**一、 选择题**

下列选项中， 不属于 Python 语言特点的是(C)。

A.简单易学 B.开源 C.面对过程 D.可移植性

————————————————————————————

下列领域中， 使用 Python 可以实现的是(ABCD)。 (多选)

A.Web 开发 B.操作系统管理 C.科学计算 D.游戏

————————————————————————————

下列关于 Python 2.x 和 Python 3.x 的说法， 正确的是(B)。

A.Python 3.x 使用 print 语句输出数据

B.Python 3.x 默认使用的编码是 UTF-8

C.Python 2.x 和 Python 3.x 使用//进行除法运算的结果不一致

D.Python 3.x 版本的异常可以直接被抛出

—————————————————————————————

下列关于 Python 的说法中， 错误的是(C)。

A.Python 是从 ABC 发展起来的

B.Python 是一门高级的计算机语言

C.Python 是一门只面向对象的语言

D.Python 是一种代表简单主义思想的语言

————————————————————————————

Python 中定义私有属性的方法是(D)。

A. 使用 private 关键字 B.使用 public 关键字

C.使用\_\_XX\_\_定义属性名 D.使用\_\_XX 定义属性名

————————————————————————————

下列选项中， 不属于面向对象程序设计的三个特征的是(A)。

A.抽象 B.封装 C. 继承 D.多态

————————————————————————————

以下 C 类继承 A 类和 B 类的格式中， 正确的是(C)。

A. class C A,B: B. class C（A:B） C.class C（A,B） D.class C A and B:

————————————————————————————

下列选项中， 与 class Person 等价的是(C)。

A. class Person（Object） B. class Person（Animal）

C. class Person（object） D. class Person: object

————————————————————————————

下列选项中， (D)的布尔值不是 Flase。

A.None

B.0

C.()

D.1

————————————————————————————

假设 a=9， b=2， 那么下列运算中， 错误的是(D)。

A.a+b 的值是 11

B.a//b 的值是 4

C.a%b 的值是 1

D.a\*\*b 的值是 18

————————————————————————————

下列标识符中， 合法的是(AD)。

A.helloWorld

B.2ndObj

C.hello#world

D.\_helloworld

————————————————————————————

下列符号中， 表示 Python 中单行注释的是(A)。

A.#

B.//

C.<!-- -->

D."“"

————————————————————————————

下列选项中， 符合 Python 命名规范的标识符是(C)。

A.user-Passwd

B.if

C.\_name

D.setup.exe

————————————————————————————

下列选项中， Python 不支持的数据类型有(B)。

A.int

B.char

C.float

D.dicitionary

————————————————————————————

下列表达式中， 返回 True 的是(B)。

A.a=2 b=2 a=b

B.3>2>1

C.True and False

D.2!=2

————————————————————————————

下列语句中， 哪个在 Python 中是非法的?(B)。

A.x = y = z = 1

B.x = (y = z + 1)

C.x, y = y, x

D.x += y

————————————————————————————

下列选项中， 幂运算的符号为(D)。

A.\*

B.++

C.%

D.\*\*

————————————————————————————

下列选项中， 会输出 1， 2， 3 三个数字的是(BC)。

A.

for i in range(3):

print(i)

——————

B.

for i in range(3):

print(i + 1)

——————

C.

a\_list = [0,1,2]

for i in a\_list:

print(i + 1)

——————

D.

i = 1

while i < 3:

print(i)

i = i + 1

————————————————————————————

阅读下面的代码：

sum = 0

for i in range(100):

if(i%10):

continue

sum = sum + i

print(sum)

上述程序的执行结果是(C)。

A.5050 B.4950 C.450 D.45

————————————————————————————

已知 x=10,y=20,z=30:以下语句执行后 x,y,z 的值是(C)。

if x < y:

z=x

x=y

y=z

A.10,20,30

B.10,20,20

C.20,10,10

D.20,10,30

————————————————————————————

当需要在字符串中使用特殊字符的时候， Python 使用(A)作为转义字符。

A.\

B./

C.#

D.%

————————————————————————————

下列数据中不属于字符串的是(D)。

A.‘ab’

B.’ ’ ‘perfect’ ’ ’

C.“52wo”

