Write code to fullfil the following requirements. 6 functions plus main It asks the directory from the user and zips all files in it

GitHub Copilot

Sure, here's a Python script that fulfills your requirements. It includes six functions plus the main function to ask the directory from the user and zip all files in it.

```python
import os
import zipfile

def get_directory():
    """Ask the user for the directory to zip."""
    return input("Enter the directory to zip: ")

def list_files(directory):
    """List all files in the given directory."""
    files = []
    for root, _, filenames in os.walk(directory):
        for filename in filenames:
            files.append(os.path.join(root, filename))
    return files

def create_zip_filename(directory):
    """Create a zip filename based on the directory name."""
    return os.path.basename(os.path.normpath(directory)) + '.zip'

def zip_files(files, zip_filename):
    """Zip all files into the given zip filename."""
    with zipfile.ZipFile(zip_filename, 'w') as zipf:
        for file in files:
            zipf.write(file, os.path.relpath(file, os.path.dirname(files[0])))
```

```python
def confirm_zip_creation(zip_filename):
    """Confirm that the zip file was created."""
    return os.path.exists(zip_filename)

def print_confirmation(zip_filename):
    """Print confirmation message."""
    print(f"Zip file '{zip_filename}' created successfully.")

def main():
    directory = get_directory()
    files = list_files(directory)
    zip_filename = create_zip_filename(directory)
    zip_files(files, zip_filename)
    if confirm_zip_creation(zip_filename):
        print_confirmation(zip_filename)
    else:
        print("Failed to create the zip file.")

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

Add a function to name the zip file

zip_example.py:1-23

GitHub Copilot

> Used 1 reference

Sure, I'll add a function to name the zip file based on user input. Here's the updated zip_example.py :

```python
import os
import zipfile

def get_directory():
    """Ask the user for the directory to zip."""
    return input("Enter the directory to zip: ")

def list_files(directory):
    """List all files in the given directory."""
    files = []
    for root, _, filenames in os.walk(directory):
        for filename in filenames:
            files.append(os.path.join(root, filename))
    return files
```

```python
def create_zip_filename(directory):
    """Create a zip filename based on the directory name."""
    return os.path.basename(os.path.normpath(directory)) + '.zip'

def name_zip_file():
    """Ask the user for the name of the output zip file."""
    zip_filename = input("Enter the name of the output .zip file: ")
    if not zip_filename.endswith('.zip'):
        zip_filename += '.zip'
    return zip_filename

def zip_files(files, zip_filename):
    """Zip all files into the given zip filename."""
    with zipfile.ZipFile(zip_filename, 'w') as zipf:
        for file in files:
            zipf.write(file, os.path.relpath(file, os.path.dirname(files[0])))

def confirm_zip_creation(zip_filename):
    """Confirm that the zip file was created."""
    return os.path.exists(zip_filename)

def print_confirmation(zip_filename):
    """Print confirmation message."""
    print(f"Zip file '{zip_filename}' created successfully.")

def main():
    directory = get_directory()
    files = list_files(directory)
    zip_filename = name_zip_file()
    zip_files(files, zip_filename)
    if confirm_zip_creation(zip_filename):
        print_confirmation(zip_filename)
```

```python
def confirm_zip_creation(zip_filename):
    """Confirm that the zip file was created."""
    return os.path.exists(zip_filename)

def print_confirmation(zip_filename):
    """Print confirmation message."""
    print(f"Zip file '{zip_filename}' created successfully.")

def main():
    directory = get_directory()
    files = list_files(directory)
    zip_filename = name_zip_file()
    zip_files(files, zip_filename)
    if confirm_zip_creation(zip_filename):
        print_confirmation(zip_filename)
    else:
        print("Failed to create the zip file.")

