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 № _3_

**Дисциплина: Backend-разработка**

**Тема:** Функции и лямбда-выражения

**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: научиться создавать и использовать различные типы функций, лямбда-выражений и замыканий в Python.

Требуется написать следующие конструкции:

1. Функция без параметров.
2. Функция с параметрами.
3. Функция с несколькими параметрами со значениями по умолчанию.
4. Функция с несколькими параметрами, у которых задан тип.
5. Функция с неопределённым количеством параметров (*args).
6. Функция с неопределённым количеством параметров (**kwargs).
7. Функция, вызывающая внутри себя другую функцию.
8. Функция, принимающая функцию как параметр (минимум 3 примера).
9. Функция с объявленной внутри локальной функцией (минимум 2 примера).
10. Лямбда-выражение без параметров.
11. Лямбда-выражение с параметрами.
12. Функция, принимающая лямбда-выражение как параметр и вызывающая его внутри себя.
13. Функция с замыканиями (минимум 3 примера).

**Дополнительные требования:**

- Все функции и лямбда-выражения должны быть объявлены в файле bar_functions.py.
- Вызовы функций и лямбда-выражений — в файле main.py с использованием конструкции if __name__ == "__main__".
- Как минимум 10 функций должны возвращать значения.
- Тема: бар и алкогольные напитки.

**Решение**

**1. Функция без параметров**

**Описание:** Функция welcome_to_bar возвращает приветственное сообщение для посетителей бара.
**Код в bar_functions.py:**

```
def welcome_to_bar():
    return "Добро пожаловать в бар!"
```

**Вызов в main.py:**

```
print(bar_functions.welcome_to_bar())
```

**Результат:** "Добро пожаловать в бар!"

**2. Функция с параметрами**

**Описание:** Функция get_drink_name принимает название напитка и возвращает строку с этим названием.
**Код в bar_functions.py:**

```
def get_drink_name(name):
    return f"Напиток: {name}"
```

**Вызов в main.py:**

```
print(bar_functions.get_drink_name("Виски"))
```

**Результат:** "Напиток: Виски"

**3. Функция с несколькими параметрами со значениями по умолчанию**

**Описание:** Функция describe_drink описывает напиток с типом и процентом алкоголя, используя значения по умолчанию.
**Код в bar_functions.py:**

```
def describe_drink(type="Коктейль",
alcohol_percent=10):
    return f"Тип: {type}, Крепость: {alcohol_percent}%"
```

**Вызов в main.py:**

```
print(bar_functions.describe_drink())
print(bar_functions.describe_drink("Водка", 40))
```

**Результат:**

- "Тип: Коктейль, Крепость: 10%"
- "Тип: Водка, Крепость: 40%"

**4. Функция с несколькими параметрами, у которых задан тип**

**Описание:** Функция set_drink_price задаёт цену напитка с указанием типов параметров и возвращаемого значения.
**Код в bar_functions.py:**

```
def set_drink_price(name: str, price: float) -> str:
    return f"Напиток '{name}' стоит {price} рублей"
```

**Вызов в main.py:**

```
print(bar_functions.set_drink_price("Пиво", 150.0))
```

**Результат:** "Напиток 'Пиво' стоит 150.0 рублей"

**5. Функция с неопределённым количеством параметров (*args)**

**Описание:** Функция list_drinks принимает любое количество названий напитков и возвращает их список.
**Код в bar_functions.py:**

```
def list_drinks(*drink_names):
    return "Доступные напитки: " + ",
".join(drink_names)
```

**Вызов в main.py:**

```
print(bar_functions.list_drinks("Ром", "Текила",
"Джин"))
```

**Результат:** "Доступные напитки: Ром, Текила, Джин"

**6. Функция с неопределённым количеством параметров (**kwargs)**

**Описание:** Функция drink_details принимает произвольные именованные параметры и возвращает их в виде строки.
**Код в bar_functions.py:**

```
def drink_details(**info):
    details = ""
    for key, value in info.items():
        details += f"{key}: {value}, "
    return details[:-2]
```

