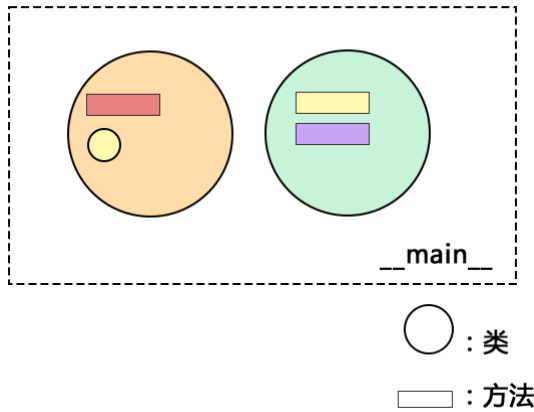


编程教育等级评测七级模拟题

理论部分

一、单选题

1. 集合思维是学习“类”的重要思维方式，请问下图使用类的方式来表示最准确的是哪个？



A.

```
class A
    def func_a
class B
    def func_b
    def func_c
class C
```

B.

```
class A
    def func_a
class C
class B
    def func_b
    def func_c
```

```
class A
class B
    def func_a
C.    def func_b
class C
    def func_c
```

```
class A
    def func_a
D.    class B
        def func_b
class C
    def func_c
```

答案：B

2. 选项中属于下图中类的实例的是哪个？

```
class Greeting:
    def __init__(self, name):
        self.name = name
    def say_hello(self):
        print(f'Hello, {self.name}!')
greeting = Greeting('xiaoan')
greeting.say_hello()
```

- A. greeting
- B. name
- C. Greeting
- D. say_hello

答案：A

3. 创建一个类D并将其实例化为d，现在需要通过一些内置函数完成以下操作：

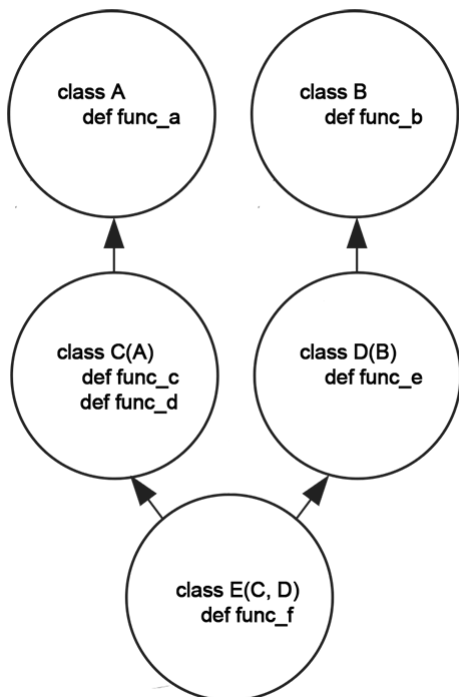
- (1)在实例d中添加一个值为'xiaohong'的属性name；
- (2)获取实例d的name属性的值；
- (3)检查实例d是否有name属性。

下面哪个选项组合的调用方法是正确的？

- setattr(d, 'name', 'xiaohong')
- A. getattr(d, 'name')
hasattr(d, 'name')
- hasattr(d, 'name', 'xiaohong')
- B. setattr(d, 'name')
getattr(d, 'name')
- hasattr(d, 'name')
- C. getattr(d, 'name', 'xiaohong')
setattr(d, 'name')
- getattr(d, 'name')
- D. setattr(d, 'name', 'xiaohong')
hasattr(d, 'name')

答案：A

4. 下图中有5个类，创建一个类E的实例e，请问e中可以调用哪些自定义方法？



- A. func_b、func_c、func_d、func_e、func_f
- B. func_c、func_d、func_e、func_f
- C. func_a、func_b、func_c、func_d、func_e、func_f

D. func_f

答案 : C

5. 下图中, 类Vehicle是所有类的基类, 类Car继承自类Vehicle, 类Sedan继承自类Car。类Vehicle和类Car都定义了一个名为fuel()的方法, 并且类Car和类Sedan还定义了一个名为drive()的方法, 实例化类Sedan并依次调用实例的方法drive()和fuel(), 输出的两种结果正确的是哪个?

```
class Vehicle:
    def fuel(self):
        print('Regular')

class Car(Vehicle):
    def fuel(self):
        print('Premium')
    def drive(self):
        print('Drive slow')

class Sedan(Car):
    def drive(self):
        print('Drive fast')
```

A. Drive fast
Premium

B. Drive slow
Regular

C. Drive fast
Regular

D. Drive slow
Premium

答案 : A

6. 在Python3中, datetime模块提供了处理日期和时间的类。假设你想要同时获取当前的日期和时间, datetime模块中的哪个方法实例化运行后可以返回该值?

