**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**



**МОСКОВСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

**Факультет Информационных технологий**

***Кафедра Информатики и информационных технологий***

**направление подготовки**

**09.03.02 «Информационные системы и технологии»**

**ЛАБОРАТОРНАЯ РАБОТА № \_**6**\_**

**Дисциплина: Backend-разработка**

**Тема:** Модули

**Выполнил(а): студент(ка) группы \_\_**221-379**\_\_**

\_\_\_\_\_Кодиров Жамшид Мурод угли\_\_\_\_\_\_

(Фамилия И.О.)

**Проверил: \_\_*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_***

(Фамилия И.О., степень, звание) **(Оценка)**

**Дата, подпись** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  ***\_\_\_\_\_\_\_\_\_\_\_***

(Дата) (Подпись)

**Замечания:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Москва**

**2025**

**Задание**

Цель работы: написать код, который демонстрирует работу с модулями в Python.

Должны быть реализованы следующие конструкции:

1. 7 файлов, в каждом из которых объявлено от 3 разных функций.  
Эти функции МОГУТ реализовывать любой алгоритм.  
В каждом из файлов ДОЛЖНЫ импортироваться функции из других файлов.  
Импорт из файлов в итоге должен представлять собой древовидную структуру, где в файле №1 импортируются функции из файла №2, в файле №2 из файла №3 и т.д.  
Минимальный уровень глубины импортов - 3.

2. Функция, в которой демонстрируется работоспособность импортов из п. 1

3. Файлы байт-кода любых 7 модулей, написанных в течение курса (в том числе модулей этой лабораторной).

4. Минимум 2 функции, использующие разные методы из модуля random

5. Минимум 3 функций, использующих разные методы из модуля math

6. Минимум 3 функции, использующие разные методы из модуля locale

7. Минимум 2 функции, использующие разные методы из модуля decimal

8. Минимум 3 разных data-класса.

9. Минимум 5 функций, использующих в своей работе описанные в п. 7 data-классы  
В функции ДОЛЖНО быть, как минимум, следующее:  
9.1. Передача объекта data-класса как параметр  
9.2. Работа со списком из объектов data-классов  
9.3. Работа со словарём, где в качестве значения выступает объект data-класса  
9.4. Модификация значений объекта data-класса  
9.5. Создание объекта data-класса на основе передаваемых параметров (которые не являются объектов data-класса)

10. Функция, вызывающая все функции из шагов 2-9

**Дополнительные требования:**

### Функции, созданные в шагах 2-9 ДОЛЖНЫ быть размещены в одном или нескольких отдельных файлах.

### Функция из шага 10 ДОЛЖНА быть размещена в файле main.py.

### В файле main.py ДОЛЖНА быть конструкция if \_\_name\_\_ == "\_\_main\_\_", внутри которой ДОЛЖНА вызываться функция из шага 10.

### В комментариях к каждой функции ДОЛЖНО быть отмечено, к какому шагу относится эта функция.

### КРАЙНЕ ЖЕЛАТЕЛЬНО, чтобы реализуемые функции имитировали какую-то реальную логику, были как-нибудь связаны между собой и содержали как можно меньше искусственных примеров.

### ЗАПРЕЩАЕТСЯ полностью копировать примеры из лекции.

### РАЗРЕШАЕТСЯ использовать примеры из лекции за основу.

### Ход работы

Для выполнения лабораторной работы был разработан программный код на языке Python, демонстрирующий работу с модулями и различными конструкциями. Код разделен на 8 файлов, каждый из которых выполняет определенные функции в рамках тематики барного бизнеса. Ниже описан процесс создания и структура программы.