**Вызов в main.py:**

```
print(bar_functions.drink_details(name="Маргарита",
type="Коктейль", price=250))
```

**Результат:** "name: Маргарита, type: Коктейль, price: 250"

**7. Функция, вызывающая внутри себя другую функцию**

**Описание:** Функция announce_drink использует welcome_to_bar для создания сообщения о напитке.
**Код в bar_functions.py:**

```
def announce_drink(drink):
    greeting = welcome_to_bar()
    return f"{greeting} Ваш напиток: {drink}"
```

**Вызов в main.py:**

```
print(bar_functions.announce_drink("Кровавая Мэри"))
```

**Результат:** "Добро пожаловать в бар! Ваш напиток: Кровавая Мэри"

**8. Функция, принимающая функцию как параметр (3 примера)**

*Пример 1*

**Описание:** Функция apply_discount принимает функцию для получения цены и применяет скидку.
**Код в bar_functions.py:**

```
def apply_discount(get_price_func, discount):
    price = float(get_price_func("Мохито",
300.0).split()[-2])
    new_price = price - (price * discount / 100)
    return f"Цена со скидкой: {new_price} рублей"
```

**Вызов в main.py:**

```
print(bar_functions.apply_discount(bar_functions.set_dr
ink_price, 15))
```

**Результат:** "Цена со скидкой: 255.0 рублей"

*Пример 2*

**Описание:** Функция format_drink_name принимает функцию для форматирования имени напитка.
**Код в bar_functions.py:**

```
def format_drink_name(format_func, name):
    return format_func(name)
```

**Вызов в main.py:**

```
print(bar_functions.format_drink_name(bar_functions.get
_drink_name, "Эль"))
```

**Результат:** "Напиток: Эль"

*Пример 3*

**Описание:** Функция check_drink принимает функцию проверки и возвращает её результат.
**Код в bar_functions.py:**

```
def check_drink(check_func, name):
    return f"Статус напитка: {check_func(name)}"
```

**Вызов в main.py:**

```
def availability_check(name):
    return f"{name} есть в наличии"
print(bar_functions.check_drink(availability_check,
"Коньяк"))
```

**Результат:** "Статус напитка: Коньяк есть в наличии"

---

**9. Функция с локальной функцией (2 примера)**

*Пример 1*

**Описание:** Функция order_counter считает заказы с помощью внутренней функции.
**Код в bar_functions.py:**

```
def order_counter():
    count = 0
    def increase_order():
        nonlocal count
        count += 1
        return count
    return increase_order()
```

**Вызов в main.py:**

```
print(f"Количество заказов:
{bar_functions.order_counter()}")
```

**Результат:** "Количество заказов: 1"

*Пример 2*

**Описание:** Функция drink_info добавляет статус к напитку через локальную функцию.
**Код в bar_functions.py:**

```
def drink_info(drink):
    def add_status():
        return f"Напиток - {drink} готов"
    return add_status()
```

**Вызов в main.py:**

```
print(bar_functions.drink_info("Лонг Айленд"))
```

**Результат:** "Напиток - Лонг Айленд готов"

**10. Лямбда-выражение без параметров**

**Описание:** Лямбда no_param_lambda возвращает сообщение о статусе бара.
**Код в bar_functions.py:**

```
no_param_lambda = lambda: "Бар открыт!"
```

**Вызов в main.py:**

```
print(bar_functions.no_param_lambda())
```

**Результат:** "Бар открыт!"

**11. Лямбда-выражение с параметрами**

**Описание:** Лямбда add_volume_lambda добавляет объём к названию напитка.
**Код в bar_functions.py:**

```
add_volume_lambda = lambda name, volume: f"{name}
({volume} мл)"
```

**Вызов в main.py:**

```
print(bar_functions.add_volume_lambda("Шот", 50))
```

**Результат:** "Шот (50 мл)"

### 12. Функция, принимающая лямбда-выражение как параметр

**Описание:** Функция process_drink вызывает переданную лямбда-функцию для обработки данных.
**Код в bar_functions.py:**

```
def process_drink(lambda_func, name, value):
    return lambda_func(name, value)
```

**Вызов в main.py:**

```
result =
bar_functions.process_drink(bar_functions.add_volume_la
mbda, "Саке", 200)
print(result)
```