A. datetime.time

B. datetime.date

C. datetime.datetime.now

D. datetime.timedelta

答案 : C

7. 有A、B、C、D四个类, 其中类D继承了类B和类C, 而类B又继承了类A。请问根据Python3中关于新类的MRO算法, 类D的实例在调用父类方法时正确的解析顺序是哪个?

A. D->C->B->A

B. D->C->B->A

C. D->B->C->A

D. A->B->C->D

答案：C

8. 在Python3中，创建了一个空字典dict_a并往里面添加一些键值对，然后通过dict_a[key]的方式获取字典中对应key的值。请问下面选项中还有哪个方法可以实现和dict_a[key]相同的功能？

A. dict_a.__getattr__(key)

B. dict_a.__getitem__(key)

C. dict_a.__setitem__(key, value)

D. dict_a.__setattr__(key, value)

答案：B

9. 使用下面的类计算'yzxyzxyz', 先将'yzxyzxyz'初始化到A的实例中，然后将实例赋值给变量a，再运行a-'xyz'，得到的结果正确的是哪个？

class A:

```
def __init__(self, str_a):
    self.str_a = str_a
def __sub__(self, str_b):
    self.str_a = self.str_a.replace(str_b, '')
    return self.str_a
```

A. 'yzxyzxyz'

B. 返回TypeError

C. 'yzyz'

D. 'yzxyzxyz'

答案：C

10. 如果对解释器进行修改，在分词逻辑中增加一条命令：将等于=符号作为“可以断开”的标点符号，发生作用就会被消耗掉（不计数）。那么下面这段程序总共可以分出几个词？（不必考虑语法的正确性）

class A:

```
_id = 0
def cal_mul(self, p_a, p_b):
    a = p_a
    b = p_b
    return a * b
```

A. 33

B. 26

C. 36

D. 23

答案：D

二、多选题

1. 创建一个类A，并在init()中初始化了一个实例属性v，然后又定义了下面选项中的四个方法，最后实例化该类，并同时传入一个v的整数类型的参数，请问当实例调用这四个类时不会返回异常的有哪些？

A.

```
def __str__(self):  
    return str(self.v)
```

B.

```
def show_number():  
    print(self.v)
```

C.

```
def _set_data(a, b):  
    a.v = b
```

D.

```
def __len__(self):  
    return len(self.v)
```

答案：AC

2. 下面关于try语句语法结构正确的有哪些？

A.

```
try:  
    ...  
except:  
    ...  
finally:  
    ...
```

B.

```
try:  
    ...  
except:  
    ...  
else:  
    ...
```

C.

```
try:  
    ...  
finally:  
    ...  
else:  
    ...
```

```

try:
    ...
else:
    ...
except:
    ...

```

答案：AB

3. 在Python3中，表达式中的比较符号可以在自定义类中通过调用特殊方法来进行重载，下面选项中的比较符号与其对应的重载方法匹配正确的有哪些？

- A. 符号<对应__lt__
- B. 符号>=对应__le__
- C. 符号>对应__gt__
- D. 符号==对应__eq__

答案：ACD

4. 通过下图中的类可以创建一个可迭代的实例对象，然后传入一个列表list_a，并允许使用for语句遍历list_a中的元素。其中的a、b、c三个空格中按顺序分别需要填入哪些代码？

```

class A:
    def __init__(self, data):
        self.data = data

    def a(self, index):
        return self.data[index]

    def b(self):
        return len(self.data)

    def c(self):
        return iter(sorted(self.data, reverse=True))

```

- A. __iter__
- B. __getitem__
- C. __len__
- D. __setitem__

答案：ABC

5. 在Python3中，使用第三方库numpy中的方法linalg.solve可以求解二元一次方程组，下面选项中箭头左边的矩阵乘法形态转换回箭头右边二元一次方程组形态，正确的有哪些？

