Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Кафедра экономической информатики

**Лабораторная работа №4**

**«Основы ООП»**

Выполнил: студент гр. 414302 К. С. Черненко

Проверил: Мацокин М П.

Минск 2025

**Цель работы:** закрепление на практике теоретических знаний по работе с ООП на языке программирования Python путем решения комплекса задач.

**Вариант 29**

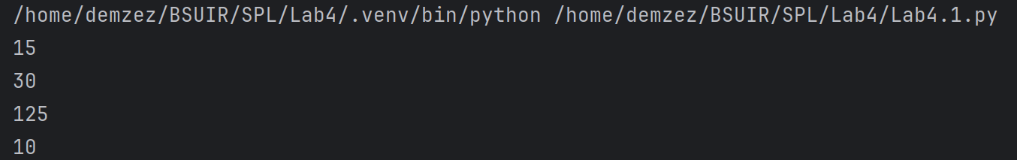
**Задание 1.**

Класс Example. В нём пропишите 3 (метода) функции. Две переменные задайте статически, две динамически. Первый метод создайте переменную и выведите её Второй метод: верните сумму 2-ух глобальных переменных. Третий метод: верните результат возведения первой динамической переменной во вторую динамическую переменную. Создайте объект класса. Напечатайте оба метода. Напечатайте переменную a.

Листинг 1

|  |
| --- |
| class Example:  static\_var1 = 10  static\_var2 = 20   def \_\_init\_\_(self):  self.dynamic\_var1 = 5  self.dynamic\_var2 = 3   def method1(self):  var = 15  print(var)   def method2(self):  return self.static\_var1 + self.static\_var2   def method3(self):  return self.dynamic\_var1 \*\* self.dynamic\_var2  obj = Example() obj.method1() print(obj.method2()) print(obj.method3()) print(obj.static\_var1) |

Результата работы программы:



**Задание 2.**

Создать класс Country: Столица, Площадь, Численность населения.

Создать список объектов. Вывести:

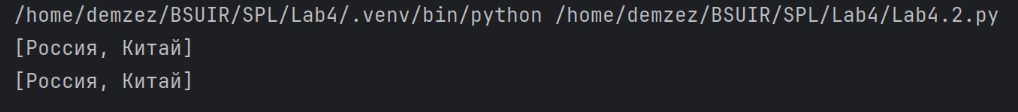
a) список стран по заданной площади;

b) список стран по заданной численности населения.

Листинг 2

|  |
| --- |
| class Country:  def \_\_init\_\_(self, name, capital, area, population):  self.name = name  self.capital = capital  self.area = area  self.population = population   def \_\_repr\_\_(self):  return self.name  countries = [  Country('Россия', 'Москва', 17125191, 146700000),  Country('Германия', 'Берлин', 357385, 83190556),  Country('Китай', 'Пекин', 9596961, 1402000000) ]  def filter\_by\_area(countries, min\_area):  return [c for c in countries if c.area >= min\_area]  def filter\_by\_population(countries, min\_population):  return [c for c in countries if c.population >= min\_population]  print(filter\_by\_area(countries, 1000000)) print(filter\_by\_population(countries, 100000000)) |

Результата работы программы:

****

**Задание 3.**

1. Реализуйте базовый класс Car.

2. У класса должны быть следующие атрибуты: speed, color, name, is\_police(булево). А также методы: go, stop, turn(direction), которые должны сообщать, что машина поехала, остановилась, повернула (куда);

3. Опишите несколько дочерних классов: TownCar, SportCar, WorkCar, PoliceCar;

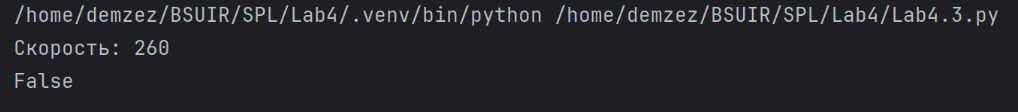
4. Добавьте в базовый класс метод show\_speed, который должен показывать текущую скорость автомобиля;

5. Для классов TownCar и WorkCar переопределите метод show\_speed. При значении скорости свыше 60 (TownCar) и 40 (WorkCar) должно выводиться сообщение о превышении скорости.

Листинг 3

|  |
| --- |
| class Car:  def \_\_init\_\_(self, speed, color, name, is\_police=False):  self.speed = speed  self.color = color  self.name = name  self.is\_police = is\_police   def go(self):  print(f"{self.name} поехала")   def stop(self):  print(f"{self.name} остановилась")   def turn(self, direction):  print(f"{self.name} повернула {direction}")   def show\_speed(self):  print(f"Скорость: {self.speed}")  class TownCar(Car):  def show\_speed(self):  super().show\_speed()  if self.speed > 60:  print("Превышение скорости!")  class SportCar(Car):  pass  class WorkCar(Car):  def show\_speed(self):  super().show\_speed()  if self.speed > 40:  print("Превышение скорости!")  class PoliceCar(Car):  def \_\_init\_\_(self, speed, color, name):  super().\_\_init\_\_(speed, color, name, True)    NewCar = SportCar(260, 'green', 'Urus', False) #PoliceCar  NewCar.show\_speed() print(NewCar.is\_police) |

Результата работы программы:

****

**Задание 4.**

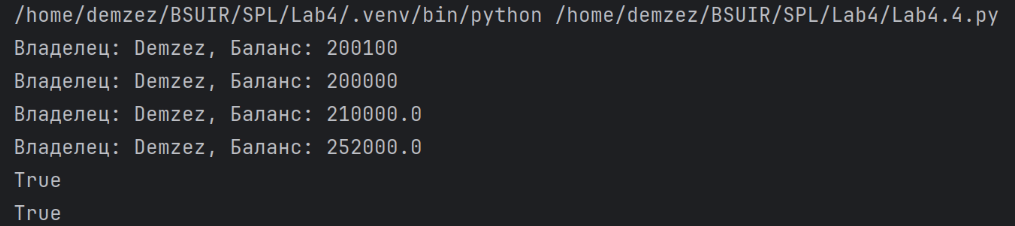
Придумать класс самостоятельно, реализовать в нем методы экземпляра

класса, статические, методы, методы класса.

Листинг 4

|  |
| --- |
| class BankAccount:  interest\_rate = 0.05   def \_\_init\_\_(self, owner, balance=0):  self.owner = owner  self.balance = balance   def deposit(self, amount):  self.balance += amount  return self.balance   def withdraw(self, amount):  if amount <= self.balance:  self.balance -= amount  return amount  print("Недостаточно средств")   @classmethod # декоратор статического метода не имеет доступа к полям экземпляря  def set\_interest\_rate(cls, rate):  cls.interest\_rate = rate   @staticmethod # декоратор статического метода не имеет доступа к полям ни класса ни экземпляря  def validate\_amount(amount):  return amount > 0   def apply\_interest(self):  self.balance += self.balance \* self.interest\_rate   def \_\_repr\_\_(self):  return f"Владелец: {self.owner}, Баланс: {self.balance}"   myAccount = BankAccount('Demzez', 200000) myAccount.deposit(100) print(myAccount)  myAccount.withdraw(100) print(myAccount)  myAccount.apply\_interest() print(myAccount)  BankAccount.set\_interest\_rate(0.2) myAccount.apply\_interest() print(myAccount)  print(BankAccount.validate\_amount(100)) print(myAccount.validate\_amount(100)) |

Результата работы программы:



**Вывод:**

В результате выполнения лабораторной работы были успешно закреплено умение работы с конструкциями ООП на языке программирования Python, освоена основная область применения ООП и синтаксис работы с ним.