DSA2025Test

March 30, 2025

[1]:
```python
# Initialize Otter
import otter
grader = otter.Notebook("DSA2025Test.ipynb")
```

```
---------------------------------------------------------------------------
ModuleNotFoundError                       Traceback (most recent call last)
Cell In[1], line 2
      1 # Initialize Otter
----> 2 import otter
      3 grader = otter.Notebook("DSA2025Test.ipynb")

ModuleNotFoundError: No module named 'otter'
```

1 DSA 2025 Summer School Admittance Check

Thanks for your interest in attending DSA 2025 Ibadan, Nigeria. To attend the summer school you have to have some level of basic Python proficiency. Completing the following notebook should ensure you have the right kind of background to benefit maximally from the Summer School. See you in Ibadan!

1.1 Instructions

1. Complete each function according to the provided specifications
2. Run all cells to verify your solutions
3. All tests must pass to generate a submission
4. Save your work before submitting

[1]:
```python
# Run these once ... To be on a safe side
!pip install nose
!pip install otter-grader
import IPython
from IPython import get_ipython
# Import the good stuff
import pandas as pd
import numpy as np
import math
from nose.tools import assert_equal
```

```python
import otter
from collections import Counter

grader = otter.Notebook()
```

```
Requirement already satisfied: nose in c:\users\homepc\daily-ml\venv\lib\site-
packages (1.3.7)


[notice] A new release of pip is available: 24.0 -> 25.0.1
[notice] To update, run: python.exe -m pip install --upgrade pip

[notice] A new release of pip is available: 24.0 -> 25.0.1
[notice] To update, run: python.exe -m pip install --upgrade pip
```

```
Requirement already satisfied: otter-grader in c:\users\homepc\daily-
ml\venv\lib\site-packages (6.1.2)
Requirement already satisfied: click<9.0.0,>=8.1.7 in c:\users\homepc\daily-
ml\venv\lib\site-packages (from otter-grader) (8.1.8)
Requirement already satisfied: dill>=0.3.0 in c:\users\homepc\daily-
ml\venv\lib\site-packages (from otter-grader) (0.3.9)
Requirement already satisfied: fica>=0.4.1 in c:\users\homepc\daily-
ml\venv\lib\site-packages (from otter-grader) (0.4.1)
Requirement already satisfied: ipylab<2.0.0,>=1.0.0 in c:\users\homepc\daily-
ml\venv\lib\site-packages (from otter-grader) (1.0.0)
Requirement already satisfied: ipython in c:\users\homepc\daily-
ml\venv\lib\site-packages (from otter-grader) (9.0.2)
Requirement already satisfied: ipywidgets<9.0.0,>=8.1.5 in
c:\users\homepc\daily-ml\venv\lib\site-packages (from otter-grader) (8.1.5)
Requirement already satisfied: jinja2<4.0,>=3.1 in c:\users\homepc\daily-
ml\venv\lib\site-packages (from otter-grader) (3.1.6)
Requirement already satisfied: jupytext<2.0.0,>=1.16.4 in c:\users\homepc\daily-
ml\venv\lib\site-packages (from otter-grader) (1.16.7)
Requirement already satisfied: nbconvert>=6.0.0 in c:\users\homepc\daily-
ml\venv\lib\site-packages (from nbconvert[webpdf]>=6.0.0; sys_platform !=
"emscripten" and sys_platform != "wasi"->otter-grader) (7.16.6)
Requirement already satisfied: nbformat>=5.0.0 in c:\users\homepc\daily-
ml\venv\lib\site-packages (from otter-grader) (5.10.4)
Requirement already satisfied: pandas>=2.0.0 in c:\users\homepc\daily-
ml\venv\lib\site-packages (from otter-grader) (2.2.3)
Requirement already satisfied: python-on-whales<1.0.0,>=0.72.0 in
c:\users\homepc\daily-ml\venv\lib\site-packages (from otter-grader) (0.76.1)
Requirement already satisfied: pyyaml<7,>=6 in c:\users\homepc\daily-
ml\venv\lib\site-packages (from otter-grader) (6.0.2)
Requirement already satisfied: requests<3.0,>=2.31 in c:\users\homepc\daily-
ml\venv\lib\site-packages (from otter-grader) (2.32.3)
Requirement already satisfied: wrapt<2.0.0,>=1.16.0 in c:\users\homepc\daily-
ml\venv\lib\site-packages (from otter-grader) (1.17.2)
Requirement already satisfied: colorama in c:\users\homepc\daily-
```

```
ml\venv\lib\site-packages (from click<9.0.0,>=8.1.7->otter-grader) (0.4.6)
Requirement already satisfied: docutils in c:\users\homepc\daily-
ml\venv\lib\site-packages (from fica>=0.4.1->otter-grader) (0.21.2)
Requirement already satisfied: sphinx in c:\users\homepc\daily-ml\venv\lib\site-
