РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ им. П. Лумумба

Факультет физико-математических и естественных наук

Кафедра теории вероятностей и кибербезопасности

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ № 1

Дисциплина: *Компьютерный практикум по моделированию*

Студент: Королёв Иван Андреевич

Группа: НКАбд-04-22

МОСКВА

2024 г.

Цель работы: Научиться работать с классами, объектами, атрибутами. Использовать конструкторы, методы.

Выполнение работы

Задание 1. В программе объявлен класс Car с некоторыми атрибутами. Добавьте в него метод drive(), который будет выводить сообщение 'Вперед!', а затем вызовите этот метод

Листинг программы на языке Python:

```
class Car:
    brand = "VW"
    model = "Polo"

    def drive():
        print("Вперед!")
```

Результаты (численные значения, графики, скриншоты):

```
: Car.drive()
  Вперед!
```

Задание 2. Создайте класс Terminator и реализуйте в нем два метода:

- say_greetings, печатающий фразу 'I am T-700, give me your ride and jacket!'
- say_goodbye, печатающий фразу 'I'll be back!'

Объявите два экземпляра вашего класс: T700 и T800. Вызовите каждый из методов для каждого экземпляра класса поочередно.

Листинг программы на языке Python:

```
class Terminator:
    def say_greetings(self):
        print("I am T-700, give me your ride and jacket!")

    def say_goodbye(self):
        print("I'll be back!")
```

Результаты (численные значения, графики, скриншоты):

```
T700 = Terminator()
T800 = Terminator()

T700.say_greetings()
T700.say_goodbye()

T800.say_greetings()
T800.say_goodbye()
```

```
I am T-700, give me your ride and jacket!
I'll be back!
I am T-700, give me your ride and jacket!
I'll be back!
```

Задание 3. Создайте класс Motorbike, имеющий инициализатор __init__, который принимает объем двигателя и год выпуска мотоцикла в качестве аргументов и сохраняет их в атрибутах engine и year. Объем двигателя и год выпуска мотоцикла передаются на вход программы пользователем.

Листинг программы на языке Python:

```
class Motorbike:

    def __init__(self, engine, year):
        self.engine = engine
        self.year = year

    def output(self): # функция для теста класса
        return f"\nОбъем двигателя мотоцикла: {self.engine}\nГод выпуска мотоцикла: {self.year}"
```

Результаты (численные значения, графики, скриншоты):

```
engine = input("Введите объем двигателя мотоцикла: ")
year = input("Введите год выпуска мотоцикла: ")

bike = Motorbike(engine, year)
print(bike.output())
```

```
Введите объем двигателя мотоцикла: 2.5
Введите год выпуска мотоцикла: 2022

Объем двигателя мотоцикла: 2.5
Год выпуска мотоцикла: 2022
```

Задание 4. Создайте класс Car, имеющий инициализатор __init__, который принимает и инициализирует атрибуты класса: brand, model, price, а также атрибут name – строку следующего вида: . Брэнд, модель и цена автомобиля передаются на вход программы пользователем.

Листинг программы на языке Python:

```
class Car:

    def __init__(self, br, mod, pr):
        self.brand = br
        self.model = mod
        self.price = price
        self.name = f"{br} {mod}"

    def __str__(self):
        return f"""\n{self.brand} - Бренд автомобиля\n{self.model} - Модель автомобиля\n{self.price} -
        Цена автомобиля\n{self.name} - Бренд и модель автомобиля
        """
```

Результаты (численные значения, графики, скриншоты):

```
brand = input("Введите марку автомобиля: ")
model = input("Введите модель автомобиля: ")
price = input("Введите цену автомобиля: ")

bmw = Car(brand, model, price)
print(bmw)
```

```
Введите марку автомобиля: BMW
Введите модель автомобиля: 5-series
Введите цену автомобиля: 3000000

BMW - Бренд автомобиля
5-series - Модель автомобиля
3000000 - Цена автомобиля
BMW 5-series - Бренд и модель автомобиля
```

Задание 5. Создайте класс HockeyPlayer, у которого есть:

- инициализатор __init__, принимающий в качестве аргументов first_name и last_name, то есть имя и фамилию хоккеиста. Также во время инициализации объекта необходимо создать два атрибута-счетчика: goals и assists. Эти атрибуты будут хранить информацию о забитых голах и передачах каждого хоккеиста. Оба атрибута необходимо проинициализировать нулями;
- метод add_goals, который добавляет количество голов в счетчик игрока. По умолчанию добавляет один гол;
- метод add_assists, который добавляет количество передач в счетчик игрока. По умолчанию добавляет одну передачу;
- метод statistics, который считает результативность игрока и возвращает строку вида: <Фамилия>, <Имя> - <голы> + 0.5*<перадачи>

Фамилия, имя, количество голов и передач хоккеиста передаются на вход программы пользователем.

Листинг программы на языке Python:

```
class HockeyPlayer:

    def __init__(self, firs_name, last_name, goals=0, assist=0):
        self.first_name = firs_name
        self.last_name = last_name
        self.goals = goals
        self.assist = assist

    def add_goals(self):
        self.goals = int(self.goals) + 1

    def add_assist(self):
        self.assist = int(self.assist) + 1

    def statistics(self):
        return f"""\n{self.first_name} {self.last_name} - {float(self.goals) + 0.5*float(self.assist)}"""
```

Результаты (численные значения, графики, скриншоты):

```
A,O = input("Введите имя хоккеиста: "), input("Введите фамилию хоккеиста: ")
goals = input("Введите кол-во забитых шайб: ")
assists = input("Введите кол-во передач: ")

Aleksandr_Ovechkin = HockeyPlayer(A, O, goals, assists)

print("До функции add_goals ", Aleksandr_Ovechkin.goals)

Aleksandr_Ovechkin.add_goals()

print("После add_goals ", Aleksandr_Ovechkin.goals)


print(Aleksandr_Ovechkin.statistics())
```

```
Введите имя хоккеиста: Александр
Введите фамилию хоккеиста: Овечкин
Введите кол-во забитых шайб: 30
Введите кол-во передач: 20
До функции add_goals  30
После add_goals  31

Александр Овечкин - 41.0
```

Задание 6. Создайте класс Array, который содержит атрибут values и 3 метода:

- Инициализатор __init__, который принимает произвольное число аргументов, фильтрует их с помощью второго метода и сохраняет в values.
- get_integers, который вызывается из инициализатора, фильтрует все его аргументы оставляя только числовые, а затем сортирует их по возрастанию и возвращает в инициализатор в виде списка.
- Переопределенный метод __str__, который возвращает строку: "Массив(числа…)", если массив не пустой, или строку "Пустой массив", если массив пуст.

Листинг программы на языке Python:

```
class Array:
    values = 0

    def __init__(self, *args):
        self.args = []
        self.args = args
        self.values = self.get_integers()

    def get_integers(self):
        result = list(filter(lambda x: type(x) is int or type(x) is float, self.args))
        return sorted(result)

    def __str__(self):
        if self.values:
            return f"\nМассив{self.values}"
        else:
            return f"\nМассив пуст"
```

Результаты (численные значения, графики, скриншоты):

```
n = Array(1, 2, 3, 4, 6, "242", 31, "ke", 2.1, "saw", 2.124)
print(n)

empty = Array()
print(empty)
```

```
Массив[1, 2, 2.1, 2.124, 3, 4, 6, 31]

Массив пуст
```

Заключение.

Я научился работать с классами, с его объектами, экземплярами и атрибутами.

Использовал методы и конструкторы.№