ПЗ №9. Работа с функциями в Python

Тема

Основы создания и использования пользовательских функций в Python. Параметры, возвращаемые значения, области видимости.

Цель работы

- **Теоретическая:** Изучить синтаксис объявления и вызова функций в Python, понять концепции параметров, возвращаемых значений и областей видимости переменных.
- **Практическая:** Приобрести навыки декомпозиции задачи на подзадачи, написания многократно используемого кода с помощью функций, отладки отдельных блоков программы.

Задачи

1. Закрепить синтаксис объявления (`def`) и вызова функций.
2. Научиться передавать в функцию аргументы различных типов (позиционные, именованные, значения по умолчанию).
3. Научиться возвращать из функции одно или несколько значений.
4. Понять разницу между глобальными и локальными переменными.
5. Применить знания для структурирования программы, решающей конкретную задачу.

Оборудование и программное обеспечение

- **Программное обеспечение:** Интерпретатор Python 3.8+, интегрированная среда разработки (IDE) Visual Studio Code (VS Code) с установленным расширением Python.
- **Оборудование:** Персональный компьютер.

Краткие теоретические сведения

Функция — это именованный блок кода, предназначенный для выполнения одной конкретной задачи. Функции позволяют структурировать программу, избегать дублирования кода и упрощать его чтение и отладку.

Объявление функции осуществляется с помощью ключевого слова `def` . После имени функции в круглых скобках указываются параметры (аргументы), которые она принимает.

```
def имя_функции(параметр1, параметр2=значение_по_умолчанию):
    """Строка документации (docstring)."""
    # Тело функции
    return результат  # Оператор return возвращает значение
```

Параметры функции могут быть:

- **Позиционными** (передаются по порядку).
- **Именованными** (передаются по имени параметра).
- **Иметь значения по умолчанию** (становятся необязательными).

Оператор `return` завершает выполнение функции и передает указанное значение в точку вызова. Если `return` отсутствует или указан без значения, функция возвращает `None`. Можно возвращать несколько значений через кортеж.

Область видимости переменных определяет, где переменная доступна.

- **Локальная переменная:** Объявлена внутри функции, доступна только в ней.
- **Глобальная переменная:** Объявлена вне функций, доступна для чтения внутри них. Для изменения глобальной переменной используется ключевое слово `global`.
- **Встроенная (built-in) область:** Содержит встроенные имена Python (например, `print`, `len`).

Порядок выполнения работы

1. Подготовительный этап

1. Запустите VS Code.
2. Создайте новый файл с именем `lab_09_[ВашаФамилия].py`.
3. В начале файла добавьте комментарий с номером работы, вашим именем и группой.

```
# Лабораторная работа №9. Работа с функциями.
# [Ваше Имя Фамилия], [Ваша группа]
```

2. Основной этап

Общий пример для всех вариантов
Создайте и протестируйте функцию, которая принимает список чисел и возвращает словарь с информацией о нем.

```python
def analyze_list(num_list):
    """Анализирует список чисел и возвращает статистику."""
    if not num_list:  # Проверка на пустой список
        return None
    stats = {
        'min': min(num_list),
        'max': max(num_list),
        'sum': sum(num_list),
        'avg': sum(num_list) / len(num_list)
    }
    return stats

# Пример вызова:
my_numbers = [5, 2, 8, 1, 9]
result = analyze_list(my_numbers)
print(f"Анализ списка {my_numbers}:")
print(f"  Минимум: {result['min']}")
print(f"  Максимум: {result['max']}")
print(f"  Сумма: {result['sum']}")
print(f"  Среднее: {result['avg']:.2f}") # :.2f - форматирование до 2 знаков
```

Задания по вариантам (выполнить только свой вариант):

Вариант 1. Работа со строками и числами.

1. Напишите функцию `is_palindrome(text)`, которая принимает строку, очищает ее от пробелов и приводит к нижнему регистру, а затем проверяет, является ли она палиндромом (читается одинаково слева направо и справа налево). Функция должна возвращать `True` или `False`.
2. Напишите функцию `power(base, exp)`, которая возвращает результат возведения `base` в степень `exp`. Учтите, что `exp` может быть отрицательным. Используйте оператор `**`.
3. Напишите функцию `process_data(*args, **kwargs)`, которая:
   - Печатает сумму всех числовых аргументов в `args`.
   - Печатает все ключи словаря `kwargs`.
   - Возвращает кортеж из двух элементов: (`сумма_из_args`, `список_ключей_из_kwargs`).

Вариант 2. Работа со списками и кортежами.

