Студент группы ИС-22 Кузнецова А.Ю.

Практическое занятие №16

Тема: Составление программ с использованием ООП в IDE PyCharm Community

Цель: закрепить усвоенные знания, понятия, алгоритмы, основные принципы составления программ с ООП в IDE PyCharm Community

Постановка задачи №1: Создайте класс 'студент', который имеет атрибуты имя, фамилия и оценки. Добавьте методы для вычисления среднего балла и определения, является ли студент отличником

Текст программы №1:

```
class Student:
    def __init__(self, name, surname, marks):
        self.name = name
        self.surname = surname
        self.marks = marks
    def srednee(self):
        vse = 0
        for i in self.marks:
            vse += i
        sred = vse / len(self.marks)
        return sred
    def otlichnik(self, sred):
        if sred == 5:
            return('Отличник')
        else:
            return('Не отличник')

student1 = Student('Stasy', 'Bread', [2, 2, 3, 4, 4])
print(student1.__dict__)
print('Среднее по оценкам', student1.srednee())
print('Отличник?', student1.otlichnik(student1.srednee()))
print('----------')
student2 = Student('Prohor', 'Patrikeev', [5, 5, 5, 5, 5])
print('Среднее по оценкам', student2.srednee())
print('Отличник?', student2.otlichnik(student2.srednee()))
```

Протокол работы программы №1:

{'name': 'Stasy', 'surname': 'Bread', 'marks': [2, 2, 3, 4, 4]}

Среднее по оценкам 3.0

Отличник? Не отличник

Среднее по оценкам 5.0

Отличник? Отличник

Process finished with exit code 0

Постановка задачи №2: Создание базового класса 'Животное' и его наследование для создания классов 'Собака' и 'Кошка'. В классе 'Животное' будут общие методы, такие как 'Дышать' и 'Питаться', а классы-наследники будут иметь свои уникальные методы и свойства, такие как 'гавкать' и 'мурлыкать'

Текст программы №2:

```
class Tier:#дышать, есть, слышать, бегать
    def __init__(self, name, atmen, essen):
        self.name = name
        self.atmen = atmen
        self.essen = essen
    def atmen(self):
        return self.atmen
    def essen(self):
        return self.essen
    #def horen(self):
        #self.horen = True
    #def laufen(self):
        #self.laufen = True

class Katze(Tier):
    def mur(self):
        return('mur mur mur')

class Hund(Tier):
    def gav(self):
        return('gav gav gav')

animal_base = Tier('Base-tier', False, False)
```

```
print(animal_base.__dict__, animal_base.essen)

cat = Katze('Lola', True, False)
print(cat.__dict__, cat.atmen, cat.mur())

dog = Hund('Lila', True, True)
print(dog.__dict__, dog.atmen, dog.gav())
```

Протокол работы программы №2:

{'name': 'Base-tier', 'atmen': False, 'essen': False} False

{'name': 'Lola', 'atmen': True, 'essen': False} True mur mur mur

{'name': 'Lila', 'atmen': True, 'essen': True} True gav gav gav

Process finished with exit code 0

Постановка задачи №3: Для задачи из блока 1 создать две функции, save_def и load_def, которые позволяют сохранять информацию из экземпляров класса(3шт) в файл и загружать ее обратно. Использовать модуль picle для сериализации и десериализации объектов Python в бинарном формате

Текст программы №3:

```
import pickle as p
class Student2:
    def __init__(self, name, surname, marks):
        self.name = name
        self.surname = surname
        self.marks = marks
    def srednee(self):
        vse = 0
        for i in self.marks:
            vse += i
        sred = vse / len(self.marks)
        return sred
    def otlichnik(self, sred):
        if sred == 5:
            return('Отличник')
        else:
            return('Не отличник')

def save_def(obj):
```

```
    with open('f1.bin', 'wb') as f:
        p.dump(obj, f)
    return('сохранено')
def load_def():
    with open('f1.bin', 'rb') as f:
        obj = p.load(f)
    return obj

student01 = Student2('Klo', 'Stu', [2, 4, 5, 3, 4, 4, 3])
student02 = Student2('Bro', 'Bin', [2, 3, 3, 3, 2])
student03 = Student2('Tru', 'Sam', [4, 5, 3, 5, 5, 4])

exempl = [student01, student02, student03]#попытка обьеденить все три обьекта
print('exempl =', exempl)
save_def(exempl)
retrieve = load_def()
for x in retrieve:
    print(x.__dict__)
```

Протокол работы программы №3:

exempl = [<__main__.Student2 object at 0x00000241CF3180D0>, <__main__.Student2 object at 0x00000241CF318490>, <__main__.Student2 object at 0x00000241CF350EE0>]

{'name': 'Klo', 'surname': 'Stu', 'marks': [2, 4, 5, 3, 4, 4, 3]}

{'name': 'Bro', 'surname': 'Bin', 'marks': [2, 3, 3, 3, 2]}

{'name': 'Tru', 'surname': 'Sam', 'marks': [4, 5, 3, 5, 5, 4]}

Process finished with exit code 0

Вывод: мной были закреплены усвоенные знания, понятия, алгоритмы, основные принципы составления программ и приобретены навыки работы с ООП в IDE PyCharm Community