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

rewite all of this file to work with To_Zip. Then remove the duplicate code

To_Zip.py test_To_Zip.py:1-22

GitHub Copilot

> Used 2 references

Sure, I'll rewrite the TO_ZIP.py file to work with the existing functions and remove any duplicate code. Here's the updated TO_ZIP.py :

```python
# Team member names
# Date
# CptS 111, Fall 2024
# Name of project
# Brief description and/or purpose of project; Copilot for code and comments
# Modules used

import os
import zipfile
from Unused_Material.lala import create_zip_file, check_directory_exists, collect_files, add_files_to_zi

def get_directory():
    """Ask the user for the directory to zip."""
    return input("Enter the directory to zip: ")

def create_zip_filename(directory):
    """Create a zip filename based on the directory name."""
    return os.path.basename(os.path.normpath(directory)) + '.zip'
```

```python
def name_zip_file():
    """Ask the user for the name of the output zip file."""
    zip_filename = input("Enter the name of the output .zip file: ")
    if not zip_filename.endswith('.zip'):
        zip_filename += '.zip'
    return zip_filename

def zip_files(directory, zip_filename):
    """Zip all files in the given directory into the given zip filename."""
    files = collect_files(directory)
    zipf = create_zip_file(zip_filename)
    if not zipf:
        return
    add_files_to_zip(zipf, directory, files)
    close_zip_file(zipf)

def confirm_zip_creation(zip_filename):
    """Confirm that the zip file was created."""
    return os.path.exists(zip_filename)

def print_confirmation(zip_filename):
    """Print confirmation message."""
    print(f"Zip file '{zip_filename}' created successfully.")

def main():
    directory = get_directory()
    if not check_directory_exists(directory):
        print("Directory does not exist.")
        return
    zip_filename = name_zip_file()
    zip_files(directory, zip_filename)
    if confirm_zip_creation(zip_filename):
```

```python
def main():
    directory = get_directory()
    if not check_directory_exists(directory):
        print("Directory does not exist.")
        return
    zip_filename = name_zip_file()
    zip_files(directory, zip_filename)
    if confirm_zip_creation(zip_filename):
        print_confirmation(zip_filename)
    else:
        print("Failed to create the zip file.")

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

write test cases for testing the output of this file

Test Code

```python
# Hans Anstey and Madelynn Fulmer
# 11/21/2024
# CptS 111, Fall 2024
# Backup directory to .zip
# Creates a zip file of a directory given by the user and allows the user to name it; Copilot for code and comments,this version tests the main file
# Modules used: os, zipfile, unittest, subprocess, To_Zip

import unittest    Gem811 [11 days ago] • added the test file    You, 2 weeks ago • added the test file
import os
import zipfile
import subprocess
from To_Zip import get_directory, list_files, create_zip_filename, name_zip_file, zip_files, confirm_zip_creation, print_confirmation
You, 2 weeks ago | 1 author (You)
class TestToZip(unittest.TestCase):

    def setUp(self):
        # Setup a test directory and a test file
        self.test_dir = 'test_dir'
        self.test_file = 'test_dir/test_file.txt'
        self.output_zip = 'test_dir.zip'
        os.makedirs(self.test_dir, exist_ok=True)
        with open(self.test_file, 'w') as f:
            f.write('This is a test file.')
```

```python
import unittest
import os
import zipfile
import subprocess
from Done.To_Zip import get_directory, list_files, create_zip_filename, name_zip_file, zip_files, confirm_zip_creation, print_confirmation
You, 2 weeks ago | 1 author (You)
class TestToZip(unittest.TestCase):

    def setUp(self):
        # Setup a test directory and a test file
        self.test_dir = 'test_dir'
        self.test_file = 'test_dir/test_file.txt'
        self.output_zip = 'test_dir.zip'
        os.makedirs(self.test_dir, exist_ok=True)
        with open(self.test_file, 'w') as f:
            f.write('This is a test file.')

    def tearDown(self):
        # Clean up the test directory and the output zip file after each test
        if os.path.exists(self.test_dir):
            for root, dirs, files in os.walk(self.test_dir, topdown=False):
                for name in files:
                    os.remove(os.path.join(root, name))
                for name in dirs:
                    os.rmdir(os.path.join(root, name))
            os.rmdir(self.test_dir)
        if os.path.exists(self.output_zip):
            os.remove(self.output_zip)

    def test_directory_creation(self):
        # Test if the directory is created and contains the test file
        self.assertTrue(os.path.exists(self.test_dir))
        self.assertTrue(os.path.exists(self.test_file))

    def test_zip_file_creation(self):
        # Test if the zip file is created
        subprocess.run(['python', 'to_zip.py'], input=f'{self.test_dir}\n{self.output_zip}\n', text=True)
        self.assertTrue(os.path.exists(self.output_zip))
```