1. Напишите функцию `find_common_elements(list1, list2)`, которая принимает два списка и возвращает новый список, содержащий общие для обоих списков элементы (без дубликатов).
2. Напишите функцию `tuple_stats(my_tuple)`, которая принимает кортеж чисел и возвращает словарь, где ключи — это позиции (индексы) элементов, а значения — сами элементы. Дополнительно функция должна печатать длину кортежа.
3. Напишите функцию `list_shift(lst, n)`, которая циклически сдвигает элементы списка `lst` на `n` позиций вправо (при положительном `n`) или влево (при отрицательном `n`). Функция должна изменять исходный список и ничего не возвращать.

Вариант 3. Работа со словарями и множествами.

1. Напишите функцию `merge_dicts(dict1, dict2)`, которая объединяет два словаря. Если ключ присутствует в обоих, значение из `dict2` должно перезаписать значение из `dict1`. Функция должна возвращать новый словарь, не изменяя исходные.
2. Напишите функцию `set_operations(set_a, set_b)`, которая выполняет и возвращает результаты трех операций над множествами: объединение, пересечение и разность (`set_a - set_b`). Возвращайте результат в виде кортежа.
3. Напишите функцию `count_words(text)`, которая принимает строку `text`, разбивает ее на слова (по пробелам), приводит слова к нижнему регистру и возвращает словарь, где ключ — слово, а значение — сколько раз оно встретилось.

Вариант 4. Комбинированные структуры данных.

1. Напишите функцию `filter_by_value(list_of_dicts, key, min_value)`, которая принимает список словарей, ключ (для поиска в каждом словаре) и минимальное значение. Функция должна возвращать новый список, содержащий только те словари из исходного списка, у которых значение по указанному ключу больше или равно `min_value`.
2. Напишите функцию `matrix_transpose(matrix)`, которая принимает матрицу (список списков) и возвращает ее транспонированную версию (строки становятся столбцами). Предполагается, что все "строки" матрицы имеют одинаковую длину.
3. Создайте функцию-генератор `number_generator(start, end, step=1)`, которая бы возвращала (yield) числа от `start` до `end` (не включая) с шагом `step`.

Вариант 5. Практическая задача.

1. Напишите функцию `bank_deposit(initial, years, rate)`, которая рассчитывает итоговую сумму на вкладе по формуле сложного процента: `initial * (1 + rate/100) ** years`. Функция должна возвращать округленный до двух знаков результат.
2. Напишите функцию `quadratic_roots(a, b, c)`, которая решает квадратное уравнение `ax^2 + bx + c = 0`. Функция должна возвращать кортеж с корнями. Учтите случаи с одним корнем и отсутствием действительных корней (вернуть `None`).
3. Создайте функцию `interactive_menu()`, которая в цикле предлагает пользователю выбрать действие из словаря `options`, где ключ — цифра, значение — строка-описание. При выборе "1" вызывается функция `bank_deposit` с запросом параметров у пользователя, при выборе "2" — `quadratic_roots`. При выборе "0" цикл прерывается.

3. Контрольный этап

1. Для каждой написанной функции из вашего варианта напишите 2-3 различных примера вызова с разными аргументами.
2. Выведите результаты вызовов с помощью `print()`, чтобы продемонстрировать работу функции.
3. Убедитесь, что функции, которые должны изменять глобальные данные или оригинальные структуры (если это требуется в задании), делают это корректно, а функции, которые не должны — не делают.
4. Запустите программу (`python lab_09_[ВашаФамилия].py` или через кнопку Run в VS Code) и убедитесь, что нет ошибок выполнения, а вывод соответствует ожиданиям.

Контрольные вопросы

1. Каково основное назначение функций в программировании?
2. Объясните разницу между параметром и аргументом функции.
3. Что такое `*args` и `**kwargs`? В каких случаях их используют?
4. Чем отличается оператор `return` от оператора `print` внутри функции?
5. Может ли функция возвращать несколько значений? Если да, то как?
6. Что произойдет, если внутри функции попытаться изменить значение глобальной переменной без использования `global`?
7. Что такое `None`? Когда функция возвращает это значение?
8. В чем разница между изменяемыми (mutable) и неизменяемыми (immutable) типами данных при передаче их в функцию в качестве аргументов?
9. Что такое рекурсивная функция? Какое обязательное условие должно быть в ней прописано?

10. Что такое `docstring` и для чего он используется?

Содержание отчета

1. Тема, цель и задачи работы.
2. Код вашей программы (файл `lab_09_[ВашаФамилия].py`) с комментариями к ключевым моментам.
3. Результаты выполнения заданий:
   - Вывод программы на экране (скриншот или текстовый вывод).
   - Объяснение, как каждая из ваших функций работает (кратко).
4. Ответы на контрольные вопросы.
5. **Выводы по работе:** Опишите, что нового вы узнали о функциях, с какими трудностями столкнулись (например, область видимости, работа с изменяемыми аргументами), как функции помогают в структурировании кода. Оцените, достигнута ли цель работы.