packages (from fica>=0.4.1->otter-grader) (8.2.3)
Requirement already satisfied: comm>=0.1.3 in c:\users\homepc\daily-
ml\venv\lib\site-packages (from ipywidgets<9.0.0,>=8.1.5->otter-grader) (0.2.2)
Requirement already satisfied: traitlets>=4.3.1 in c:\users\homepc\daily-
ml\venv\lib\site-packages (from ipywidgets<9.0.0,>=8.1.5->otter-grader) (5.14.3)
Requirement already satisfied: widgetsnbextension~=4.0.12 in
c:\users\homepc\daily-ml\venv\lib\site-packages (from
ipywidgets<9.0.0,>=8.1.5->otter-grader) (4.0.13)
Requirement already satisfied: jupyterlab-widgets~=3.0.12 in
c:\users\homepc\daily-ml\venv\lib\site-packages (from
ipywidgets<9.0.0,>=8.1.5->otter-grader) (3.0.13)
Requirement already satisfied: decorator in c:\users\homepc\daily-
ml\venv\lib\site-packages (from ipython->otter-grader) (5.2.1)
Requirement already satisfied: ipython-pygments-lexers in c:\users\homepc\daily-
ml\venv\lib\site-packages (from ipython->otter-grader) (1.1.1)
Requirement already satisfied: jedi>=0.16 in c:\users\homepc\daily-
ml\venv\lib\site-packages (from ipython->otter-grader) (0.19.2)
Requirement already satisfied: matplotlib-inline in c:\users\homepc\daily-
ml\venv\lib\site-packages (from ipython->otter-grader) (0.1.7)
Requirement already satisfied: prompt_toolkit<3.1.0,>=3.0.41 in
c:\users\homepc\daily-ml\venv\lib\site-packages (from ipython->otter-grader)
(3.0.50)
Requirement already satisfied: pygments>=2.4.0 in c:\users\homepc\daily-
ml\venv\lib\site-packages (from ipython->otter-grader) (2.19.1)
Requirement already satisfied: stack_data in c:\users\homepc\daily-
ml\venv\lib\site-packages (from ipython->otter-grader) (0.6.3)
Requirement already satisfied: typing_extensions>=4.6 in c:\users\homepc\daily-
ml\venv\lib\site-packages (from ipython->otter-grader) (4.13.0)
Requirement already satisfied: MarkupSafe>=2.0 in c:\users\homepc\daily-
ml\venv\lib\site-packages (from jinja2<4.0,>=3.1->otter-grader) (3.0.2)
Requirement already satisfied: markdown-it-py>=1.0 in c:\users\homepc\daily-
ml\venv\lib\site-packages (from jupytext<2.0.0,>=1.16.4->otter-grader) (3.0.0)
Requirement already satisfied: mdit-py-plugins in c:\users\homepc\daily-
ml\venv\lib\site-packages (from jupytext<2.0.0,>=1.16.4->otter-grader) (0.4.2)
Requirement already satisfied: packaging in c:\users\homepc\daily-
ml\venv\lib\site-packages (from jupytext<2.0.0,>=1.16.4->otter-grader) (24.2)
Requirement already satisfied: beautifulsoup4 in c:\users\homepc\daily-
ml\venv\lib\site-packages (from nbconvert>=6.0.0->nbconvert[webpdf]>=6.0.0;
sys_platform != "emscripten" and sys_platform != "wasi"->otter-grader) (4.13.3)
Requirement already satisfied: bleach!=5.0.0 in c:\users\homepc\daily-
ml\venv\lib\site-packages (from
bleach[css]!=5.0.0->nbconvert>=6.0.0->nbconvert[webpdf]>=6.0.0; sys_platform !=
"emscripten" and sys_platform != "wasi"->otter-grader) (6.2.0)
Requirement already satisfied: defusedxml in c:\users\homepc\daily-
```

```
ml\venv\lib\site-packages (from nbconvert>=6.0.0->nbconvert[webpdf]>=6.0.0;
sys_platform != "emscripten" and sys_platform != "wasi"->otter-grader) (0.7.1)
Requirement already satisfied: jupyter-core>=4.7 in c:\users\homepc\daily-
ml\venv\lib\site-packages (from nbconvert>=6.0.0->nbconvert[webpdf]>=6.0.0;
sys_platform != "emscripten" and sys_platform != "wasi"->otter-grader) (5.7.2)
Requirement already satisfied: jupyterlab-pygments in c:\users\homepc\daily-
ml\venv\lib\site-packages (from nbconvert>=6.0.0->nbconvert[webpdf]>=6.0.0;
sys_platform != "emscripten" and sys_platform != "wasi"->otter-grader) (0.3.0)
Requirement already satisfied: mistune<4,>=2.0.3 in c:\users\homepc\daily-
ml\venv\lib\site-packages (from nbconvert>=6.0.0->nbconvert[webpdf]>=6.0.0;
sys_platform != "emscripten" and sys_platform != "wasi"->otter-grader) (3.1.3)
Requirement already satisfied: nbclient>=0.5.0 in c:\users\homepc\daily-
ml\venv\lib\site-packages (from nbconvert>=6.0.0->nbconvert[webpdf]>=6.0.0;
sys_platform != "emscripten" and sys_platform != "wasi"->otter-grader) (0.10.2)
Requirement already satisfied: pandocfilters>=1.4.1 in c:\users\homepc\daily-
ml\venv\lib\site-packages (from nbconvert>=6.0.0->nbconvert[webpdf]>=6.0.0;
sys_platform != "emscripten" and sys_platform != "wasi"->otter-grader) (1.5.1)
