Министерство образования и науки Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования

«Национальный исследовательский Томский политехнический Университет»

****

самостоятельная работа № 5

курс:

# «ООП в Python»

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Исполнитель:** |  | | | | |
| студент группы | 0В21ДзебанА.А |  | ФИО студента |  | 26.12.24 |
|  |  |  |  |  |  |
| **Руководитель:** |  | | | | |
| преподаватель |  |  | Полищук В.Ю. |  |  |
|  |  |  |  |  |  |

Томск – 2024

Цель работы: обрести навыки ООП.

Задание:

1. Создать класс ThreeStates и определить у него поле state и метод hasstate. Атрибут state реализовать как приватный. В рамках метода реализовать переключение состояний: первое, второе, третье. Продолжительность первого состояния составляет 5 секунд, второго — 3 секунды, третьего — 2 секунды. Переключение между состояниями должно осуществляться только в указанном порядке (первое, второе, третье). Проверить работу примера, создав экземпляр и вызвав описанный метод.

2.Реализовать класс Volume, в котором определить защищенные поля: length (длина), width (ширина) и height (высота). Значения данных атрибутов должны передаваться при создании экземпляра класса. Определить метод расчета, использовав формулу: длина\*ширина\*высота. Проверить работу метода.

3. Реализовать базовый класс Employee, в котором определить атрибуты: name, patronymic, surname, salary. Последний атрибут должен быть защищенным и ссылаться на словарь, содержащий элементы: жалование и бонус, например, {"wage": wage, "bonus": bonus}. Создать класс Salary на базе класса Employee. В классе Salary реализовать методы получения полного имени сотрудника (get\_full\_name) и дохода с учетом премии (get\_total\_income). Создать экземпляры класса Salary, передать данные, проверить значения атрибутов, вызвать методы экземпляров.

4. Реализуйте базовый класс Airplane. У данного класса должны быть следующие атрибуты: speed, color, name, is\_jet (булево). А также методы: go, stop, direction, которые должны сообщать, что самолет летит, не летит, повернул в полете. Опишите несколько дочерних классов самолетов: FastAirplane, Biplane, ArmyAirplane. Добавьте в базовый класс метод show\_speed, который должен показывать текущую скорость самолета. Для класса FastAirplane переопределите метод show\_speed. При значении скорости свыше 1300 (FastAirplane) должно выводиться сообщение о сверхзвуковой скорости. Создайте экземпляры классов, передайте значения атрибутов. Выполните доступ к атрибутам, выведите результат. Выполните вызов методов и также покажите результат.

5. Реализовать класс MathOperations. Определить в нем атрибуты first\_num, second\_num и метод calc. Метод выводит сообщение “Запуск операции”. Создать три дочерних класса my\_sum (сложение), my\_sub (вычитание), my\_mult (умножение). В каждом из классов реализовать переопределение метода calc. Для каждого из классов метод должен выполнять математическую операцию, соответствующую названию класса. Создать экземпляры классов и проверить работу методов.

Ответ на задание необходимо предоставить в виде работающей без ошибок программы, код программы необходимо защитить перед преподавателем. К программе необходимо приложить файл в формате Word или PDF. Файл должен содержать следующие разделы: титульный лист, задание, описание хода проделанной работы, описание должно содержать получившийся код программы с пояснениями, и результаты, где описать усвоенные навыки полученные по результатам выполнения работы.

Ход работы:

1. Импорт библиотеки time для реализации класса ThreeStates:

import time

1. Определим класс ThreeStates согласно заданию и добавим соответствующий метод hasstate:

class ThreeStates:

    def \_\_init\_\_(self):

        self.\_\_state = 1

    def hasstate(self):

        while True:

            if self.\_\_state == 1:

                print("Первое состояние")

                time.sleep(5)

                self.\_\_state = 2

            elif self.\_\_state == 2:

                print("Второе состояние")

                time.sleep(3)

                self.\_\_state = 3

            elif self.\_\_state == 3:

                print("Третье состояние")

                time.sleep(2)

                self.\_\_state = 1

Для проверки корректности выполнения создадим экземпляр класса:

states = ThreeStates()

states.hasstate()

Вывод будет состоять из поочередной смены состояний, пока пользователь не прервет исполнение.