A. $\begin{bmatrix} 5 & -2 \\ 2 & 7 \end{bmatrix} * \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 3 \\ 10 \end{bmatrix} \Rightarrow \begin{cases} 5x + 2y = 3 \\ -2x + 7y = 10 \end{cases}$

B. $\begin{bmatrix} 3 & -2 \\ 4 & 1 \end{bmatrix} * \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 8 \\ 5 \end{bmatrix} \Rightarrow \begin{cases} 3x - 2y = 8 \\ 4x + y = 5 \end{cases}$

$$C. \begin{bmatrix} 6 & 4 \\ 3 & -2 \end{bmatrix} * \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 12 \\ 5 \end{bmatrix} \Rightarrow \begin{cases} 6x + 4y = 12 \\ 3y - 2x = 5 \end{cases}$$

$$D. \begin{bmatrix} 2 & 3 \\ 1 & -4 \end{bmatrix} * \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 6 \\ -7 \end{bmatrix} \Rightarrow \begin{cases} 2x + 3y = 6 \\ x - 4y = -7 \end{cases}$$

答案:BD

三、判断题

1. 在Python3中, 使用raise语句可以让程序自动引发异常。

答案:Y

2. 在Python3中, 使用import sys可以导入sys模块的功能, 当导入功能后, 可以使用sys.doc查看该模块的文档信息。

答案:Y

3. 在Python3中, 创建一个类A, 其中包含一个方法func_a, 将A实例化为a, 并调用func_a, 正确方式为a.func_a()。

答案:Y

4. 在Python3中, 当一个对象调用内部成员时, 如果属性和方法重名, 那么对象会优先调用其方法。

答案:N

5. 在Python3中, 一个类一旦实例化后便无法再修改其内部的属性。

答案:N

实操部分

四、简答题

1. 图1是一个电子自行车的app界面, 图2是自行车的体力值状态划分表。请用Python3编写一个自行车类(Bicycle), 其包含自行车型号(model)和体力值(stamina)两个实例属性; 以及骑行(ride)、休息(rest)和获取体力信息(get_stamina_status)三个实例方法。骑行和休息的行为会影响自行车体力值的变化(骑行的行为会让stamina-30, 休息的行为会让stamina+20, 但不能超过范围: $0 \leq \text{stamina} \leq 100$, 见图2)。体力信息根据自行车的体力值的变化划分为三个级别: 体力充足、体力一般和体力不足。最后实例化该类并调用相关方法。



图1

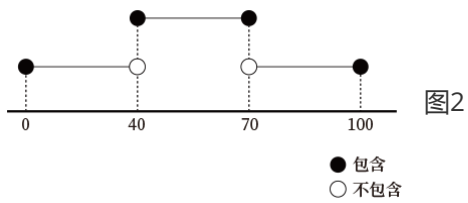


图2

输出示例:

调用Bicycle实例中的get_stamina_status(), 输出: Giant 体力充足

调用Bicycle实例中的ride(), 输出: Giant 正在骑行

调用Bicycle实例中的rest (), 输出: Giant 正在休息

...

程序结构:

请参考下面的程序结构, 然后根据题目要求在本题下方的代码输入框中编写程序。

#类创建

```
class Bicycle:
    def __init__(...):
        ...
    def ride(...):
        ...
    def rest(...):
        ...
    ...
    ...
```

```
if __name__ == '__main__':
```

```
    #类对象的创建
```

```
    ...
```

```
    #调用对象方法
```

```
    ..
```

【参考答案】

#类创建

```
class Bicycle:
```

```
    def __init__(self, model, stamina=100):
```

```
        self.model = model
```

```
        self.stamina = stamina
```

```
    def ride(self):
```

```
        self.stamina -= 30
```

```
        if self.stamina < 0:
```

```
            self.stamina = 0
```

```
        print(f"{self.model} 正在骑行")
```

```
    def rest(self):
```

```
        self.stamina += 20
```

```
        if self.stamina > 100:
```

```
            self.stamina = 100
```

```
        print(f"{self.model} 正在休息")
```

```
    def get_stamina_status(self):
```

```
        if self.stamina > 70:
```

```
            return f"{self.model} 体力充足"
```

```
        elif self.stamina > 40:
```

```
            return f"{self.model} 体力一般"
```

```
        else:
```

```
            return f"{self.model} 体力不足"
```

```
if __name__ == '__main__':
```

```
    #类对象的创建
```

```
    bike = Bicycle('Giant')
```

```
    #调用对象方法
```

```
    print(bike.get_stamina_status())
```

```
    bike.ride()
```

```
    bike.ride()
```

```
    bike.ride()
```

