Team Application Exercises (tAPP)

1 - Slicing

Using a pen and paper, write a programme that asks the user to type in the first line of their favourite song and display the length of that string. The programme should also ask a starting number and an ending number and then display just that section of the text.

```
phrase = input("Enter the first line of your favourite song: ")
length = len(phrase)
print("This has", length, "letters in it")
start = int(input("Enter a starting number: "))
end = int(input("Enter an end number: "))
part = (phrase[start:end])
print(part)
```

2 - **Exercise: Umbrella or no umbrella**

Using a pen and a paper, write a programme that asks the user if it is raining and convert their answer to lower case so that it doesn't matter what case they type in. If they answer "yes", ask if it is windy. If they answer "yes" to the second question, display the answer "It's too windy for an umbrella", otherwise display the message "Take an umbrella". If they did not answer "yes" to the first question, display the answer "Enjoy your day".

```
raining = input("Is it raining? ")
raining = str.lower(raining)
if raining == "yes":
    windy = input("Is it windy? ")
    windy = str.lower(windy)
    if windy == "yes":
        print("It is too windy for an umbrella")
    else:
        print("Take an umbrella")
else:
    print("Enjoy your day")
```

Lab5 Exercises - Solutions

Task 1: Write a Python program to draw a line with suitable label in the x axis, y axis and a title. The values of y should be twice of x. The range of x: 1 to 30.

```
import matplotlib.pyplot as plt
X = range(1, 30)
Y = [value * 2 for value in X]
print("Values of X:")
print(*range(1,30))
print("Values of Y (twice of X):")
print(Y)
# Plot lines and/or markers to the Axes.
plt.plot(X, Y)
# Set the x axis label of the current axis.
plt.xlabel('x - axis')
# Set the y axis label of the current axis.
plt.ylabel('y - axis')
# Set a title
plt.title('Draw a line.')
# Display the figure.
plt.show()
```

Task2: Write a Python program to draw line charts of the financial data of a company between October 3, 2016 to October 7, 2016.

Sample Financial data (fdata.csv) as it appears in the .csv file:
Date,Open,High,Low,Close
10-03-16,774.25,776.065002,769.5,772.559998
10-04-16,776.030029,778.710022,772.890015,776.429993
10-05-16,779.309998,782.070007,775.650024,776.469971
10-06-16,779,780.47998,775.539978,776.859985
10-07-16,779.659973,779.659973,770.75,775.080017

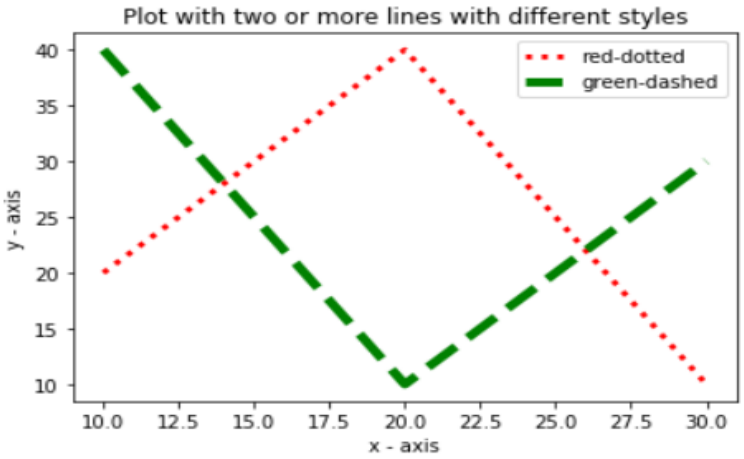
```
import matplotlib.pyplot as plt
import pandas as pd
df = pd.read_csv('fdata.csv', sep=',', parse_dates=True, index_col=0)
df.plot()
plt.show()
```

Task3: Write a Python program to plot two or more lines with legends, different widths and colours.

Sample code:

```
import matplotlib.pyplot as plt
# line 1 points
x1 = [10,20,30,70,30]
y1 = [20,40,10,5,10]
# line 2 points
x2 = [10,20,30,70,30]
y2 = [40,10,30,5,65]
# Set the x axis label of the current axis.
plt.xlabel('x - axis')
# Set the y axis label of the current axis.
plt.ylabel('y - axis')
# Set a title
plt.title('Two or more lines with different widths and colors with suitable legends ')
# Display the figure.
plt.plot(x1,y1, color='blue', linewidth = 3,  label = 'line1-width-3')
plt.plot(x2,y2, color='red', linewidth = 5,  label = 'line2-width-5')
# show a legend on the plot
plt.legend()
plt.show()
```

Task4: Write a program that will produce the following output.




```
import matplotlib.pyplot as plt
# line 1 points
x1 = [10,20,30]
y1 = [20,40,10]
# line 2 points
x2 = [10,20,30]
y2 = [40,10,30]
# Set the x axis label of the current axis.
plt.xlabel('x - axis')
# Set the y axis label of the current axis.
plt.ylabel('y - axis')
# Plot lines and/or markers to the Axes.
plt.plot(x1,y1, color='red', linewidth = 3,  label = 'red-dotted',linestyle='dotted')
plt.plot(x2,y2, color='green', linewidth = 5,  label = 'green-dashed', linestyle='dashed')
# Set a title
plt.title("Plot with two or more lines with different styles")
# show a legend on the plot
plt.legend()
# function to show the plot
plt.show()
```

Task5: Write a Python program to plot quantities which have an x and y position given the following arrays of x and y values:

x1 = [20, 30, 50, 60, 80]

y1 = [10, 50, 100, 180, 200]

x2 = [30, 40, 60, 70, 90]

y2 = [20, 60, 110, 200, 220]

```
import numpy as np
import pylab as pl
# Make an array of x values
x1 = [20, 30, 50, 60, 80]
# Make an array of y values for each x value
y1 = [10, 50, 100, 180, 200]
# Make an array of x values
x2 = [30, 40, 60, 70, 90]
# Make an array of y values for each x value
y2 = [20, 60, 110, 200, 220]
# set new axes limits
pl.axis([0, 100, 0, 230])
# use pylab to plot x and y as red circles
pl.plot(x1, y1,'b*', x2, y2, 'ro')
# show the plot on the screen
pl.show()
```

Task 6: Write a Python program to create multiple plots.

```
import matplotlib.pyplot as plt
fig = plt.figure()
fig.subplots_adjust(bottom=0.020, left=0.020, top = 0.900, right=0.800)

plt.subplot(2, 1, 1)
plt.xticks(()), plt.yticks(())

plt.subplot(2, 3, 4)
plt.xticks(())
plt.yticks(())

plt.subplot(2, 3, 5)
plt.xticks(())
plt.yticks(())

plt.subplot(2, 3, 6)
plt.xticks(())
plt.yticks(())

plt.show()
```

Task 7: Write a Python programming to display a bar chart of the popularity of programming Languages.

Sample data on the popularity of programming languages 2019 according to IEEE:
Programming languages: Python, Java, C, C++, R, JavaScript, C#
Popularity: 100, 96.3, 94.4, 87.5, 81.5, 79.4, 74.5

```
import matplotlib.pyplot as plt
x = ['Python', 'Java', 'C', 'C++', 'R', 'JavaScript', 'C#']
popularity = [100, 96.3, 94.4, 87.5, 81.5, 79.4, 74.5]
x_pos = [i for i, _ in enumerate(x)]
plt.bar(x_pos, popularity, color='blue')
plt.xlabel("Languages")
plt.ylabel("Popularity")
plt.title("PopularitY of Programming Language\n" + "IEEE survey 2019")
plt.xticks(x_pos, x)
# Turn on the grid
plt.minorticks_on()
plt.grid(which='major', linestyle='-', linewidth='0.5', color='red')
# Customize the minor grid
plt.grid(which='minor', linestyle=':', linewidth='0.5', color='black')
plt.show()
```

Task 8: Write a Python programming to display a horizontal bar chart of the popularity of programming Languages

Task 9: Write a Python programming to display a bar chart of the popularity of programming Languages. Use different colour for each bar.

```
import matplotlib.pyplot as plt
x = ['Python', 'Java', 'C', 'C++', 'R', 'JavaScript', 'C#']
popularity = [100, 96.3, 94.4, 87.5, 81.5, 79.4, 74.5]
x_pos = [i for i, _ in enumerate(x)]
plt.bar(x_pos, popularity, color=['red', 'black', 'green', 'blue', 'yellow', 'cyan'])

plt.xlabel("Languages")
plt.ylabel("Popularity")
plt.title("PopularitY of Programming Language\n" + "IEEE survey 2019")
plt.xticks(x_pos, x)
# Turn on the grid
plt.minorticks_on()
plt.grid(which='major', linestyle='-', linewidth='0.5', color='red')
# Customize the minor grid
plt.grid(which='minor', linestyle=':', linewidth='0.5', color='black')
plt.show()
```

Task 10: Write a Python programming to display a bar chart of the popularity of programming Languages. Attach a text label above each bar displaying its popularity (float value)

```
import matplotlib.pyplot as plt
x = ['Python', 'Java', 'C', 'C++', 'R', 'JavaScript', 'C#']
popularity = [100, 96.3, 94.4, 87.5, 81.5, 79.4, 74.5]
x_pos = [i for i, _ in enumerate(x)]

fig, ax = plt.subplots()
rects1 = ax.bar(x_pos, popularity, color=['red', 'black', 'green', 'blue', 'yellow', 'cyan'])
color=['red', 'black', 'green', 'blue', 'yellow', 'cyan']
plt.xlabel("Languages")
plt.ylabel("Popularity")
plt.title("PopularitY of Programming Language\n" + "IEEE survey 2019")
plt.xticks(x_pos, x)
# Turn on the grid
plt.minorticks_on()
plt.grid(which='major', linestyle='-', linewidth='0.5', color='red')
# Customize the minor grid
plt.grid(which='minor', linestyle=':', linewidth='0.5', color='black')

def autolabel(rects):
    """
    Attach a text label above each bar displaying its height
    """
    for rect in rects:
        height = rect.get_height()
        ax.text(rect.get_x() + rect.get_width()/2., 1.05*height,
                '%f' % float(height),
        ha='center', va='bottom')
autolabel(rects1)
plt.show()
```

Task11: Write a Python programming to create a pie chart of the popularity of programming Languages

```
import matplotlib.pyplot as plt

languages = 'Python', 'Java', 'C', 'C++', 'R', 'JavaScript', 'C#'
popularity = [100, 96.3, 94.4, 87.5, 81.5, 79.4, 74.5]
colors = ["#1f77b4", "#ff7f0e", "#2ca02c", "#d62728", "#9467bd","#8c564b","#8c364b"]

# explode 1st slice
explode = (0.1, 0, 0, 0,0,0,0)
# Plot
plt.pie(popularity, explode=explode, labels=languages, colors=colors,
autopct='%1.1f%%', shadow=True, startangle=140)

plt.axis('equal')
plt.show()
```

Task 12: Write a Python programming to create a pie chart with a title of the popularity of programming Languages. Make three wedges of the pie.

```
import matplotlib.pyplot as plt

languages = 'Python', 'Java', 'C', 'C++', 'R', 'JavaScript', 'C#'
popularity = [100, 96.3, 94.4, 87.5, 81.5, 79.4, 74.5]
colors = ["#1f77b4", "#ff7f0e", "#2ca02c", "#d62728", "#9467bd","#8c564b","#8c364b"]

# explode 1st slice
explode = (0.1, 0, .2, 0,.1,0,0)
# Plot
plt.pie(popularity, explode=explode, labels=languages, colors=colors,
autopct='%1.1f%%', shadow=True, startangle=140)

plt.title("PopularitY of Programming Language\n" + "IEEE survey 2019")
plt.show()
```

Task 13: Create bar plot from the following DataFrame:

a b c d e

10,40,39,30,39
80,38,24,33,50
80,36,90,25,44
70,45,30,69,15
25,45,39,30,55

```
from pandas import DataFrame
import matplotlib.pyplot as plt
import numpy as np

a=np.array([[10,40,39,30,39],[80,38,24,33,50],[80,36,90,25,44],[70,45,30,69,15],[25,45,39,30,55]])
df=DataFrame(a, columns=['a','b','c','d','e'], index=[2,4,6,8,10])

df.plot(kind='bar')
# Turn on the grid
plt.minorticks_on()
plt.grid(which='major', linestyle='-', linewidth='0.5', color='green')
plt.grid(which='minor', linestyle=':', linewidth='0.5', color='black')

plt.show()
```

Task 14: Write a Python program to create bar plots with error bars on the same figure.

Sample Date
Mean velocity: 0.2474, 0.1235, 0.1737, 0.1824
Standard deviation of velocity: 0.3314, 0.2278, 0.2836, 0.2645

```
import numpy as np
import matplotlib.pyplot as plt
N = 5
menMeans = (54.74, 42.35, 67.37, 58.24, 30.25)
menStd = (4, 3, 4, 1, 5)
# the x locations for the groups
ind = np.arange(N)
# the width of the bars
width = 0.35
plt.bar(ind, menMeans, width, yerr=menStd, color='red')
plt.ylabel('Scores')
plt.xlabel('Velocity')
plt.title('Scores by Velocity')
# Turn on the grid
plt.minorticks_on()
plt.grid(which='major', linestyle='-', linewidth='0.5', color='green')
plt.grid(which='minor', linestyle=':', linewidth='0.5', color='black')
```

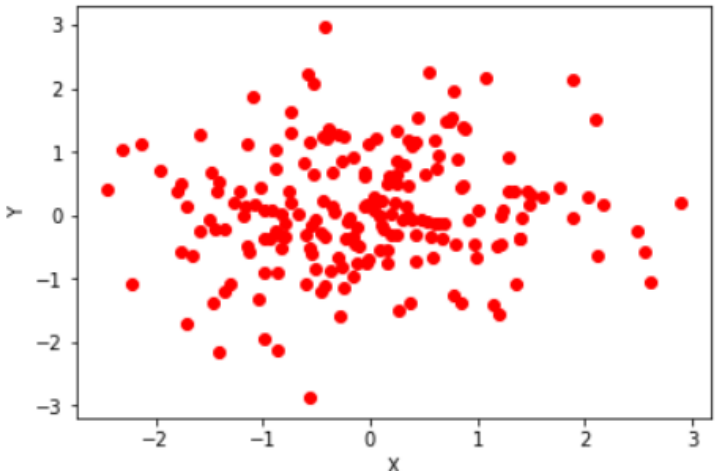
Task 15: Write a Python program to create a stacked bar plot with error bars,
Use bottom to stack the women's bars on top of the men's bars.

Sample Data:
Means (men) = (22, 30, 35, 35, 26)
Means (women) = (25, 32, 30, 35, 29)
Men Standard deviation = (4, 3, 4, 1, 5)
Women Standard deviation = (3, 5, 2, 3, 3)

```
import numpy as np
import matplotlib.pyplot as plt

N = 5
menMeans = (22, 30, 35, 35, 26)
womenMeans = (25, 32, 30, 35, 29)
menStd = (4, 3, 4, 1, 5)
womenStd = (3, 5, 2, 3, 3)
# the x locations for the groups
ind = np.arange(N)
# the width of the bars
width = 0.35

p1 = plt.bar(ind, menMeans, width, yerr=menStd, color='red')
p2 = plt.bar(ind, womenMeans, width,
bottom=menMeans, yerr=womenStd, color='green')

plt.ylabel('Scores')
plt.xlabel('Groups')
plt.title('Scores by group\n' + 'and gender')
plt.xticks(ind, ('Group1', 'Group2', 'Group3', 'Group4', 'Group5'))
plt.yticks(np.arange(0, 81, 10))
plt.legend((p1[0], p2[0]), ('Men', 'Women'))

plt.show()
```

Task 16: Write a Python program to draw a scatter graph taking a random distribution in X and Y and plotted against each other.

```
import matplotlib.pyplot as plt
from pylab import randn
X = randn(200)
Y = randn(200)
plt.scatter(X,Y, color='r')
plt.xlabel("X")
plt.ylabel("Y")
plt.show()
```



Task 17: Write a Python program to draw a scatter plot using random distributions to generate balls of different sizes.

```
import math
import random
import matplotlib.pyplot as plt
# create random data
no_of_balls = 25
x = [random.triangular() for i in range(no_of_balls)]
y = [random.gauss(0.5, 0.25) for i in range(no_of_balls)]
colors = [random.randint(1, 4) for i in range(no_of_balls)]
areas = [math.pi * random.randint(5, 15)**2 for i in range(no_of_balls)]
# draw the plot
plt.figure()
plt.scatter(x, y, s=areas, c=colors, alpha=0.85)
plt.axis([0.0, 1.0, 0.0, 1.0])
plt.xlabel("X")
plt.ylabel("Y")
plt.show()
```

Task 19: Write a Python program to draw a scatter plot comparing two subject marks of Java and Python.

