Setting Up a Python Project with GitHub, Local Environment, Logging, and Testing

Author: Hari Prasad

Date: 12/09/2023

Introduction

This document provides an overview of the steps taken to create a Python project repository on GitHub and set up the project on a local machine.
The project involves generating a basic project structure and a setup.py file for a Python application.

Step 1: Creating a GitHub Repository

1.1. Visit the GitHub website (https://github.com) and log in to your account.

1.2. Click on the '+' icon in the top-right corner and select "New Repository" to create a new GitHub repository.

1.3. Fill in the repository name and provide an optional description and other settings as needed.

1.4. Click on the "Create repository" button to create the GitHub repository.

Step 2: Cloning the Repository Locally

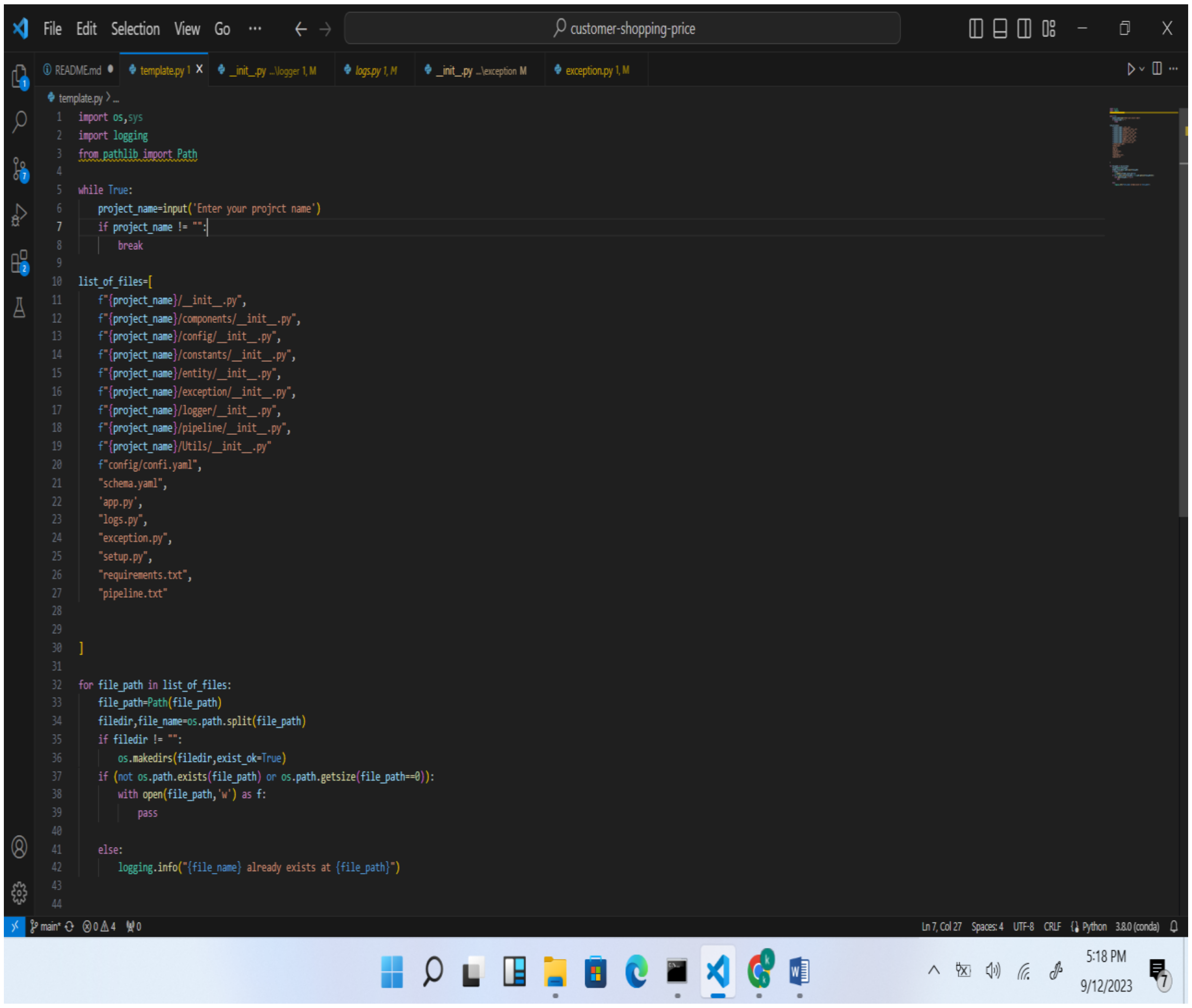
2.1. Open your terminal or command prompt on your local machine.

