PROJECT: SPREADSHEET ANALYSIS WITH PYTHON

CFG Course -Introduction to Python & Apps

Final Presentation 09.02.23

Created by Sazia Afrin

EXPLORED THE FOLLOWING MODULES:

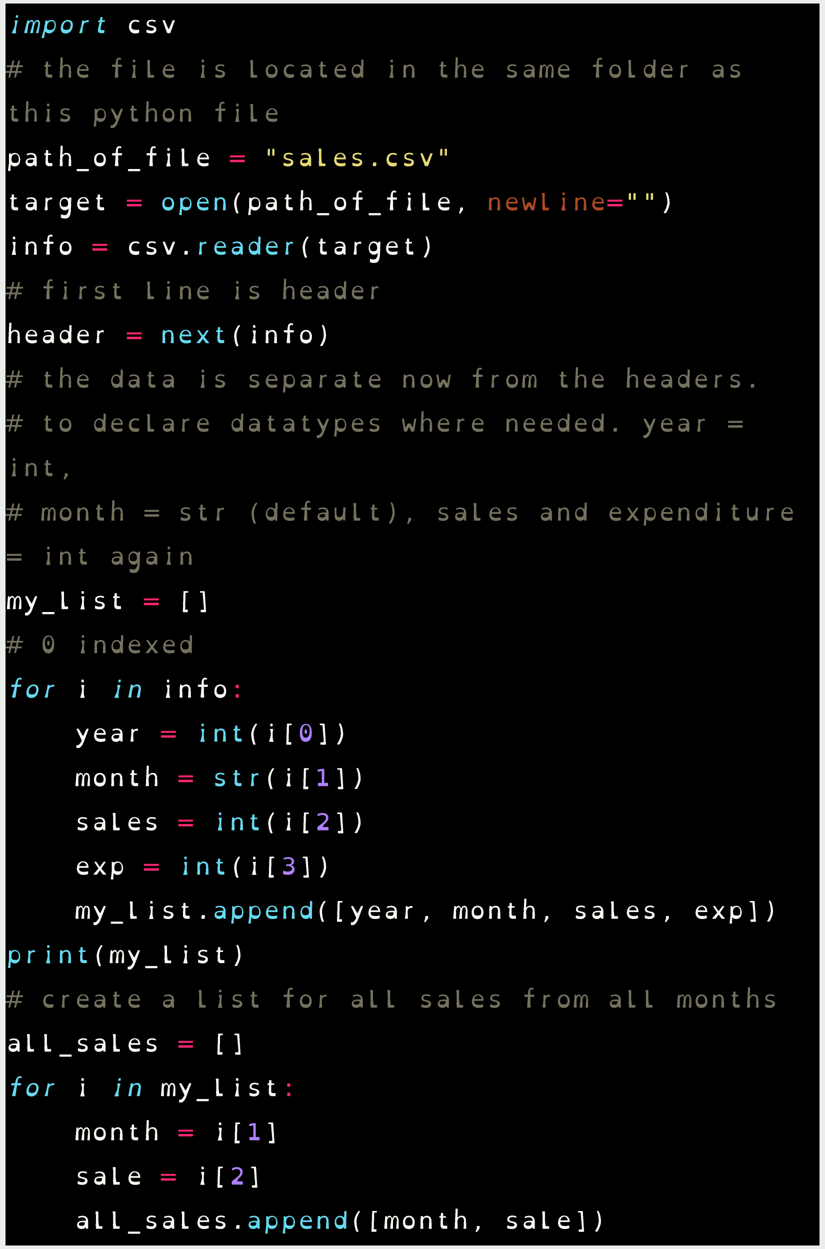
- Built-in CSV module
- Pandas
- NumPy
- Matplotlib

