Практические задания для второго блока "Модули и пакеты" по базовому курсу по Python

Предложенные практические задания помогут студентам получить навыки работы с проектами на Python используя виртуальное окружение.

Задание 1: Использование модуля `cowsay`

Условие:

Создайте программу, которая выводит сообщение пользователя с использованием ASCII-арта из модуля `cowsay`. Пользователь вводит сообщение, и программа отображает корову, "произносящую" это сообщение.

Требования:

1. **Установка виртуального окружения:**
   - Создайте и активируйте виртуальное окружение для проекта.
2. **Установка зависимостей:**
   - Установите модуль `cowsay` через `pip`.
3. **Создание файла `requirements.txt`:**
   - Сохраните установленные зависимости в файл `requirements.txt`.
4. **Написание скрипта `main.py`:**
   - Запросите у пользователя ввод сообщения.
   - Выведите сообщение с использованием `cowsay.cow()`.
5. **Запуск программы:**
   - Запустите программу и проверьте её работу в виртуальном окружении.
6. **Обработка исключений:**
   - Добавьте обработку возможных исключений при вводе данных.
7. **Логирование:**
   - Используйте модуль `logging` для записи информации о запуске программы.
8. **Форматирование кода:**
   - Соблюдайте PEP 8 для чистоты кода.
9. **Использование комментариев:**
   - Добавьте необходимые комментарии для пояснения кода.
10. **Выход из виртуального окружения:**
    - Деактивируйте виртуальное окружение после завершения работы.

Код `main.py`:

```
import cowsay
import logging

# Настройка логирования
logging.basicConfig(level=logging.INFO)

def main():
    try:
        message = input("Введите ваше сообщение: ")
        cowsay.cow(message)
        logging.info("Программа успешно выполнена")
```

```
    except Exception as e:
        logging.error(f"Произошла ошибка: {e}")

if __name__ == "__main__":
    main()
```

Содержимое `requirements.txt`:

```
cowsay==4.0
```

Задание 2: Получение данных с веб-сайта с помощью `requests`

Условие:

Создайте программу, которая получает данные с веб-сайта и выводит их содержимое. Используйте модуль `requests` для выполнения HTTP-запросов.

Требования:

1. **Установка виртуального окружения:**
   - Создайте и активируйте виртуальное окружение для проекта.
2. **Установка зависимостей:**
   - Установите модуль `requests` через `pip`.
3. **Создание файла `requirements.txt`:**
   - Сохраните установленные зависимости в файл `requirements.txt`.
4. **Написание скрипта `main.py`:**
   - Сделайте GET-запрос к странице `https://httpbin.org/get`.
   - Выведите полученный JSON-ответ.
5. **Обработка исключений:**
   - Обработайте возможные ошибки соединения.
6. **Логирование:**
   - Используйте модуль `logging` для записи информации о запросе и ответе.
7. **Форматирование вывода:**
   - Используйте модуль `json` для красивого отображения JSON-ответа.
8. **Использование функций:**
   - Оформите код запроса в виде функции.
9. **Комментарии:**
   - Добавьте комментарии для пояснения работы кода.
10. **Выход из виртуального окружения:**
    - Деактивируйте виртуальное окружение после завершения работы.

Код `main.py`:

```
import requests
import logging
import json

# Настройка логирования
logging.basicConfig(level=logging.INFO)

def fetch_data():
    url = "https://httpbin.org/get"
    try:
        response = requests.get(url)
```

```
        response.raise_for_status()  # Проверка статуса ответа
        data = response.json()
        print(json.dumps(data, indent=4))  # Красивый вывод JSON
        logging.info("Данные успешно получены")
    except requests.exceptions.RequestException as e:
        logging.error(f"Ошибка при выполнении запроса: {e}")

if __name__ == "__main__":
    fetch_data()
```

Содержимое `requirements.txt`:

```
requests==2.28.1
```

Задание 3: Обработка данных с использованием `pandas`

Условие:

Создайте программу, которая считывает данные из CSV-файла `data.csv`, выполняет простую обработку данных и выводит результат. Используйте модуль `pandas` для работы с данными.

Требования:

1. **Установка виртуального окружения:**
   - Создайте и активируйте виртуальное окружение для проекта.
2. **Установка зависимостей:**
   - Установите модуль `pandas` через `pip`.
3. **Создание файла `requirements.txt`:**
   - Сохраните установленные зависимости в файл `requirements.txt`.
4. **Подготовка данных:**
   - Создайте файл `data.csv` с произвольными данными о продажах (например, столбцы: 'Товар', 'Количество', 'Цена').
5. **Написание скрипта `main.py`:**
   - Считайте данные из `data.csv`.
   - Вычислите общую выручку по каждому товару.
   - Выведите результат.
6. **Обработка исключений:**
   - Обработайте возможные ошибки при чтении файла.
7. **Логирование:**
   - Используйте модуль `logging` для записи процесса обработки данных.
8. **Использование функций:**
   - Оформите обработку данных в виде функций.
9. **Форматирование вывода:**
   - Используйте `pandas` для красивого отображения таблицы.
10. **Выход из виртуального окружения:**
    - Деактивируйте виртуальное окружение после завершения работы.

