# Python Code Example: Workflow control and looping

A simple Python code that uses a loop and a condition:

numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

even\_numbers = []

for number in numbers:

if number % 2 == 0:

even\_numbers.append(number)

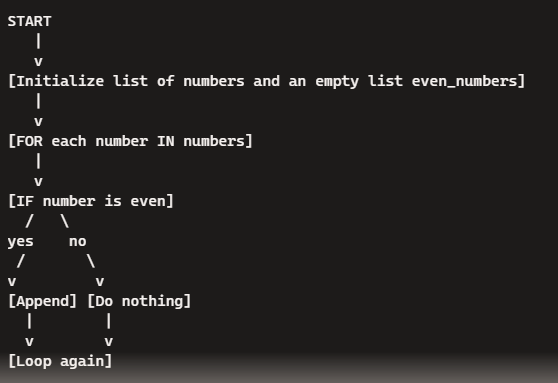
print("Even numbers:", even\_numbers)

**Explanation**

* **Loop**: This code uses a for loop to iterate over each number in the numbers list.
* **Condition**: Inside the loop, it checks if the number is even using if number % 2 == 0.
* **Action**: If the condition is true, it appends the number to the even\_numbers list.

**Diagram**

Here’s a flowchart diagram to visualize the loop and condition:



**Detailed Steps**

1. **Initialize**: The numbers list is created with values from 1 to 10. An empty list even\_numbers is also created.
2. **Iteration**: The for loop starts, iterating over each element in numbers.
3. **Condition Check**: For each number, the if statement checks if it’s even (number % 2 == 0).
4. **Action**: If the condition is true, the number is added to the even\_numbers list. If false, it moves to the next number.
5. **Output**: After the loop completes, the list of even numbers is printed.

A starting point for understanding loops and conditions in Python.

# I/O

A Python script that reads data from a file named gps in the data folder, processes it to find even numbers, and writes the output to a new file named gpscleaned.

**Python Code**

# Read from the input file, process data, and write to an output file

def read\_file(file\_path):

with open(file\_path, 'r') as file:

data = file.readlines()

return data

def process\_data(data):

even\_numbers = []

for line in data:

number = int(line.strip())

if number % 2 == 0:

even\_numbers.append(number)

return even\_numbers

def write\_file(file\_path, data):

with open(file\_path, 'w') as file:

for number in data:

file.write(f"{number}\n")

# Define file paths

input\_file\_path = './data/gpsAll.csv'

output\_file\_path = './data/gpsSelected.csv'

# Read, process, and write data

data = read\_file(input\_file\_path)

even\_numbers = process\_data(data)

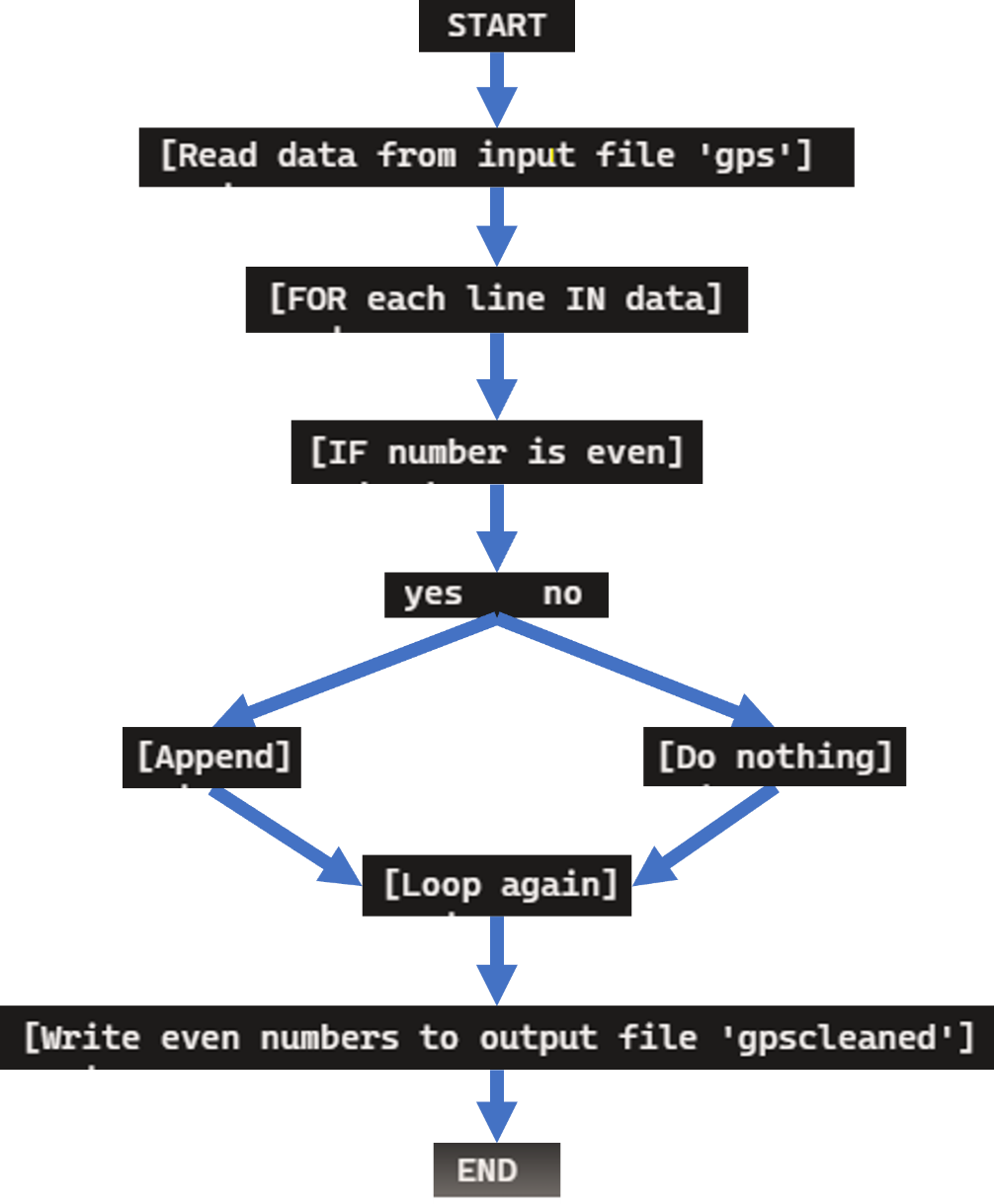
write\_file(output\_file\_path, even\_numbers)