2.2. Use the git clone command to clone the newly created GitHub repository to your local machine.
For example:
git clone https://github.com/kalehariprasad/customer-shopping-price

Step 3: Creating the Python Project Structure

3.1. In your local project directory, you've created a Python script called template.py. This script is responsible for generating the initial project structure.

3.2. The template.py script prompts you to enter a project name and then generates the following project structure snippet:



```python
import os,sys
import logging
from pathlib import Path

while True:
    project_name=input('Enter your projrct name')
    if project_name != "":
        break

list_of_files=[
    f"{project_name}/__init__.py",
    f"{project_name}/components/__init__.py",
    f"{project_name}/config/__init__.py",
    f"{project_name}/constants/__init__.py",
    f"{project_name}/entity/__init__.py",
    f"{project_name}/exception/__init__.py",
    f"{project_name}/logger/__init__.py",
    f"{project_name}/pipeline/__init__.py",
    f"{project_name}/Utils/__init__.py"
    f"config/confi.yaml",
    "schema.yaml",
    'app.py',
    "logs.py",
    "exception.py",
    "setup.py",
    "requirements.txt",
    "pipeline.txt"


]

for file_path in list_of_files:
    file_path=Path(file_path)
    filedir,file_name=os.path.split(file_path)
    if filedir != "":
        os.makedirs(filedir,exist_ok=True)
    if (not os.path.exists(file_path) or os.path.getsize(file_path==0)):
        with open(file_path,'w') as f:
            pass

    else:
        logging.info("{file_name} already exists at {file_path}")
```

3.3. The script creates directories for various project components and initializes empty __init__.py files to indicate Python packages

Step 4: Creating the setup.py File

4.1. The setup.py file is used to define project metadata and dependencies for packaging.

4.2. The script, setup.py, imports required modules and defines a function, get_requirements_list(), to read project dependencies from a requirements.txt file.

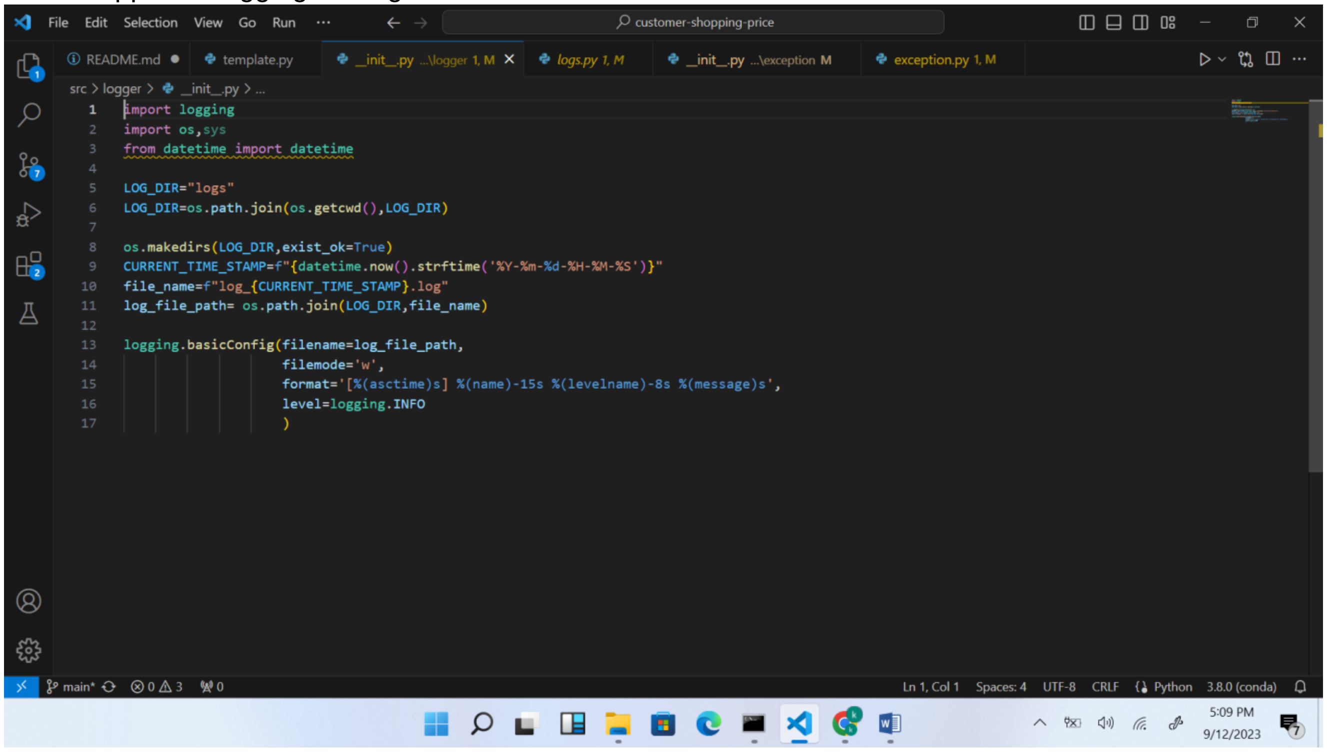
4.3. It then calls setup() from the setuptools library to configure the project with metadata such as name, version, author, and packages.