Код `main.py`:

```
import pandas as pd
import logging
```

```
# Настройка логирования
logging.basicConfig(level=logging.INFO)

def process_data():
    try:
        df = pd.read_csv('data.csv')
        df['Выручка'] = df['Количество'] * df['Цена']
        total_revenue = df['Выручка'].sum()
        print(df)
        print(f"\nОбщая выручка: {total_revenue}")
        logging.info("Данные успешно обработаны")
    except FileNotFoundError:
        logging.error("Файл data.csv не найден")
    except Exception as e:
        logging.error(f"Произошла ошибка: {e}")

if __name__ == "__main__":
    process_data()
```

Содержимое `requirements.txt`:

```
pandas==1.5.0
```

Пример содержимого `data.csv`:

```
Товар,Количество,Цена
Телефон,10,5000
Планшет,5,10000
Ноутбук,3,30000
Монитор,7,7000
```

Задание 4: Математические вычисления с использованием `numpy`

Условие:

Создайте программу, которая генерирует массив случайных чисел, вычисляет их среднее значение и стандартное отклонение. Используйте модуль `numpy`.

Требования:

1. **Установка виртуального окружения:**
   - Создайте и активируйте виртуальное окружение для проекта.
2. **Установка зависимостей:**
   - Установите модуль `numpy` через `pip`.
3. **Создание файла `requirements.txt`:**
   - Сохраните установленные зависимости в файл `requirements.txt`.
4. **Написание скрипта `main.py`:**
   - Сгенерируйте массив из 100 случайных чисел.
   - Вычислите среднее значение и стандартное отклонение.
   - Выведите результаты.
5. **Использование функций:**
   - Оформите код в виде функций.
6. **Логирование:**
   - Используйте модуль `logging`.
7. **Обработка исключений:**

    - Обработайте возможные ошибки.
8. **Комментарии:**
    - Добавьте комментарии к коду.
9. **Форматирование кода:**
    - Соблюдайте PEP 8.
10. **Выход из виртуального окружения:**
    - Деактивируйте виртуальное окружение после завершения работы.

Код `main.py`:

```
import numpy as np
import logging

# Настройка логирования
logging.basicConfig(level=logging.INFO)

def calculate_statistics():
    try:
        data = np.random.rand(100)
        mean = np.mean(data)
        std_dev = np.std(data)
        print(f"Среднее значение: {mean}")
        print(f"Стандартное отклонение: {std_dev}")
        logging.info("Вычисления успешно выполнены")
    except Exception as e:
        logging.error(f"Произошла ошибка: {e}")

if __name__ == "__main__":
    calculate_statistics()
```

Содержимое `requirements.txt`:

```
numpy==1.23.3
```

Задание 5: Обработка изображений с использованием `Pillow`

Условие:

Создайте программу, которая открывает изображение, изменяет его размер и сохраняет результат в новый файл. Используйте модуль `Pillow`.

Требования:

1. **Установка виртуального окружения:**
    - Создайте и активируйте виртуальное окружение для проекта.
2. **Установка зависимостей:**
    - Установите модуль `Pillow` через `pip`.
3. **Создание файла `requirements.txt`:**
    - Сохраните установленные зависимости в файл `requirements.txt`.
4. **Подготовка изображения:**
    - Поместите изображение `input.jpg` в папку проекта.
5. **Написание скрипта `main.py`:**
    - Откройте изображение `input.jpg`.
    - Измените размер изображения на 800x600.

    - Сохраните результат в файл `output.jpg`.
6. **Обработка исключений:**
    - Обработайте ошибки при работе с файлом.
7. **Логирование:**
    - Используйте модуль `logging`.
8. **Использование функций:**
    - Оформите код в виде функций.
9. **Комментарии:**
    - Добавьте необходимые комментарии.
10. **Выход из виртуального окружения:**
    - Деактивируйте виртуальное окружение после завершения работы.

Код `main.py`:

```
from PIL import Image
import logging

# Настройка логирования
logging.basicConfig(level=logging.INFO)

def resize_image():
    try:
        img = Image.open('input.jpg')
        img_resized = img.resize((800, 600))
        img_resized.save('output.jpg')
        logging.info("Изображение успешно обработано и сохранено как
output.jpg")
    except FileNotFoundError:
        logging.error("Файл input.jpg не найден")
    except Exception as e:
        logging.error(f"Произошла ошибка: {e}")

if __name__ == "__main__":
    resize_image()
```

Содержимое `requirements.txt`:

```
Pillow==9.2.0
```

Задание 6: Создание ASCII-арта с использованием `art`

Условие:

Создайте программу, которая принимает от пользователя строку и выводит её в виде ASCII-арта с помощью модуля `art`.