D.abc

————————————————————————————

使用(B)符号对浮点类型的数据进行格式化。

A.%c

B.%f

C.%d

D.%s

————————————————————————————

字符串’Hi,Andy’ 中， 字符’ A’ 对应的下标位置为(C)。

A.1

B.2

C.3

D.4

————————————————————————————

下列方法中， 能够返回某个子串在字符串中出现次数的是(C)。

A.length

B.index

C.count

D.find

————————————————————————————

下列方法中， 能够让所有单词的首字母变成大写的方法是(B)。

A.capitalize

B.title

C.upper

D.ljust

————————————————————————————

字符串的 strip 方法的作用是(A)。

A.删除字符串头尾指定的字符 B.删除字符串末尾的指定字符

C.删除字符串头部的指定字符 D.通过指定分隔符对字符串切片

————————————————————————————

有一个函数关系如下所示：

x y

x<0 x-1

x=0 x

x>0 x+1

下面程序段中， 能正确表示上面关系的是(C)。

A.

y = x + 1

if x >= 0

if x == 0:

y = x

else:

y = x - 1;

——————

B.

y = x - 1

if x != 0:

if x > 0:

y = x + 1

else:

y = x

——————

C.

if x <= 0:

if x < 0:

y = x - 1

else:

y = x

else:

y = x + 1

——————

D.

y = x

if x <= 0:

if x < 0:

y = x - 1

else:

y = x + 1

————————————————————————————

下列 Python 语句正确的是(D)。

A.min=x if x<y else y

B.max=x>y?x:y

C.if(x>y) print x

D.while True:pass

————————————————————————————

关于列表的说法， 描述有错误的是(D)。

A.list 是一个有序集合， 没有固定大小

B.list 可以存放任意类型的元素

C.使用 list 时， 其下标可以是负数

D.list 是不可变的数据类型

————————————————————————————

以下程序的输出结果是(B)。 (提示： ord(“a” )==97)

list\_demo=[1,2,3,4,5,'a','b']

print(list\_demo[1],list\_demo[5])

1

2

A.1 5

B.2 a

C.1 97

D.2 97

————————————————————————————

执行下面的操作后， list\_two 的值为(C)。

list\_one=[4,5,6]

list\_two=list\_one

list\_one[2]=3

1

2

3

A.[4,5,6]

B.[4,3,6]

C.[4,5,3]

D.A,B,C 都不正确

————————————————————————————

阅读下面的程序：

list\_demo=[1,2,1,3]

nums=set(list\_demo)

for i in nums:

print(i,end="")

1

2

3

4

程序执行的结果为(D)。

A.1213

B.213

C.321

D.123

————————————————————————————

下列选项中， 正确定义了一个字典的是(D)。

A.a=[‘a’ ,1,‘b’ ,2,‘c’ ,3]

B.b=(‘a’ ,1,‘b’ ,2,‘c’ ,3)

C.c={‘a’ ,1,‘b’ ,2,‘c’ ,3}

D.d={‘a’ :1,‘b’ :2,‘c’ :3}

————————————————————————————

下列选项中， 不能使用下标运算的是(C)。

A.列表

B.元组

C.集合

D.字符串

————————————————————————————

下列程序执行后输出的结果为(A)。

x = 'abc'

y = x

y = 100

print(x)

1

2

3

4

A.abc

B.100

C.97,98,99

D.以上三项均是错误的

————————————————————————————

下列删除列表中最后一个元素的函数是(B)。

A. del B. pop C. remove C. cut

————————————————————————————

下列函数中， 用于返回元祖中元素最小值的是(C)。

A. len B. max C. min D.tuple

————————————————————————————

阅读下面的程序：

def func():

print(x)

x=100

func()

1

2

3

4

执行上述语句后， 输出的结果为(C)。

A.0 B.100 C.程序出现异常 D.程序编译失败

————————————————————————————

下面关于函数的说法， 错误的是(C)。

A.函数可以减少代码的重复， 使得程序更加模块化

B.在不同的函数中可以使用相同名字的变量

C.调用函数时， 传入参数的顺序和函数定义时的顺序必须相同

D.函数体中如果没有 return 语句， 也会返回一个 None 值

————————————————————————————

下列有关函数的说法中， 正确的是(C)。

A.函数的定义必须在程序的开头

B.函数定义后， 其中的程序就可以自动执行

C.函数定义后需要调用才会执行

D.函数体与关键字 def 必须左对齐

————————————————————————————

下列函数调用使用的参数传递方式是(A)。

result = sum(num1, num2, num3)