Test Data:
java_marks = [88, 92, 80, 89, 100, 80, 60, 100, 80, 34]
python_marks = [35, 79, 79, 48, 100, 88, 32, 45, 20, 30]

marks_range = [10, 20, 30, 40, 50, 60, 70, 80, 90, 100]

```
import matplotlib.pyplot as plt
import pandas as pd
java_marks = [88, 92, 80, 89, 100, 80, 60, 100, 80, 34]
python_marks = [35, 79, 79, 48, 100, 88, 32, 45, 20, 30]
marks_range = [10, 20, 30, 40, 50, 60, 70, 80, 90, 100]
plt.scatter(marks_range, java_marks, label='Java marks', color='r')
plt.scatter(marks_range, python_marks, label='Python marks', color='g')
plt.title('Scatter Plot')
plt.xlabel('Marks Range')
plt.ylabel('Marks Scored')
plt.legend()
plt.show()
```

Team Application Exercises (tAPP-5)

A	B	C	D	E	F	G	H	I
month_number	facecream	facewash	toothpaste	bathingsoap	shampoo	moisturizer	total_units	total_profit
1	2500	1500	5200	9200	1200	1500	21100	211000
2	2630	1200	5100	6100	2100	1200	18330	183300
3	2140	1340	4550	9550	3550	1340	22470	224700
4	3400	1130	5870	8870	1870	1130	22270	222700
5	3600	1740	4560	7760	1560	1740	20960	209600
6	2760	1555	4890	7490	1890	1555	20140	201400
7	2980	1120	4780	8980	1780	1120	29550	295500
8	3700	1400	5860	9960	2860	1400	36140	361400
9	3540	1780	6100	8100	2100	1780	23400	234000
10	1990	1890	8300	10300	2300	1890	26670	266700
11	2340	2100	7300	13300	2400	2100	41280	412800
12	2900	1760	7400	14400	1800	1760	30020	300200

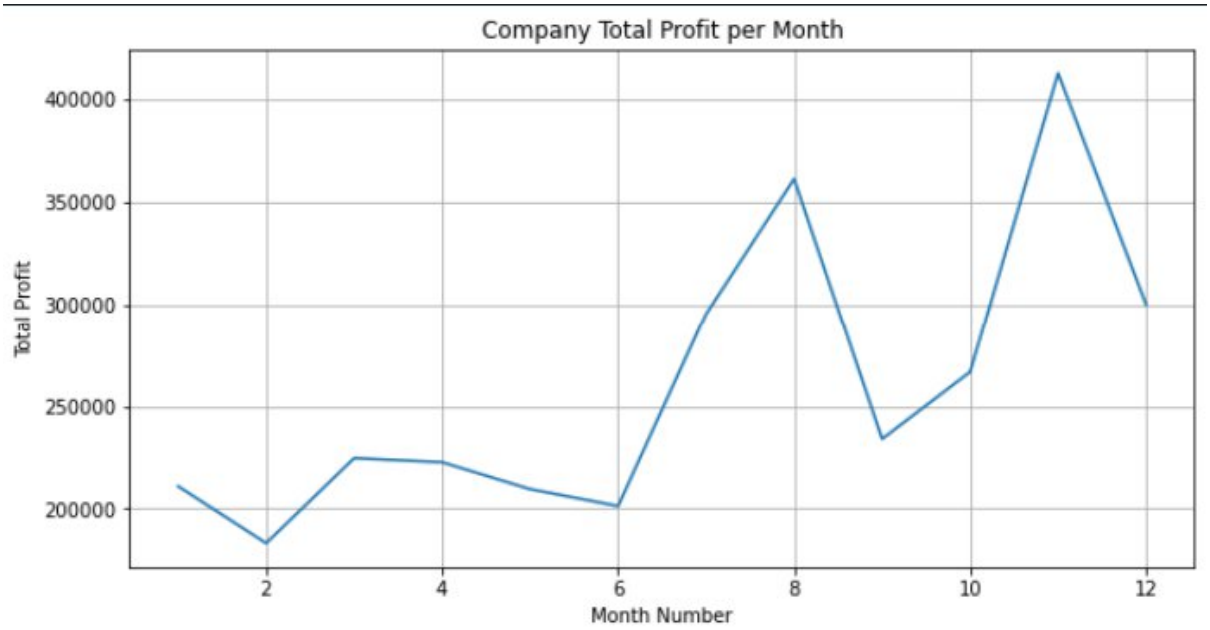
File name: company_sales_data.csv

Task 1: Write a program that reads the "Total profit" of all months and shows it using a line plot.

Total profit data provided for each month. Generated line plot must include the following properties:

- X label name = Month Number
- Y label name = Total profit

The line plot graph should look like this if run.



Solution

figsize parameter within this function specifies the size of the figure in inches.

```
import pandas as pd
import matplotlib.pyplot as plt

# Load the data
data = pd.read_csv('company_sales_data.csv')

# Task 1: Plotting Total Profit vs. Month Number
plt.figure(figsize=(10, 5))
plt.plot(data['month_number'], data['total_profit'], label='Profit')

# Adding labels and title
plt.xlabel('Month Number')
plt.ylabel('Total Profit')
plt.title('Company Total Profit per Month')

# Display the plot
plt.grid(True)
plt.show()
```

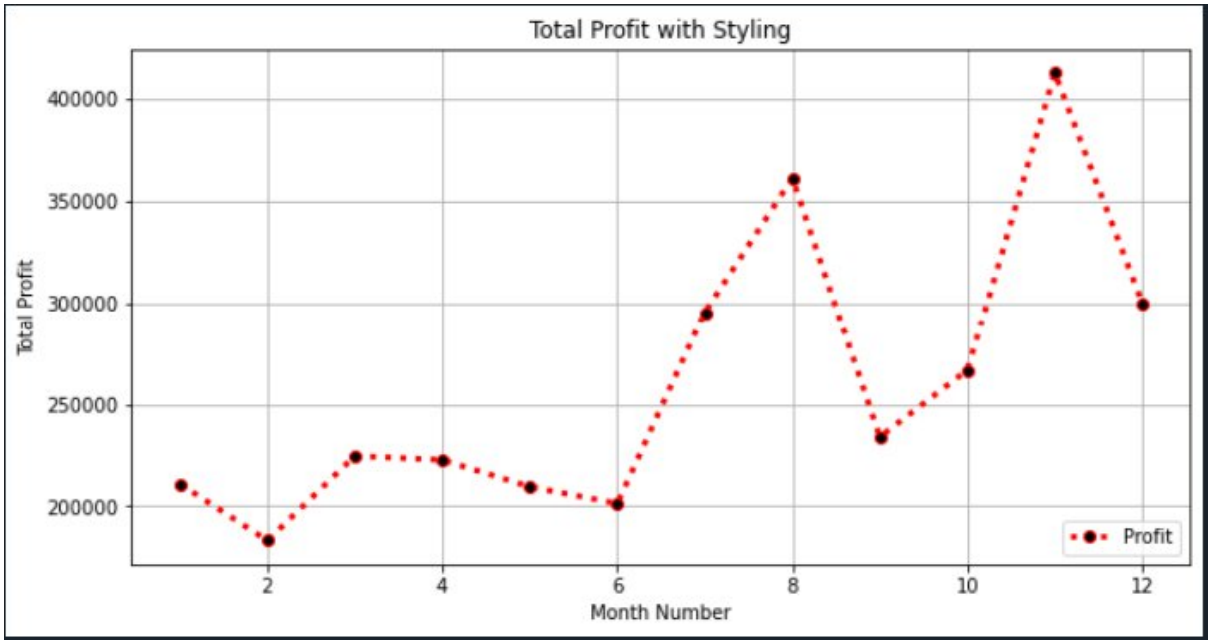
Task 2:

Create a program that will get Total profit of all months and show line plot with the following Style properties.

Generated line plot must include following Style properties: –

- Line Style dotted and Line-color should be red
- Show legend at the lower right location.
- X label name = Month Number
- Y label name = Total Profit
- Add a circle marker.
- Line marker colour as red
- Line width should be 3

The line plot graph should look like this.



Solution:

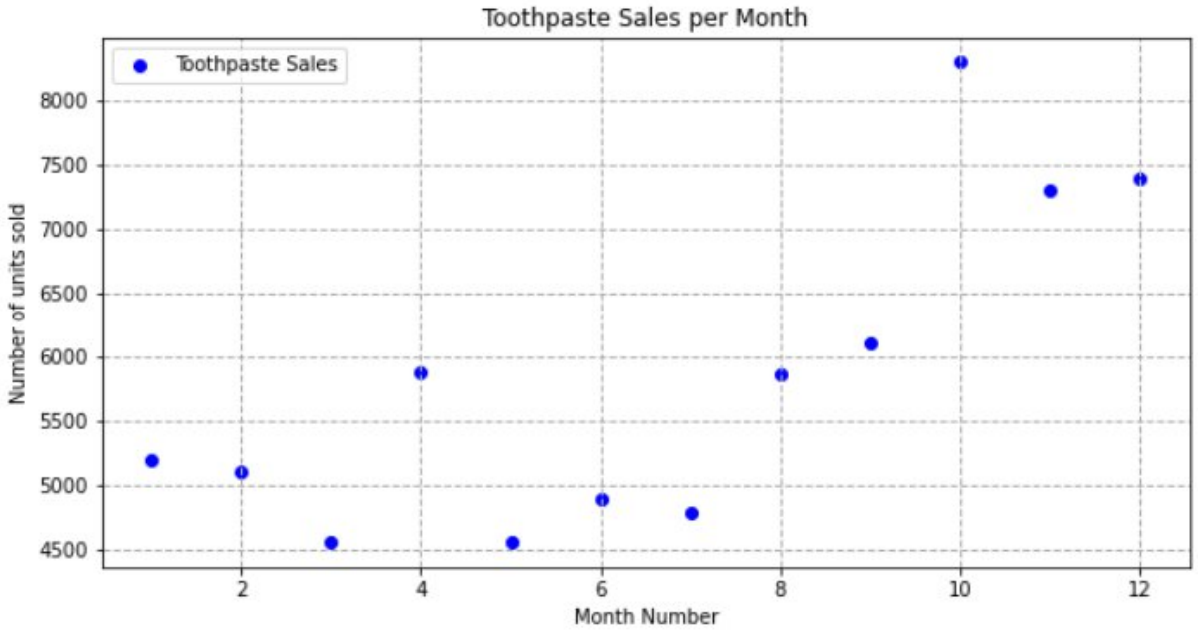
```
# Task 2: Styled Line Plot for Total Profit

plt.figure(figsize=(10, 5))
plt.plot(
    data['month_number'],
    data['total_profit'],
    label='Profit',
    color='red',
    linestyle=':',
    marker='o',
    markerfacecolor='k',
    linewidth=3
)

# Adding labels, title, and legend
plt.xlabel('Month Number')
plt.ylabel('Total Profit')
plt.title('Total Profit with Styling')
plt.legend(loc='lower right')

# Display the plot
plt.grid(True)
plt.show()
```

marker='o': Each data point on the plot is represented by a circle.

markerfacecolor='k: The inside of each circle is filled with black.

Task 3: Write a program that will read toothpaste sales data of each month and show it using a scatter plot. Also, add a grid in the plot. gridline style should be "–"

The graph should look like this when the program is run.

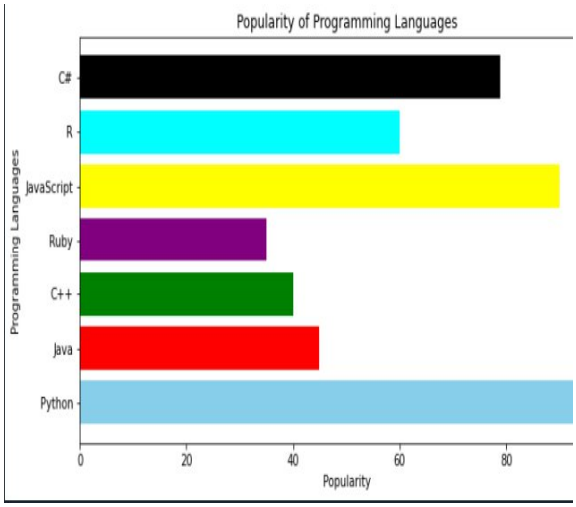


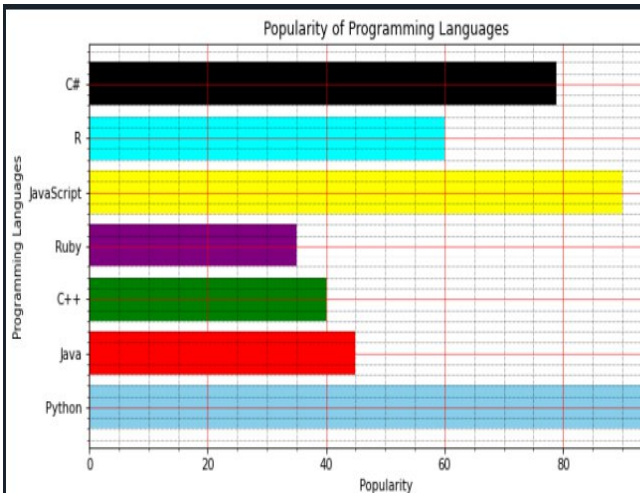
Solution

```
# Task 3: Scatter Plot for Toothpaste Sales

plt.figure(figsize=(10, 5))
plt.scatter(data['month_number'], data['toothpaste'], color='blue', label='Toothpaste Sales')

# Adding labels and title
plt.xlabel('Month Number')
plt.ylabel('Number of units sold')
plt.title('Toothpaste Sales per Month')

# Adding grid with a specific style
plt.grid(True, linewidth=1, linestyle='--')

# Adding the legend at the top-left corner
plt.legend(loc='upper left')
# Display the plot
plt.show()
```

Task 4: A Python program should display a horizontal bar chart of the popularity of programming Languages. However, after running the program, the **current output** does not look like the **intended output**.

Current output **Intended output**





Identify the issue in the code?

```
# Example data for task 4 (assuming you have a dictionary of language popularity)
languages = ['Python', 'Java', 'C++', 'Ruby', 'JavaScript', 'R','C#']
popularity = [95.5, 45, 40, 35, 90, 60, 78.9]

# Task 4: Horizontal Bar Chart
plt.figure(figsize=(10, 5))
plt.barh(languages, popularity, color=['skyblue','red','green','purple','yellow','cyan','black'])

# Adding labels and title
plt.xlabel('Popularity')
plt.ylabel('Programming Languages')
plt.title('Popularity of Programming Languages')

# ------Turn on the grid----------These have been commented out to remove the error
#plt.minorticks_on()
#plt.grid(which='major', linestyle='-',linewidth=0.5, color='red')
#plt.grid(which='minor', linestyle=':', linewidth=0.5, color='black')

# Display the plot
plt.show()
```

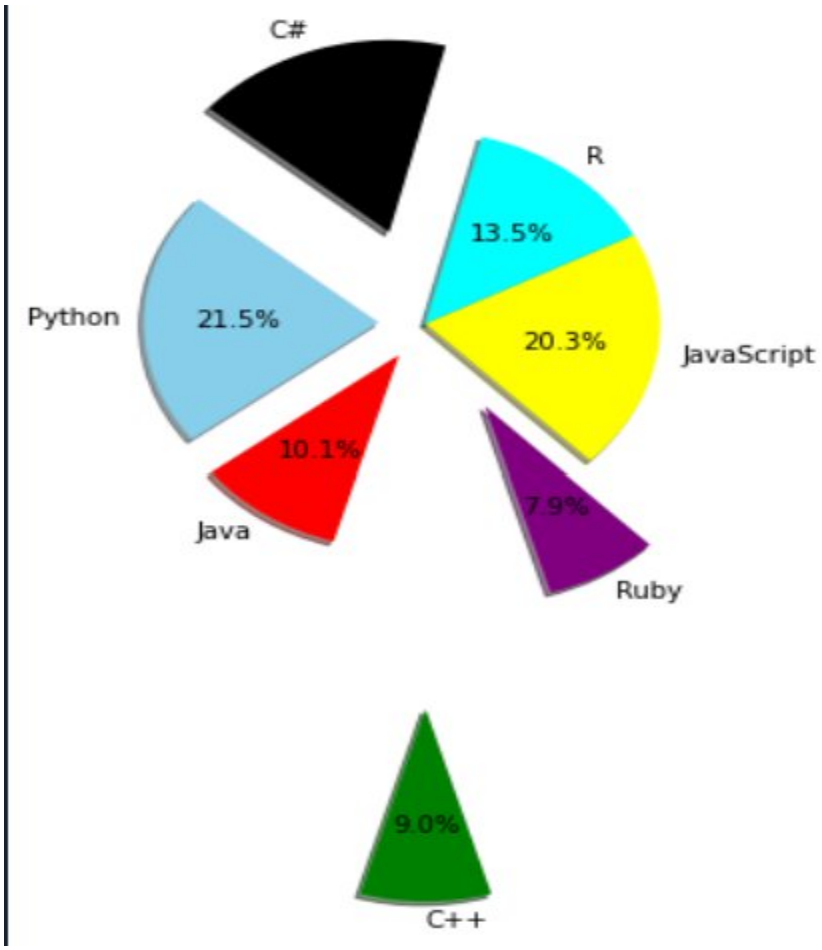
Solution:

As you can see, the grid section has been commented out. This is what was causing the issue.