Требования:

1. **Установка виртуального окружения:**
    - Создайте и активируйте виртуальное окружение для проекта.
2. **Установка зависимостей:**
    - Установите модуль `art` через `pip`.
3. **Создание файла `requirements.txt`:**
    - Сохраните установленные зависимости в файл `requirements.txt`.
4. **Написание скрипта `main.py`:**

    - Запросите у пользователя ввод строки.
    - Выведите строку в виде ASCII-арта.
5. **Обработка исключений:**
    - Обработайте возможные ошибки при вводе данных.
6. **Логирование:**
    - Используйте модуль `logging`.
7. **Использование функций:**
    - Оформите код в виде функций.
8. **Комментарии:**
    - Добавьте необходимые комментарии.
9. **Форматирование кода:**
    - Соблюдайте стандарты оформления кода.
10. **Выход из виртуального окружения:**
    - Деактивируйте виртуальное окружение после завершения работы.

Код `main.py`:

```
from art import text2art
import logging

# Настройка логирования
logging.basicConfig(level=logging.INFO)

def create_ascii_art():
    try:
        message = input("Введите сообщение для ASCII-арта: ")
        art = text2art(message)
        print(art)
        logging.info("ASCII-арт успешно создан")
    except Exception as e:
        logging.error(f"Произошла ошибка: {e}")

if __name__ == "__main__":
    create_ascii_art()
```

Содержимое `requirements.txt`:

```
art==5.8
```

Задание 7: Создание терминальных анимаций с использованием `asciimatics`

Условие:

Создайте программу, которая отображает простую анимацию в терминале с использованием модуля `asciimatics`.

Требования:

1. **Установка виртуального окружения:**
    - Создайте и активируйте виртуальное окружение для проекта.
2. **Установка зависимостей:**
    - Установите модуль `asciimatics` через `pip`.
3. **Создание файла `requirements.txt`:**
    - Сохраните установленные зависимости в файл `requirements.txt`.

4. **Написание скрипта `main.py`:**
    - Создайте анимацию бегущего текста.
5. **Обработка исключений:**
    - Обработайте возможные ошибки при запуске анимации.
6. **Логирование:**
    - Используйте модуль `logging`.
7. **Использование функций:**
    - Оформите код в виде функций.
8. **Комментарии:**
    - Добавьте необходимые комментарии.
9. **Форматирование кода:**
    - Соблюдайте стандарты оформления кода.
10. **Выход из виртуального окружения:**
    - Деактивируйте виртуальное окружение после завершения работы.

Код `main.py`:

```
from asciimatics.screen import Screen
import logging
import time

# Настройка логирования
logging.basicConfig(level=logging.INFO)

def demo(screen):
    try:
        for i in range(100):
            screen.print_at('Hello, Asciimatics!', i % screen.width,
screen.height // 2)
            screen.refresh()
            time.sleep(0.05)
            screen.clear()
        logging.info("Анимация успешно выполнена")
    except Exception as e:
        logging.error(f"Произошла ошибка: {e}")

if __name__ == "__main__":
    Screen.wrapper(demo)
```

Содержимое `requirements.txt`:

```
asciimatics==1.13.0
```

Задание 8: Астрономические расчёты с использованием `PyEphem`

Условие:

Создайте программу, которая вычисляет время восхода и захода солнца для заданной даты и местоположения с использованием модуля `PyEphem`.

Требования:

1. **Установка виртуального окружения:**
    - Создайте и активируйте виртуальное окружение для проекта.
2. **Установка зависимостей:**

    - Установите модуль `pyephem` через `pip`.
3. **Создание файла `requirements.txt`:**
    - Сохраните установленные зависимости в файл `requirements.txt`.
4. **Написание скрипта `main.py`:**
    - Запросите у пользователя широту и долготу.
    - Вычислите время восхода и захода солнца.
5. **Обработка исключений:**
    - Обработайте возможные ошибки при вводе данных.
6. **Логирование:**
    - Используйте модуль `logging`.
7. **Использование функций:**
    - Оформите код в виде функций.
8. **Комментарии:**
    - Добавьте необходимые комментарии.
9. **Форматирование кода:**
    - Соблюдайте стандарты оформления кода.
10. **Выход из виртуального окружения:**
    - Деактивируйте виртуальное окружение после завершения работы.