TO ANALISE THE DATA AS PER THE PROJECT BRIEF BELOW

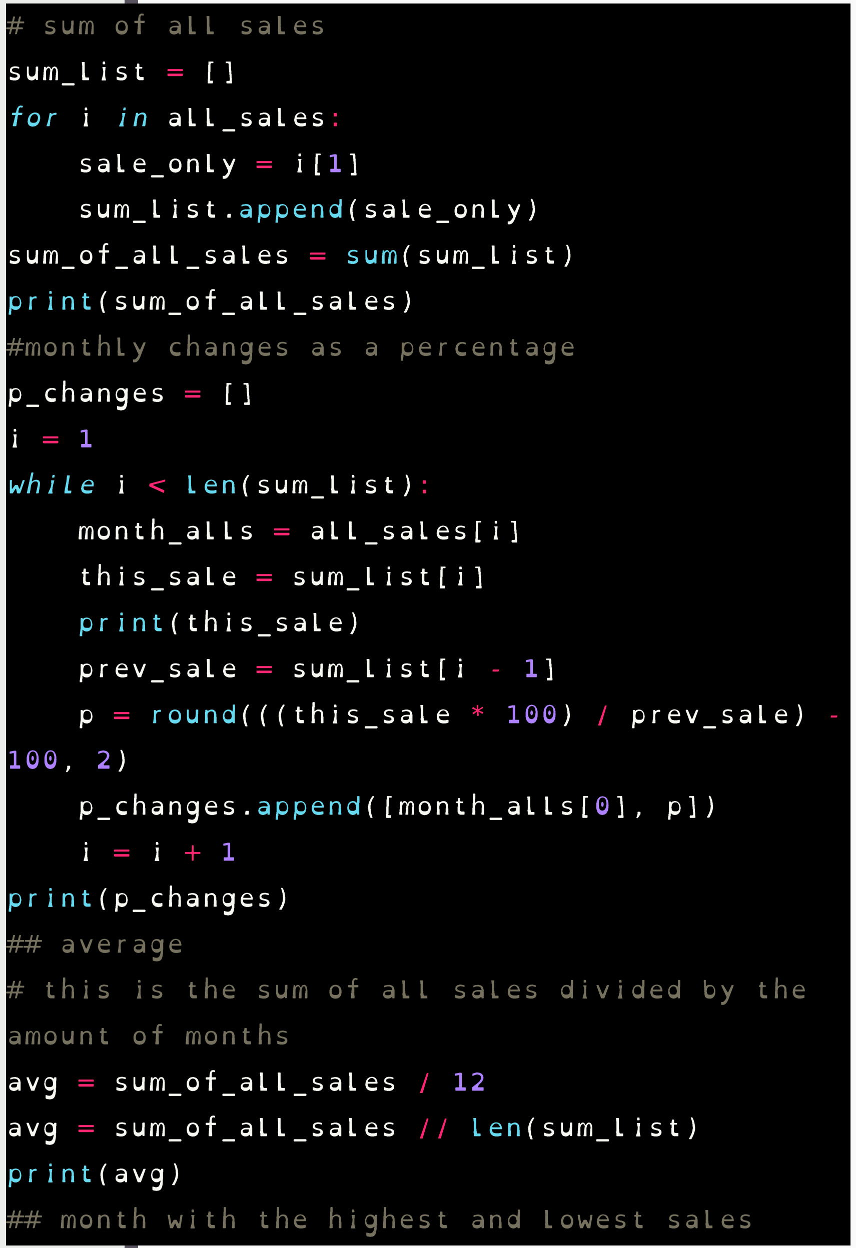
1. READ THE DATA FROM THE SPREADSHEET
2. COLLECT ALL THE SALES FROM EACH MONTH INTO A SINGLE LIST
3. OUTPUT THE TOTAL SALES ACROSS ALL MONTHS
4. OUTPUT A SUMMARY OF THE RESULTS TO A SPREADSHEET
5. CALCULATE THE FOLLOWING:
   - MONTHLY CHANGES AS A PERCENTAGE
   - THE AVERAGE
   - MONTHS WITH THE HIGHEST AND LOWEST SALES
6. USE A DATA SOURCE FROM A DIFFERENT SPREADSHEET

THE BUILT-IN CSV LIBRARY
[PROVIDES FUNCTIONALITY TO BOTH READ FROM AND WRITE TO CSV FILES]