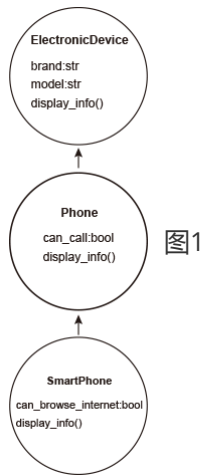
```
    print(bike.get_stamina_status())
```

```
    bike.rest()
```

```
    bike.rest()
```

```
    print(bike.get_stamina_status())
```

2. 图1展示的是电子设备类(ElectronicDevice)、手机类 (Phone) 和智能手机类 (SmartPhone) 的继承关系。其中ElectronicDevice是Phone的基类，Phone是SmartPhone的基类。请用Python3编写这三个类，然后分别实例化手机类 (Phone) 和智能手机类 (SmartPhone) ，并调用各个实例对象中的方法，然后根据图2给出的参数信息打印出相关内容。



	brand	model	can_call	can_browse_Internet
Phone	Nola	3310	True	False
SmartPhone	Orange	uPhone 12	True	True

图2

输出示例:

调用Phone实例中的display_info(), 输出:

这是一款Nola 3310电子设备

这款设备可以打电话

调用SmartPhone实例中的display_info(), 输出:

这是一款Orange uPhone12电子设备

这款设备可以打电话

这款设备可以浏览互联网

程序结构:

参考下面的程序结构，然后根据题目要求在本题下方的代码输入框中编写程序。

#类创建

```
class ElectronicDevice:
```

```
...
```

```
class Phone(...):
```

```
...
```

```
class SmartPhone(...):
```

```
...
```

```
if __name__ == '__main__':
```

```
    #类对象的创建
```

```
    ...
```

```
    #调用对象方法
```

```
    ...
```

【参考答案】

#类创建

```
class ElectronicDevice:
```

```
    def __init__(self, brand, model):
```

```
        self.brand = brand
```

```
        self.model = model
```

```
    def display_info(self):
```

```
        print(f"这是一款{self.brand} {self.model}电子设备")
```

```
class Phone(ElectronicDevice):
```

```
    def __init__(self, brand, model, can_call):
```

```
        super().__init__(brand, model)
```

```
        self.can_call = can_call
```

```
    def display_info(self):
```

```
        super().display_info()
```

```
        if self.can_call:
```

```
            print(f"这款设备可以打电话")
```

```
class SmartPhone(Phone):
```

```
    def __init__(self, brand, model, can_call, can_browse_internet):
```

```
        super().__init__(brand, model, can_call)
```

```
        self.can_browse_internet = can_browse_internet
```

```
    def display_info(self):
```

```
        super().display_info()
```

```
        if self.can_browse_internet:
```

```
            print(f"这款设备可以浏览互联网")
```

```
if __name__ == '__main__':
```

```
    #类对象的创建
```

```
    phone = Phone('Nola', '3310', True)
```

```
    smart_phone = SmartPhone('Orange', 'uPhone 12', True, True)
```

```
    #调用对象方法
```

```
    phone.display_info()
```

```
    smart_phone.display_info()
```

3. 编写一个Numbers类，它有一个整数int类型的实例属性num和一个len()方法。重写len()方法，允许使用len()函数来计算一个数字的位数（比如n是Numbers类的一个实例，当n = 12345时，len(n)返回5）。然后将该类实例化为n，传入一个整数，尝试使用len函数计算并打印结果。

输出示例:

当实例n中的num为12345，len(n)返回5

当实例n中的num为246，len(n)返回3

程序结构:

#类创建

```
class Number:
```

```
...
```

```
if __name__ == '__main__':
```

```
    #类对象的创建
```

```
...
```

```
    #打印输出
```

```
...
```