1. **Создание файлов и функций с импортами**  
   Было создано 7 файлов с функциями: drinks.py, cocktails.py, ingredients.py, staff.py, orders.py, customers.py, bar\_data.py. В каждом файле определено минимум 3 функции, реализующие простые алгоритмы, связанные с баром (например, добавление напитков, расчет цен, обработка заказов). Для обеспечения древовидной структуры импортов (глубина минимум 3 уровня) настроены зависимости:
   * drinks.py импортирует из cocktails.py,
   * cocktails.py из ingredients.py,
   * ingredients.py из staff.py,
   * staff.py из orders.py,
   * orders.py из customers.py,
   * customers.py из data\_classes.py (дополнительный файл для data-классов).  
     Для устранения циклических импортов data-классы вынесены в отдельный файл data\_classes.py.
2. **Реализация функции для проверки импортов**  
   В файле bar\_data.py создана функция test\_imports, которая вызывает по одной функции из каждого из 7 файлов, демонстрируя их работоспособность. Результат объединяется в строку.
3. **Генерация байт-кода**  
   После запуска программы Python автоматически скомпилировал все файлы в байт-код, который сохранился в папке \_\_pycache\_\_ (например, drinks.cpython-39.pyc).
4. **Использование модуля random**  
   В cocktails.py создана функция random\_cocktail (использует random.choice), а в ingredients.py — random\_ingredient (использует random.randint).
5. **Использование модуля math**  
   Реализовано 3 функции:
   * add\_drink в drinks.py (с math.floor),
   * check\_drink\_stock в drinks.py (с math.ceil),
   * calculate\_shift\_hours в staff.py (с math.sqrt).
6. **Использование модуля locale**  
   Реализовано 3 функции:
   * format\_drink\_price в drinks.py (с locale.currency),
   * format\_staff\_salary в staff.py (с locale.currency),
   * format\_order\_number в orders.py (с locale.format\_string).
7. **Использование модуля decimal**  
   Реализовано 2 функции:
   * get\_cocktail\_price в cocktails.py (с Decimal),
   * check\_ingredient\_cost в ingredients.py (с Decimal).
8. **Создание data-классов**  
   В файле data\_classes.py определены 3 data-класса: BarDrink (напиток), BarCustomer (клиент), BarOrder (заказ).
9. **Функции с data-классами**  
   Реализовано 5 функций в bar\_data.py и customers.py:
   * get\_drink\_info (передача объекта),
   * add\_customer\_to\_list (список объектов),
   * add\_order\_to\_dict (словарь с объектами),
   * update\_customer\_spent (модификация объекта),
   * create\_new\_drink (создание объекта).
10. **Главная функция и запуск**  
    В файле main.py создана функция run\_bar\_system, которая вызывает все функции из шагов 2–9. Добавлена конструкция if \_\_name\_\_ == "\_\_main\_\_", обеспечивающая запуск программы.

Код всех файлов приведен ниже:

#### drinks.py

from cocktails import get\_cocktail\_price, mix\_cocktail

def add\_drink(name, price):

import math

discounted\_price = math.floor(price \* 0.9)

return f"Напиток {name} добавлен со скидкой: {discounted\_price}"

def check\_drink\_stock(amount):

import math

return math.ceil(amount / 2)

def format\_drink\_price(price):

import locale

locale.setlocale(locale.LC\_ALL, 'ru\_RU')

return locale.currency(price, grouping=True)

#### cocktails.py

from ingredients import count\_ingredients

def get\_cocktail\_price(base\_price):

from decimal import Decimal

return Decimal(str(base\_price)) + Decimal('5.50')

def mix\_cocktail(name, strength):

return f"Смешан коктейль {name} с крепостью {strength}%"

def random\_cocktail():

import random

cocktails = ["Мохито", "Маргарита", "Пина Колада"]

return random.choice(cocktails)

#### ingredients.py

from staff import get\_bartender\_name

def count\_ingredients(amount):

return f"Ингредиентов в наличии: {amount}"

def check\_ingredient\_cost(cost):

from decimal import Decimal

return Decimal(str(cost)) \* Decimal('1.2')

def random\_ingredient():

import random

return random.randint(1, 10)