Requirement already satisfied: playwright in c:\users\homepc\daily-
ml\venv\lib\site-packages (from nbconvert[webpdf]>=6.0.0; sys_platform !=
"emscripten" and sys_platform != "wasi"->otter-grader) (1.51.0)
Requirement already satisfied: fastjsonschema>=2.15 in c:\users\homepc\daily-
ml\venv\lib\site-packages (from nbformat>=5.0.0->otter-grader) (2.21.1)
Requirement already satisfied: jsonschema>=2.6 in c:\users\homepc\daily-
ml\venv\lib\site-packages (from nbformat>=5.0.0->otter-grader) (4.23.0)
Requirement already satisfied: numpy>=1.23.2 in c:\users\homepc\daily-
ml\venv\lib\site-packages (from pandas>=2.0.0->otter-grader) (2.2.4)
Requirement already satisfied: python-dateutil>=2.8.2 in c:\users\homepc\daily-
ml\venv\lib\site-packages (from pandas>=2.0.0->otter-grader) (2.9.0.post0)
Requirement already satisfied: pytz>=2020.1 in c:\users\homepc\daily-
ml\venv\lib\site-packages (from pandas>=2.0.0->otter-grader) (2025.2)
Requirement already satisfied: tzdata>=2022.7 in c:\users\homepc\daily-
ml\venv\lib\site-packages (from pandas>=2.0.0->otter-grader) (2025.2)
Requirement already satisfied: pydantic!=2.0.*,<3,>=2 in c:\users\homepc\daily-
ml\venv\lib\site-packages (from python-on-whales<1.0.0,>=0.72.0->otter-grader)
(2.11.1)
Requirement already satisfied: charset-normalizer<4,>=2 in
c:\users\homepc\daily-ml\venv\lib\site-packages (from
requests<3.0,>=2.31->otter-grader) (3.4.1)
Requirement already satisfied: idna<4,>=2.5 in c:\users\homepc\daily-
ml\venv\lib\site-packages (from requests<3.0,>=2.31->otter-grader) (3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in c:\users\homepc\daily-
ml\venv\lib\site-packages (from requests<3.0,>=2.31->otter-grader) (2.3.0)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\homepc\daily-
ml\venv\lib\site-packages (from requests<3.0,>=2.31->otter-grader) (2025.1.31)
Requirement already satisfied: webencodings in c:\users\homepc\daily-
ml\venv\lib\site-packages (from
bleach!=5.0.0->bleach[css]!=5.0.0->nbconvert>=6.0.0->nbconvert[webpdf]>=6.0.0;
sys_platform != "emscripten" and sys_platform != "wasi"->otter-grader) (0.5.1)
```

```
Requirement already satisfied: tinycss2<1.5,>=1.1.0 in c:\users\homepc\daily-
ml\venv\lib\site-packages (from
bleach[css]!=5.0.0->nbconvert>=6.0.0->nbconvert[webpdf]>=6.0.0; sys_platform !=
"emscripten" and sys_platform != "wasi"->otter-grader) (1.4.0)
Requirement already satisfied: parso<0.9.0,>=0.8.4 in c:\users\homepc\daily-
ml\venv\lib\site-packages (from jedi>=0.16->ipython->otter-grader) (0.8.4)
Requirement already satisfied: attrs>=22.2.0 in c:\users\homepc\daily-
ml\venv\lib\site-packages (from jsonschema>=2.6->nbformat>=5.0.0->otter-grader)
(25.3.0)
Requirement already satisfied: jsonschema-specifications>=2023.03.6 in
c:\users\homepc\daily-ml\venv\lib\site-packages (from
jsonschema>=2.6->nbformat>=5.0.0->otter-grader) (2024.10.1)
Requirement already satisfied: referencing>=0.28.4 in c:\users\homepc\daily-
ml\venv\lib\site-packages (from jsonschema>=2.6->nbformat>=5.0.0->otter-grader)
(0.36.2)
Requirement already satisfied: rpds-py>=0.7.1 in c:\users\homepc\daily-
ml\venv\lib\site-packages (from jsonschema>=2.6->nbformat>=5.0.0->otter-grader)
(0.24.0)
Requirement already satisfied: platformdirs>=2.5 in c:\users\homepc\daily-
ml\venv\lib\site-packages (from jupyter-
core>=4.7->nbconvert>=6.0.0->nbconvert[webpdf]>=6.0.0; sys_platform !=
"emscripten" and sys_platform != "wasi"->otter-grader) (4.3.7)
Requirement already satisfied: pywin32>=300 in c:\users\homepc\daily-
ml\venv\lib\site-packages (from jupyter-
core>=4.7->nbconvert>=6.0.0->nbconvert[webpdf]>=6.0.0; sys_platform !=
"emscripten" and sys_platform != "wasi"->otter-grader) (310)
Requirement already satisfied: mdurl~=0.1 in c:\users\homepc\daily-
ml\venv\lib\site-packages (from markdown-it-
py>=1.0->jupytext<2.0.0,>=1.16.4->otter-grader) (0.1.2)
Requirement already satisfied: jupyter-client>=6.1.12 in c:\users\homepc\daily-
ml\venv\lib\site-packages (from
nbclient>=0.5.0->nbconvert>=6.0.0->nbconvert[webpdf]>=6.0.0; sys_platform !=
"emscripten" and sys_platform != "wasi"->otter-grader) (8.6.3)
Requirement already satisfied: wcwidth in c:\users\homepc\daily-
