**Лабораторная работа 6 Функции**

**Задание 6.1 (Выполняют все)**

(Интернет-магазин) Решите задачу об интернет-торговле.

Несколько покупателей в течении года делали покупки в интернет- магазине. При каждой

покупке фиксировались имя покупателя (строка) и потраченная сумма (действительное число).

Напишите функцию, рассчитывающую для каждого покупателя и выдающую в виде словаря по

всем покупателям (вида имя:значение) один из следующих параметров:

1. число покупок;

2. среднюю сумму покупки;

3. максимальную сумму покупки;

4. минимальную сумму покупки;

5. общую сумму всех покупок.

На вход функции передаётся:

• либо 2 списка, в первом из которых имена покупателей (могут повторяться), во втором –

суммы покупок;

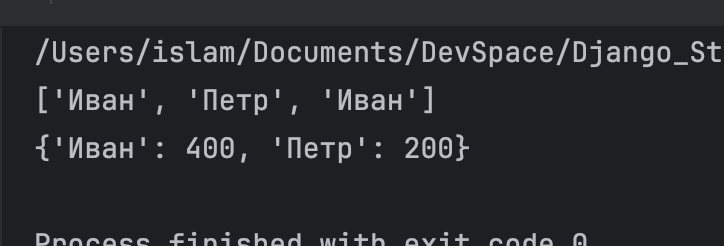
• либо 1 список, состоящий из пар вида (имя, сумма);

• либо словарь, в котором в качестве ключей используются имена, а в качестве значений \_

списки с суммами.

def analyze\_purchases(data, mode=1):  
 if isinstance(data, tuple): # (имена, суммы)  
 names, amounts = data  
 d = {}  
 for n, a in zip(names, amounts):  
 d.setdefault(n, []).append(a)  
 elif isinstance(data, list): # [(имя, сумма), ...]  
 d = {}  
 for n, a in data:  
 d.setdefault(n, []).append(a)  
 else: # {имя: [суммы]}  
 d = data  
  
 funcs = {  
 1: lambda x: len(x),  
 2: lambda x: sum(x)/len(x),  
 3: lambda x: max(x),  
 4: lambda x: min(x),  
 5: lambda x: sum(x)  
 }  
 return {k: funcs[mode](v) for k, v in d.items()}  
  
names = ["Иван", "Петр", "Иван"]  
amounts = [100, 200, 300]  
print(analyze\_purchases((names, amounts), 5))

result:



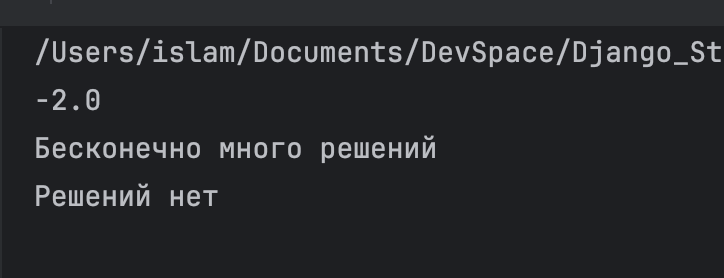
**Задание 6.2 (По индив. номеру)**

13. Написать функцию, вычисляющую корни линейного уравнения. Рассмотреть

случаи а = 0, а≠0 и т.д.

def linear\_equation(a, b):  
 if a == 0 and b == 0:  
 return "Бесконечно много решений"  
 elif a == 0 and b != 0:  
 return "Решений нет"  
 else:  
 return -b / a  
print(linear\_equation(2, 4))   
print(linear\_equation(0, 0))   
print(linear\_equation(0, 5))

result:



**Задание 6.3 (По индив. номеру)**

13. Создайте словарь, представляющий собой записную книжку с тратами по разным

категорям, где ключи - названия категорий, а значения - деньги, которые были потрачены в

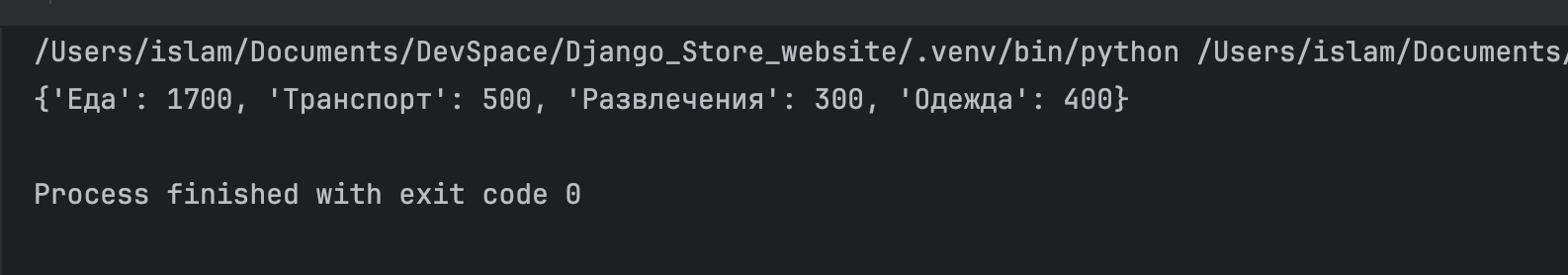
этой категории (целые положительные числа). Напишите функцию, которая принимает два

словаря (представляющих две записные книжки) и возвращает новый словарь, в котором

значения трат сложены.

def merge\_expenses(book1, book2):  
 result = book1.copy()  
 for k, v in book2.items():  
 result[k] = result.get(k, 0) + v  
 return result  
a = {"Еда": 1000, "Транспорт": 500, "Развлечения": 300}  
b = {"Еда": 700, "Одежда": 400}  
print(merge\_expenses(a, b))

result:

****

**Задание 6.4. Создать свой модуль и использовать в программе.**

Лабораторная работа 7

Работа с классами.Содать свой класс, без и с использованием

конструктора. Использовать в программе наследование, полиморфизм.

from abc import ABC, abstractmethod  
  
  
class Vehicle(ABC):  
 @abstractmethod  
 def start(self):  
 pass  
  
 @abstractmethod  
 def stop(self):  
 pass  
  
class Car(Vehicle):  
 def \_\_init\_\_(self, brand, model):  
 self.brand = brand  
 self.model = model  
  
 def start(self):  
 print(f"{self.brand} {self.model} started!")  
  
 def stop(self):  
 print(f"{self.brand} {self.model} stopped!")  
  
 def \_\_str\_\_(self):  
 return f"Car: {self.brand} {self.model}"  
  
 def \_\_add\_\_(self, other):  
 return f"{self.brand} + {other.brand} combination"  
  
class Truck(Car):  
 def \_\_init\_\_(self, brand, model, load\_capacity):  
 # self.brand = brand  
 # self.model = model  
 super().\_\_init\_\_(brand, model)  
 self.load\_capacity = load\_capacity  
  
 def \_\_str\_\_(self):  
 return f"Truck: {self.brand} {self.model}, Load: {self.load\_capacity}kg"  
  
def main():  
 car1 = Car("Toyota", "Camry")  
 car2 = Car("Honda", "Civic")  
 truck1 = Truck("Volvo", "FH", 20000)  
  
 car1.start()  
 car2.start()  
 truck1.start()  
  
 car1.stop()  
 truck1.stop()  
  
 print(car1)   
 print(truck1)   
 print(car1 + car2)   
  
if \_\_name\_\_ == "\_\_main\_\_":  
 main()

result:

