**Часть 1:**

**Строки, Списки, Словари**

Для каждого упражнения подготовить отдельный .py файл с комментариями.

**Упражнение 1: Операции со строкой**

# Дана строка: "Python Programming"

# Напишите код, который:

# 1. Выводит длину строки

# 2. Выводит символ по индексу 7

# 3. Выводит последние 3 символа

# 4. Проверяет, содержится ли в строке подстрока "gram"

**Комментарий:**

**string = "Python Programming":** Присваивает строку переменной

# 1. Выводим длину строки

**print(f"Длина строки: {len(string)}")** :

f"Длина строки: {len(string)}": Использует f-строку для вывода отформатированного текста.

# 2. Выводим символ по индексу 7

**print(f"Символ по индексу 7: {string[7]}"):**

string[7]: Получает символ из строки по индексу 7.

f"Символ по индексу 7: {string[7]}": Использует f-строку для вывода результата.

# 3. Выводим последние 3 символа

**print(f"Последние 3 символа: {string[-3:]}")**

string[-3:]: Использует срезы строк для получения последних 3 символов. -3 означает "начиная с третьего символа с конца", а : означает "до конца строки".

f"Последние 3 символа: {string[-3:]}": Использует f-строку для вывода результата.

# 4. Проверяем, содержится ли в строке подстрока "gram"

**if "gram" in string:**

**print("Подстрока 'gram' содержится в строке.")**

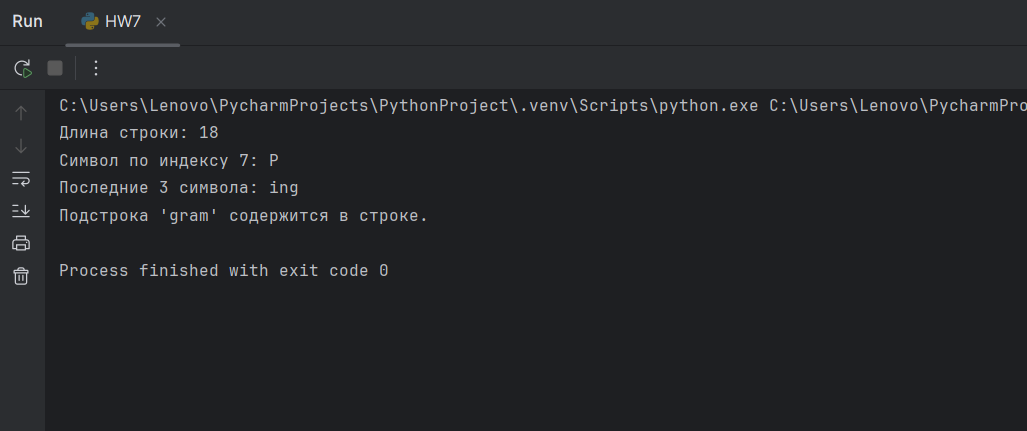
**else:**

**print("Подстрока 'gram' не содержится в строке.")**

"gram" in string: Проверяет, содержится ли подстрока "gram" в строке string. Оператор in возвращает True, если подстрока найдена, и False если не найдена.

print("Подстрока 'gram' содержится в строке."): Выводит это сообщение, если подстрока найдена.

print("Подстрока 'gram' не содержится в строке."): Выводит это сообщение, если подстрока не найдена.



**Упражнение 2:**

**Методы строк и форматинг**

Дано: email = " USER@DOMAIN.COM "

1. Очистить и отформатировать до вида: "user@domain.com"

2. Разделить на имя пользователя и домен

3. Используя f-строку, создать: "Username: user, Domain: domain.com"

**Комментарий:**

email = " USER@DOMAIN.COM ": Присваивает строку email переменной email.

# 1. Очистить и отформатировать до вида: "user@domain.com"

**email = email.strip().lower()**

**print(f"Отформатированный email: {email}")**

email.strip(): Метод strip() удаляет пробелы в начале и конце строки.

email.lower(): Метод lower() преобразует все символы строки в нижний регистр.

# 2. Разделить на имя пользователя и домен

**username, domain = email.split('@')**

email.split('@'): Метод split('@') разделяет строку email на две части по символу @. Возвращает список из двух строк.

**print(f"Имя пользователя: {username}, Домен: {domain}")**

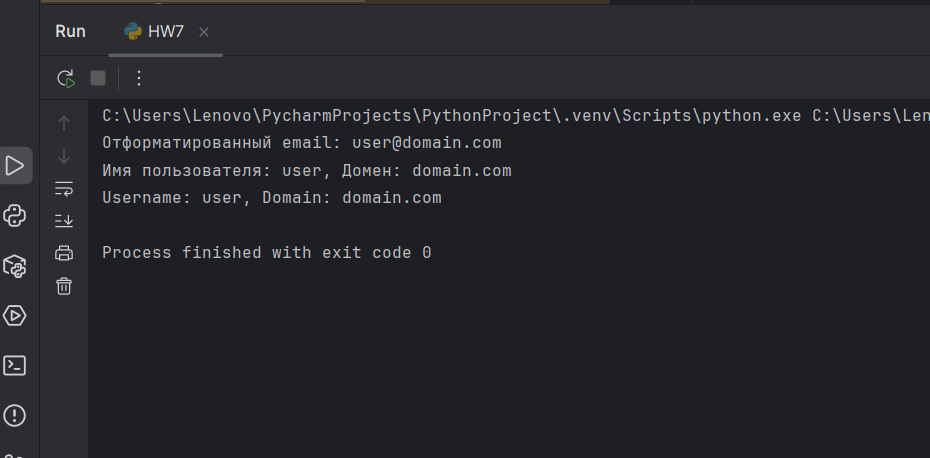
print(f"Имя пользователя: {username}, Домен: {domain}"): Выводит имя пользователя и домен.

# 3. Используя f-строку, создать: "Username: user, Domain: domain.com"

**formatted\_string = f"Username: {username}, Domain: {domain}"**

**print(formatted\_string)**

f"Username: {username}, Domain: {domain}": Создает строку с использованием f-строки, подставляя значения переменных username и domain.



**Упражнение 3: Методы списка**

# Исходный список: fruits = ["apple", "banana"]

# Выполните следующие операции по порядку:

# 1. Добавьте "orange" в конец списка

# 2. Вставьте "grape" по индексу 1

# 3. Удалите "banana"

# 4. Отсортируйте список

# 5. Переверните список

# Выводите список на печать после каждого шага

**Комментарий:**

**fruits = ["apple", "banana"]**

# 1. Добавьте "orange" в конец списка

**fruits.append("orange"):** Метод append() добавляет элемент "orange" в конец списка.

**print(f"После добавления 'orange': {fruits}"):** Выводит список после изменения.

# 2. Вставьте "grape" по индексу 1

**fruits.insert(1, "grape"):** Метод insert() вставляет элемент "grape" по индексу 1. Все элементы, которые находятся после индекса 1, сдвигаются вправо.

**print(f"После вставки 'grape': {fruits}")**

# 3. Удалите "banana"

**fruits.remove("banana"):** Метод remove() удаляет первое "banana" из списка.

**print(f"После удаления 'banana': {fruits}")**

# 4. Отсортируйте список

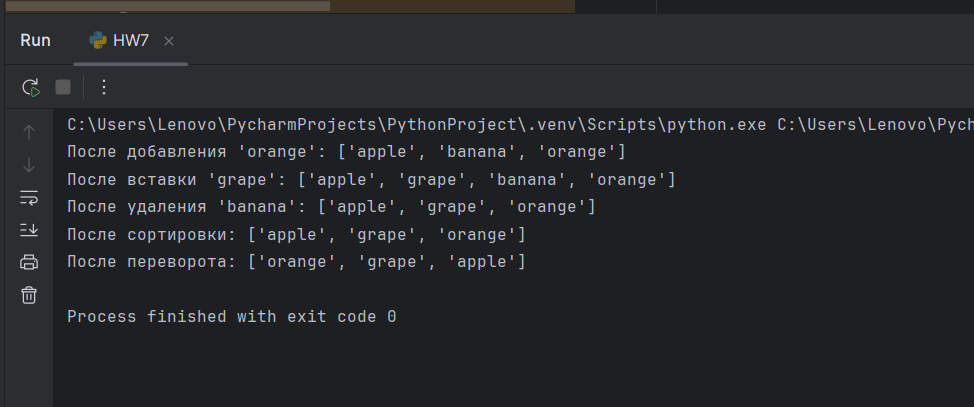
**fruits.sort():** Метод sort() сортирует элементы списка в алфавитном порядке.

**print(f"После сортировки: {fruits}")**

# 5. Переверните список

**fruits.reverse():** Метод reverse() переворачивает порядок элементов в списке (первый становится последним, второй – предпоследним).