1

A.位置绑定 B.关键字绑定 C.变量类型绑定 D.变量名称绑定

————————————————————————————

使用(C)关键字创建自定义函数。

A. Function B. func C.def D.procedure

————————————————————————————

使用(D)关键字声明匿名函数。

A. function B. func C.def D.lambda

————————————————————————————

关于装饰器， 下列说法错误的是(B)。

A.装饰器是一个包裹函数

B.装饰器智能有一个参数

C.通过在函数定义的面前加上@符号阿和装饰器名， 使得装饰器函数生效

D.如果装饰器带有参数， 则必须在装饰函数的外层再嵌套一层函数

————————————————————————————

下列函数中， 用于使用函数对制定序列进行过滤的是(C)。

A.map 函数 B.select 函数 C.filter 函数 D.reduce 函数

————————————————————————————

下列选项中， 不能作为 filter 函数参数的是(D)。

A.列表 B.元组 C.字符串 D.整数

————————————————————————————

阅读下面一段程序：

def foo():

a = 1

def bar():

a = a + 1

return a

return bar

print(foo()())

上述程序执行的结果为(A)。

A.程序出现异常

B.2

C.1

D.没有输出结果

————————————————————————————

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(C)。

A.‘r’ B.‘w’ C.‘a’ D.‘w+’

————————————————————————————

假设文件不存在， 如果使用 open 方法打开文件会报错， 那么该文件的打开方式是下列哪种模式?(A)

A.‘r’ B.‘w’ C.‘a’ D.‘w+’

————————————————————————————

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?(C)

A.file.read() B.file.read(200)

C.file.readline() D.file.readlines()

————————————————————————————

下列方法中， 用于向文件中写出内容的是(B)。

A.open B.write C.closeD.read

————————————————————————————

下列荣方法中， 用于获取当前目录的是(D)。

A.open B.write C.Getcwd D.read

————————————————————————————

下列语句打开文件的位置应该在(D)。

f = open('itheima.txt','w')

1

A.C 盘根目录下

B.D 盘根目录下

C.Python 安装目录下

D.与源文件在相同的目录下

————————————————————————————

若文本文件 abc.txt 中的内容如下：

abcdef

阅读下面的程序：

file=open("abc.txt","r")

s=file.readline()

s1=list(s)

print(s1)

上述程序执行的结果为(C)。

A.[‘abcdef’ ]

B.[‘abcdef\n’ ]

C.[‘a’ ,‘b’ ,‘c’ ,‘d’ ,‘e’ ,‘f’ ]

D.[‘a’ ,‘b’ ,‘c’ ,‘d’ ,‘e’ ,‘f’ ,’ \n’ ]