4.4. The script reads the project dependencies using the get_requirements_list() function and sets them as install requirements.

Step 5: Configuring Logging and Testing

5.1. Created a `logging` file (e.g., `logging.py`) in your project directory to handle logging configurations.

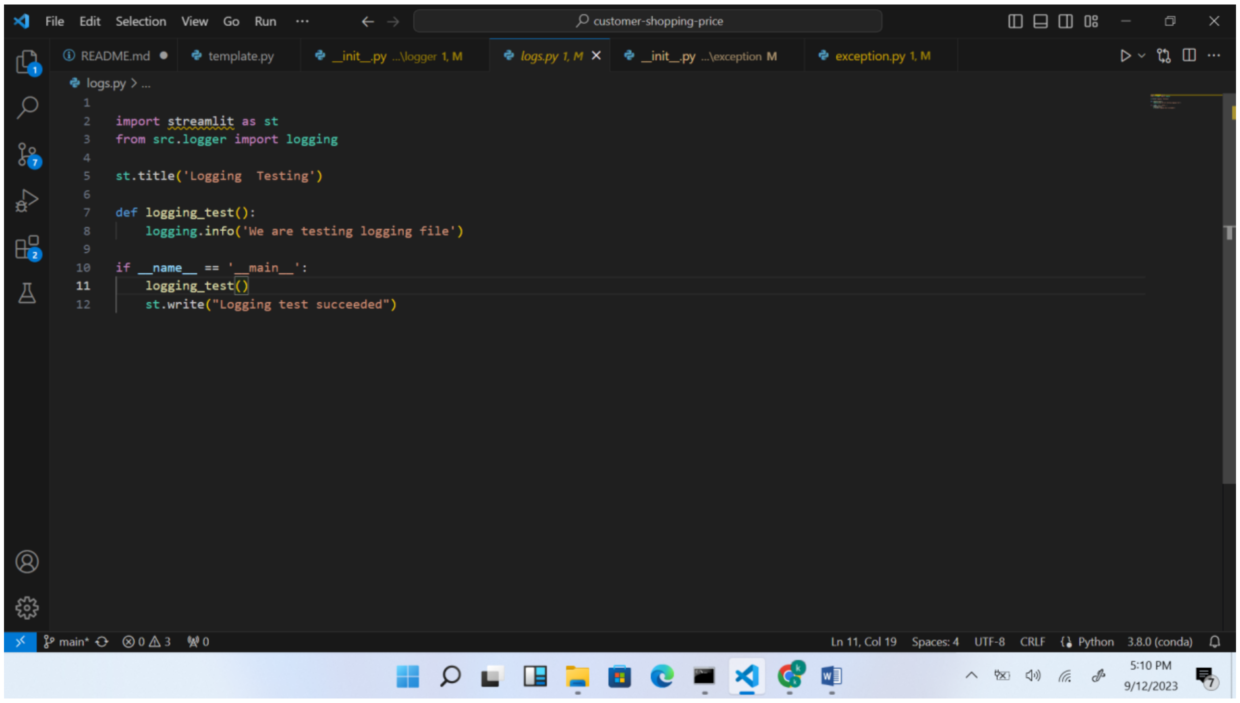
Code snippet for logging file is given bellow