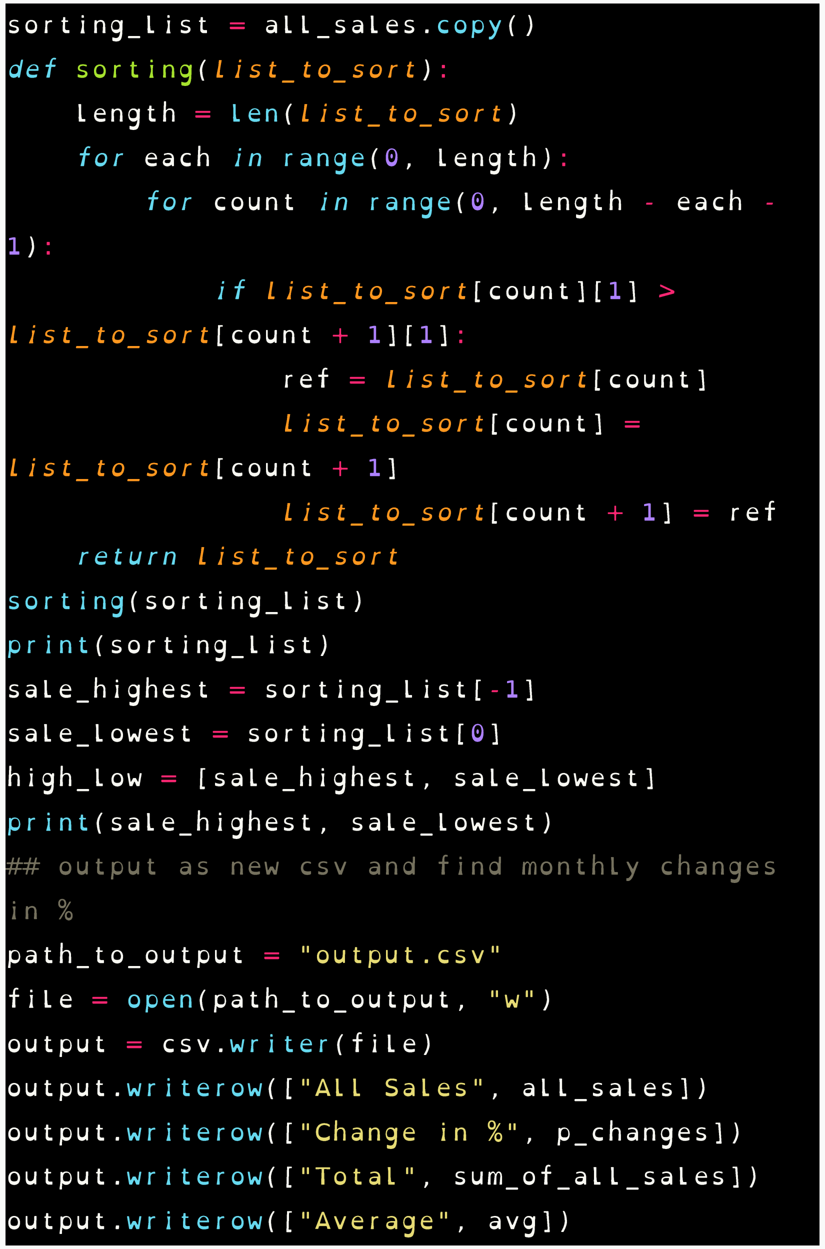
REQUIRED TASKS COMPLETED IN BUILT-IN CSV MODULE



```
import csv
# the file is located in the same folder as
this python file
path_of_file = "sales.csv"
target = open(path_of_file, newline="")
info = csv.reader(target)
# first line is header
header = next(info)
# the data is separate now from the headers.
# to declare datatypes where needed. year =
int,
# month = str (default), sales and expenditure
= int again
my_list = []
# 0 indexed
for i in info:
    year = int(i[0])
    month = str(i[1])
    sales = int(i[2])
    exp = int(i[3])
    my_list.append([year, month, sales, exp])
print(my_list)
# create a list for all sales from all months
all_sales = []
for i in my_list:
    month = i[1]
    sale = i[2]
    all_sales.append([month, sale])
```



```
# sum of all sales
sum_list = []
for i in all_sales:
    sale_only = i[1]
    sum_list.append(sale_only)
sum_of_all_sales = sum(sum_list)
print(sum_of_all_sales)
#monthly changes as a percentage
p_changes = []
i = 1
while i < len(sum_list):
    month_alls = all_sales[i]
    this_sale = sum_list[i]
    print(this_sale)
    prev_sale = sum_list[i - 1]
    p = round(((this_sale * 100) / prev_sale) -
100, 2)
    p_changes.append([month_alls[0], p])
    i = i + 1
print(p_changes)
## average
# this is the sum of all sales divided by the
amount of months
avg = sum_of_all_sales / 12
avg = sum_of_all_sales // len(sum_list)
print(avg)
## month with the highest and lowest sales
```



```
sorting_list = all_sales.copy()
def sorting(list_to_sort):
    length = len(list_to_sort)
    for each in range(0, length):
        for count in range(0, length - each -
1):
            if list_to_sort[count][1] >
list_to_sort[count + 1][1]:
                ref = list_to_sort[count]
                list_to_sort[count] =
list_to_sort[count + 1]
                list_to_sort[count + 1] = ref
    return list_to_sort
sorting(sorting_list)
print(sorting_list)
sale_highest = sorting_list[-1]
sale_lowest = sorting_list[0]
high_low = [sale_highest, sale_lowest]
print(sale_highest, sale_lowest)
## output as new csv and find monthly changes
in %
path_to_output = "output.csv"
file = open(path_to_output, "w")
output = csv.writer(file)
output.writerow(["All Sales", all_sales])
output.writerow(["Change in %", p_changes])
output.writerow(["Total", sum_of_all_sales])
output.writerow(["Average", avg])
```

