Bahria University,

Karachi Campus



LAB EXPERIMENT NO.

**11**

LIST OF TASKS

|  |  |
| --- | --- |
| TASK NO | OBJECTIVE |
| 01 | **Refactor this code to remove duplicated logic & make it working code.** |
| 02 | **Refactor this code to simplify the conditional logic** & **make it working code.** |
| 03 | **Identify code smells in the following method and refactor the code to address them. Explain your changes & make it working code.** |
| 04 | **Refactor this code to remove duplicate code smell & make it working code.** |
| 05 | **Refactor this code to remove duplicate code smell & make it working code.** |
| 06 | Identify code smells in this method and refactor to improve readability and reduce complexity & make it working code. |

Submitted On:

**20/12/2023**

**TASK # 1: Refactor this code to simplify the conditional logic & make it working code.**

**SOLUTION:**

def calculate\_differences(data):

    return [abs(data[i] - data[i-1]) for i in range(1, len(data))]

def print\_report(data):

    diff\_list = calculate\_differences(data)

    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, 9, 7, 10, 12]

print\_report(data)

A black text on a white background

Description automatically generated

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

**SOLUTION:**

def calculate\_shipping(region, weight):

    shipping\_rates = {

        "US": {2: 3, 6: 5, 10: 8, float('inf'): 10},

        "Canada": {2: 4, float('inf'): None}

    }

    region\_rates = shipping\_rates.get(region)

    if region\_rates is not None:

        for max\_weight, rate in sorted(region\_rates.items()):

            if weight <= max\_weight:

                return rate

    return None

region = "US"

weight = 12

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

if shipping\_cost is not None:

    print(f"Shipping cost to {region}: ${shipping\_cost}")

else:

    print(f"Shipping to {region} not available for weight {weight}")



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

**SOLUTION:**

def process\_data(filename):

    processed\_data = []

    with open(filename, 'r') as file:

        for line in file:

            name, date\_str = map(str.strip, line.split('\t'))

            month, day, year = map(int, date\_str.split('-'))

            entry = {'name': name, 'month': month, 'day': day, 'year': year}

            processed\_data.append(entry)

    return processed\_data

filename = "data.txt"

result = process\_data(filename)

for entry in result:

    print(entry)

A number with black text

Description automatically generated with medium confidence

There are several code smells in the provided method. Let's identify and address them:

**Unclosed File:**

**Missed Function Call:**

There is a missing function call in date\_str = parts[1].strip. It should be date\_str = parts[1].strip().

**Inconsistent Indentation:**

**Magic Values:**

The code contains magic values, such as the tab character ('\t'). It's better to use named constants or variables to make the code more readable and maintainable.

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

**SOLUTION:**

**class SaveFile**:

    def \_save\_to\_file(self, data, filename, file\_extension):

        with open(filename + file\_extension, "w") as file:

            file.write(data)

        print(f"Data saved to {filename}{file\_extension}")

    def save\_txt(self, data, filename):

        print("Printing data:")

        print(data)

        self.\_save\_to\_file(data, filename, ".txt")

    def save\_csv(self, data, filename):

        print("Printing data:")

        print(data)

        self.\_save\_to\_file(data, filename, ".csv")

save\_file\_instance = SaveFile()

data\_to\_save = "This is some data."

save\_file\_instance.save\_txt(data\_to\_save, "example\_txt")

save\_file\_instance.save\_csv(data\_to\_save, "example\_csv")

A screen shot of a computer code

Description automatically generated

**TASK # 5: Refactor this code to remove duplicate code smell & make it working code.**

**SOLUTION:**

import util

import datetime

def validate\_data(data):

    if util.is\_empty(data):

        raise ValueError("Data is empty")

    if not util.is\_valid\_format(data):

        raise ValueError("Invalid format")

    date\_str = data.get("date")

    if not util.is\_valid\_format(datetime.str\_to\_date(date\_str)):

        raise ValueError("Invalid date")

data\_to\_validate = {

    "field1": "some value",

    "date": "2023-12-17",

}

try:

    validate\_data(data\_to\_validate)

    print("Data is valid.")

except ValueError as ve:

    print(f"Validation failed: {ve}")



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

**SOLUTION:**

* **Named Constants:**

Introduced named constants for tax rate, discount rate, and priority shipping cost to improve readability.

* **Clearer Variable Names:**

Renamed variables to provide clearer meanings, such as changing subtotal to total\_price.

* **Consolidated Calculations:**

Consolidated calculations to reduce redundancy and improve code clarity.

* **Returned Order Summary:**

Instead of separate variables, returned a dictionary containing various components of the order summary.

def process\_order(order):

    TAX\_RATE = 0.10

    DISCOUNT\_RATE = 0.2

    PRIORITY\_SHIPPING\_COST = 5

    total\_price = 0

    total\_tax = 0

    total\_discount = 0

    total\_shipping = 0

    for item in order:

        qty = item["qty"]

        price = item["price"]

        subtotal = qty \* price

        total\_price += subtotal

        if item["type"] == 1:

            total\_tax += subtotal \* TAX\_RATE

        if qty > 10:

            total\_discount += subtotal \* DISCOUNT\_RATE

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

            total\_shipping += PRIORITY\_SHIPPING\_COST

    return {

        "subtotal": total\_price,

        "tax": total\_tax,

        "discount": total\_discount,

        "shipping": total\_shipping,

        "total": total\_price + total\_tax - total\_discount + total\_shipping

    }

order\_items = [

    {"qty": 7, "price": 15, "type": 1, "ship": "Standard"},

    {"qty": 10, "price": 6, "type": 2, "ship": "Priority"},

]

order\_summary = process\_order(order\_items)

print(order\_summary)