【参考答案】

类创建

```
class Number:
```

```
    def __init__(self, num):  
        self.num = num
```

```
    def __len__(self):  
        return len(str(self.num))
```

```
if __name__ == '__main__':
```

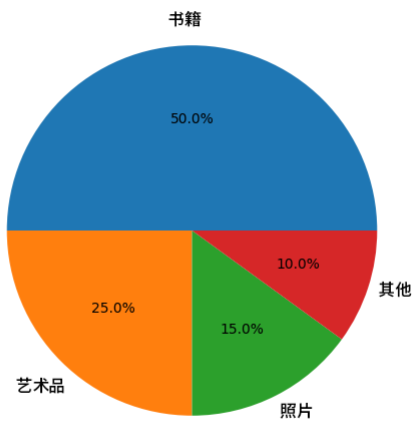
```
    # 类对象的创建
```

```
    n = Number(12345)
```

```
    # 打印输出
```

```
    print(len(n))
```

4. 某商场举办了一次公益慈善售卖活动。在这个活动中，一共有4种物品参加售卖，下面是这4种物品的收益占比(income_dict)。总收益为12500元(total_income)，请根据每种物品的收益占比计算出各自的收益金额。编写一个Charity类，完成上面的问题，将这些数据放入一个字典中，然后作为参数传入类的实例中，在方法show_income中进行转换，并按照下面输出示例的格式进行输出。



输出示例:

书籍: ...元

艺术品: ...元

...

程序结构:

请参考下面的程序结构，然后根据题目要求在本题下方的代码输入框中编写程序。

```
income_dict = {"书籍": 0.5, "艺术品": 0.25, ...}
```

```
total_income = 12500
```

```
#类创建
```

```
class Charity:
```

```
...
```

```
if __name__ == '__main__':
```

```
    #类对象的创建
```

```
...
```

```
    #打印输出
```

```
...
```

【参考答案】

```
income_dict = {"书籍": 0.5, "艺术品": 0.25, "照片": 0.15, "其他": 0.1}
total_income = 12500
```

#类创建

```
class Charity:
    def __init__(self, income_dict, total_income):
        self.income_dict = income_dict
        self.total_income = total_income

    def show_income(self):
        for item, percentage in self.income_dict.items():
            income = self.total_income * percentage
            print(f"{item}: {int(income)}元")

if __name__ == '__main__':
    #类对象的创建
    charity = Charity(income_dict, total_income)
    #打印输出
    charity.show_income()
```

5. 一只飞机的燃油箱是一个立方体，边长(length)为1.5米，那么这个燃油箱的容积(volume)有多少立方米？同时，使用防腐膜覆盖油箱内壁，那么覆盖完成后，内壁的防腐膜的面积是多少平方米？请编写一个FuelTank类，在类中建立volume()和area()两个方法，对上述两个问题分别进行计算，并返回计算结果。最后实例化类，并调用方法，对结果进行打印输出。（结果保留两位小数）

要点提示：

立方体体积计算公式：边长 \times 边长 \times 边长

立方体表面积计算公式：6 \times 边长 \times 边长

输出示例：

这个燃油箱的容积是...立方米

覆盖燃油箱内壁的防腐膜面积是...平方米

程序结构：

请参考下面的程序结构，然后根据题目要求在本题下方的代码输入框中编写程序。

#类创建

```
class FuelTank:
```

```
    ...
```

```
if __name__ == '__main__':
```

```
    #类对象的创建
```

...

#调用对象方法

...

【参考答案】

类创建

```
class FuelTank:
```

```
    def __init__(self, length):  
        self.length = length
```

```
    def volume(self):  
        # 计算并返回燃油箱体积  
        volume = self.length**3  
        return round(volume, 2)
```

```
    def area(self):  
        # 计算并返回防腐膜覆盖面积  
        area = 6 * self.length**2  
        return round(area, 2)
```

```
if __name__ == '__main__':
```

```
    # 类对象的创建
```

```
    fueltank = FuelTank(1.5)
```

```
    # 调用对象方法
```

```
    print(f"这个燃油箱的容积是{fueltank.volume()}立方米")
```

```
    print(f"覆盖燃油箱内壁的防腐膜面积是{fueltank.area()}平方米")
```