Практическая работа № 1

Тема 2. Модульность. Повторное использование

Цель работы: Изучить методы программирования на Python на основе применения модулей. Научиться создавать и применять скрипт, модуль для решения прикладных задач.

План занятия.

1.Изучить теоретическую часть.

2.Изучить примеры № 1-5.

3.Выполнить практические задания. Исходные данные (табл. 1). Выполнить один вариант из табл. 1.

Файл прикрепить на проверку.

Имя файла: **Иванов АА-пр-1.ру**

4.Ответить на контрольные вопросы (устно).

1. Теоретическая часть.

Модульность — это важный аспект проектирования программного обеспечения, который способствует созданию более управляемых, гибких и поддерживаемых систем. Вот основные критерии, правила и принципы, связанные с модульностью.

Критерии модульности

1. **Разделение ответственности**: Каждый модуль должен выполнять одну конкретную задачу или группу связанных задач. Это упрощает понимание и поддержку кода.
2. **Слабая связность**: Модули должны быть независимыми друг от друга. Изменения в одном модуле не должны требовать изменений в других модулях.
3. **Высокая связность внутри модуля**: Все компоненты внутри модуля должны быть тесно связаны и работать вместе для выполнения единой задачи.
4. **Интерфейсы**: Каждый модуль должен иметь четко определенные интерфейсы для взаимодействия с другими модулями. Это упрощает интеграцию и замену модулей.
5. **Повторное использование**: Модули должны быть спроектированы так, чтобы их можно было повторно использовать в различных частях системы или в других проектах.

Правила модульности

1. **Использование интерфейсов**: Определяйте интерфейсы для модулей, чтобы скрыть внутреннюю реализацию и предоставить только необходимые методы и свойства.
2. **Соблюдение принципов SOLID**: Эти принципы помогают создавать более модульные и поддерживаемые системы:
   - **S**: Single Responsibility Principle (Принцип единственной ответственности)
   - **O**: Open/Closed Principle (Принцип открытости/закрытости)
   - **L**: Liskov Substitution Principle (Принцип подстановки Лисков)
   - **I**: Interface Segregation Principle (Принцип разделения интерфейсов)
   - **D**: Dependency Inversion Principle (Принцип инверсии зависимостей)
3. **Структурирование кода**: Организуйте код в папки и модули, чтобы облегчить навигацию и понимание структуры проекта.
4. **Документация**: Обеспечьте хорошую документацию для каждого модуля, описывающую его функциональность, интерфейсы и зависимости.
5. **Тестирование**: Каждый модуль должен быть независимым и легко тестируемым. Используйте модульные тесты для проверки функциональности каждого модуля.

Принципы модульности

1. **Инкапсуляция**: Скрытие внутренней реализации модуля от других модулей, предоставляя только необходимые интерфейсы для взаимодействия.
2. **Композиция**: Создание сложных систем путем объединения простых модулей. Это позволяет легко изменять и заменять отдельные компоненты.
3. **Абстракция**: Сосредоточение на высокоуровневых концепциях, а не на деталях реализации. Это помогает уменьшить сложность и улучшить понимание системы.
4. **Декомпозиция**: Разделение больших задач на более мелкие, управляемые модули. Это упрощает разработку и тестирование.
5. **Согласованность**: Модули должны следовать единым стандартам и соглашениям, чтобы обеспечить согласованность в коде и облегчить его поддержку.

Python предлагает широкий спектр **модулей** для упрощения сложных задач. Среди этих полезных модулей находится модуль заголовков, который позволяет разработчикам извлекать новостные заголовки и статьи из различных источников новостей в интернете.

Этот модуль может быть чрезвычайно полезен для создания инструментов мониторинга новостей, анализа тенденций и поддержания связи с последними новостными событиями.

Рассмотрим модуль заголовков в Python и как его можно использовать для извлечения новостных заголовков из различных источников.

Модуль в Python — это файл, содержащий операторы и определения. Он может определять функции, классы и переменные, а также может включать выполняемый код. Модули используются для организации кода в логические единицы, для уменьшения сложности и увеличения возможности повторного использования.

Преимущества использования модулей:

• **Повторное использование кода**: Модули способствуют повторному использованию кода, что упрощает написание и поддержку кода. Вы можете `import` модуль в несколько программ, уменьшая необходимость писать один и тот же код снова.

