D:\user\t.txt”的内容如下，按照源程序文件方式运行下面各程序段后，输出结果与其余三项不同的是（ ）

AA

BBB

CCCC

A.

f=open(r'D:\user\t.txt')

s=[]

for i in f:

s+=[i]

print(s)

f.close()

B.

f=open(r'D:\user\t.txt','r')

s=[]

while True:

t=f.readline()

if not t:

break

s+=[t]

print(s)

f.close()

C.

f=open(r'D:\user\t.txt','r')

s=[]

t=''

while True:

i=f.read(1)

if not i:

break

if (i!='\n'):

t+=i

else:

t+='\n'

s+=[t]

t=''

print(s)

f.close()

D.

f=open(r'D:\user\t.txt','r')

s=f.readlines()

print(s)

f.close()

61. 按照源程序文件方式运行下面程序段后的输出结果是（ ）

import string

f=open(r'D:\user\t.txt','w+')

f.write(string.digits+string.ascii\_uppercase)

f.seek(0)

s=f.read(10)

print(s)

f.close()

A. 0123456789 B. ABCDEFGHIJ C. 01234 D. ABCDE

62. 按照源程序文件方式运行下面程序段后的输出结果是（ ）

import string

f=open(r'D:\user\t.txt','w+')

f.write(string.digits+'\n'+string.ascii\_uppercase+'\n'+string.ascii\_lowercase)

print(f.tell(),end=',')

f.seek(0)

s=f.read(1)

print(f.tell(),end=',')

t=f.readline()

print(f.tell(),end=',')

print(len(s+t))

f.close()

A. 65,1,12,12 B. 66,1,12,11 C. 66,1,12,12 D. 65,1,12,11

63. 按照源程序文件方式运行下面程序段后的输出结果是（ ）

import string

f=open(r'D:\user\t.dat','wb+')

s=string.digits+string.ascii\_uppercase+string.ascii\_lowercase

sb=s.encode()

f.write(sb)

print(f.tell(),end=',')

f.seek(-5,2)

t=f.read()

print(t,end=',')

print(f.tell())

f.close()

A. 0,b'vwxyz',62 B. 0,'vwxyz',62 C. 62,b'vwxyz',62 D. 62,'vwxyz',62

64. 按照源程序文件方式运行下面程序段后的输出结果是（ ）

import string

f=open(r'D:\user\t.dat','wb+')

s=string.digits+string.ascii\_uppercase+string.ascii\_lowercase

sb=s.encode()

f.write(sb)

f.seek(-40,1)

print(f.tell(),end=',')

f.seek(-10,1)

print(f.tell())

f.close()

A. 22,12 B. 21,11 C. 21,12 D. 22,13

65. 按照源程序文件方式运行下面程序段后的输出结果是（ ）

import string

f=open(r'D:\user\t.dat','wb+')

s='电子科技大学'

sb=s.encode('gbk')

f.write(sb)

f.seek(-12,1)

print(f.tell(),end=',')

t=f.read(4)

x=t.decode('gbk')

print(x,end=',')

print(f.tell())

f.close()

A. 0,电子,4 B. 0,电子科技,4 C. 6,科技大学,18 D. 4,科技,8

66. 在交互命令模式下，获取当前工作目录的程序段是（ ）

A. import os;os.getcwd() B. import os;os.chdir()

C. import os;os.listdir() D. import os;os.mkdir()

67. 在交互命令模式下，创建“D:\aa”文件夹（目录）的程序段是（ ）

A. import os;os.rmdir(r'D:\aa') B. import os;os.listdir(r'D:\aa')

C. import os;os.mkdir(r'D:\aa') D. import os;os.chdir(r'D:\aa')

68. 在交互命令模式下，列出“D:\user”文件夹（目录）中内容的程序段是（ ）

A. import os;os.rmdir(r'D:\user') B. import os;os.listdir(r'D:\user')

C. import os;os.mkdir(r'D:\user') D. import os;os.chdir(r'D:\user')

69. 在交互命令模式下，删除空目录“D:\aa”的程序段是（ ）

A. import os;os.rmdir(r'D:\aa') B. import os;os.listdir(r'D:\aa')

C. import os;os.mkdir(r'D:\aa') D. import os;os.chdir(r'D:\aa')

70. 在交互命令模式下，将系统当前目录改为“D:\aa”的程序段是（ ）

A. import os;os.rmdir(r'D:\aa') B. import os;os.listdir(r'D:\aa')

C. import os;os.mkdir(r'D:\aa') D. import os;os.chdir(r'D:\aa')

71. 功能与下面程序段相同的是（ ）

import os

path=os.getcwd()

s=os.listdir(path)

print(s)