Results

```
[[2018, 'jan', 6226, 3808], [2018, 'feb', 1521, 3373], [2018, 'mar', 1842, 3965], [2018, 'apr', 2051,
1098], [2018, 'may', 1728, 3046], [2018, 'jun', 2138, 2258], [2018, 'jul', 7479, 2084], [2018, 'aug',
4434, 2799], [2018, 'sep', 3615, 1649], [2018, 'oct', 5472, 1116], [2018, 'nov', 7224, 1431], [2018,
'dec', 1812, 3532]]

[['jan', 6226], ['feb', 1521], ['mar', 1842], ['apr', 2051], ['may', 1728], ['jun', 2138], ['jul', 7479], ['aug',
4434], ['sep', 3615], ['oct', 5472], ['nov', 7224], ['dec', 1812]]

45542

1521

1842

2051

1728

2138

7479

4434

3615

5472

7224

1812

[['feb', -75.57], ['mar', 21.1], ['apr', 11.35], ['may', -15.75], ['jun', 23.73], ['jul', 249.81], ['aug', -40.71],
['sep', -18.47], ['oct', 51.37], ['nov', 32.02], ['dec', -74.92]]

3795

[['feb', 1521], ['may', 1728], ['dec', 1812], ['mar', 1842], ['apr', 2051], ['jun', 2138], ['sep', 3615],
['aug', 4434], ['oct', 5472], ['jan', 6226], ['nov', 7224], ['jul', 7479]]

['jul', 7479] ['feb', 1521]


Process finished with exit code 0
```

Pandas

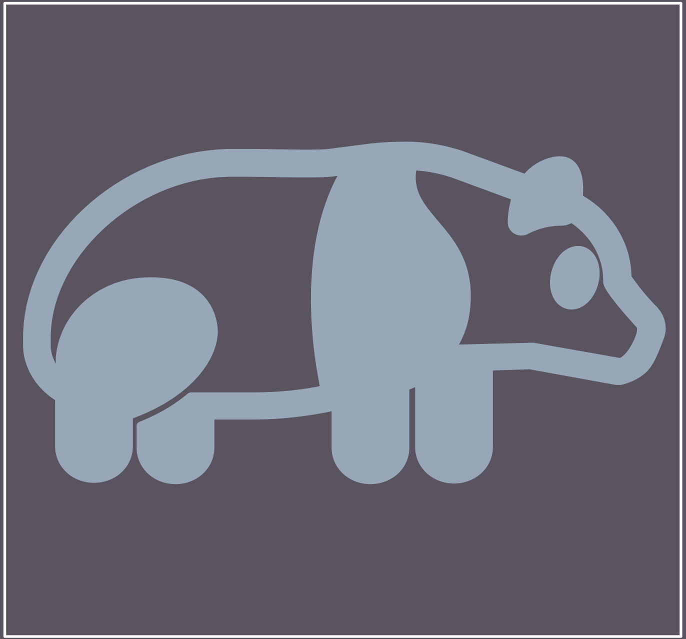
pandas is an open-source library
Pandas - providing high-performance, easy-to-use data structures and data analysis tools for the Python programming language.