• **Модульность**: Модули способствуют модульности, позволяя разработчикам разбивать большую программу на меньшие, более управляемые части. Это упрощает понимание и поддержку кода.

• **Расширение функциональности**: Модули предоставляют дополнительные функции, на написание которых иначе пришлось бы потратить много кода. Разработчики могут использовать различные предварительно написанные модули для расширения функциональности своей программы.

Соглашения о Наименовании Пакетов и Модулей

В Python **соглашения о наименовании модулей** (исходные файлы Python) следующие [1]:

1.Имена модулей должны быть в нижнем регистре.

Пример: **my_module.py**.

2.Если имя модуля состоит из нескольких слов, они должны быть разделены подчеркиваниями.

Пример: **my_module_utils.py**.

3.Имена модулей должны быть описательными и передавать цель или функциональность модуля.

Пример: **math_operations.py**.

4.Избегайте использования имен, которые конфликтуют с ключевыми словами Python или именами встроенных модулей.

Пример: **random.py** (следует избегать, так как это конфликтует с встроенным модулем random).

5.Если имя модуля совпадает с именем стандартной библиотеки или библиотеки сторонних разработчиков, рассмотрите возможность использования другого имени, чтобы избежать путаницы.

Пример: **requests.py** (следует избегать, если это совпадает с популярной библиотекой requests).

6.Избегайте использования ведущих подчеркиваний _ в именах модулей, если только это не предназначено для указания на то, что модуль предназначен для внутреннего использования или является частью приватного API пакета.

Пример: **_internal_module.py**.

7.Избегайте использования дефисов - или специальных символов в именах модулей, так как они не являются допустимыми символами в именах модулей Python.

Пример: **my-module.py** (дефис не разрешен).

Литература

1. PEP 8 – Style Guide for Python Code
https://peps.python.org/pep-0008/#package-and-module-names

Пример № 1.
Использование модуля `math` для вычисления квадратного корня.

```
import math
num = 16
result = math.sqrt(num)
print(f"Square root of {num} is: {result}")
```

Пример № 2.
Использование модуля `os` для получения текущего каталога.

```
import os
cwd = os.getcwd()
print(f"Current working directory is: {cwd}")
```

Пример № 3.
Использование модуля `random` для генерации случайных чисел.

Генерирует случайное целое число от 1 до 100:

```
import random

random_number = random.randint(1, 100)
print(random_number)
```

Пример № 4.
Использование модуля `math` для вычисления квадратного корня.

```
import math

number = 25
square_root = math.sqrt(number)
print(square_root)
```

Пример № 5.
Импортирование Функций - даты из Модуля.

```
from datetime import date

today = date.today()
print("Today's date:", today) # Output: Today's date: 2022-11-11
```

Существует несколько способов **создать модуль в Python**:

1.Способ № 1.

Создать отдельный файл .ру: Вы можете создать отдельный файл .ру, который содержит код, который вы хотите использовать в качестве модуля.

Пример:

```
### my_module.py

def greet(name):
    print(f"Hello, {name}!")
```

```
Чтобы использовать этот модуль, просто import его в свой Python-скрипт:
import my_module

my_module.greet("John")
```

Результат: **Привет, Джон!.**

2.Способ № 2.

Создайте папку с файлом __init__.py: Вы можете создать папку с файлом **__init__.py**, который определяет функции и классы, которые вы хотите использовать как модуль.

Пример:

```
my_module/
    __init__.py
    greet.py
```

В файле __init__.py вы можете определить функции и классы, которые хотите использовать:

```
### my_module/__init__.py

from .greet import greet
```

В файле greet.py вы определяете саму функцию:

```
### my_module/greet.py

def greet(name):
    print(f"Hello, {name}!")
```

Чтобы использовать этот модуль, вы можете import его так же, как и ранее:

```
import my_module

my_module.greet("John")
```

Результат: **Привет, Джон**!.

Практическое задание № 2. Модульность.
Повторное использование

Цель: Создать модуль в программе Python и дополнительный модуль. Выполнить математические и тригонометрические операции. См. Исходные данные по вариантам (таблица 1). Вариант задания выбрать по порядковому номеру списка группы.

Програмнное обеспечение:
1) Notepad++
2) Блокнот (Windows)
3) Python (ver. 3.10 и выше)

Задание № 1.