**print(f"После переворота: {fruits}")**



**Упражнение 4: List comprehension и словари**

# Дано: words = ["hello", "world", "python", "code"]

# 1. Создать список длин слов, используя списковое включение

# 2. Создать список слов длиннее 4 символов

# 3. Создать словарь: {слово: длина} для всех слов

**Комментарий:**

**words = ["hello", "world", "python", "code"]**

# 1. Создать список длин слов, используя списковое включение

**word\_lengths = [len(word) for word in words]:** Перебирает каждое слово в списке words.

**print(f"Список длин слов: {word\_lengths}"):** Вычисляет длину слова

# 2. Создать список слов длиннее 4 символов

**long\_words = [word for word in words if len(word) > 4]:** Перебирает каждое слово в списке words, проверяет, что длина слова больше 4.

**print(f"Список слов длиннее 4 символов: {long\_words}")**

# 3. Создать словарь: {слово: длина} для всех слов

**word\_length\_dict = {word: len(word) for word in words}:** Создает пару ключ-значение, где ключом является слово, а значением - его длина.

**print(f"Словарь слово:длина: {word\_length\_dict}")**



**Упражнения 5 (Опционально):**

● Дан список чисел nums и целевое число target.

Найти индексы двух чисел, сумма которых равна target.

# Пример:

nums = [2, 7, 11, 15] target = 9

# Ответ: [0, 1] (потому что nums[0] + nums[1] = 2 + 7 = 9)

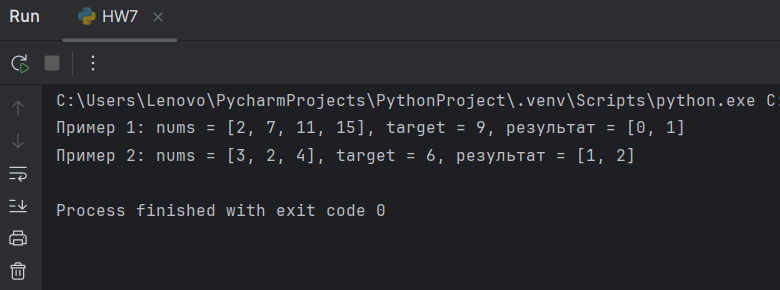
nums = [3, 2, 4]

target = 6

# Ответ: [1, 2] (потому что nums[1] + nums[2] = 2 + 4 = 6)

**Комментарий:**

**def find\_two\_sum(nums, target):  
  
 num\_map = {}** : Создаем словарь для хранения чисел и их индексов  
  
 **for index, num in enumerate(nums):  
 complement = target - num :** Вычисляет число, которое нужно прибавить к числу num, чтобы получить целевое число target.   
 **if complement in num\_map**: :Если разница уже есть в словаре, то нашли пару  
 **return [num\_map[complement], index]** :Возвращаем индексы пары  
 **num\_map[num] = index** :Добавляем текущее число и его индекс в словарь  
  
 **return None** :Если пара не найдена, возвращаем None  
  
  
**# Пример 1  
nums1 = [2, 7, 11, 15]  
target1 = 9  
result1 = find\_two\_sum(nums1, target1)  
print(f"Пример 1: nums = {nums1}, target = {target1}, результат = {result1}")  
  
# Пример 2  
nums2 = [3, 2, 4]  
target2 = 6  
result2 = find\_two\_sum(nums2, target2)  
print(f"Пример 2: nums = {nums2}, target = {target2}, результат = {result2}")**



**Часть 2: Функции и ООП**

Для каждого упражнения подготовить отдельный .py файл с комментариями.

**Упражнение 1: Функции без параметров**

# Создайте функцию без параметров show\_current\_time() — печатает текущие дату и время (используйте модуль datetime).

**Комментарий:**

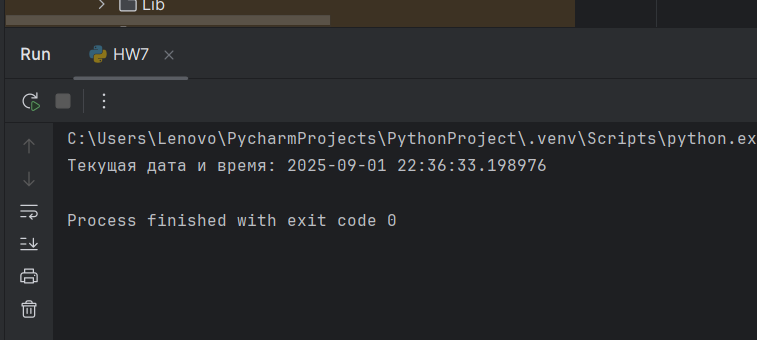
**import datetime**

**def show\_current\_time():** Определяет функцию с именем show\_current\_time

**now = datetime.datetime.now() :** Ссылается на класс datetime внутри модуля datetime.

**print("Текущая дата и время:", now) :** Печатает сообщение "Текущая дата и время:" вместе с объектом datetime, который был сохранен в переменной now.