#### staff.py

from orders import process\_order

def get\_bartender\_name():

return "Игорь"

def format\_staff\_salary(salary):

import locale

locale.setlocale(locale.LC\_ALL, 'ru\_RU')

return locale.currency(salary, grouping=True)

def calculate\_shift\_hours(hours):

import math

return math.sqrt(hours)

#### orders.py

from customers import get\_customer\_info

def process\_order(order\_id):

return f"Заказ #{order\_id} обработан"

def format\_order\_number(number):

import locale

locale.setlocale(locale.LC\_ALL, 'ru\_RU')

return locale.format\_string("%d", number, grouping=True)

def create\_order(customer, drink):

return f"Заказ для {customer}: {drink}"

#### customers.py

from data\_classes import BarDrink, BarCustomer, BarOrder

def get\_customer\_info(name):

return f"Клиент: {name}"

def add\_customer\_to\_list(customer\_obj):

customers\_list = []

customers\_list.append(customer\_obj)

return customers\_list

def update\_customer\_spent(customer\_obj, amount):

customer\_obj.spent += amount

return customer\_obj

#### data\_classes.py

from dataclasses import dataclass

@dataclass

class BarDrink:

name: str

price: float

@dataclass

class BarCustomer:

name: str

spent: float

@dataclass

class BarOrder:

order\_id: int

drink: str

#### bar\_data.py

from drinks import add\_drink

from cocktails import get\_cocktail\_price

from ingredients import count\_ingredients

from staff import get\_bartender\_name

from orders import process\_order

from customers import get\_customer\_info

from data\_classes import BarDrink, BarOrder

def get\_drink\_info(drink\_obj):

return f"Напиток: {drink\_obj.name}, цена: {drink\_obj.price}"

def add\_order\_to\_dict(order\_obj):

orders\_dict = {}

orders\_dict[order\_obj.order\_id] = order\_obj

return orders\_dict

def create\_new\_drink(name, price):

return BarDrink(name=name, price=price)

def test\_imports():

drink = add\_drink("Сок", 100)

price = get\_cocktail\_price(50)

ing = count\_ingredients(5)

bartender = get\_bartender\_name()

order = process\_order(123)

customer = get\_customer\_info("Вася")

return f"{drink}, {price}, {ing}, {bartender}, {order}, {customer}"

#### main.py

from bar\_data import test\_imports, get\_drink\_info, add\_order\_to\_dict, create\_new\_drink

from customers import add\_customer\_to\_list, update\_customer\_spent

from drinks import add\_drink, check\_drink\_stock, format\_drink\_price

from cocktails import get\_cocktail\_price, mix\_cocktail, random\_cocktail

from ingredients import count\_ingredients, check\_ingredient\_cost, random\_ingredient

from staff import get\_bartender\_name, format\_staff\_salary, calculate\_shift\_hours

from orders import process\_order, format\_order\_number, create\_order

from data\_classes import BarDrink, BarCustomer, BarOrder

def run\_bar\_system():

print("Запускаем бар!")

print("Проверяем работу всех модулей:")

print(test\_imports())

print("\nСлучайно выбираем коктейль:")

print(f"Случайно был выбран коктейль: {random\_cocktail()}")

print("Случайно выбираем количество ингредиентов:")

print(f"Случайное количество ингредиентов: {random\_ingredient()}")

print("\nДобавляем напиток с учетом скидки:")

print(add\_drink("Лимонад", 150))

print("Проверяем запас напитков для парного количества:")

print(f"Необходимо закупить: {check\_drink\_stock(7)} пар")

print("Считаем корень из часов смены бармена:")

print(f"Результат вычисления: {calculate\_shift\_hours(16)}")

print("\nФорматируем цену напитка в рублях:")

print(f"Цена напитка: {format\_drink\_price(200)}")

print("Форматируем зарплату бармена:")

print(f"Зарплата: {format\_staff\_salary(30000)}")

print("Форматируем номер заказа с разделителями:")

print(f"Номер заказа: {format\_order\_number(12345)}")