```python
    def test_directory_creation(self):
        # Test if the directory is created and contains the test file
        self.assertTrue(os.path.exists(self.test_dir))
        self.assertTrue(os.path.exists(self.test_file))

    def test_zip_file_creation(self):
        # Test if the zip file is created
        subprocess.run(['python', 'to_zip.py'], input=f'{self.test_dir}\n{self.output_zip}\n', text=True)
        self.assertTrue(os.path.exists(self.output_zip))

    def test_zip_file_contents(self):
        # Test if the zip file contains the correct files
        subprocess.run(['python', 'to_zip.py'], input=f'{self.test_dir}\n{self.output_zip}\n', text=True)
        with zipfile.ZipFile(self.output_zip, 'r') as zipf:    Gem811 [11 days ago] • added the test file      You, 2 weeks ago • added the te
            self.assertIn('test_file.txt', zipf.namelist())

    def test_custom_zip_filename(self):
        # Test if the user can specify a custom name for the zip file
        custom_zip = 'custom_name.zip'
        subprocess.run(['python', 'to_zip.py'], input=f'{self.test_dir}\n{custom_zip}\n', text=True)
        self.assertTrue(os.path.exists(custom_zip))
        if os.path.exists(custom_zip):
            os.remove(custom_zip)

    def test_confirmation_message(self):
        # Test if the confirmation message is printed correctly
        result = subprocess.run(['python', 'to_zip.py'], input=f'{self.test_dir}\n{self.output_zip}\n', text=True, capture_output=True)
        self.assertIn(f"Zip file '{self.output_zip}' created successfully.", result.stdout)

if __name__ == '__main__':
    unittest.main()
```

Test Code results

```
Enter the directory to zip: C:\Users\hans\OneDrive\Desktop\111_Final_Project2
Enter the name of the output .zip file: Test
Zip file 'Test.zip' created successfully.
.Enter the directory to zip: Enter the name of the output .zip file: Zip file 'custom_name.zip' created successfully.
..Enter the directory to zip: Enter the name of the output .zip file: Zip file 'test_dir.zip' created successfully.
.Enter the directory to zip: Enter the name of the output .zip file: Zip file 'test_dir.zip' created successfully.
.
----------------------------------------------------------------------
Ran 5 tests in 0.421s

OK
```

Finished main file code

```python
# Hans Anstey and Madelynn Fulmer
# 11/21/2024
# CptS 111, Fall 2024    Gem811 [11 days ago] • Added header template      You, 2 weeks ago • Added header template
# Backup directory to .zip
# Creates a zip file of a directory given by the user and allows the user to name it; Copilot for code and comments
# Modules used: os and zipfile

import os
import zipfile

def get_directory():
    """Ask the user for the directory to zip."""
    return input("Enter the directory to zip: ")

def list_files(directory):
    """List all files in the given directory."""
    files = []
    for root, _, filenames in os.walk(directory):
        for filename in filenames:
            files.append(os.path.join(root, filename))
    return files

def create_zip_filename(directory):
    """Create a zip filename based on the directory name."""
    return os.path.basename(os.path.normpath(directory)) + '.zip'

def name_zip_file():
    """Ask the user for the name of the output zip file."""
    zip_filename = input("Enter the name of the output .zip file: ")
    if not zip_filename.endswith('.zip'):
        zip_filename += '.zip'
    return zip_filename

def zip_files(files, zip_filename):
    """Zip all files into the given zip filename."""
    with zipfile.ZipFile(zip_filename, 'w') as zipf:
        for file in files:
            zipf.write(file, os.path.relpath(file, os.path.dirname(files[0])))
```

```python
def name_zip_file():
    """Ask the user for the name of the output zip file."""
    zip_filename = input("Enter the name of the output .zip file: ")
    if not zip_filename.endswith('.zip'):
        zip_filename += '.zip'
    return zip_filename

def zip_files(files, zip_filename):
    """Zip all files into the given zip filename."""
    with zipfile.ZipFile(zip_filename, 'w') as zipf:
        for file in files:
            zipf.write(file, os.path.relpath(file, os.path.dirname(files[0])))

def confirm_zip_creation(zip_filename):
    """Confirm that the zip file was created."""
    return os.path.exists(zip_filename)

def print_confirmation(zip_filename):
    """Print confirmation message."""
    print(f"Zip file '{zip_filename}' created successfully.")

def main():
    directory = get_directory()
    files = list_files(directory)
    zip_filename = name_zip_file()
    if zip_filename == '':
        zip_filename = create_zip_filename(directory)
    zip_files(files, zip_filename)    Gem811 [2 weeks ago] • Moved all to test_file
    if confirm_zip_creation(zip_filename):
        print_confirmation(zip_filename)
    else:
        print("Failed to create the zip file.")

main()
```