A.

import os

path=os.getcwd()

s=os.walk(path)

print(s)

B.

import os

path=os.getcwd()

s=os.walk(path)

for dirpath,dirnames,filenames in s:

print(filenames)

C.

import os

path=os.getcwd()

s=os.walk(path)

for dirpath,dirnames,filenames in s:

print(dirpath)

D.

import os

path=os.getcwd()

s=os.walk(path)

for dirpath,dirnames,filenames in s:

print(dirnames)

72. 查看文件“D:\user\a.txt”大小（单位是字节）的程序段是（ ）

A.

import os

s=os.path.getsize(r'D:\user\a.txt')

print(s)

B.

import os

s=os.path.exists(r'D:\user\a.txt')

print(s)

C.

import os

s=os.stat(r'D:\user\a.txt')

print(s)

D.

import os

s=os.path.getctime(r'D:\user\a.txt')

print(s)

73. 判断文件“D:\user\a.txt”是否存在的程序段是（ ）

A.

import os

s=os.path.getsize(r'D:\user\a.txt')

print(s)

B.

import os

s=os.path.exists(r'D:\user\a.txt')

print(s)

C.

import os

s=os.stat(r'D:\user\a.txt')

print(s)

D.

import os

s=os.path.getctime(r'D:\user\a.txt')

print(s)

74. 将文件“D:\user\a.txt”改名为“D:\user\b.txt”，不能实现该功能的程序段是（ ）

A.

import os

os.rename('D:\user\a.txt','D:\user\b.txt')

B.

import os

os.rename('D:/user/a.txt','D:/user/b.txt')

C.

import os,shutil

shutil.copyfile('D:/user/a.txt','D:/user/b.txt')

os.remove('D:/user/a.txt')

D.

import shutil

shutil.move('D:/user/a.txt','D:/user/b.txt')

75. 将文件“D:\user\a.txt”复制为“D:\b.txt”，正确的程序段是（ ）

A.

import os

os.rename('D:\\user\\a.txt','D:\\b.txt')

B.

import os

os.remove('D:\\user\\a.txt','D:\\b.txt')

C.

import shutil

shutil.copyfile('D:\\user\\a.txt','D:\\b.txt')

D.

import shutil

shutil.move('D:\\user\\a.txt','D:\\b.txt')

76. 在Python中可使用read([size])来读取文件中的数据，如果参数size省略，则读取文件中的（ ）

A. 什么也不读取 B. 一个字符 C. 一行数据 D. 所有数据

77. 在Python中可使用readline([size])来读取文件中的数据，如果参数size省略，则读取文件中的（ ）

A. 什么也不读取 B. 一个字符 C. 一行数据 D. 所有数据

78. 在Python中可使用readlines([size])来读取文件中的数据，此时返回的数据为（ ）

A. 列表 B. 字符 C. 字符串 D. 对象

79. 在Python中使用seek(offset)来移动文件位置指针，当offset为负值时，表示（ ）

A. 文件指针向文件尾部方向移动 B. 文件指针向文件头部方向移动

C. 文件指针不向任何地方移动 D. 文件指针直接移动到文件末尾

80. 在Python中，欲对文件任意位置进行写入操作，则文件的打开方式应该是（ ）

A. 'w' B. 'a' C. 'a+' D. 'w'或'a'

81. 在Python中，打开文件时没有指出打开方式，则默认的文件打开方式是（ ）

A. 'w' B. 'r' C. 'a' D. 'w+'

82. 在Python中，如果文件打开方式是'w'，当向该文件写入‘uestc’时，则（ ）

A. 将该字符串插入在文件头部 B. 将该字符串插入在文件尾部

C. 将用该字符串覆盖文件的原有内容 D. 报错

83. 在Python中，如果文件打开方式是'a'，则打开文件时文件位置指针在（ ）

A. 文件头部 B. 文件尾部 C. 文件中部 D. 随机位置

84. 在Python中，不能表示当前文件目录的是（ ）

A. '.' B. './' C. '..'或'../' D. import os;os.getcwd()

85. 按照源程序文件方式运行下面程序段后的输出结果是（ ）

try:

x=[3,4,5]

y=2+x

print(y)

except TypeError:

print('类型不匹配!')