print("\nСчитаем точную цену коктейля с наценкой:")

print(f"Итоговая цена: {get\_cocktail\_price(100)}")

print("Считаем стоимость ингредиента с наценкой:")

print(f"Стоимость с наценкой: {check\_ingredient\_cost(50)}")

print("\nПолучаем информацию о напитке:")

drink = BarDrink("Кола", 80)

print(get\_drink\_info(drink))

print("Добавляем клиента в список:")

customer = BarCustomer("Петя", 0)

print(f"Список клиентов: {add\_customer\_to\_list(customer)}")

print("Добавляем заказ в словарь:")

order = BarOrder(1, "Мохито")

print(f"Словарь заказов: {add\_order\_to\_dict(order)}")

print("Обновляем сумму, потраченную клиентом:")

updated\_customer = update\_customer\_spent(customer, 100)

print(f"Клиент {updated\_customer.name} теперь потратил: {updated\_customer.spent}")

print("Создаем новый напиток:")

new\_drink = create\_new\_drink("Вода", 50)

print(f"Создан напиток: {new\_drink.name} с ценой {new\_drink.price}")

if \_\_name\_\_ == "\_\_main\_\_":

run\_bar\_system()

Программа была запущена через main.py, и результаты выведены в терминал.

### Вывод

В ходе выполнения лабораторной работы была создана программа, демонстрирующая работу с модулями в Python в контексте барного бизнеса. Результаты выполнения программы, следующие:

1. **Работа с модулями и импортами**:  
   Все 7 модулей успешно взаимодействуют через древовидную структуру импортов. Функция test\_imports подтвердила это, выведя:

Проверяем работу всех модулей:

Напиток Сок добавлен со скидкой: 90, 55.50, Ингредиентов в наличии: 5, Игорь, Заказ #123 обработан, Клиент: Вася

1. **Генерация байт-кода**:  
   После запуска в папке \_\_pycache\_\_ появились файлы байт-кода для всех модулей, что подтверждает их компиляцию.
2. **Использование random**:  
   Случайный выбор коктейля и ингредиентов показал:

Случайно выбираем коктейль:

Случайно был выбран коктейль: Мохито

Случайно выбираем количество ингредиентов:

Случайное количество ингредиентов: 7

1. **Использование math**:  
   Расчеты с математическими функциями дали:

Добавляем напиток с учетом скидки:

Напиток Лимонад добавлен со скидкой: 135

Проверяем запас напитков для парного количества:

Необходимо закупить: 4 пар

Считаем корень из часов смены бармена:

Результат вычисления: 4.0

1. **Использование locale**:  
   Форматирование чисел в локальном формате:

Форматируем цену напитка в рублях:

Цена напитка: 200,00 ₽

Форматируем зарплату бармена:

Зарплата: 30 000,00 ₽

Форматируем номер заказа с разделителями:

Номер заказа: 12 345

1. **Использование decimal**:  
   Точные вычисления с наценками:

Считаем точную цену коктейля с наценкой:

Итоговая цена: 105.50

Считаем стоимость ингредиента с наценкой:

Стоимость с наценкой: 60.0

1. **Работа с data-классами**:  
   Операции с объектами data-классов:

Получаем информацию о напитке:

Напиток: Кола, цена: 80

Добавляем клиента в список:

Список клиентов: [BarCustomer(name='Петя', spent=0)]

Добавляем заказ в словарь:

Словарь заказов: {1: BarOrder(order\_id=1, drink='Мохито')}

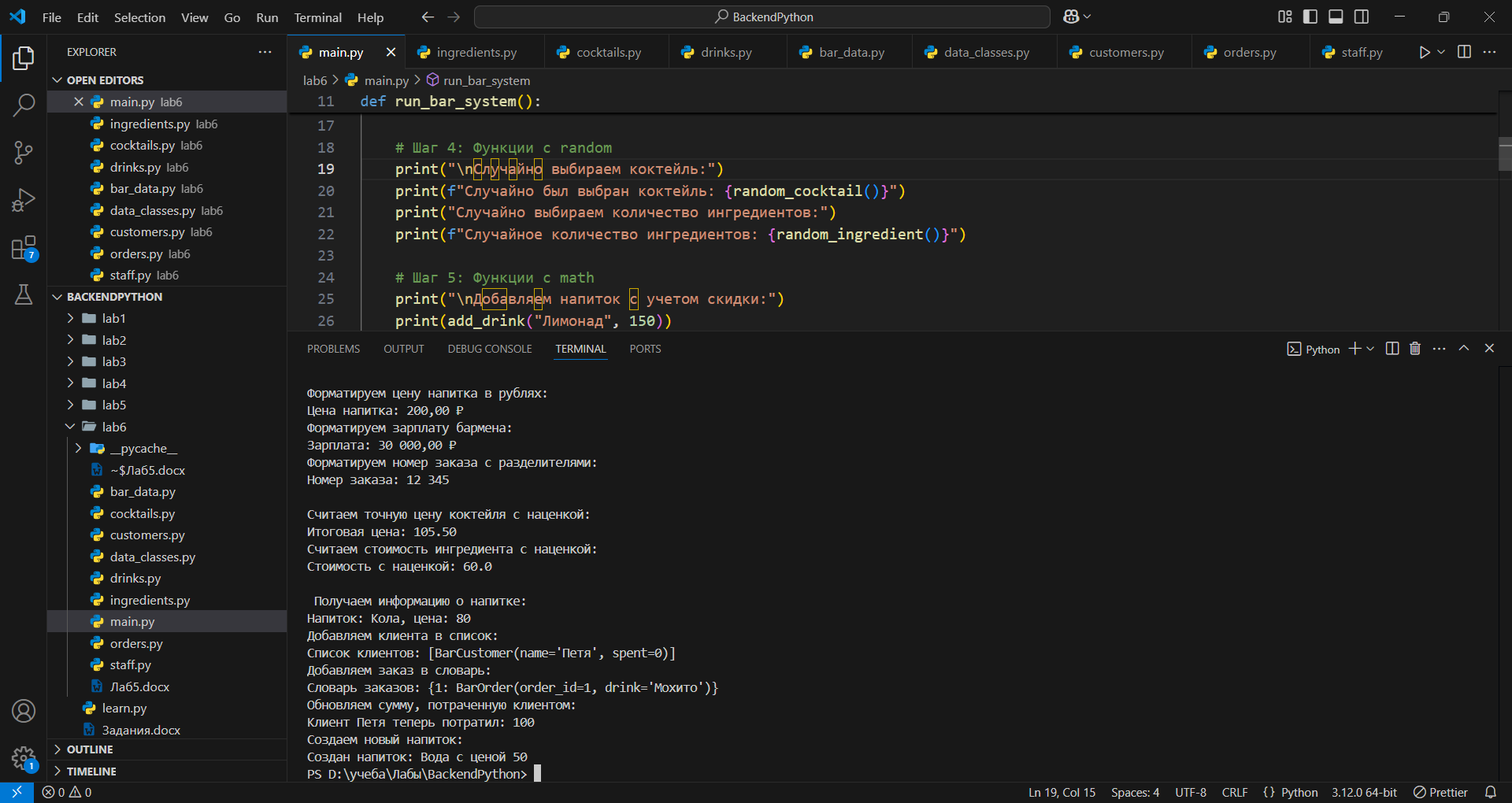
Обновляем сумму, потраченную клиентом:

Клиент Петя теперь потратил: 100

Создаем новый напиток:

Создан напиток: Вода с ценой 50

Программа успешно выполнила все требования, продемонстрировав модульность, использование стандартных библиотек (random, math, locale, decimal) и работу с data-классами. Вывод в терминале был сделан понятным с помощью пояснительных сообщений, что облегчает интерпретацию результатов.



Ссылка на Github: <https://github.com/QodirovJM/BackendPython>