File Edit Selection View Go Run ···
customer-shopping-price
README.md
template.py
__init__.py ...\logger 1, M
logs.py 1, M
__init__.py ...\exception M
exception.py 1, M
src > logger > __init__.py > ...
main* 0 3 0
Ln 1, Col 1 Spaces: 4 UTF-8 CRLF Python 3.8.0 (conda)
5:09 PM
9/12/2023

```python
import logging
import os,sys
from datetime import datetime

LOG_DIR="logs"
LOG_DIR=os.path.join(os.getcwd(),LOG_DIR)

os.makedirs(LOG_DIR,exist_ok=True)
CURRENT_TIME_STAMP=f"{datetime.now().strftime('%Y-%m-%d-%H-%M-%S')}"
file_name=f"log_{CURRENT_TIME_STAMP}.log"
log_file_path= os.path.join(LOG_DIR,file_name)

logging.basicConfig(filename=log_file_path,
                    filemode='w',
                    format='[%(asctime)s] %(name)-15s %(levelname)-8s %(message)s',
                    level=logging.INFO
                    )
```

5.2. Created a testing script (e.g., `tests.py`) in your project directory to test the logging setup.

Code snippet for testing logging file

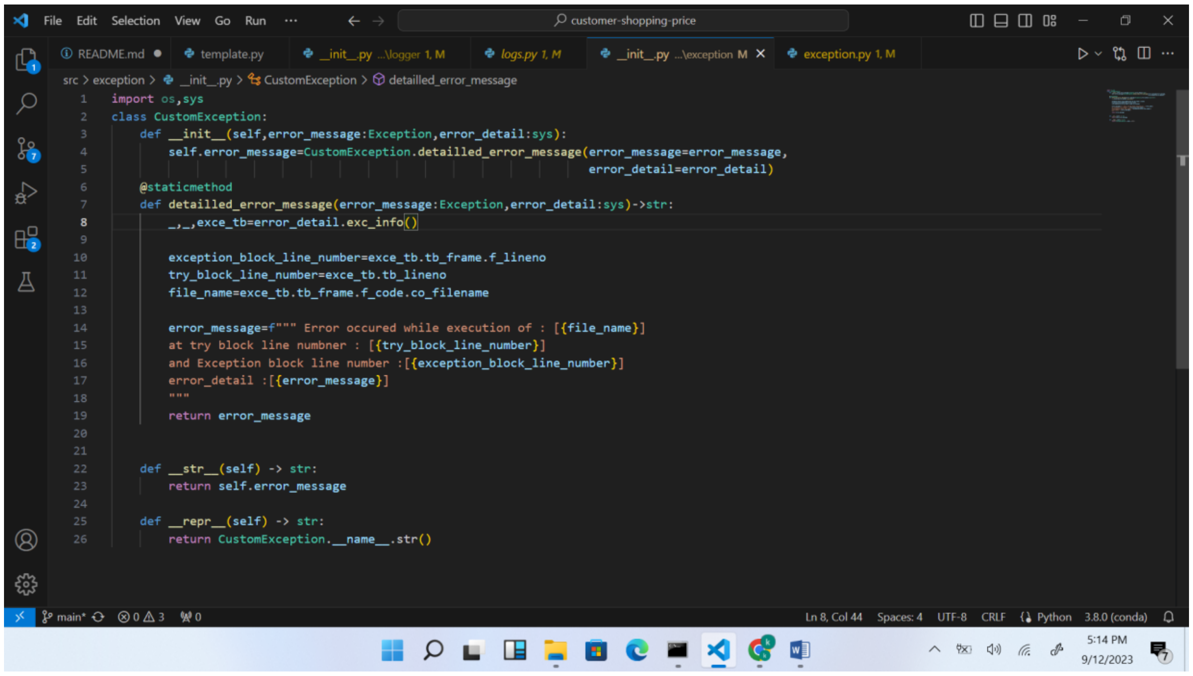


```python
import streamlit as st
from src.logger import logging

st.title('Logging  Testing')

def logging_test():
    logging.info('We are testing logging file')

if __name__ == '__main__':
    logging_test()
    st.write("Logging test succeeded")
```

