**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\_drink\_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\_lambda, "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"