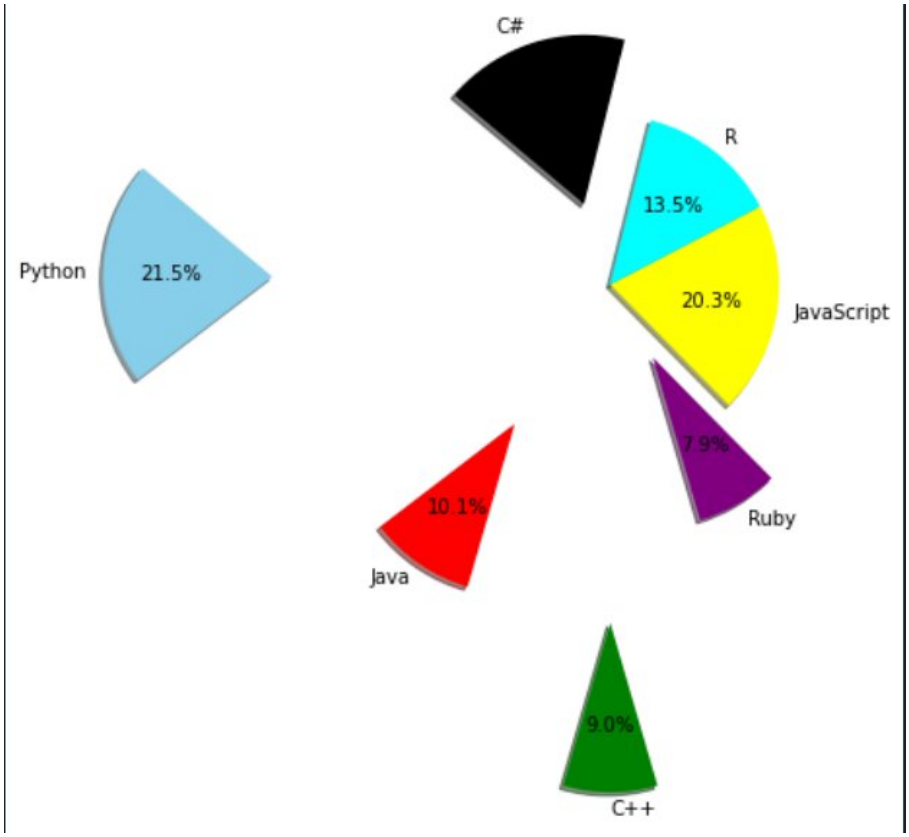
Task 5

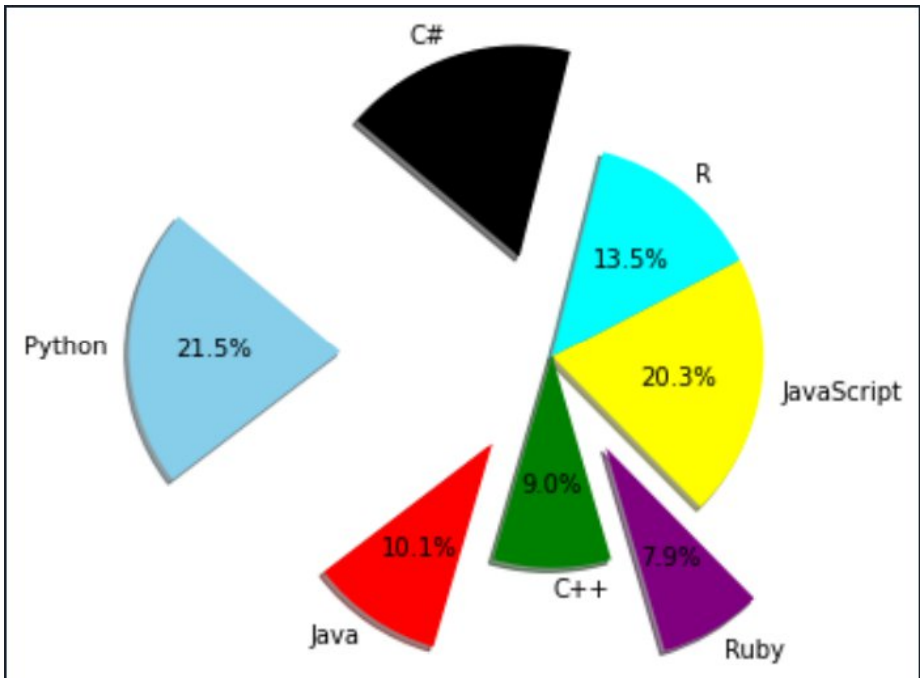
What is the output of this code?

```
languages = ['Python', 'Java', 'C++', 'Ruby', 'JavaScript', 'R','C#']
popularity = [95.5, 45, 40, 35, 90, 60, 78.9]
color=['skyblue','red','green','purple','yellow','cyan','black']
explode = (0.2, 0.2, 2, 0.5, 0, 0,0.5)
# Task 5: Pie Chart

plt.pie(popularity, explode=explode, labels=languages, colors= color,
        autopct='%1.1f%%', shadow=True, startangle=140)

# Display the plot
plt.show()
```



Wrong ones


C#
R
13.5%
Python
21.5%
20.3%
JavaScript
7.9%
10.1%
Ruby
Java
9.0%
C++


C#
R
13.5%
Python
21.5%
20.3%
JavaScript
9.0%
10.1%
7.9%
C++
Java
Ruby

Exercise 1: Lists and Arrays

Task: Create a list and an array, perform some basic operations, and print the results (list 1 contains numbers 1 to 5 and list 2 contains numbers from 6 to 10)

```python
# -*- coding: utf-8 -*-
"""
Created on Thu Aug  8 12:14:08 2024

@author: mireilla
"""
# Lists
list1 = [1, 2, 3, 4, 5]
list2 = [6, 7, 8, 9, 10]

# Concatenate lists
list3 = list1 + list2
print("Concatenated List:", list3)

# Arrays
import numpy as np

array1 = np.array([1, 2, 3, 4, 5])
array2 = np.array([6, 7, 8, 9, 10])

# Add arrays
array3 = array1 + array2
print("Added Array:", array3)

```

Exercise 2: Loading Data from Files

Task: Load data from a CSV file and print the first few rows.

```python
# -*- coding: utf-8 -*-
"""
Created on Thu Aug  8 15:44:40 2024

@author: mireilla
"""
import pandas as pd

# Load data from CSV
data = pd.read_csv('sample_data.csv')

# Print the first few rows
print(data.head())
```

	id	name	age	height	weight	income
0	1	Alice	29.0	165	55.0	50000.0
1	2	Bob	32.0	175	85.0	60000.0
2	3	Charlie	NaN	180	90.0	70000.0
3	4	David	45.0	165	80.0	NaN
4	5	Eva	37.0	160	70.0	85000.0

Exercise 3: Numpy and Array Indexing

Task: Create a numpy array and perform indexing and slicing operations.

```
"""
Created on Thu Aug  8 15:46:50 2024

@author: mireilla
"""

import numpy as np

# Create a numpy array
array = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])

# Indexing
element = array[1, 2]
print("Element at index [1, 2]:", element)

# Slicing
slice_array = array[:2, 1:]
print("Sliced Array:\n", slice_array)
```

```
Element at index [1, 2]: 6
Sliced Array:
 [[2 3]
 [5 6]]
```

Exercise 4: Creating and Indexing Numpy Arrays

Task 1: Create a 1D Numpy Array and Access Elements

1. Create a 1D numpy array with values from 0 to 9.
2. Access the 5th element of the array.
3. Access the last element of the array.

```
# -*- coding: utf-8 -*-
"""
Created on Thu Aug  8 16:22:39 2024

@author: mireilla
"""

import numpy as np

# Create a 1D numpy array
array_1d = np.arange(10)
print("1D Array:", array_1d)

# Access the 5th element
fifth_element = array_1d[4]
print("5th Element:", fifth_element)

# Access the last element
last_element = array_1d[-1]
print("Last Element:", last_element)
```

```
1D Array: [0 1 2 3 4 5 6 7 8 9]
5th Element: 4
Last Element: 9
```

Task 2: Create a 2D Numpy Array and Access Elements

1. Create a 2D numpy array with shape (3, 3) containing values from 1 to 9.
2. Access the element at the 2nd row and 3rd column.
3. Access the first row.
4. Access the first column.

```python
@author: mireilla
"""

import numpy as np

# Create a 2D numpy array
array_2d = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])
print("2D Array:\n", array_2d)

# Access the element at the 2nd row and 3rd column
element_2_3 = array_2d[1, 2]
print("Element at (2nd row, 3rd column):", element_2_3)

# Access the first row
first_row = array_2d[0, :]
print("First Row:", first_row)

# Access the first column
first_column = array_2d[:, 0]
print("First Column:", first_column)
```

```
2D Array:
 [[1 2 3]
 [4 5 6]
 [7 8 9]]
Element at (2nd row, 3rd column): 6
First Row: [1 2 3]
First Column: [1 4 7]
```

Exercise 2: Slicing and Advanced Indexing

Task 1: Slice a 1D Numpy Array

1. Create a 1D numpy array with values from 10 to 19.
2. Slice the array to get elements from index 3 to 7.
3. Slice the array to get every second element.

```python
Created on Thu Aug  8 16:22:39 2024

@author: mireilla
"""

import numpy as np

# Create a 1D numpy array
array_1d = np.arange(10, 20)
print("1D Array:", array_1d)

# Slice to get elements from index 3 to 7
slice_3_7 = array_1d[3:8]
print("Elements from index 3 to 7:", slice_3_7)

# Slice to get every second element
every_second_element = array_1d[::2]
print("Every second element:", every_second_element)
```

```
1D Array: [10 11 12 13 14 15 16 17 18 19]
Elements from index 3 to 7: [13 14 15 16 17]
Every second element: [10 12 14 16 18]
```

Task 2: Slice a 2D Numpy Array

1. Create a 2D numpy array with shape (4, 4) containing values from 1 to 16.
2. Slice the array to get the sub-array from rows 1 to 2 and columns 2 to 3.
3. Slice the array to get the last two rows.
4. Slice the array to get the first two columns.

```python
import numpy as np

# Create a 2D numpy array
array_2d = np.arange(1, 17).reshape(4, 4)
print("2D Array:\n", array_2d)

# Slice to get the sub-array from rows 1 to 2 and columns 2 to 3
sub_array = array_2d[1:3, 2:4]
print("Sub-array (rows 1-2, columns 2-3):\n", sub_array)

# Slice to get the last two rows
last_two_rows = array_2d[-2:, :]
print("Last two rows:\n", last_two_rows)

# Slice to get the first two columns
first_two_columns = array_2d[:, :2]
print("First two columns:\n", first_two_columns)
```

```
2D Array:
 [[ 1  2  3  4]
 [ 5  6  7  8]
 [ 9 10 11 12]
 [13 14 15 16]]
Sub-array (rows 1-2, columns 2-3):
 [[ 7  8]
 [11 12]]
Last two rows:
 [[ 9 10 11 12]
 [13 14 15 16]]
First two columns:
 [[ 1  2]
 [ 5  6]
 [ 9 10]
 [13 14]]
```

Exercise 3: Boolean Indexing and Fancy Indexing

Task 1: Boolean Indexing

1. Create a 1D numpy array with values from 0 to 9.
2. Use boolean indexing to get elements greater than 5.
3. Use boolean indexing to get elements that are even.

```python
import numpy as np

# Create a 1D numpy array
array_1d = np.arange(10)
print("1D Array:", array_1d)

# Boolean indexing to get elements greater than 5
greater_than_5 = array_1d[array_1d > 5]
print("Elements greater than 5:", greater_than_5)

# Boolean indexing to get elements that are even
even_elements = array_1d[array_1d % 2 == 0]
print("Even elements:", even_elements)
```

```
1D Array: [0 1 2 3 4 5 6 7 8 9]
Elements greater than 5: [6 7 8 9]
Even elements: [0 2 4 6 8]
```

Task 2: Fancy Indexing

1. Create a 2D numpy array with shape (4, 4) containing values from 1 to 16.
2. Use fancy indexing to get the elements at positions (0, 0), (1, 2), (2, 3), and (3, 1).

```python
import numpy as np

# Create a 2D numpy array
array_2d = np.arange(1, 17).reshape(4, 4)
print("2D Array:\n", array_2d)

# Fancy indexing to get specific elements
fancy_indexed_elements = array_2d[[0, 1, 2, 3], [0, 2, 3, 1]]
print("Elements at (0, 0), (1, 2), (2, 3), (3, 1):", fancy_indexed_elements)
```

```
2D Array:
 [[ 1  2  3  4]
 [ 5  6  7  8]
 [ 9 10 11 12]
 [13 14 15 16]]
Elements at (0, 0), (1, 2), (2, 3), (3, 1): [ 1  7 12 14]
```

Practical exercise for Custom package

Exercise 1: Create a Simple Package

Objective: Create a custom package with modules for math operations and string manipulations.

Instructions:

1. **Create a Directory**: Create a directory named my_package.
2. **Add an __init__.py File**: Create an empty __init__.py file in the my_package directory.
3. **Create a Module for Math Operations**:
    - Create a file named math_utils.py in the my_package directory.
    - Define two functions in math_utils.py: add(a, b) and subtract(a, b).

Create a Module for String Manipulations:

- Create a file named `string_utils.py` in the `my_package` directory.
- Define two functions in `string_utils.py`: `capitalise_words(text)` and `reverse_string(text)`.

Use the Package:

- Create a new Python script or Jupyter notebook.
- Import and use the custom package modules to perform operations.

Exercise 2: Expand the Package

Objective: Extend the custom package by adding more functionality and demonstrate importing specific functions.

Instructions:

1. **Add a New Function in `math_utils.py`**:
    - Add a function named `multiply(a, b)`.
2. **Add a New Function in `string_utils.py`**:
    - Add a function named `lowercase(text)`.
3. **Use the Extended Package**:
    - Modify the Python script or Jupyter notebook to import specific functions and use the new functionality.

Exercise 3: Creating a Custom Package for Data Analysis

Objective

The exercise will enable you to learn about the creation, structure, and usage of a custom package named **data_analysis**, which includes modules for data loading, data cleaning, and data visualisation.

A) Create the Package Directory and Files.

Create a custom package named **data_analysis** with the following structure:

data_analysis: The main package directory.

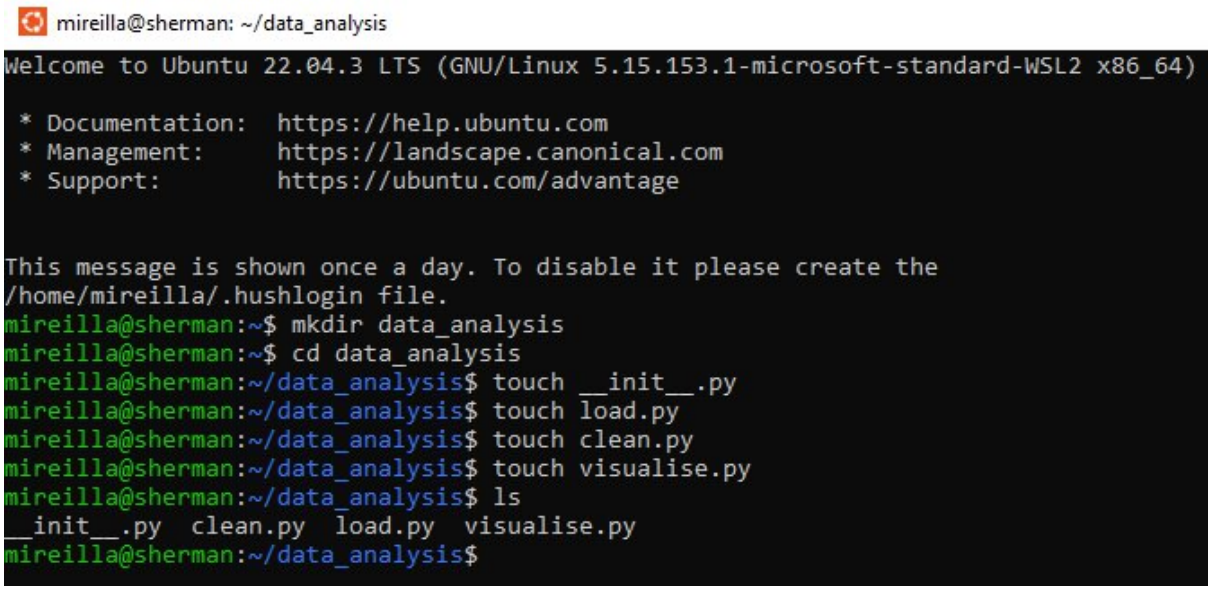
- __init__.py: To initialise the package.
- load.py: Module for loading data.
- clean.py: Module for cleaning data.
- visualise.py: Module for visualising data.

B) Implement the Modules

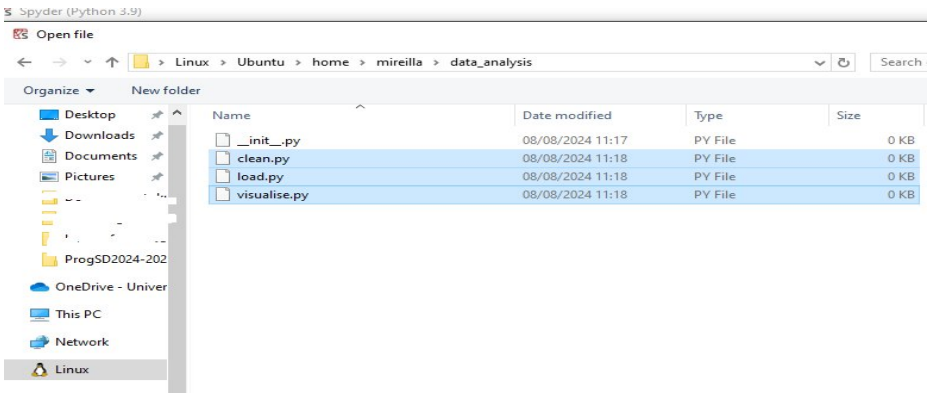
- Implement functions to load data from CSV files. (load.py).
- Implement functions to clean the data (clean.py).
- Implement functions to visualise the data (visualise.py).