ml\venv\lib\site-packages (from prompt_toolkit<3.1.0,>=3.0.41->ipython->otter-
grader) (0.2.13)
Requirement already satisfied: annotated-types>=0.6.0 in c:\users\homepc\daily-
ml\venv\lib\site-packages (from pydantic!=2.0.*,<3,>=2->python-on-
whales<1.0.0,>=0.72.0->otter-grader) (0.7.0)
Requirement already satisfied: pydantic-core==2.33.0 in c:\users\homepc\daily-
ml\venv\lib\site-packages (from pydantic!=2.0.*,<3,>=2->python-on-
whales<1.0.0,>=0.72.0->otter-grader) (2.33.0)
Requirement already satisfied: typing-inspection>=0.4.0 in
c:\users\homepc\daily-ml\venv\lib\site-packages (from
pydantic!=2.0.*,<3,>=2->python-on-whales<1.0.0,>=0.72.0->otter-grader) (0.4.0)
Requirement already satisfied: six>=1.5 in c:\users\homepc\daily-
ml\venv\lib\site-packages (from python-dateutil>=2.8.2->pandas>=2.0.0->otter-
grader) (1.17.0)
```

```
Requirement already satisfied: soupsieve>1.2 in c:\users\homepc\daily-
ml\venv\lib\site-packages (from
beautifulsoup4->nbconvert>=6.0.0->nbconvert[webpdf]>=6.0.0; sys_platform !=
"emscripten" and sys_platform != "wasi"->otter-grader) (2.6)
Requirement already satisfied: pyee<13,>=12 in c:\users\homepc\daily-
ml\venv\lib\site-packages (from playwright->nbconvert[webpdf]>=6.0.0;
sys_platform != "emscripten" and sys_platform != "wasi"->otter-grader) (12.1.1)
Requirement already satisfied: greenlet<4.0.0,>=3.1.1 in c:\users\homepc\daily-
ml\venv\lib\site-packages (from playwright->nbconvert[webpdf]>=6.0.0;
sys_platform != "emscripten" and sys_platform != "wasi"->otter-grader) (3.1.1)
Requirement already satisfied: sphinxcontrib-applehelp>=1.0.7 in
c:\users\homepc\daily-ml\venv\lib\site-packages (from
sphinx->fica>=0.4.1->otter-grader) (2.0.0)
Requirement already satisfied: sphinxcontrib-devhelp>=1.0.6 in
c:\users\homepc\daily-ml\venv\lib\site-packages (from
sphinx->fica>=0.4.1->otter-grader) (2.0.0)
Requirement already satisfied: sphinxcontrib-htmlhelp>=2.0.6 in
c:\users\homepc\daily-ml\venv\lib\site-packages (from
sphinx->fica>=0.4.1->otter-grader) (2.1.0)
Requirement already satisfied: sphinxcontrib-jsmath>=1.0.1 in
c:\users\homepc\daily-ml\venv\lib\site-packages (from
sphinx->fica>=0.4.1->otter-grader) (1.0.1)
Requirement already satisfied: sphinxcontrib-qthelp>=1.0.6 in
c:\users\homepc\daily-ml\venv\lib\site-packages (from
sphinx->fica>=0.4.1->otter-grader) (2.0.0)
Requirement already satisfied: sphinxcontrib-serializinghtml>=1.1.9 in
c:\users\homepc\daily-ml\venv\lib\site-packages (from
sphinx->fica>=0.4.1->otter-grader) (2.0.0)
Requirement already satisfied: snowballstemmer>=2.2 in c:\users\homepc\daily-
ml\venv\lib\site-packages (from sphinx->fica>=0.4.1->otter-grader) (2.2.0)
Requirement already satisfied: babel>=2.13 in c:\users\homepc\daily-
ml\venv\lib\site-packages (from sphinx->fica>=0.4.1->otter-grader) (2.17.0)
Requirement already satisfied: alabaster>=0.7.14 in c:\users\homepc\daily-
ml\venv\lib\site-packages (from sphinx->fica>=0.4.1->otter-grader) (1.0.0)
Requirement already satisfied: imagesize>=1.3 in c:\users\homepc\daily-
ml\venv\lib\site-packages (from sphinx->fica>=0.4.1->otter-grader) (1.4.1)
Requirement already satisfied: roman-numerals-py>=1.0.0 in
c:\users\homepc\daily-ml\venv\lib\site-packages (from
sphinx->fica>=0.4.1->otter-grader) (3.1.0)
Requirement already satisfied: executing>=1.2.0 in c:\users\homepc\daily-
ml\venv\lib\site-packages (from stack_data->ipython->otter-grader) (2.2.0)
Requirement already satisfied: asttokens>=2.1.0 in c:\users\homepc\daily-
ml\venv\lib\site-packages (from stack_data->ipython->otter-grader) (3.0.0)
Requirement already satisfied: pure-eval in c:\users\homepc\daily-
ml\venv\lib\site-packages (from stack_data->ipython->otter-grader) (0.2.3)
Requirement already satisfied: pyzmq>=23.0 in c:\users\homepc\daily-
ml\venv\lib\site-packages (from jupyter-
client>=6.1.12->nbclient>=0.5.0->nbconvert>=6.0.0->nbconvert[webpdf]>=6.0.0;
sys_platform != "emscripten" and sys_platform != "wasi"->otter-grader) (26.3.0)
Requirement already satisfied: tornado>=6.2 in c:\users\homepc\daily-
ml\venv\lib\site-packages (from jupyter-
client>=6.1.12->nbclient>=0.5.0->nbconvert>=6.0.0->nbconvert[webpdf]>=6.0.0;
sys_platform != "emscripten" and sys_platform != "wasi"->otter-grader) (6.4.2)
```

