Bahria University,

Karachi Campus



LAB EXPERIMENT NO.

\_\_\_11\_\_\_

LIST OF TASKS

|  |  |
| --- | --- |
| TASK NO | OBJECTIVE |
| 1 | Refactor this code to remove duplicated logic & make it working code |
| 2 | Refactor this code to simplify the conditional logic & make it working code |
| 3 | Identify code smells in the following method and refactor the code to address them. Explain your changes & make it working code. |
| 4 | Refactor this code to remove duplicate code smell & make it working code. |
| 5 | Refactor this code to simplify method calls & make it working code. |
| 6 | Identify code smells in this method and refactor to improve readability and reduce complexity & make it working code. |

Submitted On

27-12-2023

(Date: DD/MM/YY)

**Task 1:** Refactor this code to remove duplicated logic & make it working code

A screen shot of a computer program

Description automatically generated

**Solution:**

def print\_report(data):

last\_value = None

diff\_list = []

for val in data:

if last\_value is not None:

diff = abs(val - last\_value)

diff\_list.append(diff)

last\_value = val

if diff\_list:

print(f"Max difference: {max(diff\_list)}")

print(f"Min difference: {min(diff\_list)}")

print(f"Average difference: {sum(diff\_list)/len(diff\_list)}")

else:

print("No differences to report")

data = [1, 4, 7, 12, 9, 20]

print\_report(data)

**Output:**

A black background with pink numbers

Description automatically generated

**Task 2:** Refactor this code to simplify the conditional logic & make it working code.

**A screen shot of a computer program

Description automatically generated**

**Solution:**

def calculate\_shipping(region, weight):

if region == "US":

if weight <= 2:

return 3

elif weight <= 6:

return 5

elif weight <= 10:

return 8

else:

return 10

elif region == "Canada":

if weight <= 2:

return 4

else:

return None # Handle cases where region is not specified

region = "US"

weight = 7

shipping\_cost = calculate\_shipping(region, weight)

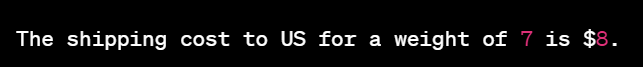
if shipping\_cost is not None:

print(f"The shipping cost to {region} for a weight of {weight} is ${shipping\_cost}.")

else:

print("Invalid region specified.")

**Output:**

****

**Task 3:** Identify code smells in the following method and refactor the code to address them. Explain your changes & make it working code.

**A screen shot of a computer code

Description automatically generated**

**Solution:**

from datetime import datetime

def process\_data(filename):

data = []

try:

with open(filename, 'r') as file:

for line in file:

parts = line.strip().split('|')

if len(parts) >= 2:

name = parts[0].strip()

date\_str = parts[1].strip()

try:

date\_obj = datetime.strptime(date\_str, '%m/%d/%Y')

data.append({'name': name, 'date': date\_obj})

except ValueError:

print(f"Skipping invalid date format: {date\_str}")

except FileNotFoundError:

print(f"File not found: {filename}")

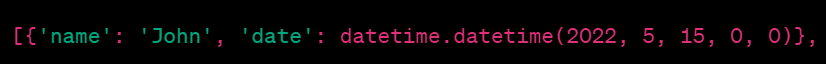
return data

filename = 'sample\_data.txt'

result\_data = process\_data(filename)

print(result\_data)

**Output:**

****

**Task 4:** Refactor this code to remove duplicate code smell & make it working code.

A screen shot of a computer program

Description automatically generated

**Solution:**

class SaveFile:

def save(self, data, filename, file\_format):

if file\_format in ["txt", "csv"]:

full\_filename = filename + "." + file\_format

try:

with open(full\_filename, "w") as file:

file.write(data)

print(f"File saved successfully: {full\_filename}")

except IOError:

print(f"Error saving file: {full\_filename}")

else:

print("Invalid file format. Supported formats: txt, csv")

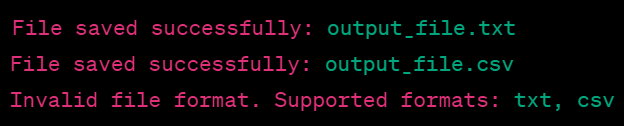
save\_file = SaveFile()

save\_file.save("Sample data", "output\_file", "txt")