Pandas was a lot of fun to use as it was swift, simple and user friendly.

Successfully completed the coding to meet the brief

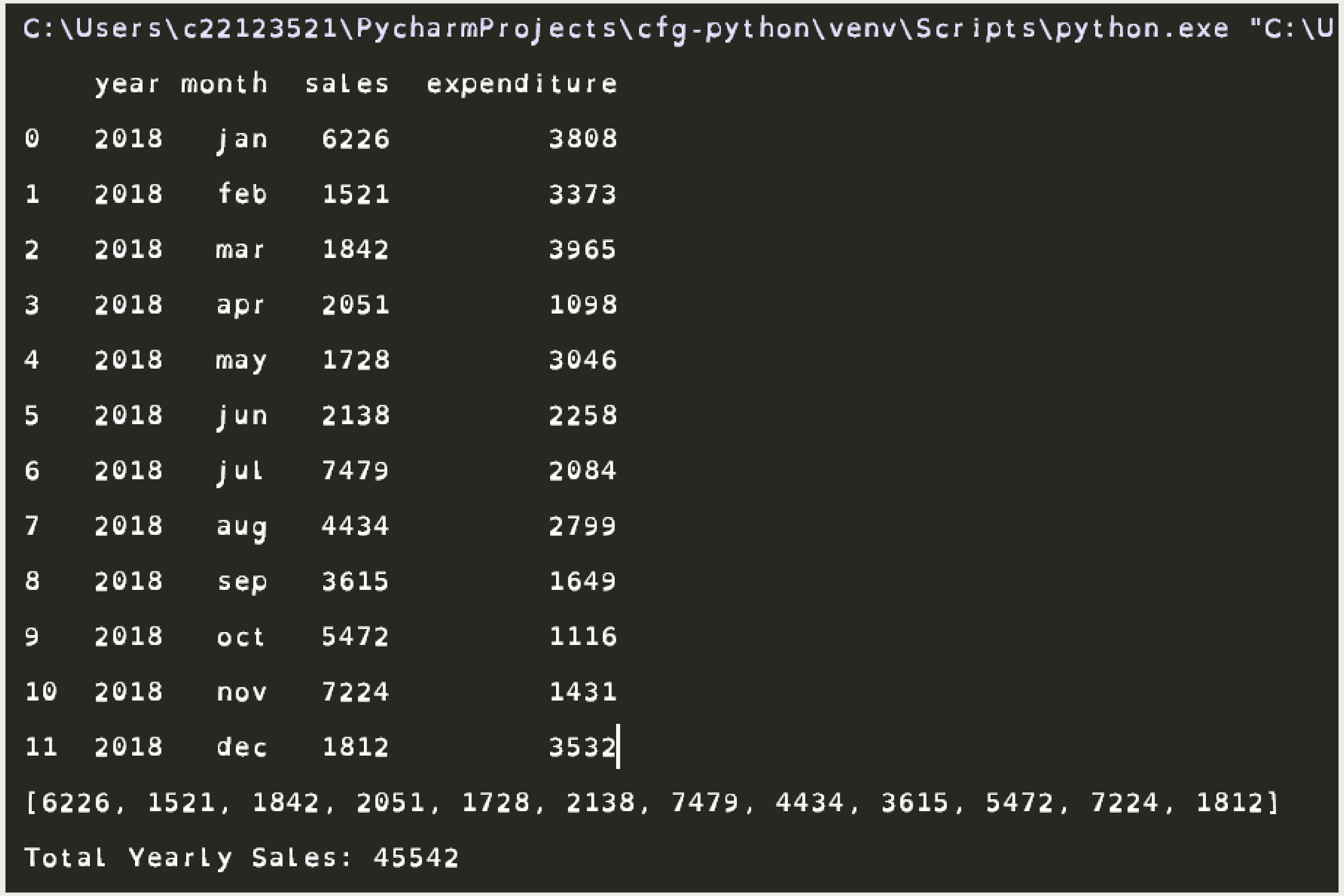
Coding with Pandas

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

#Read the data from the spreadsheet
data = pandas.read_csv("sales.csv")
print(data)
#Collect all of the sales from each month into
a single list
sales_list = data["sales"].to_list()
print(sales_list)
#Output the total sales across all months
print(‘Total Yearly Sales:‘,sum(data["sales"]))