**show\_current\_time() :** Вызывает функцию получение и печать текущего времени.



**Упражнение 2: Функции с параметрами**

# Дан повторяющийся код расчёта цены с НДС:

# prices = [1000, 3499, 250]

# nds = 0.20

# print(prices[0] + prices[0] \* nds)

# print(prices[1] + prices[1] \* nds)

# print(prices[2] + prices[2] \* nds)

#

# Задание:

# 1) Вынесите расчёт в функцию add\_vat().

# 2) Примените её ко всем элементам списка в цикле и распечатайте

итоговые цены, используйте цикл.

**Комментарий:**

**def add\_vat(price, nds=0.20)::**

Определяет функцию add\_vat с двумя параметрами:

- price: Цена без НДС.

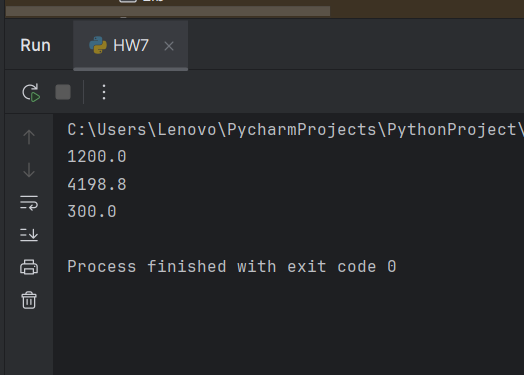
- nds: Ставка НДС .

**prices = [1000, 3499, 250]** : Создает список цен.

**for price in prices:**

**final\_price = add\_vat(price)** : Вызывает функцию add\_vat, передавая ей текущую цену price из списка. Возвращаемое значение (цена с НДС) присваивается переменной final\_price.

**print(final\_price)** : Печатает вычисленную цену с НДС.



**Упражнение 3 (Опционально)**

● Создайте функцию calculate\_average\_score(), которая будет вычислять

средний балл.

● Функция должна принимать список оценок scores как обязательный аргумент.

● Добавьте опциональный булевый параметр ignore\_lowest со значением по

умолчанию False.

● Если ignore\_lowest равен True, функция должна отбросить наименьшую оценку

перед вычислением среднего. Если в списке всего одна оценка, отбрасывать её не

нужно.

Python

● Используя цикл, пройдитесь по списку student\_data. 2 раза, первый раз

учитывая все оценки, а второй раз отбросив худшие оценки

student\_data = [

{'name': 'Алексей', 'scores': [85, 92, 78, 95]},

{'name': 'Марина', 'scores': [65, 70, 58, 82]}

{'name': 'Светлана', 'scores': [98, 95, 100]}

]

calculate\_average\_score()

**Комментарий:**

**def calculate\_average\_score(scores, ignore\_lowest=False):**

**if not scores:**

**return None** : Обработка случая пустого списка оценок

**if ignore\_lowest and len(scores) > 1:**

**scores = sorted(scores)[1:]** : Сортируем и отбрасываем наименьшую оценку

**return sum(scores) / len(scores)** : - Вычисляет средний балл, суммируя все оценки в списке scores (sum(scores)) и деля эту сумму на количество оценок (len(scores)).

- Возвращает вычисленный средний балл.

**student\_data = [**

**{'name': 'Алексей', 'scores': [85, 92, 78, 95]},**

**{'name': 'Марина', 'scores': [65, 70, 58, 82]},**

**{'name': 'Светлана', 'scores': [98, 95, 100]}**

**]**

**print("Средние баллы со всеми оценками:")**

**for student in student\_data:**

**average\_score = calculate\_average\_score(student['scores'])**

**print(f"{student['name']}: {average\_score:.2f}")** : Форматирование для 2 знаков после запятой

**print("\nСредние баллы с отброшенной худшей оценкой:")**

**for student in student\_data:**

**average\_score = calculate\_average\_score(student['scores'], ignore\_lowest=True)**

**print(f"{student['name']}: {average\_score:.2f}")**