Question 1: Write a Python function to return a tuple of even and odd numbers

1. List item
2. List item

given an integer input. Example: For input 10, the output should be ([2, 4, 6, 8, 10], [1, 3, 5, 7, 9]).

[2]:
```python
def even_odd_numbers(n):
    """
    Returns a tuple of even and odd numbers up to n.

    Args:
        n (int): The upper limit.

    Returns:
        tuple: A tuple of two lists: (evens, odds).
    """
    evens = []
    odds = []
    for i in range(1, n+1):
        if i % 2 == 0:
            evens.append(i)
        else:
            odds.append(i)
    return (evens, odds)
```

[3]:
```python
grader.check("q1")
```

[3]:
```
q1 results: All test cases passed!
```

Question 2: Create a dictionary to store the frequency of each word in the paragraph provided. Example: For the paragraph “Data science is fun. Data science is useful.”, the output should be {‘Data’: 2, ‘science’: 2, ‘is’: 2, ‘fun’: 1, ‘useful’: 1}.

[4]:
```python
def word_frequency(paragraph):
    """
    Returns a dictionary of word frequencies in a paragraph.

    Args:
        paragraph (str): The input paragraph.

    Returns:
        dict: A dictionary with words as keys and their frequencies as values.
    """
    # Handle edge case
    if not paragraph:
        return {}

    # Convert to lowercase
    paragraph = paragraph.lower()

    # Remove punctuation and split into words
    import re
    words = re.findall(r'\b\w+\b', paragraph)

    # Count frequencies
    frequency = {}
    for word in words:
        if word in frequency:
            frequency[word] += 1
        else:
            frequency[word] = 1

    return frequency
```

[5]:
```
grader.check("q2")
```

[5]:
```
q2 results: All test cases passed!
```

Question 3: Extract the values from the dictionary above into a list and sort them in descending order. Example: For the dictionary {'a': 3, 'b': 1, 'c': 2}, the output should be [3, 2, 1]

[6]:
```
def extract_and_sort_values(dictionary):
    """
    Extracts and sorts dictionary values in descending order.

    Args:
        dictionary (dict): The input dictionary.

    Returns:
        list: A list of sorted values in descending order.
    """
    # Handle edge case
    if not dictionary:
        return []

    # Extract values from dictionary
    values = list(dictionary.values())

    # Sort values in descending order
    values.sort(reverse=True)

    return values
```

