

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ
ФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«КУБАНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
(ФГБОУ ВО «КубГУ»)

Факультет математики и компьютерных наук
Кафедра математических и компьютерных методов

ЛАБОРАТОРНАЯ РАБОТА № 5
по дисциплине
СОВРЕМЕННЫЕ КОМПЬЮТЕРНЫЕ ТЕХНОЛОГИИ
Тема: «Иерархия классов»

Работу выполнила
студентка 3 курса
группы 33/2
Бахрачева А.Р.

Краснодар 2020

Задача

Напишите программу, в которой создается иерархия классов. Наполните классы соответствующими атрибутами и методами (оригинальными и перегруженными).

Вариант	Перечень классов
1.	студент, преподаватель, персона, заведующий кафедрой;

Текст программы

```
class Person:
```

```
    def __init__(self, name, age):
        self.out_str = ''
        self.values = []
        self.name = name
        self.age = age
        self.make_out_str('Person', ['name', 'age'])
        self.values.append(self.name)
        self.values.append(self.age)
        pass

    def make_out_str(self, class_name, arg_list):
        self.out_str = ''
        self.out_str += (class_name + ':\n')
        for ind, i in enumerate(arg_list):
            self.out_str += ('    {}: '.format(i) + '{}\n'.format('{0[' + str(ind) + ']}'))
        pass

    def __str__(self):
        return self.out_str.format(self.values)

    def get_name(self):
        return self.name

    def get_age(self):
        return self.age
```

```
class Student(Person):
```

```
    def __init__(self, name, age, group):
        Person.__init__(self, name, age)
        self.group = group
        self.make_out_str('Student', ['name', 'age', 'group'])
        self.values.append(self.group)
```

```
pass
```

```
def get_group(self):  
    return self.group
```

```
class Teacher(Person):
```

```
    def __init__(self, name, age, faculty):  
        Person.__init__(self, name, age)  
        self.faculty = faculty  
        self.make_out_str('Teacher', ['name', 'age', 'faculty'])  
        self.values.append(self.faculty)  
    pass
```

```
    def get_faculty(self):  
        return self.faculty
```

```
class TeacherMaster(Teacher):
```

```
    def __init__(self, name, age, faculty, main_subject):  
        Teacher.__init__(self, name, age, faculty)  
        self.main_subject = main_subject  
        self.make_out_str('TeacherMaster', ['name', 'age', 'faculty', 'main_subject'])  
        self.values.append(self.main_subject)  
    pass
```

```
    def get_main_subject(self):  
        return self.main_subject
```

```
def main():  
    person = Person('SomePerson', 28)  
    student = Student('Vadim', 19, 32)  
    teacher = Teacher('Oleg Igorevich', 42, 'FMaCS')  
    teacher_master = TeacherMaster('Vladimir Semenovich', 56, 'FMaCS', 'Python  
programming')
```

```
    print(person)  
    print(student)  
    print(teacher)  
    print(teacher_master)
```

```
pass
```

```
if __name__ == '__main__':  
    main()
```

Результат выполнения

Person:

name: SomePerson
age: 28

Student:

name: Vadim
age: 19
group: 32

Teacher:

name: Oleg Igorevich
age: 42
faculty: FMaCS

TeacherMaster:

name: Vladimir Semenovich
age: 56
faculty: FMaCS
main_subject: Python programming