Код `main.py`:

```
import ephem
import logging

# Настройка логирования
logging.basicConfig(level=logging.INFO)

def calculate_sun_times():
    try:
        lat = input("Введите широту (например, 55.75 для Москвы): ")
        lon = input("Введите долготу (например, 37.61 для Москвы): ")
        observer = ephem.Observer()
        observer.lat = lat
        observer.lon = lon
        observer.date = ephem.now()
        sun = ephem.Sun()
        sunrise = observer.next_rising(sun)
        sunset = observer.next_setting(sun)
        print(f"Восход солнца: {ephem.localtime(sunrise)}")
        print(f"Заход солнца: {ephem.localtime(sunset)}")
        logging.info("Время восхода и захода солнца успешно вычислено")
    except Exception as e:
        logging.error(f"Произошла ошибка: {e}")

if __name__ == "__main__":
    calculate_sun_times()
```

Содержимое `requirements.txt`:

```
pyephem==3.7.7.1
```

Задание 9: Создание простой игры с использованием `pygame`

Условие:

Создайте программу, которая открывает окно и отображает движущийся круг с использованием модуля `pygame`.

Требования:

1. **Установка виртуального окружения:**
   - Создайте и активируйте виртуальное окружение для проекта.
2. **Установка зависимостей:**
   - Установите модуль `pygame` через `pip`.
3. **Создание файла `requirements.txt`:**
   - Сохраните установленные зависимости в файл `requirements.txt`.
4. **Написание скрипта `main.py`:**
   - Создайте окно приложения.
   - Отобразите круг, который движется по экрану.
5. **Обработка исключений:**
   - Обработайте возможные ошибки при запуске приложения.
6. **Логирование:**
   - Используйте модуль `logging`.
7. **Использование функций:**
   - Оформите код в виде функций.
8. **Комментарии:**
   - Добавьте необходимые комментарии.
9. **Форматирование кода:**
   - Соблюдайте стандарты оформления кода.
10. **Выход из виртуального окружения:**
    - Деактивируйте виртуальное окружение после завершения работы.

Код `main.py`:

```
import pygame
import sys
import logging

# Настройка логирования
logging.basicConfig(level=logging.INFO)

def run_game():
    try:
        pygame.init()
        size = width, height = 800, 600
        speed = [2, 2]
        black = 0, 0, 0

        screen = pygame.display.set_mode(size)
        ball = pygame.Surface((50, 50), pygame.SRCALPHA)
        pygame.draw.circle(ball, (255, 0, 0), (25, 25), 25)

        ballrect = ball.get_rect()

        while True:
            for event in pygame.event.get():
                if event.type == pygame.QUIT:
                    pygame.quit()
                    sys.exit()

            ballrect = ballrect.move(speed)
            if ballrect.left < 0 or ballrect.right > width:
```

```
                speed[0] = -speed[0]
            if ballrect.top < 0 or ballrect.bottom > height:
                speed[1] = -speed[1]

            screen.fill(black)
            screen.blit(ball, ballrect)
            pygame.display.flip()
    except Exception as e:
        logging.error(f"Произошла ошибка: {e}")

if __name__ == "__main__":
    run_game()
```

Содержимое `requirements.txt`:

```
pygame==2.1.2
```

Задание 10: Вывод случайных шуток с использованием `pyjokes`

Условие:

Создайте программу, которая выводит случайную программистскую шутку с использованием модуля `pyjokes`.

Требования:

1. **Установка виртуального окружения:**
   - Создайте и активируйте виртуальное окружение для проекта.
2. **Установка зависимостей:**
   - Установите модуль `pyjokes` через `pip`.
3. **Создание файла `requirements.txt`:**
   - Сохраните установленные зависимости в файл `requirements.txt`.
4. **Написание скрипта `main.py`:**
   - Выведите случайную шутку.
5. **Обработка исключений:**
   - Обработайте возможные ошибки при получении шутки.
6. **Логирование:**
   - Используйте модуль `logging`.
7. **Использование функций:**
   - Оформите код в виде функций.
8. **Комментарии:**
   - Добавьте необходимые комментарии.
9. **Форматирование кода:**
   - Соблюдайте стандарты оформления кода.
10. **Выход из виртуального окружения:**
    - Деактивируйте виртуальное окружение после завершения работы.

Код `main.py`:

```
import pyjokes
import logging

# Настройка логирования
logging.basicConfig(level=logging.INFO)
```

```
def tell_joke():
    try:
        joke = pyjokes.get_joke(language='en', category='all')
        print(joke)
        logging.info("Шутка успешно выведена")
    except Exception as e:
        logging.error(f"Произошла ошибка: {e}")

if __name__ == "__main__":
    tell_joke()
```

Содержимое `requirements.txt`:

```
pyjokes==0.6.0
```

Эти задания предоставляют практический опыт работы с виртуальными окружениями и различными модулями Python. Выполняя их, студенты научатся устанавливать и использовать сторонние библиотеки, а также поймут, как организовывать свой код и рабочее окружение.