#Output the results to a spreadsheet
data.to_csv("new_sales_data.csv")

#Calculate Monthly changes as a percentage
data["Sales Changes in
%"]=np.round(data["sales"].pct_change()*100,0)
data=data.dropna()
print(data)
#Calculate Monthly changes as average
average = sum(sales_list)// len(sales_list)
print(‘Average Sales:‘,average)

#Calculate Monthly changes as highest
```


Coding with Pandas

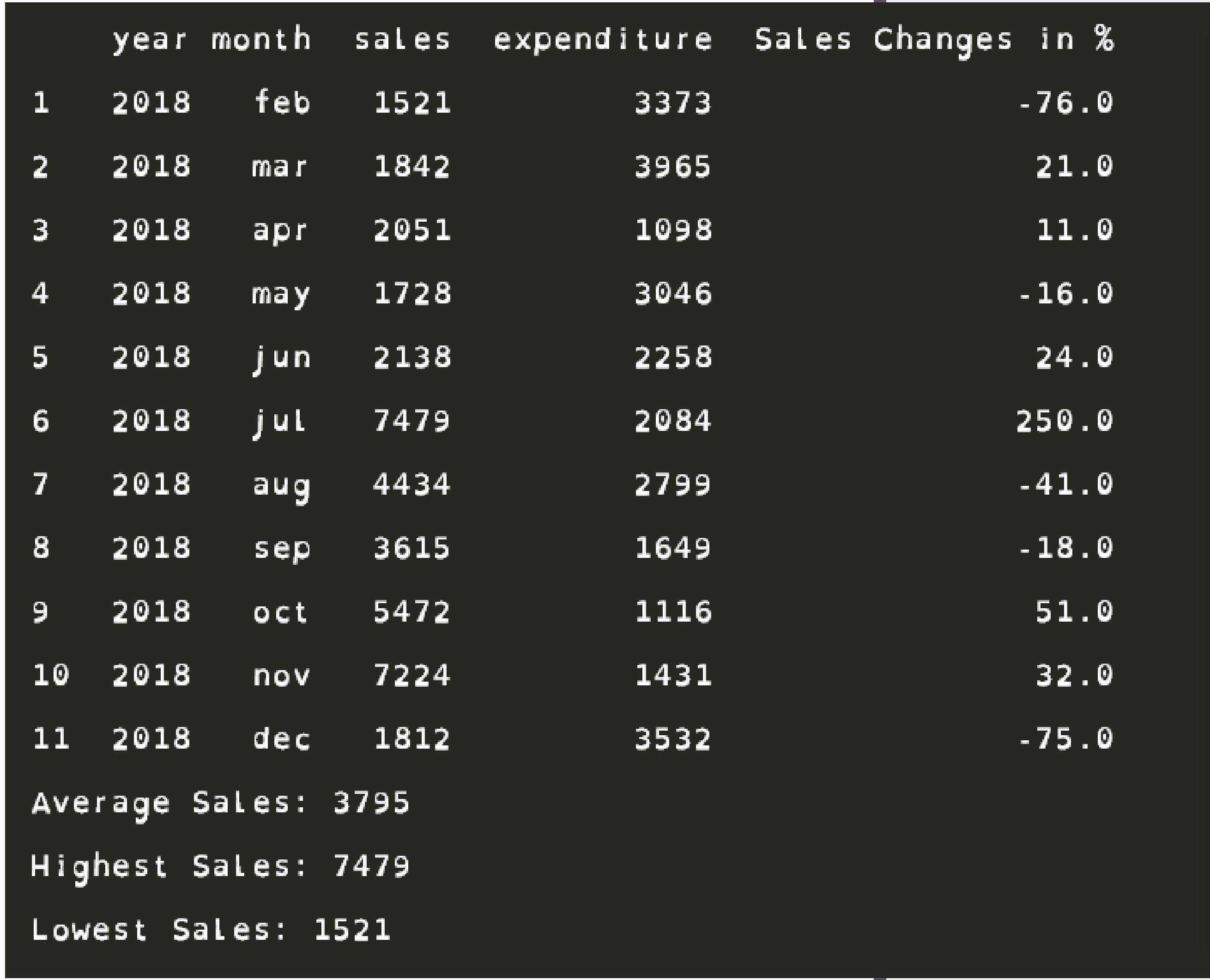
```
#Calculate Monthly changes as highest
print(‘Highest Sales:’,data["sales"].max())