- **Step 6: Configuring Exception and Testing**

6.1 Created a `exception ` file (e.g., `exception.py`) in your project directory to handle exception configurations.

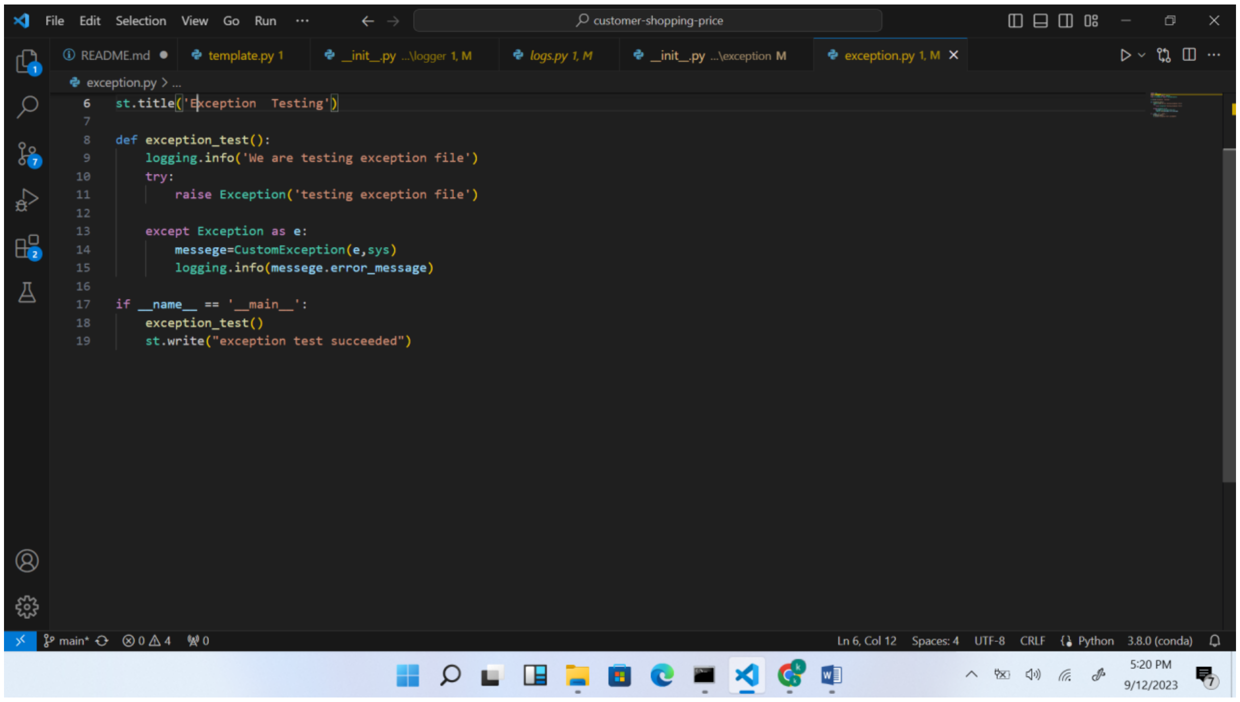
Code snippet for exception handling



```python
import os,sys
class CustomException:
    def __init__(self,error_message:Exception,error_detail:sys):
        self.error_message=CustomException.detailled_error_message(error_message=error_message,
                                                                  error_detail=error_detail)
    @staticmethod
    def detailled_error_message(error_message:Exception,error_detail:sys)->str:
        _,_,exce_tb=error_detail.exc_info()

        exception_block_line_number=exce_tb.tb_frame.f_lineno
        try_block_line_number=exce_tb.tb_lineno
        file_name=exce_tb.tb_frame.f_code.co_filename

        error_message=f""" Error occured while execution of : [{file_name}]
        at try block line numbner : [{try_block_line_number}]
        and Exception block line number :[{exception_block_line_number}]
        error_detail :[{error_message}]
        """
        return error_message


    def __str__(self) -> str:
        return self.error_message

    def __repr__(self) -> str:
        return CustomException.__name__.str()
```

6.2. Created a testing script (e.g., `exception.py`) in your project directory to test the exception setup.

Code Snippet :



```python
st.title('Exception  Testing')

def exception_test():
    logging.info('We are testing exception file')
    try:
        raise Exception('testing exception file')

    except Exception as e:
        messege=CustomException(e,sys)
        logging.info(messege.error_message)

if __name__ == '__main__':
    exception_test()
    st.write("exception test succeeded")
```