1.Создать один файл: **math_utils.py**

Или

2.Создать два файла:
- основной файл: **модульность.py**
- дополнительный файл: **math_utils.py**

В основном файле представить – исходные данные.

В дополнительном файле – представить решение, формулы в виде переменных.

Исходные данные.

Таблица 1 – «Модули в Python». Каждое задание связано с использованием модулей, их созданием или импортом

№	Задание
1	Напишите модуль string_utils.py, который содержит функции для перевода строки в верхний и нижний регистры.
2	Создайте модуль file_utils.py, который содержит функции для чтения и записи текстовых файлов.
3	Напишите модуль date_utils.py, который содержит функции для форматирования даты и времени.
4	Создайте модуль currency_converter.py, который содержит функции для конвертации валют.
5	Напишите модуль statistics_utils.py, который содержит функции для вычисления среднего, медианы и моды.
6	Создайте модуль random_utils.py, который содержит функции для генерации случайных чисел и выборки из списка.
7	Напишите модуль email_utils.py, который содержит функции для проверки корректности адреса электронной почты.
8	Создайте модуль http_utils.py, который содержит функции для отправки GET и POST запросов.
9	Напишите модуль json_utils.py, который содержит функции для сериализации и десериализации JSON данных.
10	Создайте модуль xml_utils.py, который содержит функции для работы с XML файлами.
11	Напишите модуль image_utils.py, который содержит функции для открытия и сохранения изображений.
12	Создайте модуль math_operations.py, который содержит функции для выполнения основных арифметических операций.
13	Напишите модуль logger.py, который содержит функции для логирования сообщений в файл.

14	Создайте модуль path_utils.py, который содержит функции для работы с файловыми путями.
15	Напишите модуль color_utils.py, который содержит функции для преобразования цветов из RGB в HEX и обратно.
16	Создайте модуль user_input.py, который содержит функции для безопасного получения ввода от пользователя.
17	Напишите модуль math_constants.py, который содержит константы, такие как π и e.
18	Создайте модуль math_utils.py, который содержит функции для вычисления площади круга и площади прямоугольника.
19	Создайте модуль password_generator.py, который содержит функции для генерации безопасных паролей.
20	Напишите модуль time_utils.py, который содержит функции для работы с временем (например, задержка).
21	Создайте модуль validation_utils.py, который содержит функции для валидации различных типов данных.
22	Напишите модуль geolocation_utils.py, который содержит функции для работы с геолокацией (например, получение координат).
23	Создайте модуль html_utils.py, который содержит функции для генерации HTML-кода.
24	Напишите модуль text_analysis.py, который содержит функции для анализа текста (например, подсчет слов).
25	Создайте модуль file_compressor.py, который содержит функции для сжатия и распаковки файлов.
26	Напишите модуль network_utils.py, который содержит функции для проверки доступности URL.
27	Создайте модуль shopping_cart.py, который содержит классы и функции для управления корзиной покупок.
28	Напишите модуль note_taking.py, который содержит функции для создания, редактирования и удаления заметок.
29	Создайте модуль quiz_generator.py, который содержит функции для генерации и проверки тестов.
30	Напишите модуль data_visualization.py, который содержит функции для построения графиков и диаграмм.
31	Создайте модуль weather.py, который с помощью requests получает погоду с OpenWeatherMap API. Оформите его как полноценный пакет с обработкой ошибок.
32	Создайте модуль password_checker.py. Реализуйте функцию is_strong_password(password), которая проверяет пароль на длину, наличие цифр, заглавных и строчных букв.
33	Создайте модуль report_generator.py. Реализуйте функцию create_text_report(data, filename), которая принимает список данных и записывает их в текстовый файл (*.txt) в виде структурированного отчета.

34	Создайте модуль api_client.py. Реализуйте класс APIClient с методами get() и post() для взаимодействия с внешним REST API (например, для получения курсов валют). Используйте библиотеку requests.
35	Создайте модуль task_scheduler.py. Используя библиотеку schedule, реализуйте функцию schedule_daily_report(time), которая будет ежедневно в заданное время запускать функцию генерации отчета.
36	Напишите скрипт system_info.py, который выводит информацию о системе: имя компьютера, текущий пользователь, версия ОС (используйте модуль os и platform).
37	Напишите скрипт simple_parser.py, который с помощью requests и BeautifulSoup извлекает заголовки новостей с главной страницы выбранного новостного сайта.
38	Создайте программу download_manager.py, которая параллельно скачивает полный адрес URL в формате *.m3u8) и вставляет его в файл -txt. Страна Италия. Пример записа в текстовый формат: #EXTINF:-1,Italy2 http://wms.shared.streamshow.it/italia2/mp4:italia2/playlist.m3u8 /пробел/ #EXTINF:-1,Euro TV https://5f22d76e220e1.streamlock.net/eurotv/eurotv/playlist.m3u8
39	Добавьте в ваше веб-приложение фоновую задачу, например, еженедельную рассылку отчетов по email, используя Celery и Redis в качестве брокера.