#Calculate Monthly changes as lowest
print(‘Lowest Sales:’,data["sales"].min())

#quarterly Sales summery in a piecart
Q1 = sum(sales_list[0:3])
Q2 = sum(sales_list[3:6])
Q3 = sum(sales_list[6:9])
Q4 = sum(sales_list[9:])
Quarterly_Sales = [‘Q1’,‘Q2’,‘Q3’,‘Q4’]
sales = [Q1, Q2, Q3, Q4]
#highlight the highest sales
explode = [0,0,0.1,0]
plt.pie(sales, labels=Quarterly_Sales,
autopct=‘%.2f%%’, explode=explode)
plt.show()
```

Coding Outcome [1of 2]

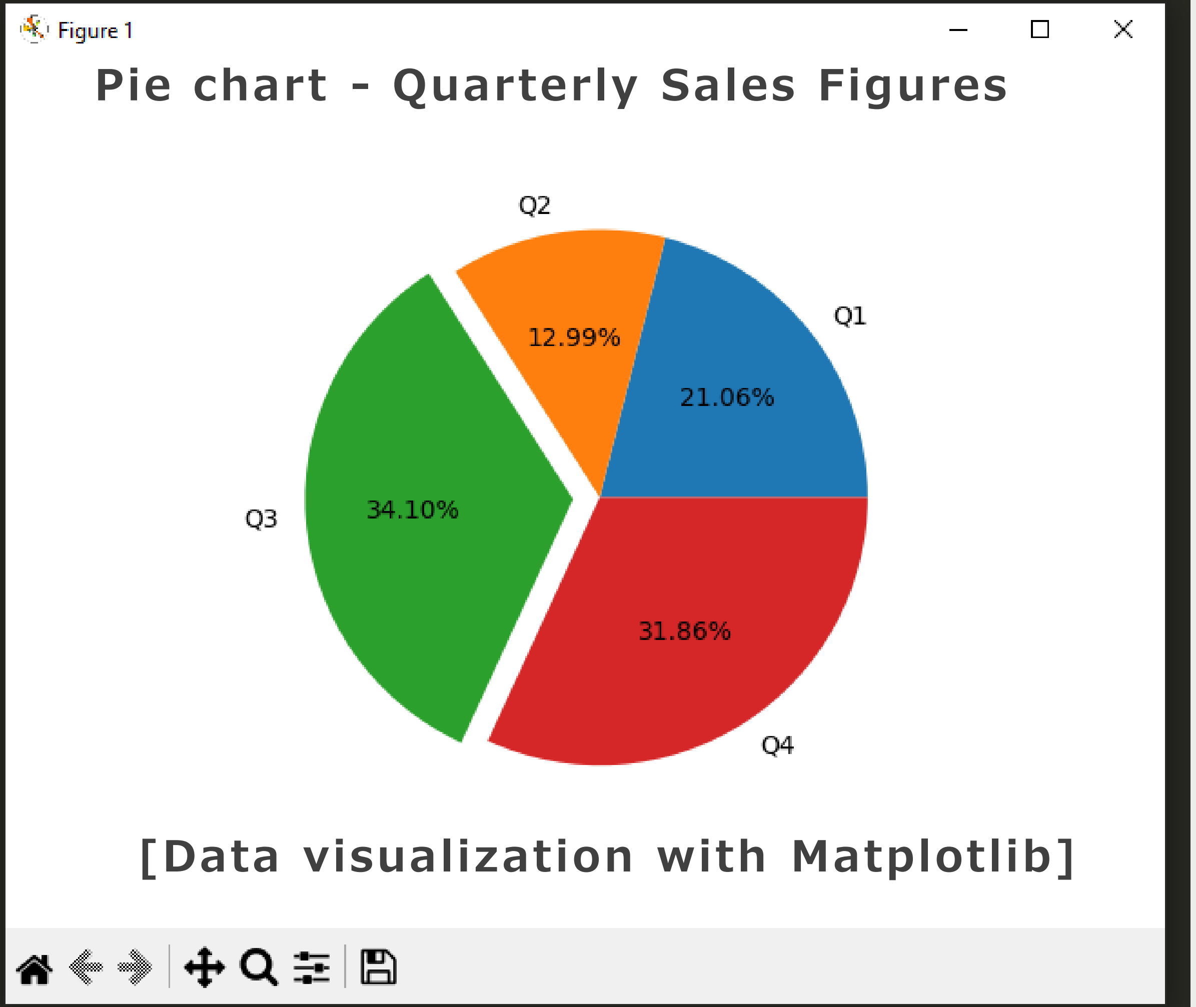