C) Use the Package
Create a separate script (main.py) to use the **data_analysis** package and run the script main.py

Exercise 3 solution:
A)



B) I could edit each of the files using the **nano** command. However, I have opened my .py files in Spyder to edit them (you can use any Python IDE). Remember how to access your package from the previous exercise.



C) **Implement functions to clean the data (clean.py).** Edit the **clean.py** file to ensure that any missing data is dropped, and data is normalised. **Normalisation** is a preprocessing step that scales the values of a column to have a mean of 0 and a standard deviation of 1. This process is often used in data analysis and machine learning to ensure that different features contribute equally to the analysis.

```python
# -*- coding: utf-8 -*-
"""
Created on Thu Aug  8 12:14:08 2024

@author: mireilla
"""

# clean.py

import pandas as pd

def drop_missing(data):
    """Drop rows with missing values."""
    cleaned_data = data.dropna()
    print("Missing values dropped.")
    return cleaned_data

# Normalisation is a preprocessing step that scales the values of a column
#to have a mean of 0 and a standard deviation of 1.
#This process is often used in data analysis and machine learning to
#ensure that different features contribute equally to the analysis.

def normalise(data, columns):
    """Normalise specified columns."""
    for column in columns:
        if column in data.columns:
            data[column] = (data[column] - data[column].mean()) / data[column].std()
            print(f"Columns {columns} normalised.")
        else:
            print(f"Column {column} not found in data.")

    return data
```

D) **Implement functions to load data from CSV files (load.py)**. Edit **load.py** to load the data from the CSV file.

```python
# -*- coding: utf-8 -*-
"""
Created on Thu Aug  8 12:14:08 2024

@author: mireilla
"""

# load.py

import pandas as pd

def load_csv(file_path):
    "Load data from a CSV file."""
    # The try block contains the code that attempts to execute an operation which MIGHT fail.
    try:
        # load the file
        data = pd.read_csv(file_path)
        print(f"Data loaded successfully from {file_path}")
        return data

    # except block is designed to catch and handle exceptions that occur within the try block
    except Exception as e:
        print(f"Error loading data: {e}")
        return None
```

E) **Implement functions to visualise the data (visualise.py)**. Edit visualise.py

- import matplotlib.pyplot and seaborn.
- Create a function for histogram.
- Create a function for a scatter plot.

```python
# -*- coding: utf-8 -*-
"""
Created on Thu Aug  8 12:14:08 2024

@author: mireilla
"""
# visualise.py
# matplotlib.pyplot is a module within the matplotlib library, which is a comprehensive library
# for creating static, animated, and interactive visualizations in Python

import matplotlib.pyplot as plt

# seaborn is a statistical data visualisation library built on top of matplotlib.
# It provides a high-level interface for drawing attractive and informative statistical graphics

import seaborn as sns

def plot_histogram(data, column):
    """Plot a histogram of a specified column."""
    plt.figure(figsize=(10, 6))
    sns.histplot(data[column], kde=True)
    plt.title(f'Histogram of {column}')
    plt.show()

def plot_scatter(data, column4, column5):
    """Plot a scatter plot of two specified columns."""
    plt.figure(figsize=(10, 6))
    sns.scatterplot(x=column4, y=column5, data=data)
    plt.title(f'Scatter plot of {column4} vs {column5}')
    plt.show()
```

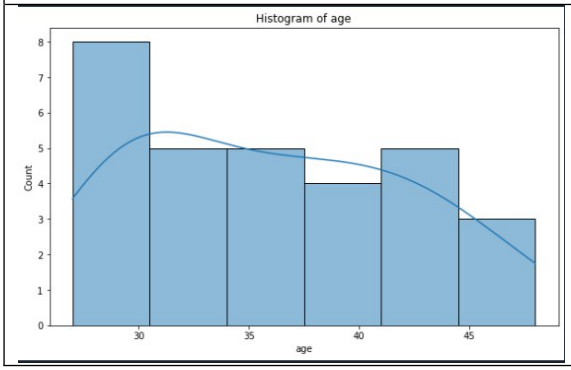
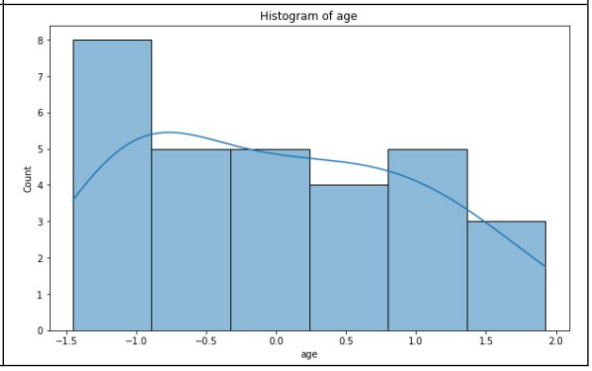
F) Create your **main.py** file using the **touch** command in Linux or directly from your Python IDE. Make sure it is saved in the same location as the other files. In main.py:

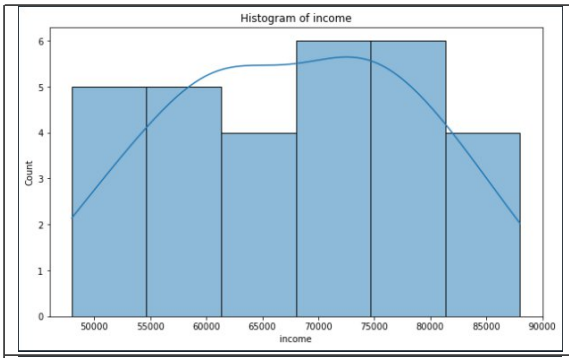
- Add the path to the custom data_analysis package
- Import the custom package modules
- Load the data (you can use this file).
- Clean the data (call clean.py).
- Visualise the data that is not normalised.
- Normalise the data (call clean.py)
- Visualise the normalised data (call visualise.py).

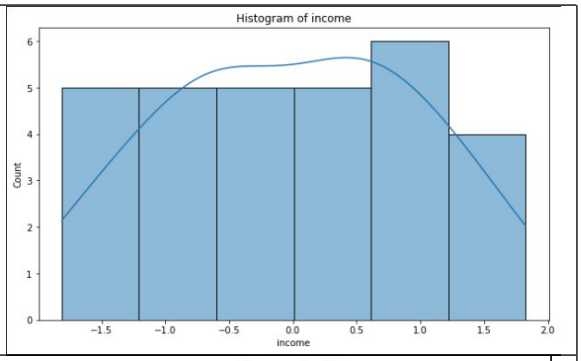
```python
# -*- coding: utf-8 -*-
"""
Created on Thu Aug  8 12:14:08 2024

@author: mireilla
"""
# main.py
import sys

# Add the path to the data_analysis package
package_path = r'\\wsl.localhost\Ubuntu\home\mireilla\data_analysis'
sys.path.append(package_path)

# Import the custom package modules
import load, clean, visualise

# Load the data
file_path = 'C:/Users/Mireilla.DCS/OneDrive - University of Glasgow/ProgSD2024-2025/sample_data1.csv'

data = load.load_csv(file_path)

# If the data loads successfully, it proceeds with cleaning and visualisation.
if data is not None:
    # Clean the data
    data = clean.drop_missing(data)

    # Visualize the data that has not been normalised
    visualise.plot_histogram(data, 'age')
    visualise.plot_histogram(data, 'income')
    visualise.plot_histogram(data, 'weight')
    visualise.plot_scatter(data, 'height', 'weight')

    # Normalise the data
    data = clean.normalise(data, ['age', 'height', 'weight', 'income'])

    # Visualize the data that has been normalised
    visualise.plot_histogram(data, 'age')
    visualise.plot_histogram(data, 'income')
    visualise.plot_histogram(data, 'weight')
    visualise.plot_scatter(data, 'height', 'weight')

# If an error occurs and None is returned, it prints "Failed to load data."
else:
    print("Failed to load data.")
```

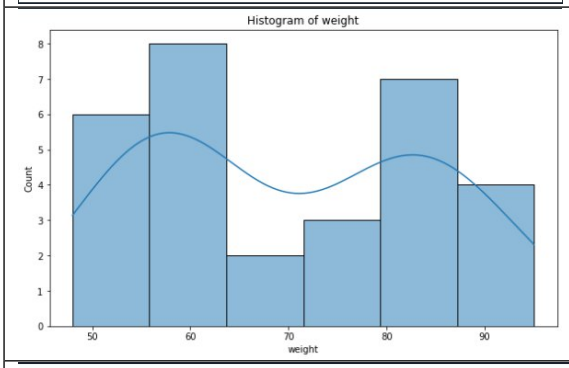
E)Outputs

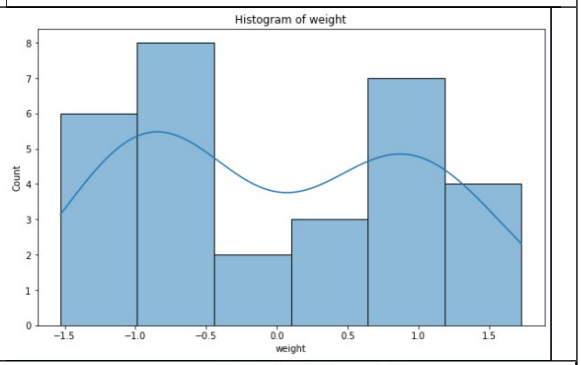
Visualise without/with normalisation

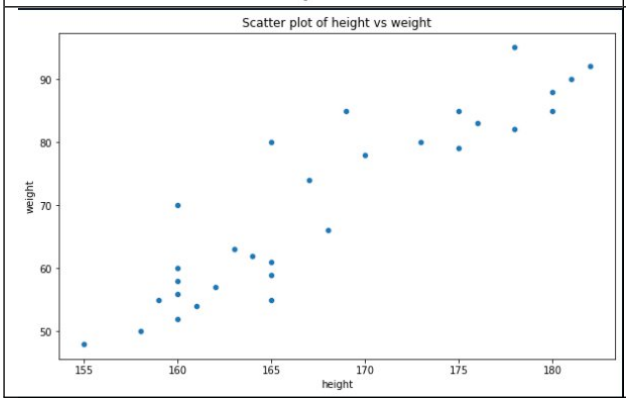
Without Normalisation	With Normalisation
	

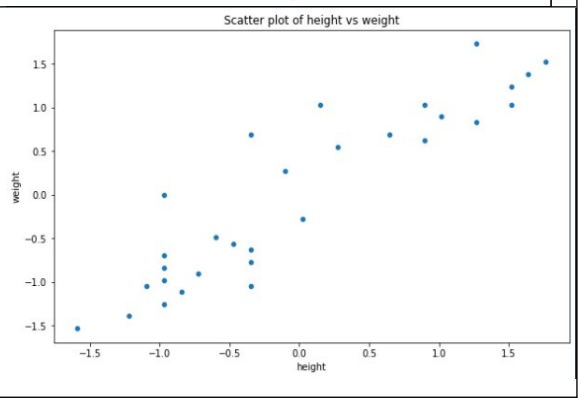

Histogram of income
Count
income


Histogram of income
Count
income


Histogram of weight
Count
weight


Histogram of weight
Count
weight


Scatter plot of height vs weight
weight
height


Scatter plot of height vs weight
weight
height

Lab Exercise 4 - Solutions

Task 1: Write a NumPy program to test whether none of the elements of a given array is zero.

```
import numpy as np
x = np.array([10, 2, 30, 45])
print("Original array:")
print(x)
print("Test if NONE of the elements of the array is zero:")
print(np.all(x))
```

Task 2: There are two arrays. The first array, array1 contains the values 45, 67, 23 and array2 contains the values 56, 23, and 89. Write a NumPy program to create an element-wise comparison (greater, greater_equal, less and less_equal) of those two arrays.

```
import numpy as np
array1 = np.array([45, 67, 23])
array2 = np.array([56, 23, 89])
print("Comparison - greater")
print(np.greater(array1, array2))
print("Comparison - greater_equal")
print(np.greater_equal(array1, array2))
print("Comparison - less")
print(np.less(array1, array2))
print("Comparison - less_equal")
print(np.less_equal(array1, array2))
```

Task3 Write a NumPy program to create an array of 8 zeros, 5 ones, 10 fives.

```
import numpy as np
array=np.zeros(8)
print("An array of 10 zeros:")
print(array)
array=np.ones(5)
print("An array of 10 ones:")
print(array)
array=np.ones(10)*5
print("An array of 10 fives:")
print(array)
```

```
An array of 10 zeros:
[0. 0. 0. 0. 0. 0. 0. 0.]
An array of 10 ones:
[1. 1. 1. 1. 1.]
An array of 10 fives:
[5. 5. 5. 5. 5. 5. 5. 5. 5. 5.]
```

Task4: Write a NumPy program to add a vector **v** with values 2, 0, 2 to each row of a matrix **m** with values:

[23 45 11]
[12 23 54]
[29 19 34]
[1 23 10]

```
import numpy as np
m = np.array([[23, 45, 11], [12, 23, 54], [29, 19, 34], [1, 23, 10]])
v = np.array([2, 0, 2])
print("Original vector:")
print(v)
print("Original matrix:")
print(m)
result = np.empty_like(m)
for i in range(4):
  result[i, :] = m[i, :] + v
print("\nAfter adding the vector v to each row of the matrix m:")
print(result)
```

Task 5: Write a NumPy program to create a 5x5 2d array with 1 on the border and 25 inside.

```
import numpy as np
x = np.ones((5,5))
print("Original array:")
print(x)
print("1 on the border and 0 inside in the array")
x[1:-1,1:-1] = 25
print(x)
```

Task 6: Write a NumPy program to find common values between two arrays.

Array1	Array2
[23, 45, 11, 5]	[23, 5, 1]

```
import numpy as np
array1 = np.array([23, 45, 11, 5])
print("Array1: ",array1)
array2 = [23, 5, 1]
print("Array2: ",array2)
print("Common values between two arrays:")
print(np.intersect1d(array1, array2))
```

Task 7: Perform the following manipulation of the two arrays below: horizontal stacking, vertical stacking; divide the individual array horizontally and vertically.

Array1	Array2
[23, 45, 11]	[3, 5, 1]
[12, 23, 54]	[2, 3, 4]
[1, 23, 10]	[9, 1, 5]

Consider the following dataframe for the remaining tasks

Sample DataFrame:
assessment_results = {'name': ['Anastasia', 'Paul', 'Kathe', 'Joseph', 'Linda', 'Michael', 'Matt', 'Laurentine', 'Chirstian', 'Jonas'],
'score': [12.5, 10, 16.5, np.nan, 9, 20, 14.5, np.nan, 8, 19],
'attempts': [1, 3, 2, 3, 2, 3, 1, 1, 2, 1],
'qualify': ['yes', 'no', 'yes', 'no', 'no', 'yes', 'yes', 'no', 'no', 'yes']}
labels = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j']

Task 8: Write a Pandas program to get the first 3 rows of a given DataFrame.

```
import pandas as pd
import numpy as np

exam_data  = {'name': ['Anastasia', 'Paul', 'Kathe', 'Joseph', 'Linda', 'Michael', 'Matt', 'Laurentine', 'Chirstian', 'Jonas'],
        'score': [12.5, 10, 16.5, np.nan, 9, 20, 14.5, np.nan, 8, 19],
        'attempts': [1, 3, 2, 3, 2, 3, 1, 1, 2, 1],
        'qualify': ['yes', 'no', 'yes', 'no', 'no', 'yes', 'yes', 'no', 'no', 'yes']}
labels = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j']

df = pd.DataFrame(exam_data , index=labels)
print("First three rows of the data frame:")
print(df.iloc[:3])
```

Task 9: Write a Pandas program to select the rows where the number of attempts in the examination is greater than 2.

```
import pandas as pd
import numpy as np

exam_data  = {'name': ['Anastasia', 'Paul', 'Kathe', 'Joseph', 'Linda', 'Michael', 'Matt', 'Laurentine', 'Chirstian', 'Jonas'],
        'score': [12.5, 10, 16.5, np.nan, 9, 20, 14.5, np.nan, 8, 19],
        'attempts': [1, 3, 2, 3, 2, 3, 1, 1, 2, 1],
        'qualify': ['yes', 'no', 'yes', 'no', 'no', 'yes', 'yes', 'no', 'no', 'yes']}
labels = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j']

df = pd.DataFrame(exam_data , index=labels)
print("First three rows of the data frame:")
print(df[df['attempts'] > 2])
```