except SyntaxError:

print('语法错误！')

except:

print('其他错误！')

A. 类型不匹配! B. 语法错误！ C. 其他错误！ D. [2,3,4,5]

86. 按照源程序文件方式运行下面程序段后的输出结果是（ ）

try:

x=x+2

except TypeError:

print('AA')

except SyntaxError:

print('BB')

except NameError:

print('CC')

except:

print('DD')

A. AA B. BB C. CC D. DD

87. 按照源程序文件方式运行下面程序段后的输出结果是（ ）

try:

s=(2,3,4)

s[1]=5

except TypeError:

print('AA')

except SyntaxError:

print('BB')

except NameError:

print('CC')

except:

print('DD')

A. AA B. BB C. CC D. DD

88. 按照源程序文件方式运行下面程序段后的输出结果是（ ）

try:

x=1+2,2+3,3+4

except:

print('AA',end='')

else:

print('BB',end='')

finally:

print('CC',end='')

print('DD',end='')

A. AA B. AACCDD C. BBCCDD D. AABBCCDD

89. 在Python语言中定义一个类时，如果省略父类名，则默认的父类是（ ）

A. object B. root C. pass D. None

90. 类中私有属性和私有方法的名称是（ ）

A. 以双下划线（\_\_）开头 B. 以双下划线（\_\_）结尾

C. 用private声明 D. 以单下划线（\_）开头

91. 在Python中，类的构造方法名是（ ）

A. 与类名相同 B. \_\_init\_\_ C. \_\_new\_\_ D. 以@开头

92. 在类中方法外定义的属性是（ ）

A. 实例属性 B. 类属性 C. 私有属性 D. 公有属性

93. 在Python中，哪种方法内只能访问类属性，不能访问实例属性？（ ）

A. 实例方法 B. 类方法 C. 静态方法 D. 析构方法

94. 在Python中，用户自定义类直接继承的父类个数是（ ）

A. 1个 B. 至少1个 C. 至少2个 D. 任意个

95. 下列方法中，哪种方法可以没有形式参数？（ ）

A. 带有装饰器@classmethod的方法 B. 实例方法

C. 带有装饰器@staticmethod的方法 D. 构造方法和析构方法

96. 在Python中，一个类有多少个类对象？（ ）

A. 1个 B. 至少1个 C. 至少2个 D. 任意个

97. 在Python中，一个类可以有多少个实例对象？（ ）

A. 1个 B. 至少1个 C. 至少2个 D. 任意个

98. 在Python中，一个类有多少个（有效的）构造方法？（ ）

A. 1个 B. 至少1个 C. 至少2个 D. 任意个

99. 在Python中，一个类可以有多少个类属性？（ ）

A. 1个 B. 至少1个 C. 至少2个 D. 任意个

100. 在Python中，一个类可以有多少个实例方法？（ ）

A. 1个 B. 至少1个 C. 至少2个 D. 任意个

-----------------------------------------------------------------------------------------------------------

附录：运算符重载

（1）什么是运算符重载

让自定义的类生成的对象(实例)能够使用运算符进行操作

（2）作用

让自定义的实例像内建对象一样进行运算符操作

让程序简洁易读

对自定义对象将运算符赋予新的规则

（3）算术运算符的重载

方法名 运算符和表达式 说明

\_\_add\_\_(self,rhs) self + rhs 加法

\_\_sub\_\_(self,rhs) self - rhs 减法

\_\_mul\_\_(self,rhs) self \* rhs 乘法

\_\_truediv\_\_(self,rhs) self / rhs 除法

\_\_floordiv\_\_(self,rhs) self //rhs 地板除

\_\_mod\_\_(self,rhs) self % rhs 取模(求余)

\_\_pow\_\_(self,rhs) self \*\*rhs 幂运算

注：rhs (right hand side) 右手边

说明：

运算符重载的方法的参数已经有了固定的含义,不建议改变原有的运算符的含义及参数的意义

二元运算符的重载方法格式：

def \_\_xx\_\_(self,other):