[7]:
```
grader.check("q3")
```

[7]:
```
q3 results: All test cases passed!
```

Question 4: Merge the two lists (from Questions 3 and 4) into a dictionary where the keys are the sorted keys and the values are the sorted values. Example: For lists ['a', 'b', 'c'] and [3, 2, 1], the output should be {'a': 3, 'b': 2, 'c': 1}

[8]:
```
def merge_lists_to_dict(keys, values):
    """
    Merges two lists into a dictionary.

    Args:
        keys (list): List of keys.
        values (list): List of values.

    Returns:
        dict: A dictionary with keys and values paired.
    """

    # Handle edge case
    if not keys or not values:
        return {}

    # Merge lists into dictionary
    dictionary = dict(zip(keys, values))

    return dictionary
```

[9]:
```
grader.check("q4")
```

[9]:
```
q4 results: All test cases passed!
```

Question 5:Write a function least_common_multiple that takes two inputs a and b and returns the least common multiple of the two numbers. Example: For input (4, 6), the output should be 12

[10]:
```
def least_common_multiple(a, b):
    """
    Returns the least common multiple of two numbers.

    Args:
        a (int): First number.
        b (int): Second number.
    Returns:
        int: The LCM of a and b.
    """

    # Handle edge case
    if a == 0 or b == 0:
        return 0

    # Compute LCM
    lcm = abs(a*b) // math.gcd(a, b)

    return lcm
```

[11]:
```
grader.check("q5")
```

[11]:
```
q5 results: All test cases passed!
```

Question 6:Write a function get_nearest_farthest that takes in a point of interest (x, y) and a list of points [(x1, y1), (x2, y2), ...] and returns the indices of the nearest and farthest points from the point of interest. Example: For pt = (0, 0) and points = [(1, 1), (3, 3), (-1, -1)], the output should be (0, 1)

[12]:
```
def get_nearest_farthest(pt, points):

    """
    Returns the indices of the nearest and farthest points from a point of␣
 ↪interest.

    Args:
        pt (tuple): The point of interest (x, y).
        points (list): List of points [(x1, y1), (x2, y2), ...].

    Returns:
        tuple: Indices of the nearest and farthest points.
    """

    # Handle edge case
    if not points:
        return None, None

    # Compute distances
    distances = [math.sqrt((pt[0]-x)**2 + (pt[1]-y)**2) for x, y in points]

    # Get nearest and farthest points
    nearest = distances.index(min(distances))
    farthest = distances.index(max(distances))
    return nearest, farthest
```

[13]:
```
grader.check("q6")
```

[13]:
```
q6 results: All test cases passed!
```

Question 7:Write a function filter_divisible to return a list of numbers between 0 and a number N that are not perfectly divisible by q.

Hint: If N is negative, use N = 20.

Example: For N = 10 and q = 2, the output should be [1, 3, 5, 7, 9]

[14]:
```
def filter_divisible(N, q):
    """
    Returns a list of numbers between 0 and N that are not divisible by q.

    Args:
        N (int): The upper limit.
        q (int): The divisor.

    Returns:
        list: A list of non-divisible numbers.
    """

    # Handle edge case
    if N == 0:
        return []

    # Filter numbers
    numbers = [i for i in range(N) if i % q != 0]

    return numbers
```

[15]:
```
grader.check("q7")
```

[15]:
```
q7 results: All test cases passed!
```

Question 8: Write a function flatten_and_unique that takes in a list of lists and outputs a sorted list of unique elements from all sublists.

[16]:
```
def flatten_and_unique(list_of_lists):
    """
    Flattens a list of lists and returns a sorted list of unique elements.

    Args:
        list_of_lists (list): A list of lists.

    Returns:
        list: A sorted list of unique elements.
    """

    # Handle edge case
    if not list_of_lists:
        return []

    # Flatten list of lists
    flat_list = [item for sublist in list_of_lists for item in sublist]

    # Get unique elements
    unique = list(set(flat_list))

    # Sort unique elements
    unique.sort()

    return unique
```

[17]:
```
grader.check("q8")
```

[17]:
```
q8 results: All test cases passed!
```

The Extra Mile!

Download the dataset Nigeria Food Prices (9.9M)

https://data.humdata.org/dataset/wfp-food-prices-for-nigeria

Question 9: Create a new column date_new from the date column, converting it to a datetime format.