————————————————————————————

下列程序运行以后， 会产生如下(B)异常。

a

1

A.SyntaxError B.NameError C.IndexError D.KeyError

————————————————————————————

下列选项中， (C)是唯一不再运行时发生的异常。

A.ZeroDivisionError B.NameError C.SyntaxError D.KeyError

————————————————————————————

当 try 语句中没有任何错误信息时， 一定不会执行(D)语句。

A.try B.else C.finaly D.except

————————————————————————————

在完整的异常语句中， 语句出现的顺序正确的是(D)。

A.try---->except----->else---->finally

B.try---->else---->except----->finally

C.try---->except----->finally—>else

D.try----->else---->else----->except

————————————————————————————

下列选项中， 用于触发异常的是(A)。

A.try B.catch C.raise D.except

————————————————————————————

关于抛出异常的说法中， 描述错误的是(C)。

A.当 raise 指定异常的类名时， 会隐式地创建异常类的实例

B.显式地创建异常类实例， 可以使用 raise 直接引发

C.不带参数的 raise 语句， 只能引发刚刚发生过的异常

D.使用 raise 抛出异常时， 无法指定描述信息

————————————————————————————

关于抛出异常的说法中， 描述错误的是(D)。

A.当 raise 指定异常的类名时， 会隐式地创建异常类的实例

B.显式地创建异常类实例， 可以使用 raise 直接引发

C.不带参数的 raise 语句， 只能引发刚刚发生过的异常

D.使用 raise 抛出异常时， 无法指定描述信息

————————————————————————————

下列关键字中， 用来引入模块的是(C)。

A.include B.from C.import D.continue

————————————————————————————

关于引入模块的方式， 错误的是(D)。

A.import math

B.from fib import fibnacci

C.form math import \*

D.from \* import fib

————————————————————————————

关于\_\_name\_\_的说法， 下列描述错误的是(A)。

A.它是 Python 提供的一个方法

B.每个模块内部都有一个\_\_name\_\_属性

C.当它的值为’ \_\_main\_\_‘时， 表示模块自身在运行

D.当它的值不为’ \_\_main\_\_'时， 表示模块被引用

————————————————————————————

关于面向过程和面向对象， 下列说法错误的是(B)。

A.面向过程和面向对象都是解决问题的一种思路

B.面向过程是基于面向对象的

C.面向过程强调的是解决问题的步骤

D.面向对象强调的是解决问题的对象

————————————————————————————

关于类和对象的关系， 下列描述正确的是(D)。

A.类和面向对象的核心

B.类是现实中事物的个体

C.对象是根据类创建的， 并且一个类只能对应一个对象

D.对象描述的是现实的个体， 它是类的实例

————————————————————————————

构造方法的作用是(C)。

A.一般成员方法 B.类的初始化

C.对象的初始化 D.对象的建立

————————————————————————————

构造方法是类的一个特殊方法， Python 中它的名称为(C)。

A.与类同名 B.\_construct C.\_init\_ D. init

————————————————————————————

Python 类中包含一个特殊的变量(A)， 它表示当前对象自身， 可以访问类的成员。

A.self B.me C.this D.与类同名

————————————————————————————

下列选项中， 符合类的命名规范的是(A)。

A.HolidayResort

B.Holiday Resort

C.hoildayResort

D.hoilidayresort

————————————————————————————

Python 中用于释放类占用资源的方法是(B)。

A.\_\_init\_\_

B.\_\_del\_\_

C.\_del

D.delete

————————————————————————————

下列关于类属性和示例属性的说法， 描述正确的是(B)。

A.类属性既可以显示定义， 又能在方法中定义

B.公有类属性可以通过类和类的实例访问

C.通过类可以获取实例属性的值

D.类的实例只能获取实例属性的值

————————————————————————————

下列选项中， 用于标识为静态方法的是(C)。

A.@classmethood B.@instancemethod

C.@staticmethod D.@privatemethod

————————————————————————————

下列方法中， 不可以使用类名访问的是(A)。

A.实例方法 B.类方法 C.静态方法D.以上 3 项都不符合

————————————————————————————————————————————————

————————————————————————————————————————————————

二、 判断题

Python 是开源的， 它可以被一直到许多平台上。 (√ )

Python 的优点之一是具有伪代码的本质。 (√ )

Python 可以开发 Web 程序， 也可以管理操作系统。 (√ )

Mac 系统自带 Python 开发环境。 (√ )

Python 自带的 shell， 其性能由于 IPython。 (×)

我们编写的 Python 代码在运行过程中， 会被编译成二进制代码。 (×)

Python 程序被解释器转换后的文件格式后缀名为.pyc。 (√ )

Python 3.x 版本的代码完全兼容 Python 2.x。 (×)

PyCharm 是开发 Python 的集成开发环境。 (√ )

代码 print(3,4)是 Python 2.x 的输出格式。 (×)

———————————————————————————————

Python 使用符号#表示单行注释。 (√ )

标识符可以以数字开头。 (×)

type()方法可以查看变量的数据类型。 (√ )

Python 中的代码块使用缩进来表示。 (√ )

Python 中的多行语句可以使用反斜杠来实现。 (×)

Python 中标识符不区分大小写。 (×)

Python 中的标识符不能使用关键字。 (√ )

使用 help()命令可以进入帮助系统。 (√ )

Python 中的成员运算符用于判断制定序列中是否包含某个值。 (√ )

比较运算符用于比较两个数， 其返回的结果智能是 True 或 False。 (√ )

———————————————————————————————

elif 可以单独使用。 (×)

pass 语句的出现是为了保持进程结构的完整性。 (√ )

在 Python 中没有 switch-case 语句。 (√ )

每个 if 条件后面都要使用冒号。 (√ )

循环语句可以嵌套使用。 (√ )

———————————————————————————————

无论使用单引号或者双引号包含字符， 使用 print 输出的结果都一样。 (√ )

无论 input 接收任何的数据， 都会以字符串的方式进行保存。 (√ )