print("Processing complete. Check the gpscleaned file for the results.")

**Explanation**

1. **Read File**: The read\_file function reads all lines from the input file gps in the data folder and returns them as a list.
2. **Process Data**: The process\_data function iterates over each line, converts it to an integer, checks if it’s even, and if so, appends it to the even\_numbers list.
3. **Write File**: The write\_file function writes the processed even numbers to a new file gpscleaned in the data folder.

**Flowchart**

This script ensures you read from a file, process it to find even numbers, and write the output to a new file.

In gpsAll.csv, you have a mix of even and odd numbers. gpsselected.csv includes only the even numbers extracted from gps.csv.

import os

import pandas as pd

import geopandas as gpd

from shapely.geometry import Point

# Define the directory containing the CSV files and the output directory for the shapefile

input\_directory = './fielddat'

output\_directory = './data\_vector'

output\_filename = 'gps\_data.shp'

# Initialize an empty list to store the data

data\_frames = []

# Loop through the CSV files in the input directory

for filename in os.listdir(input\_directory):

if filename.endswith('.csv'):

file\_path = os.path.join(input\_directory, filename)

# Read the CSV file into a pandas DataFrame

df = pd.read\_csv(file\_path)

# Ensure the columns exist and are correctly named

required\_columns = ['nr', 'x', 'y', 'coordinate\_reference\_system', 'land\_cover\_type', 'contact\_person', 'accuracy']

if all(col in df.columns for col in required\_columns):

# Append the DataFrame to the list

data\_frames.append(df)

else:

print(f"Warning: {filename} is missing required columns and will be skipped.")

# Concatenate all DataFrames into a single DataFrame

all\_data = pd.concat(data\_frames, ignore\_index=True)

# Convert the DataFrame to a GeoDataFrame

geometry = [Point(xy) for xy in zip(all\_data['x'], all\_data['y'])]

geo\_df = gpd.GeoDataFrame(all\_data, geometry=geometry)

# Set the coordinate reference system (CRS) for the GeoDataFrame

crs = all\_data['coordinate\_reference\_system'].iloc[0] # Assuming all files have the same CRS

geo\_df.set\_crs(crs, inplace=True)

# Save the GeoDataFrame to a shapefile

output\_path = os.path.join(output\_directory, output\_filename)

geo\_df.to\_file(output\_path, driver='ESRI Shapefile')

print(f"Shapefile created and saved at {output\_path}")

**Explanation**

1. **Directory Setup**: The script specifies the input (./fielddat) and output directories (./data\_vector).
2. **Reading CSV Files**: It reads all CSV files in the input directory and ensures they have the required columns.
3. **Data Aggregation**: It concatenates all DataFrames into a single DataFrame.
4. **GeoDataFrame Creation**: It converts the DataFrame to a GeoDataFrame with the correct coordinate reference system.
5. **Shapefile Saving**: It saves the GeoDataFrame as a shapefile in the specified output directory.

**Usage**

* **Ensure Dependencies**: Make sure you have pandas, geopandas, and shapely installed:

pip install pandas geopandas shapely

* **Run the Script**: Execute the script in your environment to generate the shapefile from your CSV data.