Task 10: Write a Pandas program to count the number of rows and columns of a DataFrame.

```
import pandas as pd
import numpy as np

exam_data  = {'name': ['Anastasia', 'Paul', 'Kathe', 'Joseph', 'Linda', 'Michael', 'Matt', 'Laurentine', 'Chirstian', 'Jonas'],
        'score': [12.5, 10, 16.5, np.nan, 9, 20, 14.5, np.nan, 8, 19],
        'attempts': [1, 3, 2, 3, 2, 3, 1, 1, 2, 1],
        'qualify': ['yes', 'no', 'yes', 'no', 'no', 'yes', 'yes', 'no', 'no', 'yes']}
labels = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j']

df = pd.DataFrame(exam_data , index=labels)
print("First three rows of the data frame:")
total_rows=len(df.axes[0])
total_cols=len(df.axes[1])
print("Number of Rows: "+str(total_rows))
print("Number of Columns: "+str(total_cols))
```

Task 11: Write a Pandas program to select the rows the score is between 14 and 20 (inclusive).

```
import pandas as pd
import numpy as np

exam_data  = {'name': ['Anastasia', 'Paul', 'Kathe', 'Joseph', 'Linda', 'Michael', 'Matt', 'Laurentine', 'Chirstian', 'Jonas'],
        'score': [12.5, 10, 16.5, np.nan, 9, 20, 14.5, np.nan, 8, 19],
        'attempts': [1, 3, 2, 3, 2, 3, 1, 1, 2, 1],
        'qualify': ['yes', 'no', 'yes', 'no', 'no', 'yes', 'yes', 'no', 'no', 'yes']}
labels = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j']

df = pd.DataFrame(exam_data , index=labels)
print("Rows where score between 14 and 20 (inclusive):")
print(df[df['score'].between(14, 20)])
```

Task 12: Write a Pandas program to change the score in row 'c' to 11.5

```
import pandas as pd
import numpy as np

exam_data  = {'name': ['Anastasia', 'Paul', 'Kathe', 'Joseph', 'Linda', 'Michael', 'Matt', 'Laurentine', 'Chirstian', 'Jonas'],
        'score': [12.5, 10, 16.5, np.nan, 9, 20, 14.5, np.nan, 8, 19],
        'attempts': [1, 3, 2, 3, 2, 3, 1, 1, 2, 1],
        'qualify': ['yes', 'no', 'yes', 'no', 'no', 'yes', 'yes', 'no', 'no', 'yes']}
labels = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j']

df = pd.DataFrame(exam_data , index=labels)
print("\nOriginal data frame:")
print(df)
print("\nChange the score in row 'c' to 11.5:")
df.loc['c', 'score'] = 11.5
print(df)
```

Task 13: Write a Pandas program to calculate the mean score for each different student in DataFrame

```
import pandas as pd
import numpy as np

exam_data  = {'name': ['Anastasia', 'Paul', 'Kathe', 'Joseph', 'Linda', 'Michael', 'Matt', 'Laurentine', 'Chirstian', 'Jonas'],
        'score': [12.5, 10, 16.5, np.nan, 9, 20, 14.5, np.nan, 8, 19],
        'attempts': [1, 3, 2, 3, 2, 3, 1, 1, 2, 1],
        'qualify': ['yes', 'no', 'yes', 'no', 'no', 'yes', 'yes', 'no', 'no', 'yes']}
labels = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j']

df = pd.DataFrame(exam_data , index=labels)
print("\nMean score for each different student in data frame:")
print(df['score'].mean())
```

Team Application Exercises (tAPP-4) - Solutions

Task 1

Write a NumPy program to create an array of integers from 10 to 20. Write down its output when it is run.

```
import numpy as np
array=np.arange(10,20)
print("Array of the integers from 10 to 20")
print(array)
```

```
Array of the integers from 10 to 20
[10 11 12 13 14 15 16 17 18 19]
```

Task 2:

Write a program that creates a 3x3 array. The program should the use `savetxt()` and `load txt()` to save a given array to a text file and load it.

```
import numpy as np


x = np.arange(9).reshape(3, 3)
print("Original array:")
print(x)
np.savetxt('temp.txt', x, fmt = "%d")
print("After loading, content of the text file: ")
result = np.loadtxt('temp.txt')
print (result)

# You can also do this
y = np.arange(9)
z = np.reshape(y,(3,3))
print(z)
np.savetxt('temp1.txt', z, fmt = "%d")
print("After loading, content of the text file: ")
result = np.loadtxt('temp1.txt')
print(result)
```

```
Original array:
[[0 1 2]
 [3 4 5]
 [6 7 8]]
After loading, content of the text file:
[[0. 1. 2.]
 [3. 4. 5.]
 [6. 7. 8.]]
```

```
y = np.arange(9)
print(y.reshape(3, 3))
np.savetxt('temp1.txt', y, fmt = "%d")
print("After loading, content of the text file: ")
result = np.loadtxt('temp1.txt')
print(result)
print (result.reshape(3, 3))
```

```
[[0 1 2]
 [3 4 5]
 [6 7 8]]
After loading, content of the text file:
[0. 1. 2. 3. 4. 5. 6. 7. 8.]
[[0. 1. 2.]
 [3. 4. 5.]
 [6. 7. 8.]]
```

output of print result

output of print result.reshape.

Normally, in the correct solution, we only want to load what we saved and then print it. We do not want to reshape it again.

Therefore the solution on the left is the best answer.

```
x = np.arange(200, 209).reshape(3, 3)
print("Original array:")
print(x)
np.savetxt('temp.txt', x, fmt = "%d")
print("After loading, content of the text file:")
result = np.loadtxt('temp.txt')
print (result)

# You can also do this
y = np.arange(12, 24)
z = np.reshape(y,(4,3))
print(z)
np.savetxt('temp1.txt', z, fmt = "%d")
print("After loading, content of the text file:")
result = np.loadtxt('temp1.txt')
print(result)
```

```
untitled0.py', wdir='C:/Users/mireilla/OneDrive -
Original array:
[[200 201 202]
 [203 204 205]
 [206 207 208]]
After loading, content of the text file:
[[200. 201. 202.]
 [203. 204. 205.]
 [206. 207. 208.]]
[[12 13 14]
 [15 16 17]
 [18 19 20]
 [21 22 23]]
After loading, content of the text file:
[[12. 13. 14.]
 [15. 16. 17.]
 [18. 19. 20.]
 [21. 22. 23.]]
```

Task3:

Perform the following operations: Addition, subtraction and matrix product:

array1	array2
[23, 45, 11] [12, 23, 54] [1, 23, 10]	[3, 5, 1] [2, 3, 4] [9, 1, 5]

Solutions

	Addition of two arrays			Subtraction		
Array1 -row1	23	45	11	23	45	11
	+	+	+	-	-	-
Array2-row1	3	5	1	3	5	1
Solution row1	**26**	**50**	**12**	**20**	**40**	**10**
Array1 -row2	12	23	54	12	23	54
	+	+	+	-	-	-
Array2-row2	2	3	4	2	3	4
Solution row 2	**14**	**26**	**58**	**10**	**20**	**50**
Array1 -row3	1	23	10	1	23	10
	+	+	+	-	-	-
Array2-row3	9	1	5	9	1	5
Solution row 3	**10**	**24**	**15**	**-8**	**22**	**5**

```
[[ 69 225  11]
 [ 24  69 216]
 [  9  23  50]]
```

This would be the solution if were performing a simple matrix multiplication using the "*" operator.

But this problem asked for the matrix product. So we will use numpy dot()

Addition	Subtraction
array([[26, 50, 12], [14, 26, 58], [10, 24, 15]])	array([[20, 40, 10], [10, 20, 50], [-8, 22, 5]])

Matrix Product

array1	array2
[23, 45, 11] [12, 23, 54] [1, 23, 10]	[3, 5, 1] [2, 3, 4] [9, 1, 5]

1) To get the first row of our results, we use the first row of array1 and the 3 columns of array2:

Array1 row 1 and array2 column 1: (23 * 3) + (45*2) + (11*9) = **258**
Array1 row 1 and array2 column 2: (23*5) + (45*3) + (11*1) = **261**
Array1 row 1 and array2 column 3: (23*1) + (45*4) + (11*5) = **258**

258	**261**	**258**

2) To get our second row of results, we use row2 of array1 and all 3 columns of array2.

Array1 row 2 and array2 column 1: (12 * 3) + (23*2) + (54*9) = **568**
Array1 row 2 and array2 column 2: (12 * 5) + (23*3) + (54*1) = **183**
Array1 row 2 and array2 column 3: (12 * 1) + (23*4) + (54*5) = **374**

258	**261**	**258**
568	**183**	**374**

To get our last row of results, we use **row3 of array1** and **all 3 columns of array2.**

Array1 row **3** and array2 column 1: (1 *3) + (23*2) + (10 * 9) = 139
Array1 row **3** and array2 column 2: (1 *5) + (23*3) + (10 * 1) = 84
Array1 row **3** and array2 column 3: (1 *1) + (23*4) + (10 * 5) = 143

258	**261**	**258**
568	**183**	**374**
139	**84**	**143**

So the results of our matrix dot product is:

```
[[258 261 258]
 [568 183 374]
 [139  84 143]]
```

Task4:

Perform the following operations: Addition, subtraction and matrix product:

array1	array2
[3, 5, 1] [2, 3, 5] [1, 3, 1]	[8, 5, 1] [2, 9, 4] [2, 10, 5]

Solutions

Addition	Subtraction	Matrix product
array([[11, 10, 2], [4, 12, 9], [3, 13, 6]])	array([[-5, 0, 0], [0, -6, 1], [-1, -7, -4]])	array([[36, 70, 28], [32, 87, 39], [16, 42, 18]])

array1	array2
[4, 7, 2] [2, 3, 2] [2, 3, 1]	[3, 5, 1] [5, 7, 4] [2, 5, 5]

Simulating ChatGPT Locally – Solutions

These exercises allow you to enhance the capabilities of the simulated ChatGPT function. The exercises emphasise the importance of conditional logic in determining appropriate responses.

Exercise 1: **Problem**: Expand the **simulated_chatgpt** function to handle questions about the weather, e.g., "How's the weather today?". It should return a random weather condition from a list, like ["sunny", "cloudy", "rainy"].

Exercise 2: **Problem**: Enhance the **simulated_chatgpt** function so that when asked, "Tell me a joke", it responds with a random joke from a predefined list of jokes.

```
# -*- coding: utf-8 -*-
"""
Created on Tue Aug 22 13:00:57 2023

@author: mireilla
"""

import random

def simulated_chatgpt(query):
    """
    A simple function to simulate ChatGPT's response based on given queries.

    Parameters:
    - query (str): The user's question/input

    Returns:
    - str: The simulated response from ChatGPT
    """

    # Lowercase the query for consistent matching
    query = query.lower()

    if "hello" in query:
        return "Hello there!"
    elif "how are you" in query:
        return "I'm just a simulation, so I don't have feelings. But thanks for asking!"
    elif "recommend a book" in query:
        return "I suggest reading 'To Kill a Mockingbird' by Harper Lee. It's a classic!"
# Updating function to handle questions about the weather
    elif "how's the weather" in query:
        weather_conditions = ["sunny", "cloudy", "rainy"]
        return f"It's {random.choice(weather_conditions)} today!"

# Updating function so that when asked, "Tell me a joke",
# it responds with a random joke from a predefined list of jokes.

    elif "tell me a joke" in query:
        jokes = [
            "Why did the scarecrow win an award? Because he was outstanding in his field!",
            "Why don't scientists trust atoms? Because they make up everything!",
            "Why did the math book look sad? Because it had too many problems."
        ]
        return random.choice(jokes)
    else:
        return "Sorry, I'm not sure how to respond to that."
# Let's test the simulation:
print(simulated_chatgpt("Hello"))  # Output: "Hello there!"
print(simulated_chatgpt("Can you recommend a book?"))
print(simulated_chatgpt("How are you?"))
print(simulated_chatgpt("How's the weather?"))
```

More practical tasks for Neural Network

torch.manual_seed(1337) sets the random seed for PyTorch's random number generator. That means it ensures that every time you run your script, the random values generated (for example, in torch.randn(...)) are the same each time.

Loss functions

Combination	Typical Use Case	Type of Problem
MSELoss + Adam	Regression or continuous-value prediction	Regression
BCELoss + optim.SGD	Binary classification (0/1 outcomes)	Classification

torch.nn.MSELoss() - Mean Squared Error Loss

This loss function measures the average squared difference between predictions and true values. You can use it if your model predicts continuous numerical values (e.g., temperature, price, risk score).

You can think of it as "How far are my predictions from the true numbers on average?"

torch.nn.BCELoss() – Binary Cross-Entropy Loss

This loss function measures the difference between predicted probabilities and binary labels (0 or 1). It is used when your model predicts the probability of a class (for example, "1 = attack" or "0 = safe").

Think of it as "How close are my predicted probabilities to the true labels?"

The Optimisers

torch.optim.Adam

Adam stands for Adaptive Moment Estimation. This modern optimiser automatically adjusts the learning rate for each parameter based on gradient history.

torch.optim.SGD

SGD stands for Stochastic Gradient Descent. It is the most basic and traditional optimiser. It updates weights in the direction that reduces loss, scaled by a fixed learning rate.

Aspect	MSELoss	BCELoss	Adam	SGD
Use Case	Regression (predict numbers)	Binary classification (0/1)	General optimiser (adaptive)	Basic optimiser (manual learning rate)
Output expected	Real value (no Sigmoid)	Probability (0–1)	—	—
Speed	—	—	Faster convergence	Slower (but simple)
Tuning	—	—	Needs fewer hyperparameter tweaks	Needs learning rate tuning
Common combo	MSELoss + Adam	BCELoss + SGD (or Adam)	—	—

Practical Task 1: PyTorch Regression (MSELoss + Adam)

In this practical exercise, you will build and train a simple regression model using PyTorch. You will learn how to define a neural network, generate synthetic data, use MSELoss, and train the model using the Adam optimizer.

Learning Objectives

By the end of this exercise, you will be able to:

Define a custom PyTorch neural network class using torch.nn.Module.
Generate synthetic training data for a regression task.
Configure and train a model using MSELoss and the Adam optimizer.
Track and interpret training loss across epochs.

Instructions

Step 1: Data Generation

Import torch.
Generate feature data X with 120 samples and 8 features:
  X = torch.randn(120, 8)

Create a learnable target using the formula:
  y = X @ w + b + noise, where w (8×1) and b are constants, and noise ~ N(0, 0.5).
Print X.shape and y.shape to confirm dimensions.

Step 2: Model Definition

Define a neural network class called RevenueNet that extends torch.nn.Module.
Include three layers:

- Input: 8 neurons
- Hidden: 16 → 8 neurons (ReLU activation)
- Output: 1 neuron (no activation)

Print the model architecture after initialization.

Step 3: Model Setup

Instantiate your RevenueNet model.
Use torch.nn.MSELoss() as your loss function.

Use the Adam optimizer with a learning rate of 0.010.

Step 4: Training Loop

Train the model for 30 epochs.
For each epoch:
Compute predictions with model(X).
Calculate loss using the MSELoss criterion.
Zero the gradients, perform backpropagation, and update model weights.

Print loss after each epoch in the format:
Epoch 01 | Loss: 0.123456

Step 5: Loss Visualisation

Use matplotlib to plot the recorded training loss across epochs. Label axes and provide a title. Display the chart with plt.show().

Step 6: Extension Activity (Optional)

Modify the task to deepen your understanding:
• Change the number of epochs or learning rate and observe the effect on loss.
• Add another hidden layer and compare training performance.
• Increase the noise term to see how it affects learning stability.

Reflection Questions

Answer these questions after completing the exercise:

1) What does a decreasing MSE loss tell you about your model's learning process?
2) What happens if you increase the learning rate to 0.1? Why?
3) How does adding more noise to your target (y) affect training performance?
4) Why is MSELoss appropriate for regression problems instead of BCELoss?
5) In what real-world cybersecurity scenario might regression modelling be used?

Practical Task 2: PyTorch Classification (BCELoss + SGD)

In this practical exercise, you will build and train a small binary classification model using PyTorch. You will learn how to define a neural network, generate synthetic binary labels (a learnable target), use BCELoss for classification, and train the model with the SGD optimizer.

Learning Objectives

By the end of this exercise, you will be able to:

Generate synthetic features and binary targets suitable for classification.
Define a simple feedforward neural network with Sigmoid output.
Configure BCELoss and train using optim.SGD.
Track and interpret training loss during optimization.

Instructions

Step 1: Data Generation (Learnable Binary Target)

1) Import torch.
2) Generate input features X with 200 samples and 4 features:
   X = torch.randn(200, 4)
3) Create a learnable binary target using a noisy logistic model:
   w = torch.tensor([[0.8],[0.6],[0.7],[0.5]])
   b = torch.tensor([-1.0])
   noise = 0.3 * torch.randn(200, 1)
   proba = torch.sigmoid(X @ w + b + noise) # values in (0, 1)
   y = (proba > 0.5).float() # binary labels (0/1), shape (200,1)

4) Print the shapes of X and y to confirm dimensions.

Step 2: Model Definition (RiskNN)

1. Define a neural network class RiskNN that extends torch.nn.Module.
2. Architecture:
   - Input Layer: 4 neurons
   - Hidden Layer: 8 neurons with ReLU activation
   - Output Layer: 1 neuron with Sigmoid activation

3. Implement forward(self, x) to return the network output.

Step 3: Model Setup (BCELoss + SGD)

1. Instantiate the RiskNN model.
2. Use torch.nn.BCELoss() as the loss function.
3. Use torch.optim.SGD as the optimizer with a learning rate of 0.05.
4. Print the model architecture to verify the layers.