**Результат:** "Саке (200 мл)"

### 13. Функция с замыканиями (3 примера)

*Пример 1*

**Описание:** Функция create_tip_calculator создаёт замыкание для расчёта чаевых.
**Код в bar_functions.py:**

```
def create_tip_calculator(tip_percent):
    def calculate_tip(amount):
        return amount * tip_percent / 100
    return calculate_tip
```

**Вызов в main.py:**

```
tip_10 = bar_functions.create_tip_calculator(10)
print(f"Чаевые: {tip_10(500)} рублей")
```

**Результат:** "Чаевые: 50.0 рублей"

*Пример 2*

**Описание:** Функция drink_category добавляет категорию к напитку через замыкание.
**Код в bar_functions.py:**

```
def drink_category(category):
    def add_category(name):
        return f"{name} из категории {category}"
    return add_category
```

**Вызов в main.py:**

```
cocktails = bar_functions.drink_category("Коктейли")
print(cocktails("Космополитен"))
```

**Результат:** "Космополитен из категории Коктейли"

*Пример 3*

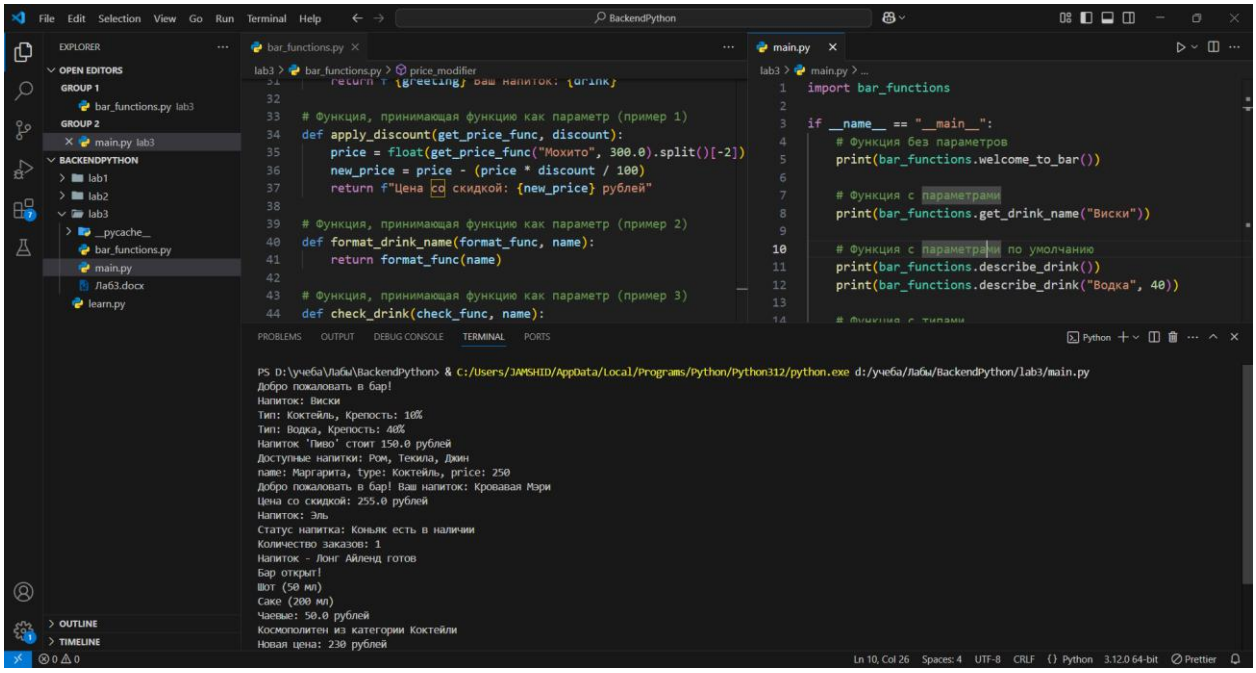
**Описание:** Функция price_modifier изменяет базовую цену через замыкание.
**Код в bar_functions.py:**

```
def price_modifier(base_price):
    def modify(amount):
        return base_price + amount
    return modify
```

**Вызов в main.py:**

```
adjust_price = bar_functions.price_modifier(200)
print(f"Новая цена: {adjust_price(30)} рублей")
```

**Результат:** "Новая цена: 230 рублей"



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