Python 中只有一个字母的字符串属于字符类型。 (×)

使用下标可以访问字符串中的每个字符。 (√ )

Python 中字符串的下表是从 1 开始的。 (×)

切片选区的区间范围是从起始位开始的， 到结束位结束。 (×)

如果 index 方法没有在字符串中找到子串， 则会返回-1。 (×)

———————————————————————————————

列表的索引是从 0 开始的。 (√ )

通过 insert 方法可以在制定位置插入元素。 (√ )

使用下标能修改列表的元素。 (√ )

列表的嵌套指的是一个列表的元素是另一个列表。 (√ )

通过下标索引可以修改和访问元祖的元素。 (×)

字典中的值只能够是字符串类型。 (×)

在字典中， 可以使用 count 方法计算键值对的个数。 (×)

———————————————————————————————

函数的名称可以随意命名。 (×)

不带 return 的函数代表返回 None。 (√ )

默认情况下， 参数值和参数名是跟函数声明定义的顺序匹配的。 (√ )

函数定义完成后， 系统会自动执行其内部的功能。 (×)

函数体以冒号起始， 并且是缩进格式的。 (√ )

带有默认值的参数一定位于参数列表的末尾。 (√ )

局部变量的作用域是整个程序， 任何时候使用都有效。 (×)

匿名函数就是没有名字的函数。 (√ )

———————————————————————————————

闭包是内部函数对外部作用域的变量进行引用。 (×)

当外部函数执行结束， 其内部闭包引用的变量一定会立即释放。 (×)

装饰器是一个变量。 (×)

装饰器函数至少要接收一个函数。 (√ )

装饰器既能装饰带参数的函数， 也能自己带参数。 (√ )

如果 map 函数传入的两个序列个数不同， 那么个数多的序列会把多余的元素删除。 (×)

map 函数只能传递一个序列。 (×)

map 传入函数的参数个数必须跟序列的个数一样。 (√ )

filter 传入的函数可以为 None。 (√ )

filter 函数智能对序列执行过滤操作。 (×)

filter 函数的返回值为字符串， 它的序列类型一定是字符串。 (√ )

———————————————————————————————

文件打开的默认方式是只读。 (√ )

打开一个可读写的文件， 如果文件存在会被覆盖。 (√ )

使用 write 方法写入文件时， 数据会追加到文件的末尾。 (√ )

实际开发中， 文件或者文件夹操作都要用到 os 模块。 (√ )

read 方法只能一次性读取文件中的所有数据。 (×)

———————————————————————————————

默认情况下， 系统检测到错误后会终止程序。 (√ )

在使用异常时必须先导入 exceptions 模块。 (× )

一个 try 语句只能对应一个 except 子句。 (×)

如果 except 子句没有指明任何异常类型， 则表示捕捉所有的异常。 (√ )

无论程序是否捕捉到异常， 一定会执行 finally 语句。 (√ )

所有的 except 子句一定在 else 和 finally 的前面。 (√ )

———————————————————————————————

Python 解释器会优先查看默认的路径搜索模块的位置。 (×)

每个 Python 文件就是一个模块。 (√ )

当\_\_name\_\_属性的值为\_\_main\_\_时， 代表该模块自身在运行。 (√ )

包目录下必须有一个\_\_init\_\_py 文件。 (√ )

外部模块都提供了自动安装的文件， 直接双击安装就行。 (×)

———————————————————————————————

面向对象是基于面向过程的。 (×)

通过类可以创建对象， 有且只有一个对象实例。 (×)

方法和杉树的格式是完全一样的。 (×)

创建类的对象时， 系统会自动调用构造方法进行初始化。 (√ )

创建完对象后， 其属性的初始值是固定的， 外界无法进行修改。 (×)

使用 del 语句删除对象， 可以手动释放它所占用的资源。 (√ )

———————————————————————————————

Python 中没有任何关键字区分公有属性和私有属性。 (√ )

继承会在原有类的基础上产生新的类， 这个新类就是父类。 (×)

带有两个下划线的方法一定是私有方法。 (√ )

子类能继承父类的一切属性和方法。 (×)

子类通过重写继承的方法， 覆盖掉跟父类同名的方法。 (√ )

如果类属性和实例属性重名， 对象有限访问类属性的值。 (×)

使用类名获取到的值一定是类属性的值。 (√ )

静态方法中一定不能访问实例属性的值。 (√ )