Step 4: Training Loop

1. Train the model for 15 epochs.
2. For each epoch:
   - Compute predictions: preds = model(X)
   - Compute loss: loss = criterion(preds, y)
   - Zero gradients, backpropagate, and step the optimizer.
3. Record and print the loss each epoch in the format:
   Epoch 01: loss=0.6931

Step 5: Loss Visualisation

Use matplotlib to plot the recorded training loss across epochs. Label axes and provide a title. Display the chart with plt.show().

Reflection Questions

Answer these questions after completing the exercise:

1) Why is BCELoss appropriate for this task, and when would you prefer MSELoss instead?
2) What happens to learning if you remove the Sigmoid from the output layer while still using BCELoss?
3) How does increasing the learning rate for SGD (e.g., to 0.2) affect convergence and stability?
4) How does increasing the noise in label generation change the loss curve and final performance?
5) If classes become imbalanced (e.g., 90% of labels are 0), what changes could you make to the loss or data generation to address this?

PyTorch Practical Exercises

Practical exercises to help solidify the concepts of using PyTorch to build neural networks:

Exercise 1: Setting Up PyTorch

1. **Task:** Install PyTorch on your local machine or cloud environment
   - **Command:** Use the appropriate command from PyTorch's official site to install the latest version compatible with your setup.
   - For Google Colab:

```
bash
Copy code
!pip install torch torchvision
```

2. **Task:** Verify the installation by importing PyTorch and checking the version.
   - **Code:**

```
python
Copy code
import torch
print(torch.__version__)
```

Exercise 2: Building a Simple Neural Network

1. **Task:** Define a simple feedforward neural network using `nn.Module`.
   - **Code:**

```
python
Copy code
import torch.nn as nn

class SimpleNN(nn.Module):
    def __init__(self):
        super(SimpleNN, self).__init__()
        self.fc1 = nn.Linear(784, 128)  # Input layer
        self.fc2 = nn.Linear(128, 64)   # Hidden layer
        self.fc3 = nn.Linear(64, 10)    # Output layer

    def forward(self, x):
        x = torch.relu(self.fc1(x))
        x = torch.relu(self.fc2(x))
        x = torch.softmax(self.fc3(x), dim=1)
        return x

# Instantiate the network
model = SimpleNN()
print(model)
```

Exercise 3: Training the Neural Network

1. **Task:** Train the neural network on a simple dataset, such as MNIST.
    - **Code:**

```
python
Copy code
import torch.optim as optim
import torch.nn.functional as F
from torchvision import datasets, transforms

# Load the MNIST dataset
transform = transforms.Compose([transforms.ToTensor()])
trainset = datasets.MNIST(root='./data', train=True,
download=True, transform=transform)
trainloader = torch.utils.data.DataLoader(trainset,
batch_size=32, shuffle=True)

# Define the loss function and optimizer
criterion = nn.CrossEntropyLoss()
optimizer = optim.SGD(model.parameters(), lr=0.01)

# Training loop
for epoch in range(1, 6):  # 5 epochs
    running_loss = 0.0
    for images, labels in trainloader:
        # Flatten the images into vectors
        images = images.view(images.shape[0], -1)

        # Zero the parameter gradients
        optimizer.zero_grad()

        # Forward pass
        outputs = model(images)

        # Compute loss
        loss = criterion(outputs, labels)

        # Backward pass and optimization
        loss.backward()
        optimizer.step()

        # Update running loss
        running_loss += loss.item()

    print(f"Epoch {epoch}, Loss:
{running_loss/len(trainloader)}")
```

Exercise 4: Evaluating the Model

1. **Task:** Evaluate the trained model on a validation set.
    - **Code:**

```
python
Copy code
testset = datasets.MNIST(root='./data', train=False,
download=True, transform=transform)
testloader = torch.utils.data.DataLoader(testset,
batch_size=32, shuffle=False)

correct = 0
```

```
total = 0

with torch.no_grad():
    for images, labels in testloader:
        images = images.view(images.shape[0], -1)
        outputs = model(images)
        _, predicted = torch.max(outputs.data, 1)
        total += labels.size(0)
        correct += (predicted == labels).sum().item()

print(f"Accuracy: {100 * correct / total}%")
```

Exercise 5: Advanced Practice

1. **Task:** Implement a Convolutional Neural Network (CNN) using PyTorch and train it on the CIFAR-10 dataset.

The CIFAR dataset is a popular image dataset used for machine learning and computer vision tasks. There are two versions of the CIFAR dataset:

1. **CIFAR-10**: Consists of 60,000 32x32 color images in 10 classes, with 6,000 images per class.
2. **CIFAR-100**: Consists of 60,000 32x32 color images in 100 classes, with 600 images per class.

Accessing the CIFAR Dataset in PyTorch

In PyTorch, you can easily access the CIFAR-10 or CIFAR-100 dataset using the `torchvision.datasets` module, which provides a straightforward way to load and preprocess these datasets.

Here's an example of how to load the CIFAR-10 dataset:

```
python
Copy code
import torch
import torchvision
import torchvision.transforms as transforms

# Define a transformation to apply to the images
transform = transforms.Compose([
    transforms.ToTensor(),
    transforms.Normalize((0.5, 0.5, 0.5), (0.5, 0.5, 0.5))
])

# Download and load the training dataset
trainset = torchvision.datasets.CIFAR10(root='./data', train=True,
download=True, transform=transform)
trainloader = torch.utils.data.DataLoader(trainset, batch_size=32,
shuffle=True)

# Download and load the test dataset
testset = torchvision.datasets.CIFAR10(root='./data', train=False,
download=True, transform=transform)
```

```
testloader = torch.utils.data.DataLoader(testset, batch_size=32,
shuffle=False)

# Classes in CIFAR-10
classes = ('plane', 'car', 'bird', 'cat', 'deer', 'dog', 'frog', 'horse',
'ship', 'truck')
```

Accessing CIFAR-100:

If you want to work with the CIFAR-100 dataset, you can simply replace `CIFAR10` with `CIFAR100` in the code above:

```
python
Copy code
trainset = torchvision.datasets.CIFAR100(root='./data', train=True,
download=True, transform=transform)
testset = torchvision.datasets.CIFAR100(root='./data', train=False,
download=True, transform=transform)
```

This will load the CIFAR-100 dataset, which has 100 classes instead of 10.

2. **Task:** Implement transfer learning by fine-tuning a pretrained model like ResNet on a custom dataset.

Lab Exercises - 3 Solutions

Files

Exercise 1 – Display the following menu to the user:

1) Create a new file
2) Display the file
3) Add a new item to the file

Make a selection to 1, 2 or 3

Ask the user to enter 1, 2 or 3. If they select anything other than 1, 2 and 3, it should display a suitable error message.

If they select 1, ask the user to enter a school subject and save it to a new file called “Subjects.txt”. It should overwrite any existing file with a new file. If they select 2, display the contents of “Subjects.txt” file. If they select 3, ask the user to enter a new subject and save it to the file and then display the entire content of the file.

Run the programme several times to test the options.

```
print ("1) Create a new file")
print ("2) Display the file")
print("3) Add a new item to the file")
selection = int(input("Make a selection 1, 2 or 3: "))
if selection == 1:
    subject = input("Enter a school subject: ")
    file = open("Subject.txt", "w")
    file.write(subject + "\n")
    file.close()
elif selection == 2:
    file = open("Subject.txt", "r")
    print(file.read())
elif selection == 3:
    file = open("Subject.txt", "a")
    subject = input("Enter a school subject: ")
    file.write(subject + "\n")
    file.close()
    file = open("Subject.txt", "r")
    print(file.read())
else:
    print("Invalid option")
```

Exercise 2 - Create a simple maths quiz that will ask the user for their name and then generate two random questions. Store their name, the questions that were asked, their answers and their final score in a .csv file. Whenever the programme is run it should add to the csv file, and not overwrite anything.

```
import csv
import random

score = 0
name = input("What is your name: ")
q1_num1 = random.randint(10,50)
q1_num2 = random.randint(10,50)
question1 = str(q1_num1) + "+" + str(q1_num2) + " = "
ans1 = int(input(question1))
realans1 = q1_num1 + q1_num2
if ans1 == realans1:
    score = score + 1
q2_num1 = random.randint(10,50)
q2_num2 = random.randint(10,50)
question2 = str(q1_num1) + " + " + str(q2_num2) + " = "
ans2 = int(input(question2))
realans2 = q2_num1 + q2_num2
if ans2 == realans2:
    score = score + 1

file = open("QuizScore.csv","a")
newrecord = name+","+question1+","+str(ans1)+","+question2+","+str(ans2)+","+str(score)+"\n"
file.write(str(newrecord))
```

Exercise 3 - (**Function and .csv files**) Create the following menu:

1) Add to file
2) View all records
3) Quit programme

Enter the number of your selection:

If the user selects 1, allow them to add to a file called Salaries.csv which will store their name and salary. If they select 2 it should display all records in the Salaries.csv file. If they select 3 it should stop the programme. If they select an incorrect option, they should see an error message. They should keep returning to the menu until they select option 3.

```
import csv

def addtofile():
    file = open("Salaries.csv", "a")
    name = input("Enter name: ")
    salary = int(input("Enter Salary: "))
    newrecord = name + ", " + str(salary) + "\n"
    file.write(str(newrecord))
    file.close()

def viewrecords():
    file = open("salaries.csv", "r")
    for  row in file:
        print(row)
    file.close()

tryagain = True
while tryagain == True:
    print("1) Add to file")
    print("2) View all records")
    print("3) Quit program")
    print()
    selection = input("Enter the number of your selection: ")
    if selection == "1":
        addtofile()
    elif selection == "2":
        viewrecords()
    elif selection == "3":
        tryagain = False
    else:
        print("Incorrect option")
```

Exercise 4 – (function and csv) In Python, it is not technically possible to directly delete a record from a .csv file. Instead, you need to save the file to a temporary list in Python, make the changes to the list and then overwrite the original file with the temporary list.

Change the previous programme to allow you to do this. Your menu should now look like this:

1) Add to file
2) View all records
3) Delete a record
4) Quit programme

```
import csv

def addtofile():
    file = open("Salaries.csv", "a")
    name = input("Enter name: ")
    salary = int(input("Enter Salary: "))
    newrecord = name + ", " + str(salary) + "\n"
    file.write(str(newrecord))
    file.close()

def viewrecords():
    file = open("salaries.csv", "r")
    for  row in file:
        print(row)
    file.close()

def deleterecord():
    file = open("Salaries.csv", "r")
    x = 0
    tmplist = []
    for row in file:
        tmplist.append(row)
    file.close()
    for row in tmplist:
        print(x, row)
        x = x + 1
    rowtodelete = int(input("Enter thr row number to delete: "))
    del tmplist[rowtodelete]
    file = open("Salaries.csv", "w")
    for row in tmplist:
        file.write(row)
    file.close()


tryagain = True
while tryagain == True:
    print("1) Add to file")
    print("2) View all records")
    print("3) Delete a record")
    print("4) Quit programme")
    print()
    selection = input("Enter the number of your selection: ")
    if selection == "1":
        addtofile()
    elif selection == "2":
        viewrecords()
    elif selection == "3":
        deleterecord()
    elif selection == "4":
        tryagain = False
    else:
        print("Incorrect option")
```

Exercise 5 - Sum a Collection of Numbers

Create a program that sums all of the numbers entered by the user while ignoring any input that is not a valid number. Your program should display the current sum after each number is entered. It should display an appropriate message after each non-numeric input, and then continue to sum any additional numbers entered by the user. Exit the program when the user enters a blank line. Ensures that your program works correctly for both integer and floating-point numbers.

```
# Read the first line of input from the user
line = input("Enter a number: ")
total = 0

# Keep reading until the user enters a blank line
while line != "":
    try:
        # Try and convert the line to a number
        num = float(line)
        # If the conversion succeds then add it to the total and display it
        total = total + num
        print("The total is now", total)

    except ValueError:
        # Display an error message before going on to read the next value
        print("That wasn't a number.")

    line = input("Enter a number: ")

# Display the total
print("The grand total is", total)
```

SQlite3

Exercise 6 – Create an SQL database called PhoneBook1 that contains a table called Names with the following data:

ID	First Name	Surname	Phone Number
1	Simon	Pierre	0142678 9056
2	Katarina	Iglesias	0203456 7078
3	Derrick	Brown	0122345 8765
4	John	Smith	0112653 2312
5	Mark	Isaac	01416571383

```
import sqlite3
# Connect to the database called PhoneBook or create one if there is none
with sqlite3.connect("PhoneBook1.db") as db:
    cursor = db.cursor()

#    Create a table called Names with foru fields
cursor.execute(""" CREATE TABLE IF NOT EXISTS Names(
id integer PRIMARY KEY,
firstname text,
surname text,
phonenumber text); """)

# Insert data into the table
cursor.execute(""" INSERT INTO Names(id,firstname,surname,phonenumber)
VALUES ("60", "Simon","Pierre","0142678 9056")""")
db.commit() # Saves the changes

# Insert data into the table Names
cursor.execute(""" INSERT INTO Names(id, firstname,surname,phonenumber)
VALUES ("70", "Katarina","Iglesias","0203456 7078")""")
db.commit() # saves the chnages

# Insert data into a table called Names
cursor.execute(""" INSERT INTO Names(id,firstname,surname,phonenumber)
VALUES ("30", "Derrick", "Brown", "0122345 8765")""")
db.commit() # saves the changes

# Insert data into a table called Names
cursor.execute(""" INSERT INTO Names(id,firstname,surname,phonenumber)
VALUES ("40", "John", "Smith", "0112653 2312")""")
db.commit() # saves the changes
# Insert data into a table called Names
cursor.execute(""" INSERT INTO Names(id,firstname,surname,phonenumber)
VALUES ("50", "Mark", "Isaac", "0141657 1383")""")
db.commit() # saves the changes

db.close() # close the database
```

Exercise 7 – Using the phonebook database, write a programme that will display the following menu

```
Main menu

1) View phone book
2) Add to phone book
3) Search for surname
4) Delete person from phone book
5) Quit
Enter your selection
```

If the user selects 1, they should be able to view the entire phonebook. If they select 2, it should allow them to add a new person to the phonebook. If they select 3, it should ask them for a surname and then display only the record of people with the same surname. If they select 4, it should ask for an ID and then delete that record from the table. If they select 5, it should end the programme.

Finally, the programme should display a suitable message if they enter an incorrect selection from the menu. They should return to the menu after each action, until they select 5.

```
def viewphonebook():
    cursor.execute("SELECT * FROM Names")
    for x in cursor.fetchall():
        print(x)
def addtophonebook():
    newid = int(input("Enter ID: "))
    newfname = input("Enter first name: ")
    newsname = input("Enter surname: ")
    newpnum = input("Enter phone number: ")
    cursor.execute("""INSERT INTO Names (id,firstname,surname,phonenumber)
        VALUES (?,?,?,?)""", (newid,newfname,newsname,newpnum))
    db.commit()

def selectname():
    selectsurname = input("Enter a surname: ")
    cursor.execute("SELECT * FROM Names WHERE surname = ?", [selectsurname])
    for x in cursor.fetchall():
        print(x)

def deletedata():
    selectid = int(input("Enter ID: "))
    cursor.execute("DELETE FROM Names WHERE id = ?", [selectid])
    cursor.execute("SELECT * FROM Names")
    for x in cursor.fetchall():
        print(x)
    db.commit()

with sqlite3.connect("PhoneBook1.db") as db:
    cursor = db.cursor()

def main():
    again = "y"
    while again == "y":
        print()
        print("Main Menu")
        print()
        print("1) View phone book")
        print("2) Add to phone book")
        print("3) Search for surname")
        print("4) Delete person from phone book")
        print("5) Quit")
        print()
        selection = int(input("Enter your selection: "))
        print()

        if selection == 1:
            viewphonebook()
        elif selection == 2:
            addtophonebook()
        elif selection == 3:
            selectname()
        elif selection == 4:
            deletedata()
        elif selection == 5:
            again = "n"
        else:
            print("Incorrect selection entered")
main()
db.close()
```

Team Application Exercises (tAPP-3)

Problem 1

Write a code that creates a database called employees. That database contains a table called contactDetails. The following output will be produced when the program is executed.

```
(42, 'Derrick', 'Brown', '0122345 8765')
(62, 'Simon', 'Pierre', '0142678 9056')
(72, 'Katarina', 'Iglesias', '0203456 7078')
```

Problem 1 Solution

```
import sqlite3
# Connect to the database called PhoneBook or create one if there is none
with sqlite3.connect("employees.db") as db:
    cursor = db.cursor()

#    Create a table called Names with foru fields
cursor.execute(""" CREATE TABLE IF NOT EXISTS contactDetails(
id integer PRIMARY KEY,
firstname text,
surname text,
phonenumber text); """)

# Insert data into the table
cursor.execute(""" INSERT INTO contactDetails(id,firstname,surname,phonenumber)
VALUES ("42", "Simon","Pierre","0142678 9056")""")
db.commit() # Saves the changes

# Insert data into the table Names
cursor.execute(""" INSERT INTO contactDetails(id, firstname,surname,phonenumber)
VALUES ("62", "Katarina","Iglesias","0203456 7078")""")
db.commit() # saves the chnages

# Insert data into a table called Names
cursor.execute(""" INSERT INTO contactDetails(id,firstname,surname,phonenumber)
VALUES ("72", "Derrick", "Brown", "0122345 8765")""")
db.commit() # saves the changes


cursor.execute("SELECT * FROM contactDetails")
for x in cursor.fetchall():
    print(x)

db.close() # close the database
```

<u>Problem 2</u> - Debugging code

A user has written a Python script that is supposed to create a new CSV file, ask the user to input the name and age of a person, and then add this information to the end of the file. The script is also supposed to allow multiple records to be added in one session. However, when running the code, it does not execute due to some errors.

