



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
```

2. Генерация байт-кода:

После запуска в папке __pycache__ появились файлы байт-кода для всех модулей, что подтверждает их компиляцию.

3. Использование random:

Случайный выбор коктейля и ингредиентов показал:

```
Случайно выбираем коктейль:
```

```
Случайно был выбран коктейль: Мохито
```

```
Случайно выбираем количество ингредиентов:
```

```
Случайное количество ингредиентов: 7
```

4. 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
```

5. 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
```

6. 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
```

7. Работа с data-классами:

Операции с объектами data-классов:

```
Получаем информацию о напитке:

Напиток: Кола, цена: 80

Добавляем клиента в список:
```

Создан напиток: Вода с ценой 50

The image shows a VS Code editor window with a Python file named `main.py` open. The file contains a script for a bar system simulation. The script defines a `run_bar_system()` function that performs several tasks:

- It sets a seed for random selection.
- It prints a message indicating a drink is being selected.
- It randomly selects a drink from a list of cocktails.
- It prints a message indicating a drink is being selected.
- It randomly selects a quantity of ingredients.
- It prints a message indicating a quantity of ingredients is being selected.
- It sets a seed for random selection.
- It prints a message indicating a drink is being selected.
- It adds a drink to the list with a specific price.

The terminal output shows the execution results of the script:

```
форматируем цену напитка в рублях:
Цена напитка: 200,00 P
форматируем зарплату бармена:
Зарплата: 30 000,00 P
форматируем номер заказа с разделителями:
Номер заказа: 12 345

Считаем точную цену коктейля с наценкой:
Итоговая цена: 105,50
Считаем стоимость ингредиента с наценкой:
Стоимость с наценкой: 60,0

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
PS D:\учеба\lab5\BackendPython>
```

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