—————————————————————————————————————————————————

—————————————————————————————————————————————————

三、 填空题

在 Python 中， int 表示的数据类型是整型。

布尔类型的值包括 True 和 False。

Python 的浮点数占 8 个字节。

如果要在计算机中表示浮点数 1.2 × 1 0 5 1.2\times10^51.2×10

5

， 则表示方法为 1.2e5。

00001000>>2 的结果是 00000010。

若 a=20， 那么 bin(a)的值为 0b10100。

如果想测试变量的类型， 可以使用 type()来实现。

若 a=1， b=2， 那么(a or b)的值为 1。

若 a=10， b=20， 那么(a and b)结果为 20。

4.34E5 表示的是 4.34×10^5。

———————————————————————————————

在循环体中使用 break 语句可以跳出循环体。

elif 语句是 else 语句和 if 语句的组合。

在循环体中可以使用 continue 语句跳过本次循环后面的代码， 重新开始下一次循环。

如果希望循环是无限的， 我们可以通过设置条件表达式永远为 True 来实现无限循环。

Python 中的 pass 表示的是空语句。

———————————————————————————————

字符串是一种表示文本数据的类型。

像双引号这样的特殊符号， 需要对它进行转义输出。

Python3 提供了 input 函数从标准输入(如键盘)读入一行文本。

切片指的是对操作的对象截取其中的一部分。

切片选取的区间是左闭右开型的， 不包含结束位的值。

———————————————————————————————

Python 序列类型包括字符串、 列表和元组三种， 字典是 Python 中唯一的映射类型。

Python 中的可变数据类型有字典和列表。

在列表中查找元素时可以使用 not in 和 in 运算符。

如果要从小到大的排列列表的元素， 可以使用 sort 方法实现。

元组使用圆括号存放元素， 列表使用的是方括号。

———————————————————————————————

函数可以有多个参数， 参数之间使用逗号分隔。

使用 return 语句可以返回函数值并退出函数。

通过 return 结束函数， 从而选择性地返回一个值给调用方。

函数能处理比声明时更多的参数， 它们是不定长参数。

在函数里面调用另外一个函数， 这就是函数嵌套调用。

在函数的内部定义的变量称作局部变量。

全局变量定义在函数外， 可以在整个程序范围内访问。

如果想在函数中修改全部变量， 需要在变量的前面加上 global 关键字。

———————————————————————————————

内部函数引用了外部函数作用域的变量， 那么内部函数叫作闭包。

装饰器本质上是一个函数。

装饰器函数需要接收一个参数， 这个参数表示被修饰的函数。

在函数定义的前面添加装饰器名和@符号， 实现对函数的包装。

支持参数的装饰器函数需要再多一层内嵌函数。

map 函数会根据提供的函数对制定的序列做映射。

map 的两个序列的元素个数不一致， 那么元素少的序列会以 None 补齐。

filter 函数会对制定序列执行过滤操作。

filter 传入的函数的返回值是布尔值。

reduce 传入的是带有两个参数的函数， 该函数不能为 None。

———————————————————————————————

打开文件对文件进行读写， 操作完成后应该调用 close()方法关闭文件， 以释放资源。

seek 方法用于移动指针到制定位置， 该方法中 offset 参数表示要偏移的字节数。

使用 readlines 方法把整个文件中的内容进行一次性读取， 返回的是一个列表。

os 模块中的 mkdir 方法用于创建文件夹。

在读写文件的过程中， tell 方法可以获取当前的读写位置。

———————————————————————————————

Python 中所有的异常类都是 Exception 子类。

当使用序列中不存在的索引时， 会引发 IndexError 异常。

一个 try 语句智能对应一个 finally 子句。

当约束条件不满足时， assert 语句会触发 AssertionError 异常。

如果在没有 except 的 try 语句中使用 else 语句， 会引发语法错误。

———————————————————————————————

要调用 random 模块的 randint 函数， 书写形式为 random. randint。

每个 Python 文件都可以作为一个模块， 模块的名字就是文件的名字。

每个模块都有一个\_\_name\_\_属性， 使程序块智能在模块自身运行时执行。

为了更好地组织模块， 通常会把多个模块放在一个包中。

当程序中需要引入外部模块时， 需要从外面下载并安装。

如果要搜索模块的路径， 可以使用 sys 模块的 path 变量。

———————————————————————————————

在 Python 中， 可以使用 class 关键字来声明一个类。

面向对象需要把问题划分多个独立的对象， 然后调用其方法解决问题。

类的方法中必须有一个 self 参数， 位于参数列表的开头。

Python 提供了名称为\_\_init\_\_的构造方法， 实现让类的对象完成初始化。

如果想修改属性的默认值， 可以在构造方法中使用参数设置。

———————————————————————————————

如果属性名的前面加上了两个下划线， 就表明它是私有属性。

在现有类基础上构建新类， 新的类称作子类， 现有的类称作父类。

父类的私有属性和方法是不能被子类继承的， 更不能被子类访问。

Python 语言既支持单继承， 也支持多继承。

子类想按照自己的方式实现方法， 需要重写从父类继承的方法。

子类通过 super()可以成功地访问父类的成员。