```
C:\Users\c22123521\PycharmProjects\cfg-python\venv\Scripts\python.exe "C:\U
    year month  sales  expenditure
0   2018   jan   6226         3808
1   2018   feb   1521         3373
2   2018   mar   1842         3965
3   2018   apr   2051         1098
4   2018   may   1728         3046
5   2018   jun   2138         2258
6   2018   jul   7479         2084
7   2018   aug   4434         2799
8   2018   sep   3615         1649
9   2018   oct   5472         1116
10  2018   nov   7224         1431
11  2018   dec   1812         3532
[6226, 1521, 1842, 2051, 1728, 2138, 7479, 4434, 3615, 5472, 7224, 1812]
Total Yearly Sales: 45542
```

Coding Outcome [2 of 2]



```
     year  month   sales   expenditure  Sales Changes  in %
1    2018    feb    1521          3373                -76.0
2    2018    mar    1842          3965                 21.0
3    2018    apr    2051          1098                 11.0
4    2018    may    1728          3046                -16.0
5    2018    jun    2138          2258                 24.0
6    2018    jul    7479          2084                250.0
7    2018    aug    4434          2799                -41.0
8    2018    sep    3615          1649                -18.0
9    2018    oct    5472          1116                 51.0
10   2018    nov    7224          1431                 32.0
11   2018    dec    1812          3532                -75.0
Average Sales: 3795
Highest Sales: 7479
Lowest Sales: 1521
```



[Data visualization with Matplotlib]

ALSO EXPLORED – NUMPY & MATPLOTLIB

Matplotlib: Visualization with Python

Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python. Matplotlib makes easy things easy and hard things possible.

NUMPY:

NUMPY CAN BE USED TO PERFORM A WIDE VARIETY OF MATHEMATICAL OPERATIONS ON ARRAYS. IT ADDS POWERFUL DATA STRUCTURES TO PYTHON THAT GUARANTEE EFFICIENT CALCULATIONS WITH ARRAYS AND MATRICES, AND IT SUPPLIES AN ENORMOUS LIBRARY OF HIGH-LEVEL MATHEMATICAL FUNCTIONS THAT OPERATE ON THESE ARRAYS AND MATRICES.

Thank you

Panagiota

&

Zack

GOOD LUCK EVERYBODY.

HAPPY CODING!