

第三次作业讲评

郭明非

2022.03.07

1.1

创建一个名为PkuClass 的类， 其方法__init__() 设置两个属性：
student_num 和teacher_num 。

创建一个名为describe_class() 的方法， 能够返回student_num 和
teacher_num。

创建一个quit()方法， 能够减去相应的退课学生数。

```
# 叶风灿 1900017806
class PkuClass():
    def __init__(self, s_n, t_n):
        self.student_num, self.teacher_num = s_n, t_n

    def describe_class(self):
        return self.student_num, self.teacher_num

    def quit(self, s_n):
        self.student_num -= s_n
        assert self.student_num >= 0

pku_class = PkuClass(100, 1)
pku_class.quit(10)
assert pku_class.describe_class() == (90, 1)
print('Test passed!')
```

1.2 继承1.1的PkuClass类实现一个PythonClass，修改相应默认函数，使得直接print该类就能在屏幕中打出该课程的学生人数和教师人数

```
# 杨昊翔 2000017741
class PythonClass(PkuClass):
    def __init__(self, s_n, t_n):
        super(PythonClass, self).__init__(s_n, t_n)

    def __str__(self):
        return "该课共有学生%d人, 老师%d人" % (self.student_num, self.teacher_num)

python_class = PythonClass(100, 1)
print(python_class)
```

1.3 用类实现装饰器@deprecated。

```
# 郝俊霖 2000013140
class deprecated(object):

    def __init__(self, since, reason):
        self.since = since
        self.reason = reason

    def __call__(self, func):
        def wrapper(*args, **kwargs):
            print(f"The function {func.__name__} version {self.since} is deprecated! {self.reason}.")
            return func(*args, **kwargs)
        return wrapper

# since: 指定已被弃用的版本
# forRemoval: 表示在将来的既定版本中会被删除
@deprecated(since = "1.1", reason = "This function will be removed soon")
def f():
    print('Foo!')

f()
```

提示：记得写函数的返回值。

1.4

类之间的继承关系如图，可以看成是一棵倒着的树。请实现按照 F -> E -> C -> B -> D -> A 的顺序遍历这棵树。建议使用super。



提示：MRO即Method Resolution Order（方法解析顺序），在调用方法时，会对当前类以及所有的基类进行一个搜索，以确定顺序。在该题（树形继承）的情况下，MRO的顺序是深度优先的。super以当前所处类为基准，调用MRO中的下一个类的函数。

<https://zh.wikipedia.org/wiki/C3线性化>

```
# 张子韞 2000012970
```

```
class A():  
    # TODO  
    def __init__(self):  
        print('A')  
        super(A, self).__init__()
```

```
class B():  
    # TODO  
    def __init__(self):  
        print('B')  
        super(B, self).__init__()
```

```
class C():  
    # TODO  
    def __init__(self):  
        print('C')  
        super(C, self).__init__()
```

```
class D(A):  
    # TODO  
    def __init__(self):  
        print('D')  
        super(D, self).__init__()
```

```
class E(C, B):  
    # TODO  
    def __init__(self):  
        print('E')  
        super(E, self).__init__()
```

```
class F(E, D):  
    # TODO  
    def __init__(self):  
        print('F')  
        super(F, self).__init__()
```

```
f = F()
```

```
class A():
    def __init__(self):
        print("A")
    # TODO

class B():
    def __init__(self):
        print("B")
    # TODO

class C():
    def __init__(self):
        print("C")
    # TODO
```

```
# 王田雨 2000012903
class D(A):
    def __init__(self):
        print("D")
        super(D,self).__init__()
    # TODO

class E(C, B):
    def __init__(self):
        print("E")
        super(E,self).__init__()
        super(C,self).__init__()
    # TODO

class F(E, D):
    def __init__(self):
        print("F")
        super(F,self).__init__()
        super(B,self).__init__()
```

1.5 最小堆insert。

```
def insert(self, k):
    # 杜思娴 2000017403
    self.current_size += 1
    self.heap_list.append(k)
    self.sift_up(self.current_size)

def sift_up(self, i):
    # 向上移动二叉树中的值，维护最小堆
    while i // 2 > 0:
        if self.heap_list[i] < self.heap_list[i // 2]:
            self.heap_list[i], self.heap_list[i // 2] = self.heap_list[i // 2], self.heap_list[i]
        i = i // 2
```

提示：记得调用sift_up，维护最小堆。

1.5 最小堆sort。

```
def delete_min(self):
    # 删除最小值
    if len(self.heap_list) == 1:
        return 'Empty heap'

    root = self.heap_list[1]
    self.heap_list[1] = self.heap_list[self.current_size]
    *self.heap_list, _ = self.heap_list
    self.current_size -= 1

    self.sift_down(1)
    return root

def sort(self):
    # 李昊洋 2000012918
    ans = []
    while self.current_size > 0:
        ans.append(self.delete_min())
    print(ans)
```

提示：从小到大排序，依次取出堆的根节点的数值。

*self.heap_list表示将不定长度的数字重新放进self.heap_list中，类似*args。

1.5 最小堆iter。

```
def __iter__(self):  
    # 邓朝萌 1900013039  
    return self  
  
def __next__(self):  
    if self.current_size == 0:  
        raise StopIteration  
    return self.delete_min()
```

提示：重写__iter__和__next__，每次调用__next__弹出堆顶值。