Task 1: Identify the errors (making sure you indicate the line number) and how many errors have you found?

Code with errors

Line 14: csv and “w” instead of “m”

Line 22: mode should be a

Line 29: it should be age =

Line 42: it should be create_csv_file

```python
import csv

# Function to create a new CSV file
def create_csv_file():
    # Create and open the CSV file for writing
    file = open("ages.csv", "w")
    writer = csv.writer(file)
    file.close()
    print(f"File 'ages.csv' created successfully.")

# Function to add a new record to the CSV file
def add_record_to_csv():
    # Open the CSV file in append mode
    file = open("ages.csv", mode='a')
    writer = csv.writer(file)

    # Ask the user to enter the name and age
    add_more = 'yes'
    while add_more == 'yes':
        name = input("Enter the name: ")
        age = input("Enter the age: ")
        writer.writerow([name, age])
        print(f"Record for {name}, aged {age}, added successfully.")

        # Ask if the user wants to add another record
        add_more = input("Do you want to add another record? (yes/no): ").strip().lower()

    # Close the file after adding all records
    file.close()

# Main function to execute the script
def main():
    # Create the CSV file
    create_csv_file()

    # Add records to the CSV file
    add_record_to_csv()

# Run the main function
if __name__ == "__main__":
    main()
```

Problem 3 - Parsons puzzle

Task:

Problem 3 Solution

1	`import sqlite3`
2	`conn = sqlite3.connect('EmployeeDB.db')`
3	`cursor = conn.cursor()`

4	`cursor.execute('''`
	`CREATE TABLE IF NOT EXISTS EmployeePerformance (`
	`EmployeeID TEXT,`
	`Date TEXT,`
	`Department TEXT,`
	`TasksCompleted INTEGER,`
	`HoursWorked INTEGER,`
	`PerformanceScore REAL`
	`)`
	`''')`

5	`for _, row in df.iterrows():`

```
6       cursor.execute('''
            INSERT OR IGNORE INTO EmployeePerformance
            (EmployeeID, Date, Department, TasksCompleted, HoursWorked, Performance
            VALUES (?, ?, ?, ?, ?, ?)
        ''', (row['EmployeeID'], row['Date'].strftime('%Y-%m-%d'), row['Department'
        row['TasksCompleted'], row['HoursWorked'], row['PerformanceScore']))
7   conn.commit()
8   conn.close()
```

Problem 4:

This program below should allow the user to convert between miles and kilometres. 1 kilometre = 0.6214 miles and 1 mile = 1.6093 kilometres.

```
Choose an option:
1. Convert Miles to Kilometers
2. Convert Kilometers to Miles
3. Exit
Enter your choice (1/2/3):
```

Task: Using a pen and paper, complete the code (with correct indentation).

```
52      def miles_to_kilometers(miles):
53          return miles * 1.6093
54
55      def kilometers_to_miles(kilometers):
56          return kilometers * 0.6214
57
58      def main():
59          print("Welcome to the Miles and Kilometers Converter!")
60
61          while True:
62              print("\nChoose an option:")
63              print("1. Convert Miles to Kilometers")
64              print("2. Convert Kilometers to Miles")
65              print("3. Exit")
66
67              choice = input("Enter your choice (1/2/3): ")
68
69              if choice == '1':
70                  miles = float(input("Enter distance in miles: "))
71                  kilometers = miles_to_kilometers(miles)
72                  print(f"{miles} miles is equal to {kilometers:.2f} kilometers.")
73
74              elif choice == '2':
75                  kilometers = float(input("Enter distance in kilometers: "))
76                  miles = kilometers_to_miles(kilometers)
77                  print(f"{kilometers} kilometers is equal to {miles:.2f} miles.")
78
79              elif choice == '3':
80                  print("Exiting the converter. Goodbye!")
81                  break
82
83              else:
84                  print("Invalid choice, please try again.")
85
86      if __name__ == "__main__":
87          main()
```

Problem 5

The following code is intended to load a CSV file and filter data by the most recent year. However, it throws an error when trying to access the ‘Date’ column. Debug the code to fix the issue.

Solution

The error occurred because the 'Date' column was not parsed as a datetime object. The correct approach is to use the parse_dates parameter in pd.read_csv() to ensure the 'Date' column is correctly parsed.

```
import pandas as pd

# Load the data
df = pd.read_csv('employee_data.csv', parse_dates=['Date'])

# Filter data to include only the most recent year
most_recent_year = df['Date'].dt.year.max()
df_recent = df[df['Date'].dt.year == most_recent_year]

print(df_recent.head())
```

Problem 6 - Debugging

Which of the following codes intended to create a PyTorch tensor and apply the ReLU activation function will throw an error? What is the error?

(1)

```
#######Debug problem 3 ######
import torch

# Create a PyTorch tensor
x = torch.linspace(-10, 10, 100)

# Apply the ReLU activation function
y = torch.nn.functional.relu(x)

print(y)
```

(2)

```
import torch
# Create a PyTorch tensor
z = torch.linspace(-10, 10, 100)

# Apply the ReLU activation function
y = torch.n.functional.relu(x)

print(y)
```

Solution

(2)

AttributeError: module 'torch' has no attribute 'n'.

```
import torch
# Create a PyTorch tensor
z = torch.linspace(-10, 10, 100)

# Apply the ReLU activation function
y = torch.nn.functional.relu(x)

print(y)
```

Lab Exercises - 2

Python Repetition

For Loops Exercises

1 - Write a programme that asks the user to enter their name and a number. If the number is less than 10, then display their name that number times; otherwise, display the message “Too high” three times.

```
name = input("Enter your name: ")
num = int(input("Enter a number: "))
if num < 10:
    for i in range (0, num):
        print(name)
else:
    for i in range (0,3):
        print("Too high")
```

2 – Set a variable called total to 0. Your programme should ask the user to enter five numbers and after each input, ask them if they want that number to be included. If they do, the programme should then add the number to the total. If they do not want it included, the programme shouldn’t add the number to the total. After the user has entered all five numbers, display the total.

```
total = 0
for i in range (0,5):
    num = int(input("Enter a number: "))
    ans = input("Do you want this number included? (y/n) ")
    if ans == "y":
        total = total + num
print(total)
```

3 - Write a programme that asks which direction the user wants to count (up or down). If they select up, then the programme should ask them for the top number and count from 1 to that number. If the user selects down, the programme should ask them to enter a number below 20 and then count down from 20 to that number. If the user entered something other than up or down, display the message “I don’t understand”.

```
direction = input("Do you want to count up or down? (u/d) ")
if direction == "u":
    num = int(input("What is the top number? "))
    for i in range(1, num+1):
        print(i)
elif direction == "d":
    num = int(input("Enter a number below 20: "))
    for i in range(20,num-1, -1):
        print(i)
else:
    print("I don't understand")
```

While Loop exercises

4 - Write a programme that asks the user to enter a number and then enter another number. The programme should add these two numbers together and then ask if they want to add another number. If the user enters “y”, the programme will ask them to enter another number and will keep adding numbers until the user stops answering “y”. Once the loop has stopped, display the total.

```
num1 = int(input("Enter a number: "))
total = num1
again = "y"
while again == "y":
    num2 = int(input("Enter another number: "))
    total = total + num2
    again = input("Do you want to add another number? (y/n) ")
print("The total is ", total)
```

5 – Create a variable called compnum and set the value to 50. Ask the user to enter a number. While their guess is not the same as the compnum value, tell them if their guess is too high or too low and ask them to have another guess. If they enter the same value as compnum, display the message “Well done, you took [count] attempts”.

```
compnum = 50
guess = int(input("Can you guess the number I am thinking of? "))
count = 1
while guess != compnum:
    if guess < compnum:
        print("Too low")
    else:
        print("Too high")
    count = count+1
    guess = int(input("Have another guess: "))
print("Well done, you took ", count, "attempts")
```

6 – Using the song “10 green bottles”, display the lines “There are [num] green bottles hanging on the wall, [num] green bottle hanging on the wall, and is 1 green bottle should accidentally fall”. Then ask the question “how many green bottles will be hanging on the wall?” If the user answers correctly, display the message “There will be [num] green bottles hanging on the wall”. If they answer incorrectly, display the message “No, try again” until they get it right. When the number of green bottles gets down to 0, display the message “There are no more green bottles hanging on the wall”.

```
num = 10
while num >0:
    print("There are ", num, "green bottles hanging on the wall.")
    print(num, "green bottles hanging on the wall.")
    print("And if 1 green bottle should accidentally fall, ")
    num = num - 1
    answer = int(input("How many green bottles will be hanging on the wall? "))
    if answer == num:
        print("There will be ", num, "green bottles hanging on the wall.")
    else:
        while answer != num:
            answer = int(input("No, try again: "))
print("There are no more green bottles hanging on the wall.")
```

Function Exercises

7 – Define a function that will ask the user to enter a number and save it as the variable “num”. Define another function that will use “num” and count 1 to that number.

```
def ask_value():
    num = int(input("Enter a number: "))
    return num

def count(num):
    n = 1
    while n <= num:
        print(n)
        n = n + 1

def main():
    num = ask_value()
    count(num)

main()
```

8 – Create a programme that will allow the user to easily manage a list of names. You should display a menu that will allow them to add a name to the list, change the name in the list, delete a name from the list from the list or view all the names in the list. There should be a menu option to allow the user to end the programme. If they select an option that is not relevant, then it should display a suitable message. After they have made a selection to either, add a name, change a name delete a name or view all the names, they should see the menu again without having to restart the programme. The programme should be as easy to use as possible.

```
def add_name():
    name = input("Enter a new name: ")
    names.append(name)
    return names

def change_name():
    num = 0
    for x in names:
        print(num, x)
        num = num +1
    select_num = int(input("Enter the number of the name you want to change: "))
    name = input("Enter new name: ")
    names[select_num] = name
    return names

def delete_name():
    num = 0
    for x in names:
        print(num, x)
        num = num + 1
    select_num = int(input("Enter the number of the name you want to delete: "))
    del names[select_num]
    return names

def view_names():
    for x in names:
        print(x)
    print()

def main():
    again = "y"
    while again == "y":
        print("1) Add a name")
        print("2) Change a name")
        print("3) Delete a name")
        print("4) View a name")
        print("5) Quit")
        selection = int(input("What do you want to do? "))
        if selection == 1:
            names = add_name()
        elif selection == 2:
            names = change_name()
        elif selection == 3:
            names = delete_name()
        elif selection == 4:
            names = view_names()
        elif selection == 5:
            again = "n"
        else:
            print("Incorrect option: ")
        data = (names, again)
names = []
main()
```

Random Exercises

9 – Write a programme that displays a random fruit from a list of 5 fruits.

```
import random
### Random exercises
fruit = random.choice(['apple', 'orange','mango','grape','banana'])
print(fruit)
```

10 – Write a basic programme that randomly chooses either heads or tails (“h” or “t”). The programme asks the user to make a choice. If their choice is the same as the randomly selected value, display the message “You win”, otherwise display “Bad luck”. At the end, tell the user if the computer selected heads or tails.

```
coin = random.choice(["h", "t"])
guess = input("enter (h)ead or (t)ail: ")
if (guess == coin):
    print("you win")
else:
    print("bad luck")
if coin == "h":
    print(" the computer selected head")
else:
    print("The computer selected tail")
```

11 - Write a programme that randomly chooses a number between 1 and 5. The programme should ask the user to pick a number. If they guess correctly, display the message “Well done”, otherwise the programme should tell them if they are too high or too low and should ask them to pick a second number. If they guess correctly on their second guess, display the message “Correct”, otherwise display “You lose”.

```
num = random.randint(1,5)
guess = int(input("Enter a number: "))
if guess == num:
    print("Well done")
elif guess > num:
    print("Too high")
    guess = int(input("Guess again: "))
    if guess == num:
        print("Correct")
    else:
        print("You lose")
elif guess < num:
    print("Too low")
    guess = int(input("Guess again: "))
    if guess == num:
        print("Correct")
    else:
        print("You lose")
```

12 – Write a programme that displays five colours and asks the user to pick one. If they pick the same as the programme has chosen, say “Well done”, otherwise display a witty answer which involves the correct colour, e.g. “I bet you are GREEN with envy” or “You are probably feeling BLUE right now”. The programme should ask them to guess again; if the user has still not got it right, it keeps giving them the same clue and it asks the user to enter a colour until they guess it correctly.

```
colour = random.choice(["red", "blue", "green", "white", "pink"])
print("Select from red, blue, green, white or pink")
tryagain = True
while tryagain == True:
    theirchoice = input("Enter a colour: ")
    theirchoice = theirchoice.lower()
    if colour == theirchoice:
        print("Well done!")
        tryagain = False
    else:
        if colour == "red":
            print("I bet you are seeing RED right now")
        elif colour == "blue":
            print("Don't feel BLUE")
        elif colour == "green":
            print("I bet you are GREEN with envy right now.")
        elif colour == "white":
            print(" Your brain is blank as a WHITE sheet, as you did not guess correctly" )
        elif colour == "pink":
            print("Shame you are not feeling in the PINK, as you got it wrong")
```

Turtle Graphics exercises

13 - Draw a circle.

```
import turtle

for i in range (0,360):

    turtle.forward(1)
    turtle.right(1)

turtle.exitonclick()
```

14 - Draw an octagon that uses a different colour (randomly selected from a list of six possible colours) for each line.

```
import turtle
import random

turtle.pensize(3)

for i in range (0,8):
    turtle.color(random.choice(["red","blue","yellow","green","pink","orange"]))
    turtle.forward(50)
    turtle.right(45)
```

15 - Draw a pattern that will change each time the programme is run. Use random function to pick the number of lines, the length of each line and the angle of each turn.

```
import turtle
import random

lines = random.randint(5,20)

for x in range (0,lines):
    length = random.randint(25,100)
    rotate = random.randint(1,365)
    turtle.forward(length)
    turtle.right(rotate)

turtle.exitonclick()
```

Data structure exercises

Data structure – List Exercise

16 - Create a list of four three-digit numbers. Display the list to the user, showing each item from the list on a separate line. Ask the user to enter a three-digit number. If the number they have typed in matches one in the list, display the position of that number in the list otherwise display the message "That's not in the list"

```
nums = [123,345,234,765]
for i in nums:
    print(i)
selection = int(input("Enter a number from the list: "))
if selection in nums:
    print(selection, "is in position", nums.index(selection))
else:
    print("That is not in the list")
```

17 - Write a programme that asks the user to enter the names of three people they want to invite to a party and store them in a list. After they have entered all three names, ask them if they want to add another. If they do, allow them to add more names until they say "no". When they say "no", display how many people they have invited to the party.

```
name1 = input("Enter a name of somebody you want to invite to your party: ")
name2 = input("Enter another name: ")
name3 = input("Enter a third name: ")
party = [name1,name2,name3]
another = input("Do you want to invite another (y/n)?: ")
while another == "y":
    newname = party.append(input("Enter another name: "))
    another = input("Do you want to invite another (y/n)?: ")
print("You have ", len(party), "people coming to your party")
```

Data structure – Dictionary Exercises

18 - Create a programme that asks the user to enter four of their favourite foods and store them in a dictionary so that they are associated with keys starting from 1. Display the dictionary in full, showing the key-value pairs. The programme should ask the user which key-value pair they want to get rid of and remove it from the dictionary. Finally, the programme should sort the remaining data and display the dictionary.

```
food_dictionary = {}
food1 = input("Enter a food you like: ")
food_dictionary[1] = food1
food2 = input("Enter another food you like: ")
food_dictionary[2] = food2
food3 = input("Enter another food you like: ")
food_dictionary[3] = food3
food4 = input("Enter another food you like: ")
food_dictionary[4] = food4
print(food_dictionary)
dislike = int(input("Which of these do you want to get rid of? "))
del food_dictionary[dislike]
print(sorted(food_dictionary.values()))
```

Data structure – tuple exercises

19 - Create a tuple containing the names of five countries and display the whole tuple. Ask the user to enter one of the countries that have been shown to them and then display the index number (i.e. position in the list) of that item in the tuple.