位于类内部、 方法外部的方法是类方法。

类方法是类拥有的方法， 使用修饰器@classmethod 来标识。

————————————————————————————————————————————————

————————————————————————————————————————————————

四、 程序分析题

阅读下面的程序， 分析代码是否可以编译通过。 如果编译通过， 请列出运行的结果， 否则说明编译失败的原因。

代码一:

num\_one = input("请输入一个整数： ")

num\_two = input("请输入一个整数： ")

if num\_one % num\_two == 0:

print("验证码正确")

答： 不能编译通过。 因为 num1 和 num2 属于字符串类型， 不能执行取余操作。

—————————————————————————————————————

代码二:

name = 'Steve Jobs'

print(name[6])

结果为： J

—————————————————————————————————————

代码三:

string\_example = 'hello world itheima'

index = string\_example.index("itheima",0,10)

print(index)

答： 由于没有在字符串中找到子串， index 方法默认会抛出 ValueError 异常。

—————————————————————————————————————

代码四:

string\_example = " hello world "

print(string\_example.strip())

结果为： Hello World (默认删除两头的空格)

—————————————————————————————————————

代码五:

string\_example = "Hello" + 'Python'

print(string\_example)

结果为： HelloPython

阅读下面的程序， 分析代码是否能够编译通过。 如果能编译通过， 请列出运行的结果， 否则请说明编译失败的原因。

代码一:

tup = ('a','b','c')

tup[3] = 'd'

print(tup)

答： 程序运行错误， 元组不能使用下标增加元素。

—————————————————————————————————————

代码二:

dict\_demo={'a':1,'b':2,'c':3}

print(dict\_demo['a'])

结果为： 1

—————————————————————————————————————

代码三:

list\_demo=[10,23,66,26,35,1,76,88,58]

list\_demo.reverse()

print(list\_demo[3])

list\_demo.sort()

print(list\_demo[3])

结果为：

1

26

—————————————————————————————————————

阅读下面的程序， 分析代码是否能够编译通过。 如果能编译通过， 请列出运行的结果， 否则请说明编译失败的原因。

代码一:

def func():

x = 200

x = 100

func()

print(x)

结果为： 100

—————————————————————————————————————

代码二:

def func():

global x

x = 200

x = 100

func()

print(x)

结果为： 200

—————————————————————————————————————

代码三:

def func():

x = 200

def func2():

print(x)

func2()

x = 100

func()

print(x)

结果为：

200

100

—————————————————————————————————————

阅读下面的程序， 分析代码是否能够编译通过。 如果能编译通过， 请列出运行的结果， 否则请说明编译失败的原因。

代码一:

def funX():

x = 5

def funY():

nonlocal x

x += 1

return x

return funY

a = funX()

print(a())

print(a())

print(a())

结果为：

6

7

8

—————————————————————————————————————

代码二:

def funX():

x = 5

def funY():

nonlocal x

x += 1

return x

return funY

return funY

a = funX

print(a()())

print(a()())

print(a()())

结果为：

6

6

6

—————————————————————————————————————

阅读下面的程序， 分析代码是否能够编译通过。 如果能编译通过， 请列出运行的结果， 否则请说明编译失败的原因。

代码一:

class Person:

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

self.name = name

def \_\_str\_\_(self):

return "我的名字是" + self.name

person = Person("小明")

print(person)

结果为： 我的名字是小明

—————————————————————————————————————

代码二:

class Person:

def \_\_del\_\_(self):

print("--del--")

person = Person()

del person

print("--end--")

结果为：

--del--

--end--