Интернет ресурсы
URL: https://metanit.com/python/tutorial/2.3.php

Контрольные вопросы

1.Назовите определение «Модульность».
2.Назовите критерии модульности.
3.Назовите принципы модульности.
4.Для какой цели применяют модули в программе Python.
5.Назовите преимущества использования модулей в программе Python.

Основная литература

1.Е.В. Курехин. Конспект лекций по дисциплине «Объектно-ориентированное программирование». Для студентов, обучающихся по направлениям 01.03.02 – Прикладная математика и информатика, ОП «Прикладное машинное обучение». Профиль: «Прикладное машинное

обучение» (программа подготовки бакалавра). — М.: Финансовый университет, департамент анализа данных и машинного обучения, 2025. — 143 с.
https://campus.fa.ru/my/

2.Р.И. Горохова, Е.П. Догадина, В.И. Долгов, С.В. Макрушин. Практикум по программированию на языке Python. Учебное пособие. Для студентов, обучающихся по направлениям 01.03.02 «Прикладная математика и информатика», 09.03.03 «Прикладная информатика», 10.03.01 «Информационная безопасность», 38.03.05 «Бизнес-информатика», (программа подготовки бакалавра). — М.: Финансовый универси-тет, департамент анализа данных и машинного обучения, 2023. — 320 с.
https://elib.fa.ru/fbook/books137296.pdf/view

3.Краткий курс ООП на Python: как избежать путаницы в коде
https://skillbox.ru/media/code/kak-izbezhat-putanitsy-v-kode-ili-kratkiy-kurs-oop-na-python/

4.ООП на Python: концепции, принципы и примеры реализации
https://proglib.io/p/python-oop

5.5.1. Объектная модель Python. Классы, поля и методы. Примеры
https://education.yandex.ru/handbook/python/article/obuektnaya-model-python-klassy-polya-i-metody

6. Урок 6. Принципы ООП. Классы, объекты, поля и методы. Уровни доступа.
https://smartiqa.ru/courses/python/lesson-6

7.Статьи (Периодические публикации, Журналы РФ, зарубежные).

Интернет ресурсы

URL:
https://pythonworld.ru/osnovy
https://metanit.com/python/tutorial/1.1.php
https://www.opennet.ru/docs/RUS/python/python_b.html

Дополнительная литература

1. Объектно-ориентированное программирование. В 3-х частях. Ч.1: учебное пособие / П.П. Степанов, А.А. Кабанов, В.А. Никонов, Т.С. Павлюченко. – Омск: Омский государственный технический университет, 2021. – 112 с. – Режим доступа: https://www.iprbookshop.ru/124850.html (ЭБС «IPRbooks»).

2. Букунов, С.В. Объектно-ориентированное программирование на

языке Python: учебное пособие / С.В. Букунов, О.В. Букунова. – Санкт-Петербург: Санкт-Петербургский государственный архитектурно-строительный университет, ЭБС АСВ, 2020. – 119 с. – Режим доступа: https://www.iprbookshop.ru/117194.html (ЭБС «IPRbooks»).

3. Зыков, С.В. Введение в теорию программирования. Объектно-ориентированный подход: учебное пособие / С.В. Зыков. – 3-е изд. – Москва: Интернет-Университет Информационных Технологий (ИНТУИТ), Ай Пи Ар Медиа, 2021. – 187 с. – Режим доступа: ttps://www.iprbookshop.ru/102007.html (ЭБС «IPRbooks»).

4. Щерба, А.В. Программирование на Python: первые шаги / А.В. Щерба. – Москва: Лаборатория знаний, 2022. – 251 с. – Режим доступа: https://www.iprbookshop.ru/120878.html (ЭБС «IPRbooks»).