1. Определим класс вычисления объема volume согласно заданию:

class Volume:

    def \_\_init\_\_(self, length, width, height):

        self.\_length = length

        self.\_width = width

        self.\_height = height

    def calculate\_volume(self):

        return self.\_length \* self.\_width \* self.\_height

Для проверки корректности создадим экземпляр класса, и вычислим объем используя соответствующий метод:

volume = Volume(2, 3, 4)

print("Объем:", volume.calculate\_volume())

Объем: 24

1. Создадим соответствующие классы Employee и дочерние классы Salary(Employee):

class Employee:

    def \_\_init\_\_(self, name, patronymic, surname, wage, bonus):

        self.name = name

        self.patronymic = patronymic

        self.surname = surname

        self.\_salary = {"wage": wage, "bonus": bonus}

class Salary(Employee):

    def get\_full\_name(self):

        return f"{self.surname} {self.name} {self.patronymic}"

    def get\_total\_income(self):

        return self.\_salary["wage"] + self.\_salary["bonus"]

# Проверка

emp = Salary("Арсений", "Андреевич", "Андреевич", 50000, 10000)

print("Полное имя:", emp.get\_full\_name())

print("Общий доход:", emp.get\_total\_income())

Вывод:   
Полное имя: Андреевич Дзебан Арсений

Общий доход: 60000

Для решения задачи 4 определим материнский класс Airplane, его методы и дочерние классы FastAirplane, Biplane, ArmyAirplane:

# Задача 4

class Airplane:

    def \_\_init\_\_(self, speed, color, name, is\_jet, is\_biplane, is\_army):

        self.speed = speed

        self.color = color

        self.name = name

        self.is\_jet = is\_jet

        self.get\_biplane = is\_biplane

        self.get\_army = is\_army

    def go(self):

        print(f"{self.name} летит")

    def stop(self):

        print(f"{self.name} не летит")

    def direction(self, direction):

        print(f"{self.name} повернул {direction}")

    def show\_speed(self):

        print(f"Текущая скорость {self.name}: {self.speed}")

class FastAirplane(Airplane):

    def show\_speed(self):

        if self.speed > 1300:

            print(f"{self.name} имеет сверхзвуковую скорость: {self.speed}")

        else:

            super().show\_speed()

class Biplane(Airplane):

    def is\_biplane(self):

        print(f"{self.get\_biplane}")

class ArmyAirplane(Airplane):

    def is\_army(self):

        print(f"{self.get\_army}")

Причем метод show\_speed будет являться переопределением метода из материнского класса:

# Проверка

fast\_plane = FastAirplane(1500, "белый", "Стриж", True, True, False)

biplane = Biplane(200, "красный", "Ан-2", False, True, False)

army\_plane = ArmyAirplane(800, "зеленый", "Су-35", True, True, True)

fast\_plane.go()

fast\_plane.show\_speed()

biplane.stop()

army\_plane.direction("налево")

army\_plane.is\_army()

Вывод:

Стриж летит

Стриж имеет сверхзвуковую скорость: 1500

Ан-2 не летит

Су-35 повернул налево

True

1. Для решение задачи 5 создадим материнский класс MathOperations (self), содержащий метод calc и 3 дочерних класса MySum, MySub, MyMult, переопределяющие метод calc в каждом случае.

# Задача 5

class MathOperations:

    def \_\_init\_\_(self, first\_num, second\_num):

        self.first\_num = first\_num

        self.second\_num = second\_num

    def calc(self):

        print("Запуск операции")

class MySum(MathOperations):

    def calc(self):

        super().calc()

        return self.first\_num + self.second\_num

class MySub(MathOperations):

    def calc(self):

        super().calc()

        return self.first\_num - self.second\_num

class MyMult(MathOperations):

    def calc(self):

        super().calc()

        return self.first\_num \* self.second\_num

Проверка:

# Проверка

sum\_op = MySum(10, 5)

sub\_op = MySub(10, 5)

mult\_op = MyMult(10, 5)

print("Сложение:", sum\_op.calc())

print("Вычитание:", sub\_op.calc())

print("Умножение:", mult\_op.calc())

Вывод:

Запуск операции

Сложение: 15

Запуск операции

Вычитание: 5

Запуск операции

Умножение: 50

**Вывод:** был изучен ООП подход к программированию в python.