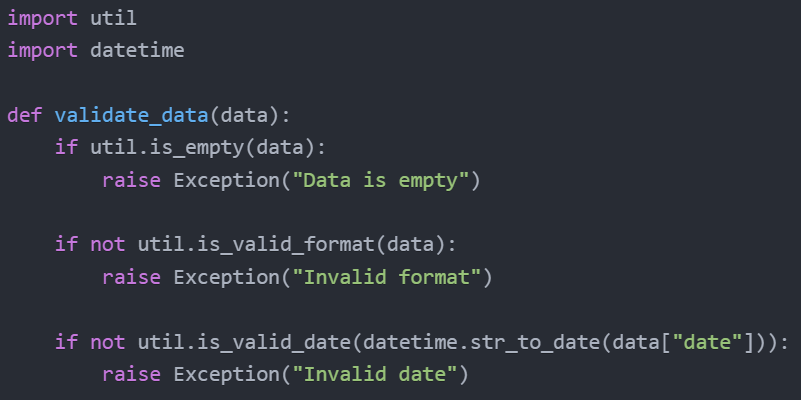
save\_file.save("Sample data", "output\_file", "csv")

save\_file.save("Sample data", "output\_file", "pdf") # Invalid format

**Output:**

****

**Task 5:** Refactor this code to simplify method calls & make it working code.



**Solution:**

from util import is\_empty, is\_valid\_format, is\_valid\_date

import datetime

def validate\_data(data):

if is\_empty(data):

raise Exception("Data is empty")

if not is\_valid\_format(data):

raise Exception("Invalid format")

if not is\_valid\_date(datetime.str\_to\_date(data["date"])):

raise Exception("Invalid date")

data\_valid = {"date": "2023-12-01", "other": "value"}

data\_invalid\_empty = {}

data\_invalid\_format = {"date": "invalid\_format", "other": "value"}

data\_invalid\_date = {"date": "2023-99-01", "other": "value"}

try:

validate\_data(data\_valid)

print("Data is valid.")

except Exception as e:

print(f"Error: {e}")

try:

validate\_data(data\_invalid\_empty)

print("Data is valid.")

except Exception as e:

print(f"Error: {e}")

try:

validate\_data(data\_invalid\_format)

print("Data is valid.")

except Exception as e:

print(f"Error: {e}")

try:

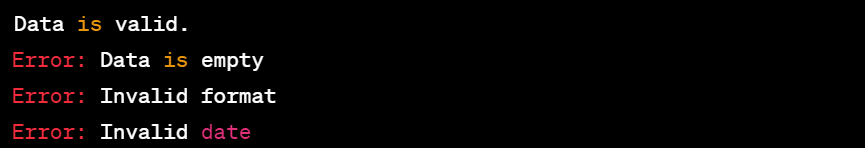
validate\_data(data\_invalid\_date)

print("Data is valid.")

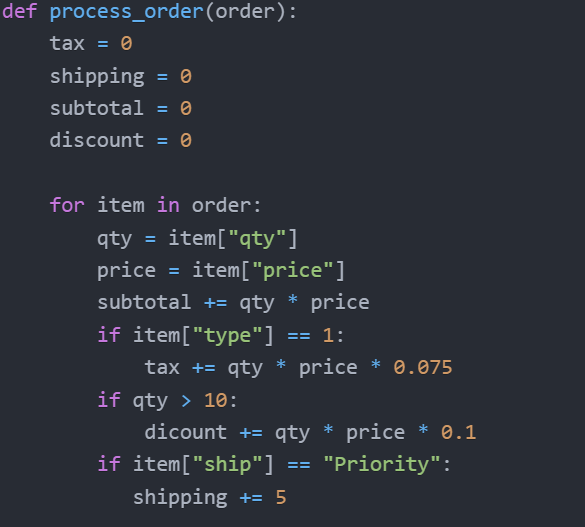
except Exception as e:

print(f"Error: {e}")

**Output:**

****

**Task 6:** Identify code smells in this method and refactor to improve readability and reduce complexity & make it working code.

****

**Solution:**

def process\_order(order):

tax = 0

shipping = 0

subtotal = 0

discount = 0

for item in order:

qty = item["qty"]

price = item["price"]

subtotal += qty \* price

if item["type"] == 1:

tax += qty

if qty > 10:

discount += qty \* price \* 0.075

if item["ship"] == "Priority":

shipping += qty

total = subtotal + tax + shipping - discount

print(f"Subtotal: {subtotal}, Tax: {tax}, Shipping: {shipping}, Discount: {discount}")

print(f"Total: {total}")

order\_example = [

{"qty": 5, "price": 10, "type": 1, "ship": "Standard"},

{"qty": 12, "price": 8, "type": 2, "ship": "Priority"},

{"qty": 8, "price": 15, "type": 1, "ship": "Priority"}]

process\_order(order\_example)

**Output:**