Next, the programme should ask the user to enter a number (index) and display the country in that position.

```
country_tuple = ("Cameroon", "Equatorial Guinea", "Guinea-Bissao", "Papua New Guinea", "Guinea-Conakry")
print(country_tuple)
print()
country = input("Please enter one of the countries from above: ")
print(country, "has index number", country_tuple.index(country))
print()
num = int(input("Enter a number between 0 and 4: "))
print(country_tuple[num])
```

Numeric Arrays

20 – Write a programme that asks the user for a list of five integers. Store them in an array. Sort the list and displays it in reverse order.

```
from array import *

nums = array('i',[])

for i in range (0,5):
    num = int(input("Enter a number: "))
    nums.append(num)
nums = sorted(nums)
nums.reverse()

print(nums)
```

21 – Create an array which will store a list of integers. Generate five random numbers and store them in the array. Display the array (showing each item on a separate line).

```
from array import *
import random

nums = array('i',[])

for i in range (0,5):
    num = random.randint(1,100)
    nums.append(num)

for i in nums:
    print(i)
```

22 – Create two arrays (one containing three numbers that the user enters and one containing a set of five random numbers). Join these two arrays together into one large array. Sort this large array and display it so that each number appears on a separate line.

```
from array import *
import random

num1 = array('i',[])
num2 = array('i',[])


for i in range (0,3):
    num = int(input("Enter a number: "))
    num1.append(num)

for i in range (0,5):
    num = random.randint(1,100)
    num2.append(num)

num1.extend(num2)
num1 = sorted(num1)

for i in num1:
    print(i)
```

23 – Ask the user to enter five numbers. Sort them in order and present them to the user. Ask them to select one of the numbers. Remove it from the original array and save it in a new array.

```
from array import *
import random

nums = array('i',[])


for i in range (0,5):
    num = int(input("Enter a number: "))
    nums.append(num)
nums = sorted(nums)
for i in nums:
    print(i)

num = int(input("Select a number from the array: "))
if num in nums:
    nums.remove(num)
    num2 = array('i',[])
    num2.append(num)
    print(nums)
    print(num2)
else:
    print("That is not a value in the array")
```

24 – Display and array of five numbers. Ask the user to select one of the numbers. Once they have selected a number, display the position of that number in the array. It they enter something that is not in the array, ask them to try again until they select a relevant number.

```
nums = array('i',[4,6,8,2,5])


for i in nums:
    print(i)

num = int(input("Select one of the numbers : "))
tryagain = True
while tryagain == True:
    if num in nums:
        print("This is in position", nums.index(num))
        tryagain = False
    else:
        print("Not in array")
        num = int(input("Select on of the numbers"))
```

25 – Create an array of five numbers between 10 and 100 which each have two decimal places. Ask the user to enter a whole number between 2 and 5. If they enter something outside of that range, display a suitable error message and ask them again to try until they enter a valid number. Divide each of the numbers in the array by the number entered and display the answers shown to two decimal places.

```
from array import *
import math

nums = array('f',[34.76,35.56,67.09, 23.12, 89.34])
tryagain = True
while tryagain == True:
    num = int(input("Enter a number between 2 and 5: "))
    if num <2 or num >5:
        print("Incorrect value, try again.")
    else:
        tryagain = False
for i in range(0,5):
    ans = nums[i]/num
    print(round(ans,2))
```

2 D Lists and Dictionaries

26 – Create the following using a simple 2D list and ask the user to select a row and a column and display that value.

	0	1	2
0	2	5	8
1	3	7	4
2	1	6	9
3	4	2	0

```
list = [[2,5,8],[3,7,4],[1,6,9],[4,2,0]]
row = int(input("Select a row: "))
col = int(input("Select a column: "))
print(list[row][col])
```

27 - Using the 2D list in the previous exercise, ask the user which row they would like displayed and display just that row. Ask them to enter a new value and add it to the end of the row and display row again.

```
list = [[2,5,8],[3,7,4],[1,6,9],[4,2,0]]
row = int(input("Select a row: "))
print(list[row])
newvalue = int(input("Enter a new number: "))
list[row].append(newvalue)
print(list[row])
```

28 – Change the previous programme to ask the user which row they want displayed. Display that row. Ask the user which column in that row they want displayed and display the value in held in there. Ask the user if they want to change the value and change the data. Finally, display the whole row again.

```
list = [[2,5,8],[3,7,4],[1,6,9],[4,2,0]]
row = int(input("Select a row: "))
print(list[row])
col = int(input("Select a column: "))
print(list[row][col])
chnage = input("Do you want to chnage the value? (y/n) ")
if chnage == "y":
    newvalue = int(input("Enter new value: "))
    list[row][col] = newvalue
print(list[row])
```

29 – Ask the user to enter the name age and show size for the four people in a list and display the names and ages of all the people in the list but do not show their show size.

```
list = {}
for i in range(0,4):
    name = input("Enter name: ")
    age = int(input("Enter age: "))
    shoe = int(input("Enter showe size: "))
    list[name] = {"Age":age,"Shoe size":shoe}

for name in list:
    print((name),list[name]["Age"])
```

30 – After gathering the four names, ages and shoe sizes, ask the user to enter the name of the person they want to remove from the list. Delete this row from the data and display the other rows on separate lines.

```
list = {}
for i in range(0,4):
    name = input("Enter name: ")
    age = int(input("Enter age: "))
    shoe = int(input("Enter showe size: "))
    list[name] = {"Age":age,"Shoe size":shoe}

getrid = input("Who do you want to remove from the list? ")
del list[getrid]

for name in list:
    print((name),list[name]["Age"], list[name]["Shoe size"])
```

Team Application Exercises (tAPP-2) - Solutions

Problem 1

The code below should repeat the user's message the number of times that user wants. The code has 5 lines that are not in the correct order. Rearrange the code by entering the number that corresponds to each line of code.

Solution

3	`n = int(input("How many times should it be reapted? "))`
4	`for i in range(n):`
5	`print(message)`
2	`while message != "":`
1	`message = input("Enter a message (blank to quit): ")`

Problem 2

This code will determine whether or not a string entered by the user is a palindrome or not is not. The code has 11 lines that are not in the correct order. Rearrange the code by entering the number that corresponds to each line of code (whatever you think should be the first line of code will have number 1 in front of it).

Solution

6	`is_palindrome = False`
8	`if is_palindrome:`
10	`else:`
3	`i = 0`
5	`if line[i] != line[len(line) - i - 1]:`
1	`line = input("Enter a string: ")`
11	`print(line, "is not a palindrome")`
7	`i = i + 1`
2	`is_palindrome = True`
4	`while i < len(line) / 2 and is_palindrome:`
9	`print(line, "is a palindrome")`

Problem 3

When the following code runs, it asks for the name and then the age. If the age is under or equal to 10, the message is Hi followed by the name. The code has 13 lines that are not in the correct order. Rearrange the code by entering the number that corresponds to each line of code.

Solution

11	`def main():`

7	`if age <= 10:`
5	`return data_tuple`
8	`print("Hi", username)`
2	`username = input("Enter your user name: ")`
13	`message(username, age)`
1	`def get_data():`
6	`def message (username, age):`
3	`age = int(input("Enter your age: "))`
4	`data_tuple = (username, age)`
12	`username, age = get_data()`
10	`print("Hello", username)`
9	`else:`

Problem 4: Debugging code

The code should display a menu, add or subtract two numbers and display the corresponding messages. However, the code is not working. Identify the issues that affect this code.

Solution: There are 10 errors in total.

Line 11: attribute error – It should be randint.

Line 16: Name error – it should be answers (s at the end)

Line 20: Type error – it should be randint instead of random.

Line 23: Name error – it should be assignment operator instead of equality operator

Line 24: Syntax error – remove the colon

Line 28: Indentation error

Line 37: Syntax error –it should be equality operator instead of assignment operator

Line 38: UnboundLocalError – the local variable user_answer should not have the ‘s’ at the end.

Line 42: Synthax error – missing comma

Line 43: indentation error

```python
import random

def addition():
    num1 = random.randint(5,20)
    num2 = random.randint(5,20)
    print(num1, "+", num2, "= ")
    user_answer = int(input("Your answer: "))
    actual_answer = num1 + num2
    answers = (user_answer, actual_answer)
    return answers

def subtraction():
    num3 = random.randint(25,50)
    num4 = random.randint(1,25)
    print(num3, "-", num4, "= ")
    user_answer = int(input("Your answer: "))
    actual_answer = num3 - num4
    answers = (user_answer, actual_answer)
    return answers

def check_answer(user_answer,actual_answer):
    if user_answer == actual_answer:
        print("Correct")
    else:
        print("Incorrect, the answer is ", actual_answer)

def main():
    print("1) Addition")
    print("2) Subtraction")
    selection = int(input("Enter 1 or 2: "))
    if selection == 1:
        user_answer, actual_answer = addition()
        check_answer(user_answer, actual_answer)
    elif selection == 2:
        user_answer, actual_answer = subtraction()
        check_answer(user_answer, actual_answer)
    else:
        print("Incorrect selection")

main()
```

Problem 5:

This programme will ask the user to pick a low and high number, and then generate a random number between those two values and store it in a variable called “comp_num”. It will give the instruction “I am thinking of a number …” and then ask the user to guess the number they are thinking of.

`Incomplete function`: The third function should check to see if the `comp_num` is the same as the user’s guess. If it is, it should display the message “Correct, you win”, otherwise it should keep looking, telling the user if they are too low or too high and asking them to guess again (using a variable called `try_again`) until they guess correctly.

Task: Using a pen and paper, complete the function (with correct indentation).

Solution

```
import random

def pick_number():
    low = int(input("Enter the bottom of the range: "))
    high = int(input("Enter the top of the range: "))
    comp_num = random.randint(low, high)
    return comp_num

def first_guess():
    print("I am thinking of a number ... ")
    guess = int(input("What am I thinking of: "))
    return guess

def check_answer(comp_num, guess):
    try_again = True
    while try_again == True:
        if comp_num == guess:
            print("Correct, you win.")
            try_again = False
        elif comp_num > guess:
            guess = int(input("Too low, try again: "))
        else:
            guess = int(input("Too high, try again: "))

def main():
    comp_num = pick_number()
    guess = first_guess()
    check_answer(comp_num, guess)


main()
```

Lab Exercises – 1: Solutions

Python Basics - Exercises

The following exercise should use Python Basics

Exercise: Day old Bread

A bakery sells loaves of bread for £ 3.49 each. Day old bread is discounted by 60 percent. Write a programme that begins by reading the number of loaves of day old bread being purchased from the user. Then your programme should display the regular price for the bread, the discount because it is a day old, and the total price. Each of these amounts should be displayed on its own line with an appropriate label. All of the values should be displayed using two decimal places, and the decimal points in all the numbers should be aligned when reasonable values are entered by the user.

```
##
# DAY OLD BREAD
#
# Compute the price of a day old bread order.
#

BREAD_PRICE = 3.49
DISCOUNT_RATE = 0.60

# Read the number of loaves from the user
num_loaves = int(input("Enter the number of day old loaves: "))

# Compute the discount and total price
regular_price = num_loaves * BREAD_PRICE
discount = regular_price * DISCOUNT_RATE
total = regular_price - discount

# Display the result
print("Regular price: %5.2f" % regular_price)
print("Discount:      %5.2f" % discount)
print("Total:         %5.2f" % total)
```

Exercise: Length and Slicing

Write a programme that asks the user to type in the first line of their favourite song and display the length of that string. The programme should also ask a starting number and an ending number and then display just that section of the text.

```
phrase = input("Enter the first line of your favourite song: ")
length = len(phrase)
print("This has", length, "letters in it")
start = int(input("Enter a starting number: "))
end = int(input("Enter an end number: "))
part = (phrase[start:end])
print(part)
```

Exercise: Upper or Lower case name

Write a programme that asks the user to enter their first name. If the length of the first name is under five characters, the programme should ask them to enter their surname and join them together (without a space) and display the name in upper case. If the length of their first name is five or more characters, display their first name in lower case.

```
name = input("Enter your firstname: ")
if len(name) < 5:
    surname = input("Enter your surname: ")
    name = name+surname
    print(name.upper())
else:
    print(name.lower())
```

Exercise: Pig Latin

Pig Latin takes the first consonant of a word, moves it to the end of the word and adds on an "`ay`". If a word begins with a vowel you just add "`way`" to the end. For example, `pig` becomes `igpay` banana becomes `ananabay`, and aardvark becomes `aarvarkway`. Create a programme that will ask the user to enter a word and change it into Pig Latin. Make sure the new word is displayed in lower case.

```
word = input("Please enter a word: ")
first = word[0]
length = len(word)
rest = word[1: length]
if first != "a" and first != "e" and first != "i" and first !="o" and first !="u":
    newword = rest + first + "ay"
else:
    newword = word + "way"
print (newword.lower())
```

<u>Decision Making Exercises</u>

Exercise: Umbrella or no umbrella

Write a programme that asks the user if it is raining and convert their answer to lower case so that it doesn't matter what case they type in. If they answer "yes", ask if it is windy. If they answer "yes" to the second question, display the answer "It's too windy for an umbrella", otherwise display the message "Take an umbrella". If they did not answer "yes" to the first question, display the answer "Enjoy your day".

```
raining = input("Is it raining? ")
raining = str.lower(raining)
if raining == "yes":
    windy = input("Is it windy? ")
    windy = str.lower(windy)
    if windy == "yes":
        print("It is too windy for an umbrella")
    else:
        print("Take an umbrella")
else:
    print("Enjoy your day")
```

Exercise: Your age

Write a programme that asks the user's age. If they are 18 or over, display the message "you can vote". If they are 17, display the message "You can learn how to drive". If they are 16, display the message "you can buy a lottery ticket" and if they are under 16, display the message "You can go Trick-or-Treating".

```
##
# Your age
#

age = int(input("What is your age? "))
if age >= 18:
    print("you can vote")
elif age == 17:
    print("You can learn to drive")
elif age == 16:
    print("You can buy a lottery ticket")
else:
    print("You can go Trick-or-Treating")
```

Exercise: Even or Odd

Write a programme that reads an integer from the user. Then your programme should display a message indicating whether the integer is even or odd.

```
##
# EVEN OR ODD
#
# Determine and display whether an integer entered by the user is even or odd.
#

# Read the integer from the user
num = int(input("Enter an integer: "))

# Determine whether it is even or odd by using the modulus (remainder) operator
if num % 2 == 1:
    print(num, "is odd.")
else:
    print(num, "is even.")
```

Exercise: Vowel or Consonant

In the exercise, you will create a programme that reads a letter of the alphabet from the user. If the user enters `a`, `e`, `i`, `o` or `u` then your programme should display a message indicating that the

entered letter is a vowel. If a user enters y then your programme should display a message indicating that sometimes y is a vowel, and sometimes y is a consonant. Otherwise, your programme should display a message indicating that the letter is a consonant.

```
##
# VOWEL OR CONONANT
#
# Determine if a letter is a vowel or consonant
#

# Read a letter from the user
letter = input("Enter an letter: ")

# Classify the letter and report the  result
if letter == "a" or letter == "e" or \
    letter == "i" or letter == "o" or \
    letter == "u":
    print("It's a vowel.")
elif letter == "y":
    print("Sometimes it's a vowel.. Sometimes it's a consonant.")
else:
    print("It's a consonant.")
```

Exercise: Name the shape

Write a programme that determines the name of a shape from its number of sides. Read the number of sides from the user and then report the appropriate name as part of a meaningful message. Your programme should support shapes with anywhere from 3 to up to (and including) 10 sides. If a number of sides outside of this range is entered then your programme should display an appropriate error message.

```
##
# NAME THAT SHAPE
#
# Report the name of that shape from the number of sides.
#

# Read the number of sides from the user
nsides = int(input("Enter number of sides: "))

# Determine the name, leaving it empty if an unsupported number of sides
# was entered
name = ""
if nsides == 3:
    name = "triangle"
elif nsides == 4:
    name = "quadrilateral"
elif nsides == 5:
    name = "pentagon"
elif nsides == 6:
    name = "hexagon"
elif nsides == 7:
    name = "heptagon"
elif nsides == 8:
    name = "octagon"
elif nsides == 9:
    name = "nonagon"
elif nsides == 10:
    name = "decagon"

# Display an error message or the name of the polygon
if name == "":
    print("That number of sides is not supported by this program.")
else:
    print("That's a", name)
```

Exercise: Classifying Triangles

A triangle can be classified based on the lengths of its sides as equilateral, isosceles or scalene. All three sides of an equilateral triangle have the same length. And isosceles triangle has two sides that are all the same length and a third side that is a different length. If all of the sides have different lengths then the triangle is scalene.

Write a programme that reads the lengths of the three sides of a triangle from the user. Then display a message that states the triangle's type.

```
##
# CLASSIFYING TRIANGLES
#
# Classify a triangle based on the length of its sides
#

# Read a letter from the user
side1 = float(input("Enter the length of side 1: "))
side2 = float(input("Enter the length of side 2: "))
side3 = float(input("Enter the length of side 3: "))

# Determine the triangle's type
if side1 == side2 and side2 == side3:
    tri_type = "equilateral"
elif side1 == side2 or side1 == side3 or \
        side2 == side3:
    tri_type = "isosceles"
else:
    tri_type = "scalene"

# Display the triangle's type
print("That's a", tri_type, "triangle")
```

Exercise: Find the area

Write a programme that displays the following message:

1) Square
2) Triangle

   Enter a number:

If a user enters 1, then it should ask them for the length of one of its sides and display the area.

If they select 2, it should ask for the base and height of the triangle and displays the area. If they type in anything else, it should give them a suitable error.

```
print("1) Square")
print("2) Triangle")
print()
menuselection = int(input("Enter a number: "))
if menuselection == 1:
    side = int(input("Enter the length of one side: "))
    area = side*side
    print("the area of your chosen shape is: ", area)
elif menuselection == 2:
    base = int(input("Enter the length of the base: "))
    height = int(input("Enter the height of the triangle: "))
    area = (base * height) / 2
    print("the area of your chosen shape is: ", area)
else:
    print("Incorrect option selected.")
```