[18]:
```
import pandas as pd

# Load the dataset
file_path = 'wfp_food_prices_nga.csv'  # Replace with the actual file path
df = pd.read_csv(file_path,skiprows=[1])
df = df.drop(index=2)
df = df.reset_index(drop=True)

# Display the first few rows of the dataset
print("First 5 rows of the dataset:")
print(df.head())
```

```
First 5 rows of the dataset:
         date   admin1 admin2        market  latitude  longitude  \
0  2002-01-15  Katsina  Jibia   Jibia (CBM)    13.080      7.240
1  2002-01-15  Katsina  Jibia   Jibia (CBM)    13.080      7.240
2  2002-01-15  Katsina  Jibia   Jibia (CBM)    13.080      7.240
3  2002-01-15  Katsina  Jibia   Jibia (CBM)    13.080      7.240
4  2002-01-15   Sokoto   Gada  Illela (CBM)    13.645      5.278
```

```
             category      commodity unit priceflag  pricetype currency  \
0  cereals and tubers          Maize   KG    actual  Wholesale      NGN
1  cereals and tubers         Millet   KG    actual  Wholesale      NGN
2  cereals and tubers        Sorghum   KG    actual  Wholesale      NGN
3     pulses and nuts  Beans (niebe)   KG    actual  Wholesale      NGN
4  cereals and tubers          Maize   KG    actual  Wholesale      NGN

    price  usdprice
0  175.92    1.5398
1  150.18    1.3145
2  155.61    1.3620
3  196.87    1.7231
4  153.35    1.3422
```

[19]:
```
def create_date_new(df):
    """
    Creates a new column `date_new` from the `date` column, converting it to a␣
 ↪datetime format.

    Args:
        df (pd.DataFrame): The input dataframe.

    Returns:
        pd.DataFrame: The dataframe with the new `date_new` column.
    """

    # Handle edge case
    if 'date' not in df.columns:
        return df

    # Convert to datetime
    df['date_new'] = pd.to_datetime(df['date'])

    return df
```

[20]:
```
grader.check("q9")
```

[20]:
```
q9 results: All test cases passed!
```

Question 10:Split the dataframe into two separate dataframes based on the category column: one for cereals and tubers and another for pulses and nuts

[21]:
```
def split_dataframes(df):
    """
    Splits the dataframe into two separate dataframes based on the `category`␣
 ↪column.
```

```
    Args:
        df (pd.DataFrame): The input dataframe.

    Returns:
        tuple: A tuple of two dataframes: (cereals_df, pulses_df).
    """
    global cereals_df, pulses_df  # Make them global to survive separate test␣
↪runs

    if df is None or df.empty or 'category' not in df.columns:
        cereals_df, pulses_df = pd.DataFrame(), pd.DataFrame()
        return cereals_df, pulses_df

    cereals_df = df[df['category'] == 'cereals and tubers'].copy()
    pulses_df = df[df['category'] == 'pulses and nuts'].copy()

    return cereals_df, pulses_df
```

[22]:
```
grader.check("q10")
```

[22]:
```
q10 results: All test cases passed!
```

Question 11:Calculate the mean, median, and mode of the price and usdprice columns for each category

[23]:
```
def calculate_price_stats(df):
    """
    Calculates the mean, median, and mode of the `price` and `usdprice`
    columns for each `category`.

    Returns: (price_stats, usdprice_stats)
    """
    import pandas as pd
    required_cols = {'category', 'price', 'usdprice'}

    global price_stats, usdprice_stats  # Ensure availability across separate␣
↪checks

    if df is None or df.empty or not required_cols.issubset(df.columns):
        price_stats, usdprice_stats = pd.DataFrame(), pd.DataFrame()
        return price_stats, usdprice_stats

    # Helper to compute stats
    def compute_stats(df_group, col):
        agg_df = df_group[col].agg(['mean', 'median'])
        mode_vals = df_group[col].agg(lambda x: x.mode().iloc[0] if not x.
↪mode().empty else None)
```

```
        agg_df['mode'] = mode_vals
        return agg_df.reset_index()

    # Group by category
    grouped = df.groupby('category')
    price_stats = compute_stats(grouped, 'price')
    usdprice_stats = compute_stats(grouped, 'usdprice')

    # Rename columns
    price_stats.columns = ['category', 'Mean', 'Median', 'Mode']
    usdprice_stats.columns = ['category', 'Mean', 'Median', 'Mode']

    return price_stats, usdprice_stats
```

[24]:
```
grader.check("q11")
```

[24]:
```
q11 results: All test cases passed!
```

Question 12:Merge the two dataframes (from Question 11) back into one dataframe and save it as `merged_data.csv`.

