**Python与数据分析**

**实验报告**

|  |  |  |
| --- | --- | --- |
| **课程名称：**Python与数据分析 | **学期：**2022~2023学年上学期 | **成绩：** |
| **指导教师：**王灿 | **学生姓名：** | **学生学号：** |
| **实验名称：**Python中的类与对象 | | |
| **实验编号：**7 | **实验日期：**10月3日 | **实验学时：**NA |
| **学院：**理学院 | **专业：**信息与计算科学 | **班级：**信计xxxx |

**一、实验目的**

1．熟悉Python中面向对象编程。

2．熟悉Python中类的继承，组合。

**二、实验内容**

1. 类的继承和组合
   1. 编写Date类
      1. 属性包括：年，月，日
      2. 方法包括：返回格式化的日期字符串，如2022-10-03
   2. 编写People类
      1. 属性包括：姓名，性别，出生日期（使用Date类定义对象）
      2. 方法包括：打印姓名，性别，生日（调用Date类的方法）
   3. 编写Teacher类，需要继承People类
      1. 属性包括：姓名，性别，出生日期，课程（需要用集合表示）
      2. 方法包括：teach(self, place)（打印xxx正在place教课，其中place是一个地点的字符串）
   4. 编写Student类，需要继承People类
      1. 属性包括：姓名，性别，出生日期，每门课的分数（需要用字典表示）
      2. 方法包括：study(self, place)（打印yyy正在place学习，其中place是一个地点的字符串）
   5. 分别对上述4个类建立1个实例，并运行他们的所有方法，展示运行效果
   6. 对Teacher的实例添加一门课程，展示添加后的结果（用集合的add方法）
   7. 对Student的实例添加一门课的分数，展示添加后的结果（用字典的相关方法）
2. 运行下面的代码，体会各类的属性和方法含义，回答下面的问题
   1. Hero和Switch是什么关系？（继承/组合）
   2. Hero和Skill是什么关系？（继承/组合）
   3. 为什么swap不放在类的定义内？（从函数是否需要依赖类和函数使用范围等方面考虑）
   4. 定义一个新的英雄类Change，具有技能change\_hp(self, another\_hero)，技能作用是交换当前英雄和另一英雄的hp，需要用到swap函数。

def swap(a, b):

"""交换两个变量的值"""

return b, a

class Skill(object):

"""能造成伤害的攻击或技能"""

def \_\_init\_\_(self, name, attack, hurt, user):

self.name = name # 技能名称

self.attack = attack # 造成的伤害

self.hurt = hurt # 对自身的伤害

self.user = user # 技能使用者

def apply(self, down):

if not self.user.is\_alive():

print('当前英雄 {} 已阵亡，无法发动 {}'.format(self.user.name, self.name))

return

if not down.is\_alive():

print('{} 已阵亡，无法攻击'.format(down.name))

return

print('{} 向 {} 发动 {}, 造成伤害 {}'.format(self.user.name, down.name, self.name, self.attack))

down.update(0, 0, -self.attack)

self.user.update(0, 0, -self.hurt)

class Hero(object):

"""英雄"""

def \_\_init\_\_(self, name, money, attack, hp):

self.name = name

self.money = money

self.attack = attack

self.hp = hp

self.normal\_attack = Skill('普通攻击', self.attack, 0, self)

def show\_info(self):

print('英雄名: {}, 金钱: {}, 攻击力: {}, 血量: {}'.format(

self.name, self.money, self.attack, self.hp))

def is\_alive(self):

return self.hp > 0

def update(self, money, attack, hp):

if self.hp > 0 and self.hp + hp <= 0:

print('{}阵亡'.format(self.name))

self.money += money

self.attack += attack

self.hp += hp

self.money = max(0, self.money)

self.attack = max(0, self.attack)

self.hp = max(0, self.hp)

class Switch(Hero):

"""英雄Switch"""

def switch\_attack\_hp(self):

skill\_name = '攻击hp交换技能'

if self.hp == 0:

print('当前英雄 {} 已阵亡，无法发动 {}'.format(self.name, skill\_name))

return

print('{} 发动 {}'.format(self.name, skill\_name))

self.attack, self.hp = swap(self.attack, self.hp)

class Destroy(Hero):

"""英雄Destroy"""

def \_\_init\_\_(self, name, money, attack, hp):

Hero.\_\_init\_\_(self, name, money, attack, hp)

self.hurt\_both = Skill('两败俱伤', 750, 750, self)

def show\_status(\*heros):

"""展示多个英雄的当前状态"""

print('-' \* 52)

for hero in heros:

hero.show\_info()

print('-' \* 52)

a = Hero('普通英雄', 2000, 500, 700)

b = Switch('攻击hp交换师', 2000, 1200, 800)

c = Destroy('搏命人', 2000, 1000, 900)

show\_status(a, b, c)

a.normal\_attack.apply(b)

show\_status(a, b, c)

b.switch\_attack\_hp()

show\_status(a, b, c)

c.hurt\_both.apply(a)

show\_status(a, b, c)

c.normal\_attack.apply(a)

show\_status(a, b, c)

c.hurt\_both.apply(b)

show\_status(a, b, c)

c.hurt\_both.apply(b)

show\_status(a, b, c)

**三、实验环境**

Windows/Linux/Mac OS操作系统；conda创建的python3.10环境。

**四、实验过程和结果**

**1．实验步骤、结果和结论（包括必要的截图）**

**2．关键代码及其解释**

**3．调试过程（如果和后面的实验总结重复，可以只写实验总结）**

**五、实验总结（不能为空）**

**1．遇到的问题及解决过程**

**2．产生的错误及原因分析**

**3****．体会和收获**

**六、参考文献**

[1]柳毅等著，Python数据分析与实践，北京：清华大学出版社，2019.7.

**七、教师评语**