语句块

例：实现两个自定义列表的相加和列表自连接（乘整数）

class Mylist:

def \_\_init\_\_(self,iterable=()):

self.data=list(iterable)

def \_\_repr\_\_(self):

return "Mylist({})".format(self.data)

def \_\_add\_\_(self,lst):

return Mylist(self.data+lst.data)

def \_\_mul\_\_(self,rhs):

# rhs为int类型，不能用rhs.data

return Mylist(self.data\*rhs)

def main():

L1=Mylist([1,2,3])

L2=Mylist([4,5,6])

L3=L1+L2

L4=L2+L1

L5=L1\*3

print(L3) # Mylist([1,2,3,4,5,6])

print(L4) # Mylist([4,5,6,1,2,3])

print(L5) # Mylist([1,2,3,1,2,3,1,2,3])

main()

（4）反向运算符的重载

当运算符的左侧为内建类型时,右侧为自定义类型进行算术匀算符运算时会出现TypeError错误，因为无法修改内建类型的代码

实现运算符重载，此时需要使用反向运算符的重载

反向算术运算符的重载:

方法名 运算符和表达式 说明

\_\_radd\_\_(self,lhs) lhs + self 加法

\_\_rsub\_\_(self,lhs) lhs - self 减法

\_\_rmul\_\_(self,lhs) lhs \* self 乘法

\_\_rtruediv\_\_(self,lhs) lhs / self 除法

\_\_rfloordiv\_\_(self,lhs) lhs // self 地板除

\_\_rmod\_\_(self,lhs) lhs % self 取模(求余)

\_\_rpow\_\_(self,lhs) lhs \*\* self 幂运算

例：

class Mylist:

def \_\_init\_\_(self,iterable=()):

self.data=list(iterable)

def \_\_repr\_\_(self):

return "Mylist({})".format(self.data)

def \_\_add\_\_(self,lst):

return Mylist(self.data+lst.data)

def \_\_mul\_\_(self, rhs):

# rhs为int类型,不能用rhs.data

return Mylist(self.data\*rhs)

def \_\_rmul\_\_(self,lhs):

return Mylist(self.data\*lhs)

def main():

L1=Mylist([1,2,3])

L2=Mylist([4,5,6])

L3=3\*L1

print(L3)

L1+=L2

print(L1)

L2\*=3

print(L2)

main()

（5）复合赋值算术运算符的重载

以复合赋值算术运算符 x += y为例，此运算符会优先调用x.\_\_iadd\_\_(y)方法，如果没有\_\_iadd\_\_方法时，则会将复合赋值算术运算拆解为：x = x + y

然后调用x = x.\_\_add\_\_(y)方法,如果再不存在\_\_add\_\_方法则会触发TypeError类型的错误异常

复合赋值算术运算符的重载:

方法名 运算符和表达式 说明

\_\_iadd\_\_(self,rhs) self += rhs 加法

\_\_isub\_\_(self,rhs) self -= rhs 减法

\_\_imul\_\_(self,rhs) self \*= rhs 乘法

\_\_itruediv\_\_(self,rhs) self /= rhs 除法

\_\_ifloordiv\_\_(self,rhs) self //=rhs 地板除

\_\_imod\_\_(self,rhs) self %= rhs 取模(求余)

\_\_ipow\_\_(self,rhs) self \*\*=rhs 幂运算

（6）比较算术运算符的重载

比较算术运算符的重载:

方法名 算符和表达式 说明

\_\_lt\_\_(self,rhs) self < rhs 小于

\_\_le\_\_(self,rhs) self <= rhs 小于等于

\_\_gt\_\_(self,rhs) self > rhs 大于

\_\_ge\_\_(self,rhs) self >= rhs 大于等于

\_\_eq\_\_(self,rhs) self == rhs 等于

\_\_ne\_\_(self,rhs) self != rhs 不等于

（7）位运算符重载

方法名 运算符和表达式 说明

\_\_and\_\_(self,rhs) self & rhs 位与

\_\_or\_\_(self,rhs) self | rhs 位或

\_\_xor\_\_(self,rhs) self ^ rhs 位异或

\_\_lshift\_\_(self,rhs) self <<rhs 左移

\_\_rshift\_\_(self,rhs) self >>rhs 右移

（8）反向位运算符重载

方法名 运算符和表达式 说明

\_\_and\_\_(self,lhs) lhs & rhs 位与

\_\_or\_\_(self,lhs) lhs | rhs 位或

\_\_xor\_\_(self,lhs) lhs ^ rhs 位异或

\_\_lshift\_\_(self,lhs) lhs <<rhs 左移

\_\_rshift\_\_(self,lhs) lhs >>rhs 右移

（9）复合赋值位相关运算符重载

方法名 运算符和表达式 说明

\_\_iand\_\_(self,rhs) self & rhs 位与

\_\_ior\_\_(self,rhs) self | rhs 位或

\_\_ixor\_\_(self,rhs) self ^ rhs 位异或

\_\_ilshift\_\_(self,rhs) self <<rhs 左移

\_\_irshift\_\_(self,rhs) self >>rhs 右移

（10）一元运算符的重载

方法名 运算符和表达式 说明

\_\_neg\_\_(self) - self 负号

\_\_pos\_\_(self) + self 正号

\_\_invert\_\_(self) ~ self 取反

语法:

class 类名:

def \_\_xxx\_\_(self):

pass

例：

class Mylist:

def \_\_init\_\_(self,iterable=()):

self.data=list(iterable)

def \_\_repr\_\_(self):

return 'Mylist(%s)' % self.data

def \_\_neg\_\_(self):

g=(-x for x in self.data)

return Mylist(g)

def \_\_pos\_\_(self):

g=(abs(x) for x in self.data)

return Mylist(g)

def main():

s1=Mylist([1,-2,3,-4,5,-6])

s2=-s1

print(s2)

s3=+s1

print(s3)

main()

（11）in/not in 运算符重载

格式:

def \_\_contains\_\_(self,e):

语句

注: in / not in 返回布尔值 True / False

当重载了\_\_contains\_\_后,in和not in运算符都可用

not in 运算符的返回值与 in相反

例：

class Mylist:

def \_\_init\_\_(self,iterable=()):

self.data=list(iterable)

def \_\_repr\_\_(self):

return "Mylist({})".format(self.data)

def \_\_contains\_\_(self, e):

return True if e in self.data else False

def main():

s1=Mylist([1,2,3,4,-5,6])

if 2 in s1: # 等同于if s1.\_\_contains\_\_(4)

print('2在s1内')

else:

print('2不在s1内')

if -4 not in s1: # 等同于if not s1.\_\_contains\_\_(4)

print('-4不在s1内')

else:

print('-4在s1内')

main()

（12）索引和切片运算符重载方法

方法名 运算符和表达式 说明

\_\_getitem\_\_(self,i) x = self(i) 索引/切片取值

\_\_setitem\_\_(self,i,v) self[i] = v 索引/切片赋值

\_\_delitem\_\_(self,i) del self[i] del语句删除索引/切片

作用:

让自定义的类型的对象能够支持索引和切片操作

例：

class Mylist:

def \_\_init\_\_(self,iterable=()):

self.\_\_data=list(iterable)

def \_\_repr\_\_(self):

return "Mylist({})".format(self.\_\_data)

def \_\_getitem\_\_(self,i):

'索引取值,i绑定[]内的元素'

print('i的值',i)

return self.\_\_data[i] # 返回data绑定列表中的第i个元素

def \_\_setitem\_\_(self,i,v):

'''此方法可以让自定义的列表支持索引赋值操作'''

print('\_\_setitem\_\_被调用,i=',i,'v=',v)

self.\_\_data[i]=v

def \_\_delitem\_\_(self,i):

del self.\_\_data[i] # self.\_\_data.pop(i)

return self

if type(i) is int:

print('用户正在用索引取值')

elif type(i) is slice:

print('用户正在用切片取值')

print('切片的起点是:', i.start)

print('切片的终点是:', i.stop)

print('切片的步长是:', i.step)

elif type(i) is str:

print('用户正在用字符串进行索引操作')

# raise KeyError

return self.\_\_data[i] # 返回data绑定的第i个元素

def main():

s=Mylist([1,2,3,4,-5,6])

print(s[3]) # 4

s[3]=400

print(s) # Mylist([1,2,3,400,-5,6])

del s[3]

print(s) # Mylist([1,2,3,-5,6])

print(s[::2]) # [1,3,6]

main()