[25]:
```
def merge_dataframes(cereals_df, pulses_df):
    """
    Merges the two dataframes back into one dataframe
    and saves it as `merged_data.csv`.

    Args:
        cereals_df (pd.DataFrame): The cereals and tubers dataframe.
        pulses_df (pd.DataFrame): The pulses and nuts dataframe.

    Returns:
        pd.DataFrame: The merged dataframe.
    """
    import pandas as pd

    # Make merged_df global so it persists across tests
    global merged_df

    # Handle edge cases
    if cereals_df.empty and pulses_df.empty:
        merged_df = pd.DataFrame()
        merged_df.to_csv('merged_data.csv', index=False)
        return merged_df

    # Merge DataFrames
    merged_df = pd.concat([cereals_df, pulses_df], ignore_index=True)

    # Save to CSV
    merged_df.to_csv('merged_data.csv', index=False)

    return merged_df
```

```
[26]: grader.check("q12")
```

```
[26]: q12 results: All test cases passed!
```

Question 13:Open the merged_data.csv file and select only the date_new, market, commodity, price, and usdprice columns

```
[27]: def select_columns():
          """
          Opens the `merged_data.csv` file and selects only the `date_new`, `market`,␣
       ↪`commodity`,
          `price`, and `usdprice` columns.

          Returns:
              pd.DataFrame: The dataframe with selected columns.
          """

          global selected_columns  # Use the name the test expects

          # Read merged data
          df = pd.read_csv('merged_data.csv')

          # Select specified columns
          required_columns = ["date_new", "market", "commodity", "price", "usdprice"]
          selected_columns = df[required_columns].copy()

          return selected_columns
```

```
[28]: grader.check("q13")
```

```
[28]: q13 results: All test cases passed!
```

Question 14:Group the data by admin1 (state) and calculate the average price and usdprice for each state

```
[29]: def calculate_state_avg_prices(df):
          """
          Groups the data by `admin1` (state) and calculates the average `price` and␣
       ↪`usdprice` for each state.

          Args:
              df (pd.DataFrame): The input dataframe.

          Returns:
```

```
        pd.DataFrame: The dataframe with average prices by state.
    """
    import pandas as pd

    global state_avg_prices  # Make it global so subsequent tests can see it

    # Required columns
    required_cols = {'admin1', 'price', 'usdprice'}
    if df is None or df.empty or not required_cols.issubset(df.columns):
        state_avg_prices = pd.DataFrame()
        return state_avg_prices

    # Group by 'admin1' and compute mean
    state_avg_prices = (
        df
        .groupby('admin1', as_index=False)
        .agg({'price': 'mean', 'usdprice': 'mean'})
    )

    # Keep column names as "price" and "usdprice"
    return state_avg_prices
```

[30]:
```
grader.check("q14")
```

[30]:
```
q14 results: All test cases passed!
```

Question 15:Identify the top 5 markets with the highest average usdprice for Rice (imported).

[31]:
```
def identify_top_markets(df):
    """
    Identifies the top 5 markets with the highest average `usdprice` for `Rice␣
 ↪(imported)`.

    Args:
        df (pd.DataFrame): The input dataframe.

    Returns:
        pd.Series: A series with the top 5 markets and their average `usdprice`.
    """
    import pandas as pd

    global top_markets  # Make available across tests

    required_cols = {'commodity', 'market', 'usdprice'}
    if df is None or df.empty or not required_cols.issubset(df.columns):
        top_markets = pd.Series(dtype=float)
        return top_markets
    # Filter for Rice (imported)
    rice_data = df[df['commodity'] == 'Rice (imported)']
    if rice_data.empty:
        top_markets = pd.Series(dtype=float)
        return top_markets

    # Calculate average usdprice by market
    market_avg = rice_data.groupby('market')['usdprice'].mean()

    # Sort in descending order of usdprice
    market_avg_sorted = market_avg.sort_values(ascending=False)

    # Select the top 5
    top_markets = market_avg_sorted.head(5)

    return top_markets
```

[32]:
```
grader.check("q15")
```

[32]:
```
q15 results: All test cases passed!
```

To double-check your work, the cell below will rerun all of the autograder tests.

[33]:
```
grader.check_all()
```

[33]:
```
q1 results: All test cases passed!

q10 results: All test cases passed!

q11 results: All test cases passed!

q12 results: All test cases passed!

q13 results: All test cases passed!

q14 results: All test cases passed!

q15 results: All test cases passed!

q2 results: All test cases passed!

q3 results: All test cases passed!

q4 results: All test cases passed!
```

```
q5 results: All test cases passed!

q6 results: All test cases passed!

q7 results: All test cases passed!

q8 results: All test cases passed!

q9 results: All test cases passed!
```

1.2 Submission

Make sure you have run all cells in your notebook in order before running the cell below, so that all images/graphs appear in the output. The cell below will generate a zip file for you to submit. **Please save before exporting!**

Complete each function according to the provided specifications. Make sure all tests pass.

[34]:
```python
# Save your notebook first, then run this cell to export your submission.
grader.export(run_tests=True)
```

```
c:\Users\HomePC\Daily-ML\venv\Lib\site-packages\otter\check\notebook.py:494:
UserWarning: Could not locate a PDF to include
  warnings.warn("Could not locate a PDF to include")
```

```
VBox(children=(HTML(value='<p style="margin: 0">A PDF of your notebook could not␣
 ↪be generated. Please acknowle…
```